Engineering Agent Systems for Decision Support [*]

Sascha Ossowski [1], Josefa Z. Hernández [2], Carlos A. Iglesias [3], Alberto Fernández [1]

[1]AI Group, University Rey Juan Carlos, Campus de Móstoles s/n, E-28933 Madrid,
{s.ossowski,al.fernandez}@escet.urjc.es

[2]AI Dpt., Tech. University of Madrid, Campus de Montegancedo s/n, E-28660 Madrid
phernan@dia.fi.upm.es

[3] Technical Innovation Dpt., Germinus XXI, Gran Vía 1 - 2º Izq, E-28013 Madrid
cif@germinus.com

Abstract. This paper discusses how agent technology can be applied to the design of advanced Information Systems for Decision Support. In particular, it describes the different steps and models that are necessary to engineer Decision Support Systems based on a multiagent architecture. The approach is illustrated by a case study in the traffic management domain.

1 Introduction

Decision Support Systems (DSS) are information systems that provide assistance to humans involved in complex decision-making processes. Early DSS were conceived as simple databases for storage and recovery of decision relevant information [19]. Despite some intends to improve organisation and presentation of such data by means of additional deductive facilities, it soon became apparent that the key problem for a decision-maker is not such much to *access* pertinent data but rather to *understand* its significance. So, the fundamental task for modern DSS is to help decision-makers in building up and exploring the implications of their judgements, so as to take decisions based on understanding [6].

DSS are particularly relevant in domains where human operators have to take decisions regarding the management of complex industrial or environmental processes: controlling road traffic flows [7], managing dams in a watershed basin [8], or administering large computer networks [20]. The increasing data volume and the decreasing time horizon within which control decisions have to be taken, have generated a need for DSS; they evaluate data about the system state, collected either directly or through real-time databases, so as to warn operators of any undesired evolution and to answer their questions concerning potential reasons, effects and countermeasures [4].

Recently, the concept of *Intelligent DEcision-making Assistants* (IDEA) has been proposed [18]. These are intelligent agents that render support to their human opera-

[*] Work supported by the Spanish Ministry of Science and Technology (MCyT) under grant TIC2000-1370-C04

tors in the various stages of their decision-making processes by means of flexible dialogues. Suppose a DSS that assists operators in a traffic control centre to generate and choose among alternative signal plans for traffic control devices (Variable Message Panels, Traffic lights etc.), so as to assure a smooth flow of traffic, as well as to avoid and/or overcome potentially critical situations. It is important for such a system to support a *reactive* mode of interaction. Once an operator detects some abnormal system parameters or receives alarm messages (e.g. from traffic observers), she initiates a dialogue with the DSS in order to prepare her decision. There are several questions that she will put forward to the agent, e.g. respecting the cause of the current situation (*What is happening*?), the reason for it (*Why is it happening*?), as well as action alternatives (*What can be done*?) and their potential effect of different alternatives (*What may happen if* ?). These questions will trigger several processes in the course of which the agent autonomously gathers relevant data from the different information sources available. On the basis of this information, it will apply its domain knowledge to generate answers to the questions, taking into account rationality constraints with respect to the operator's objective of minimising negative impacts on the part of the road network that she is responsible for. Another scenario is given when the intelligent assistant monitoring the road network detects symptoms of deterioration of the traffic situation or of dysfunctional active signal plans (initiated, or not adequately updated, for instance, by a less trained or experienced operator). In this case, it is to *proactively* issue warnings, initiating a dialogue with the decision-maker, in the course of which the latter explores the reasons and implications of that warning in a flexible and intuitive manner.

This paper discusses how agent technology can be applied to the design of advanced DSS. In particular, setting out from our past experience in this field, it describes the different steps and models that are necessary to engineer intelligent multi-agent systems for Decision Support. It is organised as follows: Section 2 describes how agent-based methodologies can be used for a principled design of IDEAs. Section 3 presents a case study respecting analysis, design, implementation and operation of such systems. We conclude this paper pointing to present and future lines of research.

2 Engineering IDEAs

Current research in Agent Oriented Software Engineering examines principled ways for constructing complex software systems based on the agent metaphor [10,12,21]. Often, traditional *object-oriented methodologies* such as UML [11] are extended to account for the characteristics of agent systems, paying particular attention to the modelling of agent interactions and agent architecture. Such methodologies also appear to be an adequate choice for the construction of IDEAs, but need to be complemented by techniques from the *knowledge engineering* field, as decision support agents draw heavily upon complex, knowledge-based reasoning processes. To this respect, the well-known *CommonKADS* approach [1], that supports advanced knowledge modelling, is particularly relevant.

In the sequel we describe how to apply agent-oriented software engineering techniques from both fields in order to engineer intelligent assistants for decision support. First, we use the UER modelling technique, an extension of UML, for analysis. Then, a *CommonKADS*-like approach is used for the identification of basic reasoning tasks of IDEAs. Finally, we show how to obtain and structure reasoning methods based on such a knowledge-oriented approach [5].

2.1 Analysis

In this section we describe how to obtain a conceptualisation of IDEAs on the basis of the UER (User-Environment-Responsibility) technique [9]. It is particularly convenient for our purposes, as it allows for both, monolithic decision support agents as well as for IDEAs that are themselves based on a multiagent architecture which, according to our experiences, will often be the case. The UER technique analyses the system from three different perspectives:

– *User-Centred Analysis*. The potential users (called actors) of the agent system are identified, together with their possible tasks or functions. The result of this analysis is the set of use cases. This analysis answers the question: What are the possible uses of the multiagent system?
– *Environment-centred Analysis*. Agents are situated in an environment, and this environment needs to be modelled. In particular, we are interested in modelling how the system can act and react to this environment. The result of this analysis is the set of reaction cases. This analysis answers the question: How does the agent system react to the environment?
– *Responsibility-driven Analysis*. In contrast with usual software systems, agent systems can act proactively. The user can desire that the system has some responsibilities, that is, the user can assign some goals or responsibilities to the system and the system carries out these responsibilities without a direct demand. This analysis answers the question: What are the goals of the system? The main difference between *goal cases* and use cases, is that the latter show how the system gives an answer to a user request, while the former show how the system behaves when some condition is fulfilled.

UER defines different UML stereotypes (see Fig. 1) for every modelling element: use cases are UML standard ellipses, goal cases are ellipses with wider lines, reactive cases are ellipses with discontinued lines, environment objects have an irregular form, human actors are standard UML actors and software actors are square headed actors). In the sequel, we model a generic Intelligent Decision Support Agent System in these terms. Fig. 1 summarises the results.

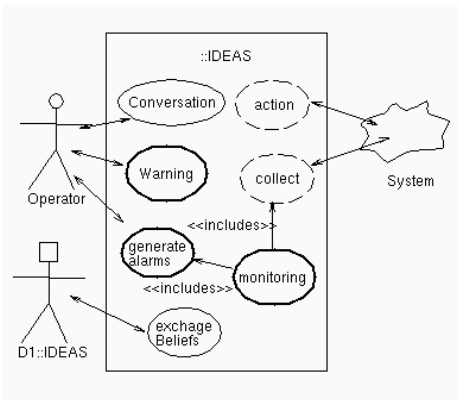


Fig. 1. UER Model of IDEAs

Not surprisingly, in the general case,

user-centred analysis identifies at least two actors: the decision-maker and the decision support system, i.e. the IDEA. The most relevant generic use cases that these actors are involved in include:

- *Conversation*: The user requests some explanation from the system.
- *Exchange-beliefs*: another DSS interchanges plans with the systems to get a broader understanding of the situation.

The use case conversation can be detailed as:

- *Why*: the decision-maker requests an explanation of information provided by the DSS.
- *What*: the decision-maker requests relevant data to take a decision.
- *What-if*: the decision-maker asks to evaluate the results of a decision and its consequences**.**
- *Suggestions*: the decision-maker requests possible actions to be taken**.**

Depending on the particular domain about which the IDEA is knowledgeable, the *environment-centred analysis* identifies relevant environment objects, and every possible event generated from these objects and possible actions carried out on them. For example, in a financial DSS, we could identify an object *share*. The possible input events are new values of this share, and the possible actions to buy or to sell.

Finally, in the *goal-driven analysis* we need to identify the autonomous behaviours of an IDeA. Its most important responsibilities usually are:

- *Monitoring*: observe the environment and detect problematic behaviours;
- *Alarm generation*: raise alarms if there is a critical situation;
- *Warning*: warning respecting undesired consequences of "bad" actions and potentially suggesting better ones.

2.2 Task Design

In methodologies that go back to the knowledge engineering field, a task is usually conceived as an abstract description of how the world (or an agent's "mental model" of it) needs to be transformed in order to achieve a desired behaviour or functionality. In this section we identify the different tasks that an IDEA needs to cope with so as to be able to behave successfully in the aforementioned conversation cases.

To generate answers for the different classes of questions in our conversation framework, we have identified essentially four tasks.

- *Problem identification.* From the analysis of the information received from a communication infrastructure or directly from the operator, an IDEA classifies the state of the monitored system.
- *Diagnosis*. The presence of unacceptable events or situations requires an explanation in terms of causal features of the situation.
- *Action planning*. Once a problem has been identified, a possible sequence of actions applicable on the causes may be established.
- *Prediction*. the consequences of events and operator actions are simulated.

Suppose some system components *S*, some external events *E* and operator actions *A*. By combining the above tasks in different manners, several questions that a decision-maker typically faces can be answered. For instance:

- "What is happening in *S* ?"*: problem identification* + *diagnosis*.
  A diagnosis *D* for some potential malfunction is produced.
- "What to do on *D* in *S* ?": *action planning* + *prediction*
  Decision options are shaped and their potential effects evaluated.
- "What may happen if *E* in *S* ?": *prediction* + *problem identification* + *diagnosis*
  Potential future problems in evolution of the system are identified.
- "What to do if *E* in *S* ?": *prediction* + *problem ident.* + *diagnosis* + *planning*
  Decision options respecting potential future problems are outlined.

Still, in complex domains, like the ones that DSS are usually being applied to, the system designer often has to deal with different types of *a priori* distribution [17], which makes it unfeasible to cope directly with these tasks. This distribution may be implied by physical requirements, as many environments show a natural spatial distribution. DSS that assist decision-makers in managing the spill gates of several dams in a watershed basin, for instance, rely on data on weather conditions and water levels from sensors that are geographically distributed. But distribution may also be implied by organisational requirements. To be successful, a DSS aimed at helping managers to maintain a smooth flow of work in a company will have to respect the context of its existing human organisation, e.g. its *a priori* distribution of responsibilities and tasks. Another source of distribution is the availability of knowledge. If, as in the road traffic management domain, a DSS relies on the knowledge elicited from human experts, and by experience these experts conceive traffic behaviour in terms of certain problem areas, then the system will have to reflect this a priori distribution.

A common way of dealing with these issues it to conceive an IDEA itself as a *multiagent* system, where each distributed entity is controlled by an agent. By consequence, any of the aforementioned tasks of problem identification, diagnosis, action planning and prediction can be performed locally by each agent within the multiagent system that makes up an IDeA. These local tasks will be of less complexity, but they are also interdependent: the cause of a rising water level at a certain dam may be the opening of a spill gate further upstream, the predicted completion of a work process in a company will depend on the timeliness of any of its work cells, and the effectiveness of action plan in road traffic management will rely on a globally consistent use of control devices. The co-ordination task, that such a multiagent system faces, refers to the management of these dependencies between local tasks.

2.3 Method Design

Most knowledge-oriented methodologies make use of the concept of problem-solving methods in order to cope with tasks. In particular, such methods indicate *how* a task is achieved, by describing the different steps by which its inputs are transformed into its outputs. The problem-solving process associated to a task is structured: each of its steps may set up several subtasks, which again are to be solved by simpler methods etc, until some elementary tasks can be achieved directly. In the sequel, we identify different such “reasoning skeletons” that our IDEAs will need to apply so as to cope effectively with the tasks identified in the previous section.

Problem identification methods

A classification method with two options may be applied:

- Identification of a reference situation and classification of the differences between the reference and the current situation.
- Direct classification of the current situation based on a predefined taxonomy where problems of different types are described.

The first approach requires: (1) to infer from the current situation the evolution of parameters consistent with the functional and structural constraints which optimises a collection of predefined criteria (e.g. in a given congested situation an ideal assignment of traffic flows could be identified, adapted to the available capacity of the network and to the traffic demand between entries and exits to the network), and (2) to classify the differences between the observed situation and the resulting class of situations according to a hierarchy similar to the one previously commented.

The first approach, then, applies a method in two steps. The first step derives a possible new state from the current situation that may be supported by an *ad hoc* procedure, adapted to the characteristics of the domain model. For the second subtask a primary representation based on rules and/or frames may be applied in a hierarchical *establish & refine* model [3]. The second approach is similar to the first one, but in this case a complete description of the situation is required, not only the differences with the reference situation.

Diagnosis methods

This task infers a collection of causes explaining the problems identified by the previous one. Several methods may be directly applied: (1) The *classification* method, which extends problem type frames by additional cause attributes in such a way that once a problem pattern has been selected, the cause features assumed for this problem type are assumed. (2) A version of the *cover & differentiate* method [16] where a hierarchical approach to an explanatory set of causes is generated through the following reasoning steps: (i) from the attributes of the type of problem detected a collection of possible causes may be inferred *covering* these values (i.e. if the causes inferred are true the problem feature values are also true), (ii) since this first set of causes may be too large, a deeper analysis to *differentiate* subsets explanatory enough is necessary. To do this, knowledge about the individual impact of sets of causes should be used to decide which causes may be erased. To obtain those enlarged impact estimates knowledge relating cause sets and impacts must be available in terms of relations or in terms of simulator.

A hierarchy of conjunctive cause sets where the resulting impacts are known may be established from aggregated sets to more disaggregated ones, every node with the condition that there is a method to evaluate the resulting effects of the causes included in the node.

This second analysis may be done in several steps in such a way that several reasonable partitions in subsets of interacting causes are selected in the hierarchy as potentially explanatory. It may be followed by a step of selection of the more efficient subset (i.e. which explains better the problem features) that may be also partitioned until a minimal explanatory set is found in the hierarchy. For instance, in the domain of traffic it may be identified a problem with three potential congested critical sections, the initial candidate causes are sets of paths between entry and exit points in the area network passing through the congested points. The process of differentiation aims to identify, at the different points, the participation in the traffic excess of a subset of those paths which provide the significant part of this excess and hence, the paths where the traffic flow must be withdrawn using control devices (messages or traffic lights) to reduce the congested demand.

Action planning methods

After the problem identification and diagnosis tasks, some scenarios of causes of problems have been deduced together with its impacts. The *action planning* task must generate a consistent set of actions oriented toward the reduction or elimination of causes and/or toward the reduction of impact damages where no possible cause reduction may be produced.

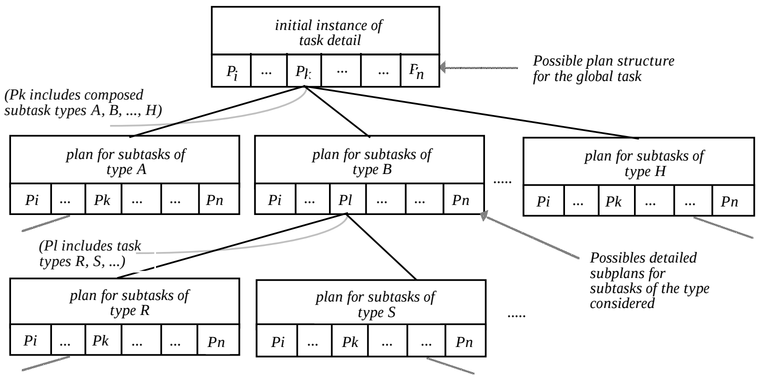
Specifying this task in a general way requires defining the elementary actions that will be the basis for definition of acceptable decision plans together with their models. The action planning task may be performed by a method integrating consistent sequences of actions to transit to a situation where most damages have been alleviated and all the problem causes have been cancelled.

The general reasoning method to deal with this functionality is a planner. However, it is assumed that the area of expertise to be modelled embodies structured knowledge criteria that are sufficiently precise to generate a plan in a more simple way, using classification reasoning on predefined plans and subplans. According to this assumption, it is reasonable to apply a stepwise reasoning method derived from the routine design method proposed by [2].

The declarative knowledge may be organised with collections of plans capable to perform subtasks that may be integrated to build plans aiming to perform more complex tasks. A plan may be defined by sequences of elementary actions or may include other subtasks supported by other collections of plans. The method using this domain knowledge is a type of *skeletal plan refinement* supporting a progressive refinement of the subtasks from intermediate partial versions of the plan in the following steps:

– A basic method M is applied which introduces a collection of consistent actions together with a decomposition of the general task in plans of actions a_i integrated with some subtasks not yet formulated as plans $< a_i, a_j, ..., T_l, T_m, ..., a_s >$ where T_l, T_m are subtasks with no plan of elementary actions specified.

– For every subtask not yet described in terms of basic actions *(a_i, a_j,...)* such as T_l or T_m an inference step of the same type is applied by using an instantiation of M with other premises and domain knowledge. A tree exploring different steps of plan detail is produced until detailed plans are obtained or an application of M fails and a backtracking is produced to alternative task decompositions.



initial instance of task detail
Pi ... Pl ... ... Pn
Possible plan structure for the global task
(Pk includes composed subtask types A, B, ..., H)
plan for subtasks of type A
Pi ... Pk ... ... Pn
plan for subtasks of type B
Pi ... Pl ... ... Pn
.....
plan for subtasks of type H
Pi ... Pk ... ... Pn
(Pl includes task types R, S, ...)
Possibles detailed subplans for subtasks of the type considered
plan for subtasks of type R
Pi ... Pk ... ... Pn
plan for subtasks of type S
Pi ... Pk ... ... Pn
.....

Fig. 2. The general plan refinement strategy

The general search process of the method *M* is summarised in Fig. 2 where every node box symbolises the application of *M* with different declarative knowledge blocks of the four types commented before. The interest in using this reasoning strategy is that it may embed the simple case where only a classification step is considered when a collection of totally defined plans (i.e. with no T_l or T_m undefined tasks) is given in the domain model. In this case, the general reasoning strategy will stop after the first step once some task detail modules are applied proposing several complete plan options.

Behaviour prediction methods
This task has as main goal to propose scenarios of short-term future behaviour of the different components of the model. There may be specific simulation methods performing this type of task. A library could be considered to support a class of applications including a collection of typical physical components. However, it could be considered a simplified model to perform short term general *black box* behaviour simulation task, by using an envisionment graph formed by classes of component situations, described every one by a collection of slot values corresponding to every state attribute, and transition links representing types of external actions or decisions producing the change from one state to the other. In simple systems this type of representations may be useful. They may be generated through the abstraction of the results obtained from previous numerical model application. It must not be forgotten that the level of precision required for these models is not very high. It should be enough to differentiate the possible types of short-term evolution of the controlled system. The model of reasoning may take the current state from the information system and the assumptions about the external actions and match it with some node in the graph. As a result, for every matched situation the predictable short-term changes are described by the downstream connected states.

Co-ordination methods
Co-ordination is best conceived of as the management of dependencies between activities [15,17]. Methods that perform this type of management usually comprise three steps:

– *Dependency detection*: using domain knowledge about the different dependencies that may occur (producer-consumer relationships, resource limitations etc. [15]) positive and negative relationships between the different local tasks of the agents are detected. For instance, two local traffic agents may want to display different warning messages on the same traffic message panel.
– *Option generation*: for every dependency, the set of possible management actions is generated. In our traffic example, any of the involved agents may change its local action plan, or we may merge the incompatible messages "incident at *A*" and "congestion at *B*" to be displayed on the same panel into the message "traffic problems at *A* and *B*".
– *Management decision*: finally, a decision must be taken respecting the dependency management action to be applied. In the traffic example, one possible criterion for taking this decision is the aim of distributing traffic load equally among the different problem areas.

3 An Example: IDEAs for Traffic Control

We now illustrate the above framework for engineering Intelligent Decision-making Assistants by an example. We will describe the construction of IDEAS for road traffic management, a real-world domain, and an example of the high degree of complexity that decision support applications have to deal with. For this purpose, we set on from the TRYS family of system (e.g. [5,7,17]), agent-based DSS that have been build and used experimentally in different Spanish towns. In this section, we provide a principled "redesign" of the architecture of these systems, along the lines proposed in the previous section. First, our particular traffic management domain is sketched and the requirements for traffic management IDEAs are analysed in terms of an UER model. Then, the design of such IDEAs from a knowledge-oriented point of view is described, so as to finally sketch some implementation issues.

3.1 The traffic problem: analysis

In Barcelona, the local traffic control centre JPT is in charge of managing urban road transport, so as to maintain and restore the "smooth" flow of vehicles. Traffic engineers continuously receive information about the traffic state, identify potential problems, and act upon control devices to overcome them. It has become particularly difficult for the JPT engineers to perform this job in real time, as in the follow-up of the 1992 Olympic Games the traffic management infrastructure in Barcelona has become increasingly complex. Nowadays, information about the traffic state of the urban motorway network, consisting of one ring road and seven adjacent motorways, is provided by over 300 telemetered sensors ("loop detectors") via fibre optics communication links. Control actions can be taken by means of 52 Variable Message Signals (VMS), 3 traffic lights for junction control, as well as by ramp metering on 7 ring-road drives. Fig. 3 illustrates typical elements of this traffic management infrastructure.

JPT traffic controllers logically subdivide the road network into *problem areas*, for which they are able to generate efficient signal plans. Still, problem areas overlap, so potential conflicts between local signal plans need to be taken into account in order to obtain globally consistent signal plans..

Fig. 4 outlines our model of an IDEA-based DSS for this domain. The ultimate goal of the traffic decision support system is to assist human operators in the management decision by increasing their awareness of the traffic situation and the options to influence it. For the purposes of this article, we consider only a human actor, the operator. A complete analysis might consider additional actors such as administrator of the system or different levels of operators. The main use cases of an operator are:

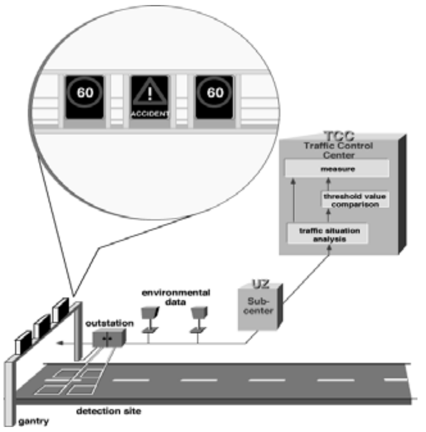


Fig. 3. Traffic infrastructure ([13])

– *EnterCaseFromSensors*: the operator receives data from the sensors from an external system and he/she introduces the new case manually.
– *EnterAlarmFromPolice*: the operator receives an alarm from the police.

Signal plans generated by operators comprise two classes of actions:

– *PutAMsgOnPanel*: set a warning message on a panel (e.g. “congestion at *X*”)
– *ModifyRegulators*: modify a regulator in order to prevent congestion

When communicating with the DSS, the operator is involved in flexible dialogues, asking for the reason of a diagnosis (*why*), what is happening (*what*) or a specific simulation (*what-if*). The following goal cases are assigned to the DSS:

– *DetectPhysicalConflict*: detect when two actions from adjacent areas generate different actions over the same area elements.
– *DetectLogicalConflict*: detect if actions from adjacent areas generate a bad overall solution.

The DSS can modify an environment problem (*publishAPanel*, *switchRegulator*). There are no current requirements for real reactivity, but one might consider reactive extensions like speed limitations in case a severe incident is detected. Once the UER cases have been defined, a first identification of the agents can be done. In this case, since we had distributed expertise in the problem, each one agent has been assigned each problem area of the road network.

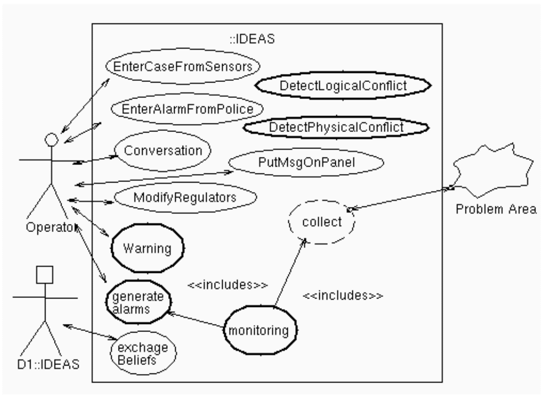


Fig. 4. UER Diagram for the traffic domain

3.2 IDEAS for traffic management: Design

The magnitude of a traffic problem in certain part of the road network can be expressed by the amount of traffic demand that exceeds the capacity of a certain road segment (in vehicles per hour). This is called the segment's *traffic excess*. The quality of a traffic management action can be measured by the reduction of traffic excess, that they are expected to produce. In consequence, traffic management agents use the overall reduction of excess in their problem areas (i.e. the sum over all road segments that belong to their area) as a local utility measure. Setting out from this notion, in the sequel we show how to design methods that cope with an IDEA's tasks.

Problem identification and diagnosis

Every couple of minutes, a traffic management agent receives temporal series of magnitudes such as traffic speed, flow and occupancy from the road sensors of its area. This raw data is initially pre-processed in order to filter out noisy and erroneous data. Subsequently, fuzzy data abstraction is performed (see Fig. 5), and aggregate magnitudes such as temporal and spatial gradients are calculated for the different sections.

The actual problem identification is performed by matching the abstracted traffic data against a knowledge base of frames, which model problem scenarios. Fig. 6 shows one such frame that matches the abstracted traffic data. Suppose that as a result of data abstraction low speed and high occupancy are identified in Ronda de Dalt en Diagonal and medium to high speed and low occupancy in Ronda en d'Eslugues. These facts match the frame shown in Fig. 6, so that an incident in the central lane of Diagonal road is identified, which manifests itself as a traffic excess (with respect to the road's capacity) of 2200 veh/h between Diagonal and Llobregat in the Dalt ring-road. Traffic from Collcerola to Llobregat and, in a minor degree, from Diagonal heading towards Llobregat contributes to this excess.

Action Planning and Prediction tasks

The action planning and prediction tasks adhere to the following line of reasoning: first, the historic traffic demand between nodes is retrieved and the contribution of each path to the problem in the critical section calculated. This is done by matching the current abstract traffic state and the state of the control devices against a knowledge base of frames, representing traffic distribution scenarios.

Finally, coherent alternative signal plans are generated by using the distribution scenario frames once again: every frame applicable to the current situation is preselected. Assume that this is the case for the frame shown in Fig. 6. Its short-term effects are estimated by simulating its impact on the current traffic situation. This is done by using network structure knowledge to assign traffic demand to the road network, in accordance with the distribution of traffic volume among paths that the frame specifies. In the example about one half of the traffic volume from Collcerola to Llobregat will pass through the Dalt ring-road, while a smaller amount chooses a path through Can Caralleu or other alternative paths, if the corresponding signal plan is set. If the simulation shows a reasonable decrease of excess in the critical section, the

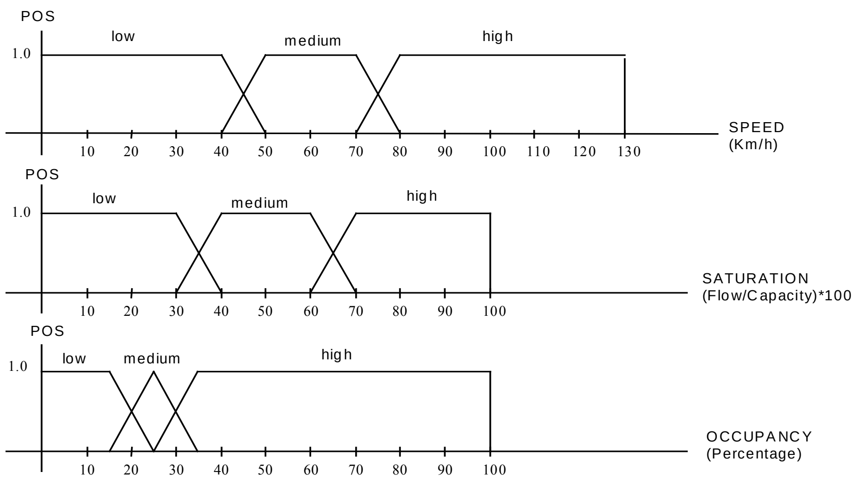
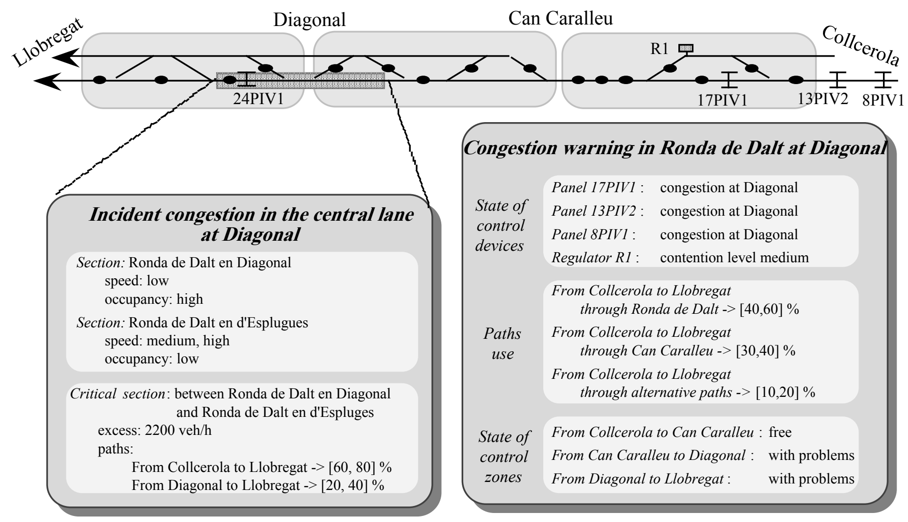


Fig. 5. Possibility functions to abstract traffic values

frame's signal plan constitutes one recommendation of the system. In the example, it is suggested to display congestion warnings at Diagonal for panels 17PIV1, 13PIV2 and 8PIV1, while setting the contention level of regulator R1 to medium. As a result of this process, a set of alternative signal plan recommendations, together with their utility (i.e. their expected reduction of local traffic excess) is produced.

Co-ordination

Methods for co-ordinating the different traffic management agents may be either centralised or decentralised, leading to different architectures for traffic management IDEAs. The InTRYS system [5] relies on a distinguished co-ordinator agent to perform co-ordination. Local traffic management agents send their control plans to that agent. Dependency detection and option generation relies on a knowledge base of rules. The dependency management decision is taken on the basis of priority knowledge, indicating which problem area is most critical, and thus determining which traffic agent needs to revise its control proposal. By contrast, the TRYSA$_2$ system [17] relies on a decentralised co-ordination mechanism. Dependency detection and option generation is done in a similar fashion as in the InTRYS system, although the corresponding knowledge bases are distributed and contain only the information necessary to deal with dependencies among neighbouring agents. Still, the management decision emerges from a negotiation among agents, which is based on a game-theoretic framework. The traffic management agents are supposed to be self-interested and mutually "threaten" each other with the potential consequences of failing to reach on agreement (although this never happens, as in this domain agents are always better off if the cooperate). The compromise, that they finally converge on, is a reflection of the relevance of an agent in a particular situation. When tuning the system, the designer can bias this basic compromise in a desired direction be issuing certain "prescriptions" on the use of resource by agents [17].



Llobregat
Diagonal
Can Caralleu
Collcerola
R1
24PIV1
17PIV1
13PIV2
8PIV1
Incident congestion in the central lane at Diagonal
Section: Ronda de Dalt en Diagonal
speed: low
occupancy: high
Section: Ronda de Dalt en d'Esplugues
speed: medium, high
occupancy: low
Critical section: between Ronda de Dalt en Diagonal and Ronda de Dalt en d'Espluges
excess: 2200 veh/h
paths:
From Collcerola to Llobregat -> [60, 80] %
From Diagonal to Llobregat -> [20, 40] %
Congestion warning in Ronda de Dalt at Diagonal
State of control devices
Panel 17PIV1 : congestion at Diagonal
Panel 13PIV2 : congestion at Diagonal
Panel 8PIV1 : congestion at Diagonal
Regulator R1 : contention level medium
Paths use
From Collcerola to Llobregat through Ronda de Dalt -> [40,60] %
From Collcerola to Llobregat through Can Caralleu -> [30,40] %
From Collcerola to Llobregat through alternative paths -> [10,20] %
State of control zones
From Collcerola to Can Caralleu : free
From Can Caralleu to Diagonal : with problems
From Diagonal to Llobregat : with problems

Fig. 6. An example scenario

4 Conclusions

In this paper we have argued in favour of an agent-based approach to the construction of advanced Decision Support Systems. We have introduced the concept of an Intelligent Decision-making Assistant (IDEA), and outlined how agent-oriented knowledge and software engineering techniques can be used to guide the process of a principled construction of IDEAs based on a (multi-)agent architecture. Our approach has been illustrated in the road traffic management domain.

We are aware of the fact that an IDEA-based design of DSS puts much burden on the agents' knowledge models, which leads to rather "fat" agents and puts limits to the scalability of the approach in this present shape. This is especially true for our traffic management agents, as each of them is to be endowed with domain knowledge to manage a complete problem area. Still, this seems hard to avoid, as real-world traffic engineers conceive traffic flows in this manner, and the decision support dialogues are to follow their lines of reasoning, so as to provide explanations that are understandable, and acceptable, to human operators. Furthermore, the decentralised coordination mechanism of the TRYSA$_2$ system already allows for a significant reduction in the need for coordination knowledge. Nevertheless, improving the scalability of our approach is a major line of future work.

In addition, we are current working on a tighter integration of the different phases of engineering IDEAs. For this purpose, particular attention is paid to the extension of the knowledge-oriented methods for design that have been outlined in this paper [4]. The principled and structured construction of the domain knowledge bases of IDEAs will be complemented by additional structured models (e.g. deeper interaction models, acquaintance models, strategic models etc.) which will eventually lead to a particular (multi-)agent architecture for IDEAs. We are particularly interested in different methods that support the design of structured interaction plans that allow for a dynamic configuration of the questions and answers produced during a decision support dialog between user and IDEA [18]. Decision support for road traffic control in an urban motorway network in the Basque Country, as well as for the management of bus fleets in Southern Spain, are target domains for the evaluation of this approach.

References

1. Breuker, J.; van de Velde, W. (1994): *CommonKADS Library for Expertise Modelling*. IOS Press.
2. Brown D.C., Chandrasekaran B. (1989): *Design Problem Solving*, Morgan Kaufmann,
3. Chandrasekaran B. (1983): Towards a Taxonomy of Problem Solving Types. *A.I. Magazine 4 (1)*, 9–17.
4. Cuena J., Ossowski S. (1999): Distributed Models for Decision Support. In: Weiß (ed.): *Multi-Agent Systems — A Modern Approach to DAI.* MIT Press, 459–504
5. Cuena, J.; Hernández, J.; Molina, M. (1996): Knowledge-Oriented Design of an Application for Real Time Traffic Management. In: *Proc. Europ. Conf. on Artificial Intelligence (ECAI-96),* Wiley & Sons, 217–245
6. French, S. (2000): *Decision Analysis and Decision Support*. John Wiley & Sons
7. Hernández, J.; Ossowski, S.; García-Serrano, A. (2002): Multiagent Architectures for Intelligent Traffic Management Systems. *Transportation Research C*, Kluwer
8. Hernández, J.; Serrano, J. (2000): Environmental Emergency Management Supported by Knowledge Modelling Techniques. *AI Communications 14 (1)*
9. Iglesias, C.A.; Garijo Ayestarán, M. (1999): UER Technique - Conceptualisation for Agent Oriented Development. In: *Proc. Intl. Conf. on IS Analysis and Synthesis (ISAS'99)*
10. Iglesias, C.A.; Garijo Ayestarán, M.; González, J.C. (1999): A Survey of Agent-Oriented Methodologies. In: *Intelligent Agents V*. Springer-Verlag.
11. Jacobson, I.; Booch, G.; Rumbaugh, J. (1999): *The Unified Software Development Process*. Addison-Wesley
12. Kinny, D.; Georgeff, M. Rao (1996): A. A methodology and modelling technique for systems of BDI agents. In: *Agents Breaking Away*, Springer-Verlag
13. Kirschfink H. (1999) Collective Traffic Control in Motorways. Tutorial at the *11th EURO-Mini Conference on AI in Transportation Systems and Science*. Helsinki
14. Klein, M.; Methlie, L. (1995): *Knowledge-Based Decision Support Systems*. John Wiley
15. Malone, T.; Crowston, K. (1994): The Interdisciplinary Study of Co-ordination. *Computing Surveys 26 (1),* 87–119
16. McDermott J.(1988): Preliminary Steps Toward a Taxonomy of PSMs. In: Marcus (ed.), *Automating Knowledge Acquisition for Expert Systems*, Kluwer
17. Ossowski, S. (1999): *Co-ordination in Artificial Agent Societies*, Springer-Verlag
18. Ossowski, S.; Serrano, J.M. (2001): Agent-based Architectures for Advanced Decision Support. In: *Proc. Workshop on Intelligent Physical Agents (WAF)*, URJC
19. Silver, M. (1991): *Systems that Support Decision Makers*. John Wiley & Sons
20. Vlahavas, I. et al (2002): An Intelligent Multiagent System for WAN Management. *IEEE Intelligence Systems 17(1),* 62–72
21. Wooldridge, M.; Jennings, N.; Kinny, D. (2000): The Gaia Methodology for Agent-oriented Analysis and Design. *Autonomous Agents and Multiagent Systems 3(3)*. Kluwer Academic Publishers, 285–312