11 Distributed Models for Decision Support

Jose Cuena and Sascha Ossowski

11.1 Introduction

The outside world is full of systems which are governed by complex laws of behavior. Independently of whether these systems are made of unanimated entities, governed by the laws of physics, organizations of humans with predefined artificial process rules or a mixture of both, often there is a need to influence their dynamics and to bias their evolution into a desired direction. The consequences of natural disasters, such as floods, can be alleviated if the spill gates of dams are managed to distribute the water volume in the watershed basin, keeping rivers and channels from overflowing. In industrial plants the production processes need to be monitored and adapted in order to ensure the quality of the final product, economic efficiency and security. Faults in chemical industries require actions that restore normal conditions and prevent the formation of toxic clouds. Computer networks have to be managed in order to maintain a certain quality of service to the users, which requires upper bounds on message delays etc. In much the same way, companies are interested in maintaining their business processes effective and robust. The flow of work in an office is to be kept smooth despite the illness of employees; material flow on car assembly lines should not be disrupted by any kind of contingencies. Traffic domains comprise a mixture of natural and artificial laws of behavior: road traffic flows have to be influenced so as to avoid traffic jams, to reduce travel times etc. In air traffic control, the primary concern is to influence the planes' routes so as to avoid accidents. This list could be extended easily.

The above examples motivate that it is a major challenge—for both economic and social reasons—to take the adequate control decisions to maximize the efficiency of the systems and to minimize the negative impact of faults. Sometimes one single person, but usually a group of *control personnel* is in charge of taking such decisions and is responsible for their effects. This implies to monitor continuously the system to be managed and requires to take decisions respecting changes in control variables, usually in *real time*. The increasing data volume and the decreasing time horizon within which control decisions have to be taken, have generated a need for computer applications that support the responsible persons in this task. These *decision support systems (DSS)* acquire data about the system state either directly or through a (real-time) database, on top of which they device an intelligent monitoring system that warns the control personnel of undesired evolution and answers their questions concerning potential reasons, effects and therapies. For instance, DSSs assist hydrology engineers to decide upon actions on spill gates, chemical engineers to manage modifications on valves etc. and network administrators on configuring routers, leasing lines and identifying faulty equipment; support to business managers is rendered by suggesting modifications in the usual work process, e.g. by reassigning work tasks or by modifying the route of some product through a jobshop; advice for traffic engineers may consist in suggesting which road traffic signals should be set or which air corridors should be assigned to certain flights.

This chapter is concerned with the principled construction of such DSSs. In particular, it points out how distributed artificial intelligence (DAI) models and architectures can be applied to the domain of Decision Support (DS). In addition, some reflections on the adequacy of DAI models for DS problems are presented: do DAI models and multiagent architectures really produce an "added value" for DSSs; or can everything that DAI systems contribute to the design and implementation of DSS be achieved better and cheaper by carefully engineered systems which are based on a conventional centralized approach?

Section 11.2 characterizes DS problems formally and informally and motivates the potential role of multiagent DS architectures in this domain. Section 11.3 presents a model for distributed DSS and a simple but powerful architecture for the agents that constitute it. The main body of this chapter is section 11.4, where this multiagent architecture is applied to the domains of environmental emergency management, energy management and traffic management. For any of these cases the characteristic domain features are highlighted first, then the generic agent architecture is instantiated to the domain, and finally the dynamics of the distributed DSS is illustrated by "simulating" the agent interactions that an example problem scenario generates. Section 11.5 summarizes the conclusions that can be drawn from these case studies.

11.2 Decision Support Systems

This section provides a model of the DS problem and introduces the knowledge-based approach to DSS design. Finally, the adequacy of a distributed approach to DS, based on a multiagent architecture, is discussed.

11.2.1 The Decision Support Problem

DSSs render support to the personnel that is in charge of managing natural or artificial systems, each governed by its particular physical and/or organizational behavior laws. So, the "area of interest" of DSS designers spreads from Control

Theory for industrial installations up to Computer Supported Cooperative Work (CSCW) in organizations and groups. Given this diverse nature of the DS domains, it does not surprise that the requirements for DS applications are as manifold as there are complex systems to control. Still, the core of the problem faced by DSS is captured by the following simple model:

- A set of *world states* S
  The relevant state of the world with respect to a DS problem is given by the values of the state and control variables of the managed system.
- A set of *ideal states* $S^+$and/or a set of *undesired states* S^-, where $S^+, S^- \subset S$
  Ideal and undesired states determine configurations of values for state and control variables that shall be achieved or are to be avoided respectively.
- A notion of *preference* $\prec$ on states
  The notion of preference expresses "how close" one state is to another. It can be expressed either in qualitative (e.g. a partial order) or in quantitative (e.g. a metric) fashion.
- A set Π of *control actions*
  Control actions can be performed on the system to be controlled which change the values of certain control variables directly. State variables are modified indirectly during the evolution of the system as implied by its behavior laws.

In real-world DS applications ideal states are almost never fully achievable. Instead, the objective of a DSS is to generate sets or sequences of control plans π so as to transform the current world state s into a state s' that is "as close as possible" with respect to $\prec$ to some ideal state s_i^+ and "as far away as possible" with respect to $\prec$ from any undesired state s_i^-.

The above semi-formal model is useful to characterize the problem that DSSs face in a less ambiguous way. Still, a whole bunch of quite different domains could be described by a similar formalism. This is because certain characteristic features of DS problems cannot be expressed easily in a formal fashion:

- Critical domains: some of the complex systems to be monitored are sensitive with respect to failures: wrong management decisions may have disastrous economic or environmental consequences.
- Understandable results and explanations: the responsible persons must assume the impact of the management actions that are taken on the system. They will be more inclined to accept suggestions from a DSS that explains its proposals using reasoning schemes and concepts that it is familiar with.
- Heterogeneous information and knowledge sources: The data, that the system needs to cope with, usually ranges from huge amounts of numeric data streams arriving in real time from a variety of different sensors, over visual information from video cameras, to quite informal messages such as telephone calls: incomplete, erroneous or even contradictory data may arrive. In much the same way, a DSS needs to integrate the different and even partially contradictory knowledge and reasoning methods, elicited from several different experts.

- Environmental contingencies: DSSs are embedded in dynamic domains, whose state changes due to events that cannot be anticipated. As a result, a DSS continuously needs to monitor its predictions and plans in order to maintain them in line with the current world state.

Although the degree of automation in the area is increasing, DSSs—as the name indicates—rarely *implement* control actions, in the sense that they seldom manipulate the world directly. Rather, they are expected to interact with the control personnel which is finally in charge of taking certain actions. Though a DSS needs to generate warnings proactively, its main functionality is to assist in the development of an adequate set of control actions and this is done on request of the control personnel. In consequence, the functionality of DSSs is greatly determined by the questions that can be put to a system and the kind of answers that they are able to supply. Crucial questions in the area of DS include [15]:

- *What is happening?*
  The system needs to analyze a situation and "understand" it by identifying advantageous and problematic aspects.
- *What may happen?*
  In many domains, there has to be an ability to reason about the evolution of the systems if no intervention takes place, i.e. in case that no management actions are performed. The main objective of this question is to foresee the decay of the present circumstances into an undesirable future situation, in order to be given the possibility to undertake appropriate counteractions. A variant of this type of question is *what may happen if...* some decision scenarios are undertaken.
- *What should be done?*
  The final aim of a DSS is to answer the questions respecting which are the most convenient actions to improve the results of system operation.

11.2.2 Knowledge-Based Decision Support

Keeping in mind the above peculiarities of DS, it becomes clear that the design of such systems actually "calls for" a knowledge-based approach. Knowledge-based systems model the expertise of the control engineers and explicitly represent it in a declarative fashion. By this, the system is not just capable of generating understandable explanations, but it also allows the designer to incrementally improve the expertise model, which has shown to be crucial in complex real world domains. Simulating the control personnel's reasoning process, which has proved to work well in the domain at hand, the system manages to cope with heterogeneous information and knowledge sources and to react adequately to environmental contingencies.

Modern methodologies for the construction of large-scale knowledge systems suggest that the expertise is organized in accordance with some structuring principle, in order to facilitate knowledge acquisition and the maintenance of knowledge mod-

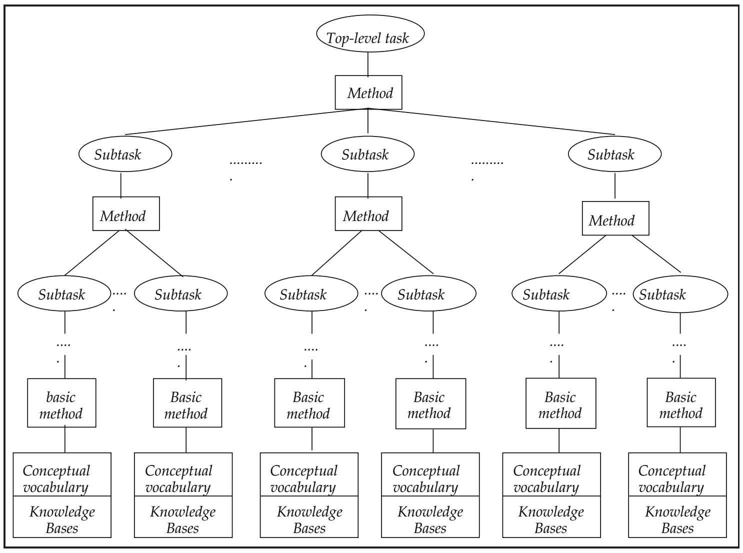

Figure 11.1 An example of a task-methods-subtasks tree.

els on the basis of a "divide and conquer" strategy[1]. The task-oriented stance is a way to express the result of such knowledge structuring. A *task* is an abstract description of how the world (or the "mental model" that the intelligent system maintains of it) needs to be transformed in order to achieve a desired behavior or functionality. Obviously, the top-level task of a DSS is to manage a complex system, answering any of the aforementioned questions that the control personnel might pose to it. *Problem solving methods* are used to cope with the tasks. They indicate *how* a task is achieved, by describing the different steps by which its inputs are transformed into its outputs. Simple tasks can be attained directly by means of *basic methods*. They rely on a *conceptual vocabulary* (or "ontology"), which defines the entities that the methods deal with, together with a *knowledge base* modeling the declarative domain knowledge, that describes how these entities are related. The method enactment uses these relations to solve elementary tasks. Still, the complexity of the DS task requires *compound problem-solving methods* that decompose the task into subtasks. These subtasks may again be decomposed by some method etc., giving rise to a task-methods-substasks (TMST) tree, whose leaves are given by basic methods. Figure 11.1 illustrates this approach, which is widely accepted as a general knowledge structure description. [7] [5]

1. The resulting structural model is often called a knowledge-level model, based on the idea of the *knowledge level*, as introduced by Newell [21]. A knowledge-level model of intelligent behavior does not focus on *how* it is achieved, but rather on *what* knowledge a system needs to be endowed with, so as to be able to reproduce that behavior, given that the knowledge is applied in a "rational way". The actual represention of the model and its "mechanization" through reasoning procedures is done at the *symbol level*.

When modeling a knowledge-based DSS from a task-oriented stance, the following set of tasks can usually be identified:

- The *classification* task takes as an input all available data about the state of the world and classifies the situation with respect to its desirability (i.e. its distance to S^- resp. S^+ with respect to $\prec$). Its output comprises the set of problematic features of the current situation. For instance, a traffic management system will receive numerical data from road sensors, on the basis of which it may classify situations as "fluid", "slight delays at junction x" or "medium congestion in area Λ".
- On the basis of the problematic features, the "symptoms" that indicate that something is going wrong in the modelled system, the *diagnosis* task comes up with an explanation that identifies the causes of such undesirable behavior. For instance, diagnosis might explain the "medium congestion in area Λ" by an incident at the outlets of that area.
- The *prediction* task evaluates how the state s of the modelled system will evolve into a state s' given certain values for the control variables. For instance, this task might conclude that if no actions are taken, the congestion in area Λ will become heavy, or that a change in the traffic light cycles will lighten the problem.
- The *option generation* task generates a set of plans (i.e. sequences of actions out of Π) that are considered to be adequate to overcome the problems identified previously. In the example, this will be the different plans' traffic guidance that deviate traffic from area Λ.
- The *action selection* task decides which of the potential plans proposed previously will be the outcome of the management process.

A major research line in the knowledge-based systems community is concerned with building up libraries of generic problem-solving methods (e.g. the European KADS project [5]). Ideally, for any concise task description, the library should contain methods to cope with it. So, the design process for a traditional "monolithic" knowledge-based DSS can be sketched as follows: first, the fundamental tasks for a specific domain are identified; subsequently, adequate problem-solving methods are selected; then, the knowledge types necessary for defining the functionality of the basic methods are elicited; and finally the knowledge is represented and the reasoning procedures of the basic methods operationalized.

11.2.3 Distributed Decision Support Models

So far, the use of a knowledge-based approach to DS has been motivated and a formalism to express the structure of the knowledge of the resulting DSSs has been introduced. Still, it remains to be shown where this structure comes from. In order to design complex DSSs there needs to be a *decomposition criterion*, that determines the shape of the TMST tree. The traditional solution is a hierarchical decomposition of the tasks via methods which require other tasks to be performed by other methods, etc. until a basic level is attained where the modules are simple enough. This is a version of the traditional *functional* decomposition standard in AI for many years.

When distributed models were considered, taking into account the low efficiency of the available hardware, this hierarchical organization was adapted in structures specialized in different tasks that may be computed in parallel. In some cases, however, this functional organization may not be easily understandable, which makes the process of knowledge modeling difficult, because the analogy between the agents' contents and the commonsense understanding of the expert is insufficient.

The search for a good modular structure in the applications is traditional in computer science. The first approach was a hierarchical functional decomposition with the already commented drawbacks. In order to improve the integration of different functionalities in understandable and maintenable entities, the following step towards modularization was the concept of *object*, where a collection of functions sharing data structures were integrated in a module. Still, although the organization in objects is an adequate organization for conventional software units, in many cases it turned out not to be intuitive enough from the point of view of knowledge models .

Multiagent Decision Support

Agent-based structuring introduces a more complex notion of modularity to computer science. This idea has evolved from Hewitt's actors concept [19] to the modern concept of agents which integrates a collection of functionalities, achieved by the interplay of both knowledge about certain problem types and about the environment in which the agent operates. By this, the agent can react to the environment situation and can interact with other agents to look for solution to its problems. The notion of agents allows a design of modules that balance two aspects:

- Level of *specialty*
  it is possible to model a detailed functional decomposition by designing agents that specialized in basic functions.
- Level of *autonomy*
  it is possible to integrate in an agent a significant set of the functions required for the whole application but limited in scope (for instance, in time if the agent performs in specific periods or in space if the agent acts on a reduced spatial environment).

In fact, the idea of agents represents the highest level in *modularity* that is used in computer science so far. It gives rise to a new generation of application structures, with a high degree of *antropomorphy* and, hence, *understandability* by the users. Additionally, the fact that agents embody a collection of functions that may serve in different societies is a design principle supporting the potential *reuse* of these advanced modules. Moreover, the generality of the agent concept allows to use human principles for structuring organizations as design criteria: an agent is an

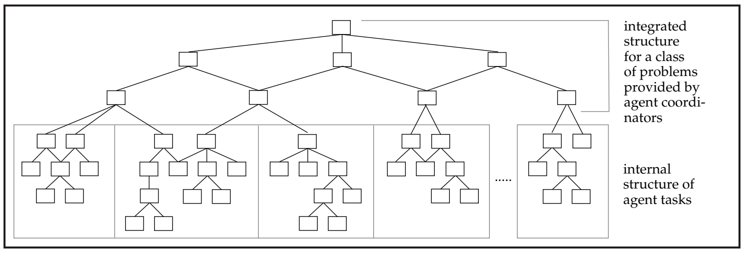

Figure 11.2 Centralized coordination: hierarchically integrated agent structure.

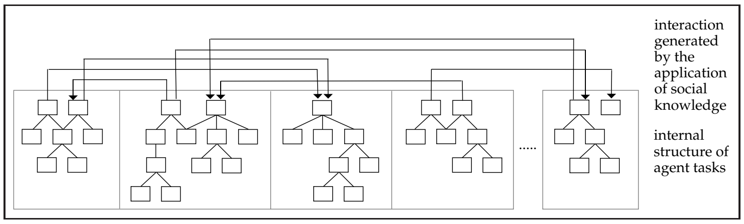

Figure 11.3 Decentralized coordination: dynamic agent interaction by social knowledge.

entity specialized in playing a social role in an organization which is a new basis for design provided by the social science area.

The experience with DSSs build so far prove that it is feasible to design the local problem-solving capabilities of agents by means of TMST trees in a principled fashion. Still, methods for modeling the interaction between agents, in order to ensure an intelligent global response of the whole society for certain tasks, are still maturing. This is the challenge of this type of models: to achieve intelligence by means of an adequate level of coordination.

The Coordination Problem

As shown in Figure 11.2, to achieve intelligent coordination a special coordinator agent can be designed, that is responsible for detecting interdependencies between the local agents' activities at successive levels of abstraction. This approach is contrasted by a decentralized stance, depicted in Figure 11.3, where no such special agent exists and agents interact laterally: agents are endowed with the knowledge to discover inconsistencies between their intended actions and interchange messages to mutually adapt their local decisions, so as to converge on one or several sets of consistent local control plans. The former coordination model leads to a hierarchical integration of control plans as determined by the upper level functions, while in the latter this integration emerges from agent interactions as implied by the agents' social knowledge. From an abstract point of view both approaches seem feasible. However, the first seems more reliable with respect to operation, while the second appears more adequate from a design perspective.

Once the knowledge to reason at the upper levels is elicited, the centralized approach provides a model of predictable behavior where all possible cases of inconsistencies are all analyzed a priori and are taken into account by the upper level modules. However, the bottleneck of this type of models is precisely the knowledge elicitation of the different inconsistencies: in many cases it is difficult to identify the precise way in which methods and domain models need to be integrated in order to solve a problem. In addition, once such a centralized model is built, the maintenance process is complex because, if additional lower level models are introduced, a sequence of changes has to be produced in the upper level models to take into account the potential modification of the situations produced by the new element.

The decentralized approach promises systems that are easier to build, because the model needs to be defined very accurately only at the local level, where it is more feasible to elicitate the knowledge to solve the specific problems of each agent. Normative coexistence knowledge for coordination may be defined in a more abstract way. Once both elements are sufficiently tuned to cope with the problems, the system may be maintained easily because the norms of coexistence are stable and independent of the number of agents in society and, if a new agent is introduced in the society, the only thing to be done is to include norms as a part of the external world model for the agent. No problems of propagation to upper levels appear. The problem of this decentralized approach is the quality of the intelligence of the whole society of agents, as it is difficult to identify the impact of normative knowledge in the quality of the global task achievement.

11.3 An Agent Architecture for Distributed DSSs

This section outlines an architecture for distributed DSSs. In particular, it is concerned with an "anatomy" of agents which, as in society, are capable of solving DS problems. The architecture does not pay special attention to computational and efficiency considerations, but comprises just features that are necessary to describe the different case studies, that will be presented in section 11.4, from a unifying view [2]. In line with the recommendations concerning design structures for agent architectures that are presented in chapter 1 of this book, the architecture is built around three major components, which are summarized in Figure 11.4.

- a *perception subsystem* allows the agent to be situated in the environment by data acquisition and in the society by perceiving agent messages. It implements the function *see*, described in chapter 1.

2. We assume that the reader is familiar with basic knowledge representation techniques and reasoning methods from symbolic AI, such as rules used by forward and backward chaining or frames used by pattern matching. An introduction to these issues is given in [26] and [27] .

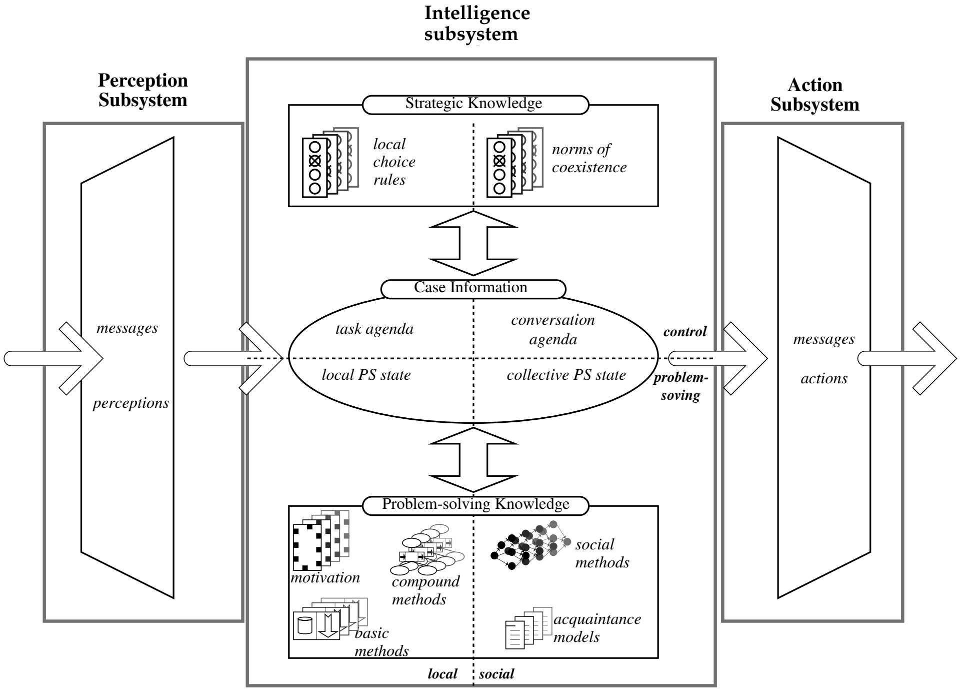

Figure 11.4 Agent architecture for a DS agent.

- an *intelligence subsystem* manages the different aspects of information processing as well as individual and social problem-solving. It implements functionality similar to chapter 1's function *next*.
- an *action subsystem* enacts the plans produced by the intelligence subsystem, displaying messages to the control personnel, sending messages to other agents or activating robotic effectors. In terms of chapter 1, it models the function *action*.

The subsequent description of the architecture focuses on the *intelligence* subsystem. First, the knowledge requirements of the DS agent will be presented. Subsequently, the structure of its information model is sketched, and finally the control model that makes the above components operational is described.

11.3.1 Information Model

The agents' dynamic beliefs about the world itself and the others are stored in the information model. The perception subsystem writes data on it according to perceptions and received messages, when the intelligence subsystem's knowledge is enacted, the information model is modified, while the action subsystem reads from it. We can distinguish two types of information in this model:

- *Problem-solving information* refers to inputs, outputs and intermediate results of tasks. Depending on whether the information refers to tasks pursued by the agent itself or by members of the society, *local* problem-solving tasks information and *social* problem-solving information can be distinguished.

- The second important part of the model contains *control information*, specifying in an *agenda* what is "intended" to be done. Respecting the agent itself there is a *task agenda* that keeps track of the tasks that are to be achieved locally. The *conversation agenda* keeps track of the "conversations" (see below) that the agent participates in, i.e. the social method in which it is involved somehow.

11.3.2 Knowledge Model

Agent knowledge can be classified from two perspectives. On the one hand, *problem-solving* knowledge, which is used to determine which actions to take, *is* distinguished from *strategic* knowledge that helps to choose among different options (the tasks or conversations, see below) that the intelligence subsystem is to process next. This knowledge may also be classified according to its role: there is *individual* agent knowledge, modeling the capacity of an agent to propose decisions and *social* knowledge, modeling the interaction knowledge framework representing the regulation of the society of agents.

For the DS problem, the individual knowledge comprises the following areas of knowledge:

- *Motivation* knowledge. It will be described by a collection of patterns modeling different classes of events considered by the agent as relevant in the external world. They are formulated by a collection of frames where the slots are classes of values of the information variables provided by the perception subsystem. When the perceptions have modified the information model in a way that some motivation frame can be matched against it, then this frame is activated. It represents an undesirable scenario in the world from the point of view of the agent. The difference between this undesirable situation and some no problem scenario, defines the task that the agent creates.
- *Local problem-solving knowledge*. The agent is endowed with the knowledge necessary to enact problem-solving methods so as to achieve tasks. This will be organized in the following form:
  - *Basic methods* perform elementary functions which are implemented either by specific algorithms or in terms of some knowledge representation and reasoning formalism as rules, frames or constraints. In addition, conceptual vocabularies model the entities that are manipulated by basic methods. A domain ontology, which comprises the different conceptual vocabularies together with the declarative relations between concepts, supports the inference operation of the different simple and composed methods.
  - *Compound methods* describe how a task is coped with by solving its subtasks. Thus, they establish a task-methods relations which, for each problem to be solved, gives rise to a TMST tree of the type shown in Figure 11.1. Compound

methods can be represented as rules, where the antecedent represents some intermediate state of task execution (e.g. the result of a previously executed subtask) and the consequent a sequence of subtasks to be executed in consequence, and relations that establish how the inputs and outputs of subtasks are related. As an alternative, this procedure might just be hard-coded in a simple algorithm.

- *Local Strategic knowledge.* This type of knowledge has as main goal to guide the process of generation of the TMST tree by selecting at every level and for every task the adequate method to be used for its performance. Then, it will be organized by a priority list of methods for the different types of tasks. The method of reasoning will recommend at a given level of the TMST tree to select the adequate options to continue tree formation.

Social knowledge comprises all expertise respecting the interaction between agents. Its components are:

- *Acquaintance models.* Knowledge about other agents is stored in these models, which can be supported by a *frame base.* Each frame comprises what is known about others in a collection of slots. Most important, such slots will represent features of every method known of a given agent (every agent models its acquaintances by its problem-solving capacity, i.e. in terms of characteristics such as levels of inference of the methods, the classes of subtasks that require to be supported, the kind of domain models supporting the basic methods, etc.). By application of a *pattern matching method* it can be deduced whether and up to which degree some acquaintance provides some desired characteristics. For instance, some task can be matched against the method slots of some acquaintance's frame, in order to evaluate its adequacy to perform that task.
- *Social methods.* These methods constitute the social problem-solving knowledge of the agent. As in the case of compound individual methods, a social method copes with a task by solving its subtasks (which will usually be handled by different agents). Still, social methods just specify at a very high level how these subtasks are to be integrated. Essentially, they configure an interaction protocol, which determines a set of meaningful *conversations* between agents [31]. A simple way of specifying such conversations is by means of a finite state automaton, whose nodes determine conversation states (in which some agents are supposed to perform tasks) and whose transitions are labelled with messages. Still, usually more flexible and more powerful techniques are used [25] [4].
  Social methods usually need to specify which agent has to solve a certain subtask, when and to whom to send task-related information and how to integrate the outcomes of tasks. So, social methods usually comprise three "meta tasks":
  - Task assignment. Through the individual selection of an agent, when several of them may be available to perform the same task. This may be done in different ways: by direct order, when one agent is endowed with acquaintance knowledge to choose the most suitable agent; by a "contract net," where

one agent asks all potential executors of a task for their adequacy and then grants the task to the best agent etc. In the remaining part of this chapter, it is assumed that tasks are assigned directly.

- Task synchronization. Once tasks are assigned, the flow of information between them needs to be configured and constraints on their execution posed. For instance, it needs to be specified what information needs to be present in order to initiate a tasks, and to which other tasks the result of one task are to be sent.
- Solution integration. Contrary to compound methods, whose subtasks have been defined in a way that their solutions neatly fit together and make up a solution for the task, the results of subtasks of a social method often need to be adapted to each other in order to constitute a consistent whole. As this adaptation usually leads to different alternatives, some choice mechanism needs to be present. Again this can be done in different ways: one agent can impose its preference in a dictatorial fashion, or a process of group decision making can be initiated (e.g. voting or negotiation).

- *Social strategic knowledge.* If there are several conversations in course, this knowledge determines the next one to work on. Again, this type of knowledge has as main goal to guide the process of generation of the TMST tree that, when several agents are involved, integrates methods of different agents integrated as a result of the conversation among agents.

11.3.3 Control Model

The mode of operation of the agent model is given by a simple reasoning cycle. It contains the following steps:

1. the perception subsystem captures percepts and messages from other agents, and updates the information model accordingly;
2. the conversation agenda is updated and reordered in accordance with the social strategic knowledge. As a result of the selection of some conversation, new tasks are added to the task agenda;
3. the motivation is matched against the information model and eventually more new tasks are created on the task agenda;
4. Using the local strategic knowledge the task agenda is reordered and some tasks are chosen for execution;
5. for every task two approaches are to be followed:
   (a) the local problem-solving approach where, using the knowledge about relation between tasks and methods, a method is chosen for execution. Usually, basic methods are preferred to compound methods, and the latter are given priority over social methods;

   (b) the delegation approach, if in the previous process no method is available in the internal problem-solving knowledge to cope with a task. In this case, the agent consults its acquaintance models and identifies a collection of agents that may perform the required tasks. The agent then assigns the task to the most adequate agent;

6. the action subsystem performs actions and sends messages as indicated by the intelligence subsystem in the information model.

This pattern of operation will be simulated and illustrated in the following examples.

11.4 Application Case Studies

The objective of this section is to illustrate the possibilities of the distributed approach for DSS design. Three examples of distributed DS in the domains of environmental emergency management, energy management and traffic management are provided, instantiating the previously described agent architecture to the particular domain and illustrating the modes of operation of the resulting multiagent DSS. You might see the agent and interaction model as the result of some multiagent design methodology, as discussed in chapter 9 of this book. This chapter also gives an overview of the architecture and operation of other industrial DAI applications.

The examples to be presented in the sequel go back to real-world designs and systems: they try to capture the "spirit" of the applications that they are based on. The reader be warned, however, that the presented models and modes of operation need not be identical with the original systems, due to lack of published information on the one hand and for educational purposes on the other.

11.4.1 Environmental Emergency Management

The Problem

Environmental emergency management is concerned with events that alter the usual mode of operation of an installation, having a negative impact on its environment. Often, even human and material damages and losses may be produced. Examples of such events are forest fires, river floods or gas dispersions from chemical plants.

The current state of sensor and communication technology allows for a dramatic change in the mode of management of these catastrophic events. Today it is possible to get on-line information about the wind state, temperatures and smoke in a forest data grid for fire control purposes, or to obtain directly data respecting rainfall and water levels in a watershed where floods may happen. On this basis, a decision-maker can evaluate the current situation (*what is happening?*) as well as its short term evolution (*what may happen if... ?*) within different scenarios, and elaborate potential action plans to apply (*what to do?*), so that an adequate *real-time* risk management is performed. So, in summary, the emergency management problem consists in generating an "understanding" of a potentially problematic situation and in elaborating consistent action proposals *in time*, so as to avoid damages when this is possible and to reduce their effects when no control action is applicable to eliminate them.

The nature of the problem comprises centralized aspects (the services to support large areas are usually located in a central place) and distributed aspects (the impacts of problems are spatially distributed, as in the case of floods).

Advanced systems can greatly improve the efficiency of environmental emergency management. In [10] the application of AI models to flood management is proposed, which have been developed along the sixties [1] [11]. Current advances in knowledge modeling promise to improve these techniques, as they support a better structuring. The European Union has promoted this line of applications in several projects such as ARTEMIS [17] in the area of Telematics for Environment, which is concerned with the management of heavy gas dispersions and river floods. In [2] a cooperation architecture for chemical plant emergencies is proposed.

A simple architecture is proposed in the following paragraph taking into account both features.

The Agent Architecture

Setting out from this problem description, the following types of agents can be identified:

- The Local Emergency Management Agent (LEMA), responsible for understanding the problem situation in a predefined area, and for proposing local decisions respecting initial management plans on this basis.
- The Dam Management Agent (DMA), responsible for taking decisions about dam control, on the basis of the needs of other agents as well as the situation of the dam and its water resources.
- The Fire Brigade Management Agent (FBMA), responsible for population evacuation as well as for the provision of manpower and other resources for protection works.
- Transport and Ambulance Management Agent (TAMA), responsible for the viability in the road transport network and ambulance resources management.

In line with these concepts an intelligent system for emergency management is designed by defining an instance of the Local Emergency Management Agent for every area where damages may be done to the population and the surrounding region where floods may impact. Furthermore, there is one agent of each of the above types, i.e. one Dam Management Agent, one Fire Brigade Management Agent and one Transport and Ambulance Management Agent (i.e. if there are nine endangered areas, a model of 12 agents is required).

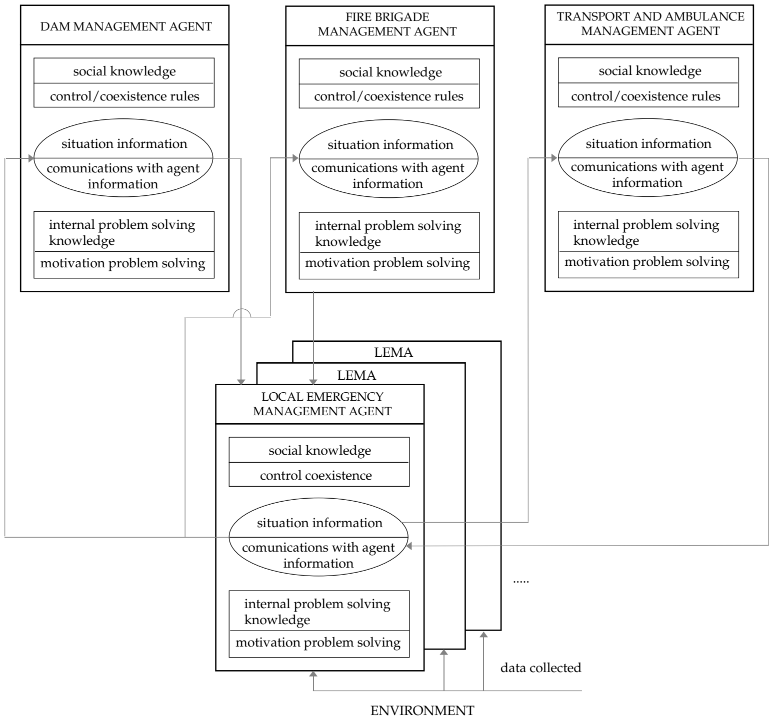

Figure 11.5 Summary of agent interactions in emergency management.

The society configuration is a simple one (summarized in Figure 11.5): every LEMA maintains two-way communications with any of the FBMA, TAMA and DMA agents. In general, no communication is designed between the latter three agents.

According with these design criteria the strategic knowledge will be based on:

- Local choice rules at the LEMA where the tasks to be provided by every general agent will be defined (given the degree of specialization of the general agents this knowledge will be expressed by statements of assignment of tasks to general agents).
- Local choice rules at the level of general agents where the preferences between the different LEMAs are established for sending messages according with general considerations of the agents and types of messages to be communicated.
- Norms of coexistence establish the preferences of some agents for task performance. In this case, given the specialty of every agent by function and by location, there are no doubts respecting which agent will perform a task.

The social methods in this case are also general enough:

- Task assignment, will be based on the specialty of every agent defined by the corresponding rule set.

- Task synchronization and solution integration. As will be commented later the general agents perform a task of activity integration constrained by the limitation of resources of the corresponding agent. A dialogue will be established between the LEMAs and the general agents where some proposed tasks by the LEMAs may be proposed to modify (see Figure 11.5).

In the following paragraphs the internal knowledge structure for every type of agent will be commented. Finally, an example of LEMA agent reasoning integrated in the society may serve to clarify the proposed specifications outline.

Every agent will receive total or partial information from the environment and from the society of agents. In the following paragraphs some features of information and knowledge operating in every agent will be described:

- Local Emergency Management Agent (LEMA):
  - The information received by the agent about the situation is the state of a collection of variables of rainfall and water levels that may influence the area controlled by the agent.
  - The motivation knowledge is used to detect if a current situation is problematic for the area. This knowledge may be described by a domain model of frames of situations describing patterns of potential damages together with a pattern matching method to infer the potentially activated problem frames for a given situation.
  - The local problem solving knowledge will propose local action plans to solve the detected problems:
    * Diagnosis knowledge. Once a subset of frames is activated by a given situation, the knowledge of diagnosis is based on a method of simulation and a rule base proposing potential causes in such a way that a procedure of generate and test supported by the simulator is applicable until a collection of explanatory factors is identified as likely enough.
    * Repair knowledge. Once a collection of causes has been identified this area of knowledge defines plans of actions. The objective is to eliminate the causes or, if this cannot be achieved entirely on the basis of local actions, to alleviate the effects through protection actions such as building flood walls or evacuating people.

    The domain model will be described by a library of plans together with a rule base for plan selection and a method of progressive plan refinement such as proposed by [6]. The kind of basic actions to be included in the plan are protection works to retain the water or transport actions using some available resources such as vans and ambulances to send people to hospital. To perform a transport action of this type requires an answer from the transport and ambulance agent about the accessibility between the desired origin and destination.

  - Social knowledge. This knowledge has to be built based on the unsolved task resulting from the planning phase in the problem solving knowledge. In fact, when the agent tries to look for a solution using its own knowledge, it tries to build a task method tree to act on the causes of problems. Still, when this is not feasible there will be unsolved task nodes, so there is a need the help of other specialized agents such as the dam control agent, the transport ambulance or fire brigade. To cope with this type of problems a domain model will describe for every unsolved task which messages will be sent to these three types of agents. For instance, if there is a problem of evacuation that cannot be solved by the local resources transport, there will be a message to the fire brigade establishing the conditions of the task to be performed.
  - In this case the knowledge will be very simple because the only task to be performed is the preference attribution to the possible agent of the same task.

- The Fire Brigade Management Agent and Transport and Ambulance Management Agent. Both agents have a similar structure so the general description of their contents will be the following:
  - Information:
    - Receives data about the state of its resources in terms of machinery vehicles and manpower.
    - Receives from the LEMA demands on manpower or transport according with the resulting needs in the local analysis.
  - Knowledge:
    - Motivation knowledge. After the reception of the environment situation and the other agent demands it may happen that a major complex event is inferred (if many demands come from one area it may be inferred that some catastrophe is happening). Accordingly, domain knowledge defining this type of global events may be included.It will be used by a method of pattern matching to detect its occurrence in a given situation.
    - Problem solving knowledge. The domain model will be described by a constraint set representing the condition of resource application for every local task (or for the more global task inferred as major event). The reasoning method will infer possible answers to the questions compatible with the constraint base. After this first step of reasoning it may happen that no feasible solution exists capable of giving a compatible answer to the current demand of the different agents. In these cases, the reasoning method will be able to propose alternative options by introducing adequate delays. To decide these delays the knowledge about control and social knowledge will be applied to reformulate the constraint base until some solutions are found (i.e. it may happen that the model answers in terms of no provision of a task now but is positive for the same task after some predefined delay). The type of constraint modification will be the object

of the conversation with the corresponding LEMA agents in a loop that will end up in an acceptable and feasible solution.

- * Social knowledge. This knowledge will model the preferences between the LEMA agents' demands to perform the delay generation described before.
- * Control knowledge. This knowledge will select options to guide the process of constraint reformulation.

- The Dam Management Agent:
  - Information. Receives data about the state in the dam system of the water levels and spill flows in every dam. Receives from LEMA limitations of the outflows from dams proposed by the LEMA according with its flooding problems.
  - Knowledge.
    - * Motivation knowledge. As with the other agents, it may happen that some major event such as a dam break or a local flood must be detected, so a collection of frames with the corresponding pattern matching methods will be designed modeling this type of events.
    - * Problem solving knowledge. The main task to be performed by this agent is to decide the outflows from every dam including the case of spill null. The way of performing this task will be similar at the previous agent by defining a domain model using constraint and rules representing the exploitation conditions of the dam system in such a way that if there is a solution according with the demands of the different agents the reasoning method will produce it or if there is not a solution guided by the control knowledge or social knowledge there will be a process of constraint relaxation introducing modifications in the demands of the LEMAagents according with their category until a solution is met.

Example of Operation

Figure 11.6 summarizes the mode of operation of a LEMA agent. As commented before, this agent is primarily concerned with management tasks for its area: (1) *what is happening,* (2) *why it happens* and (3) *what to do.*

The *what is happening* task may be performed by the event detection method of the agent supported by the domain model of a collection of predefined event patterns. The explanation of the detected events is also obtained using the internal knowledge of the agent based on a generate and test method supported by a production rule base and a simulator. The *what to do* task is to be performed by a more complex method using social and individual resources. Two options for action are considered:

- The *damage suppression* option where it is possible by applying actions from the agent and the society to eliminate the causes of damage. The figure shows the option of the LEMA problem solving method and the option to call DMA to help

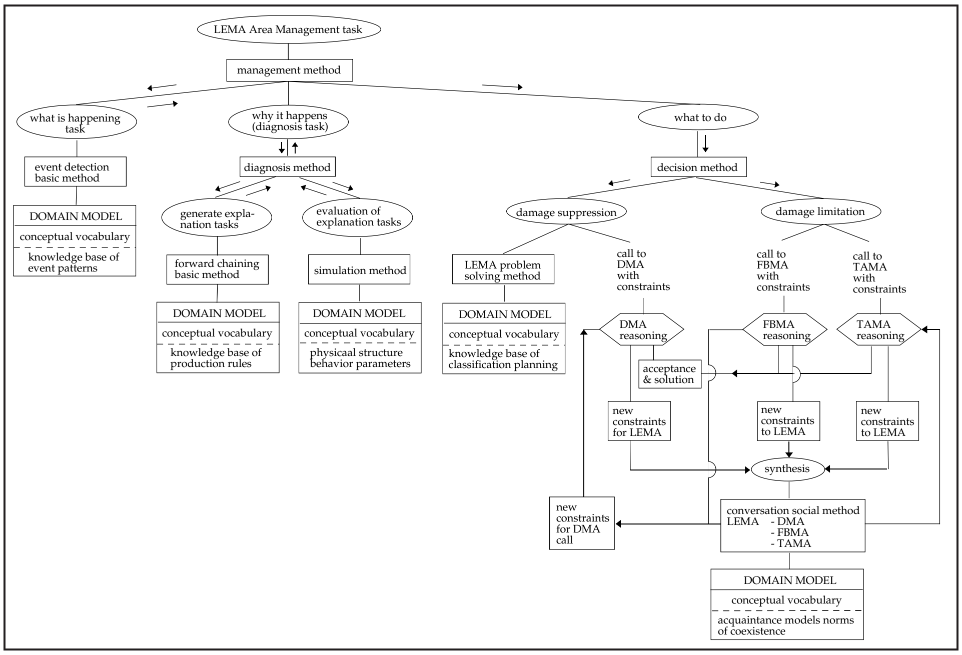

Figure 11.6 The LEMA general reasoning method.

by providing dam control actions. Two results are possible: (1) the DMA actions are sufficient to suppress damage; in this case the process is finished, (2) the DMA cannot attend totally the needs of the LEMA and proposes actions which solve the problem partially. In this case a conversation between both agents is started as commented previously in the DMA description, where several proposals are interchanged in terms of constraint size and delay until a balanced situation is obtained according with the LEMA social method.

- The *damage limitation* option where an estimate of damages and the needs to support its effects is provided by LEMA to the FBMA and TAMA agents. As in the previous case a dialogue is established depending on the available resources in FBMA and TAMA together with the norms of coexistence with LEMA (both agents propose reduced answers in time and size and the LEMA proposes alternative options).

In Figure 11.6 the conversation effect has been represented by loops of arrows derived from the LEMA synthesis task execution by using the conversation social method.

In this section, a simple model of agent interaction has been considered for illustration purposes. In the following case studies more complex interactions will be considered.

11.4.2 Energy Management

Energy management aims to assure a permanent and high quality supply of electrical energy. It is a crucial factor for the maintenance and success of modern economies.

Electricity is generated at power plants, as the result of the transformation of (thermal, nuclear, solar etc.) raw energy. For a variety technical, economical and political reasons, power plants have grown in size and generation capacity, so that the final electricity consumption takes place rather far away from the place of its production. In consequence, in nowadays electricity infrastructure, producer and consumers may be separated by several thousands of kilometers, so there is a need for an electricity network that connects the end users to the generation sites. Two types of such networks may be distinguished:

- Electricity transport networks
  Transport networks cover wide areas and assure that all regions of a country are connected to electricity supply. These networks are run at high-voltage (132 kV or more) in order to minimize losses during transportation. At transmission substations different high-voltage lines may be joined and electricity is fed from the transport network to the regional low voltage electricity distribution networks.
- Electricity distribution networks
  Distribution networks cover a much smaller geographical area than the transport network, but are more dense, as it needs to deliver electricity to all customers. They are usually run at lower voltage levels for security reasons and to reduce installation costs (lower voltages allows for "thinner" lines and thus save copper). Furthermore, as different classes of customers need different voltage levels, the network comprises a series of transformers that sustain a variety of different voltages on the net. For instance, in the UK large industries are connected to 33 kV lines, commercials and smaller industries are fed by 11 kV, while domestic customers are connected to a 240 V part [30].

Generation, transportation and distribution of electrical energy to industrial and domestic customers needs must be saved from damages due to equipment damage due to wind icing, lightning and other disasters, which are unpredictable and thus have to be coped with in real-time when they occur, there are quite frequent factors that might unbalance the network state and which allow for an extenuation of their effects well in advance [30]: fluctuations of temperature may cause changes of load; the overall demand changes according to the time of the day and of the week, to the season and the weather; *maintenance work* requires to check, exchange and install equipment safely etc.

Some of the above circumstances just affect the quality of supply (voltage peaks etc.), but others may imply real emergency situations, where problematic areas need to be isolated from the rest of the net as soon as possible: short circuits, for instance, may produce a chain reaction: an overload in a certain line increases the load in neighboring lines, giving rise to a new overload situation etc.; the situation deteriorates even more if power stations become disconnected, as this will cause an imbalance in the network's power. Still, it also has to be assured that the corresponding black-out area is minimal, that is, that as few customers as possible are affected by the incident.

Electricity networks are usually managed from a control room. High voltage distribution networks are usually telemetered so that the control engineers are provided with on-line information. They are endowed with protection equipment that can be operated remotely from the control room. In low voltage networks, this is often too expensive. So, information respecting deviant network states is usually obtained by observations from a field engineer and by telephone call of customers, reporting loss of supply. Work on the network is also performed "manually" by the field engineer. Distributed DSSs have been developed for both high-voltage and low-voltage networks [9].

The Problem: Fault Management in Electricity Transport Networks

This section is concerned with an almost fully telecontroled high-voltage electricity transport network. The model of such a network, that will be developed in the sequel, sets out from a set of *substations*. Each such substation hosts one or several *busbars*. Two busbars can be connected either by a *switch* or by a conductor *line*. For security reasons, in the latter case both edges of the line are to be connected to circuit *breaker*s, that interrupt electricity flow on the line when opened. Breakers usually come with a *protection* (relay), which automatically opens the breaker in case of a short circuit. Such a distribution of breakers makes it possible to disconnect every single line or busbar from the rest of the network and, in theory, this isolation is done automatically through the protections. The firing of breakers and protections causes alarm messages to be sent to a control room. From there, control engineers have the possibility to control the state of breakers remotely.

Figure 11.7 shows a transport network modelled in these terms. It actually represents a part of the high voltage distribution network of the Iberdrola company, which supplies large regions of northern and central Spain with electricity. This network with a generation capacity of 16.715 MW and a maximum demand of 10.000 MW is managed from the control room, located in Bilbao in the Basque Country. The transport network comprises three voltage levels and amounts to 401 busbars, 296 lines, 294 transformers, 939 breakers and 2322 switches. [20]

Emergency situations in this network are usually caused by a short circuit in a line or busbar, which is coped with immediately by isolating the affected area as soon as possible. The effects of such an incident can become worse in case of equipment malfunctioning: if a breaker fails to open, the affected area out of service is much bigger. The final objective of fault management is to restore supply for a maximum number of customers as fast as possible. As there are different "routes" along which electricity may be delivered, according to current network topology (which is determined by the state of breakers and switches), this is done

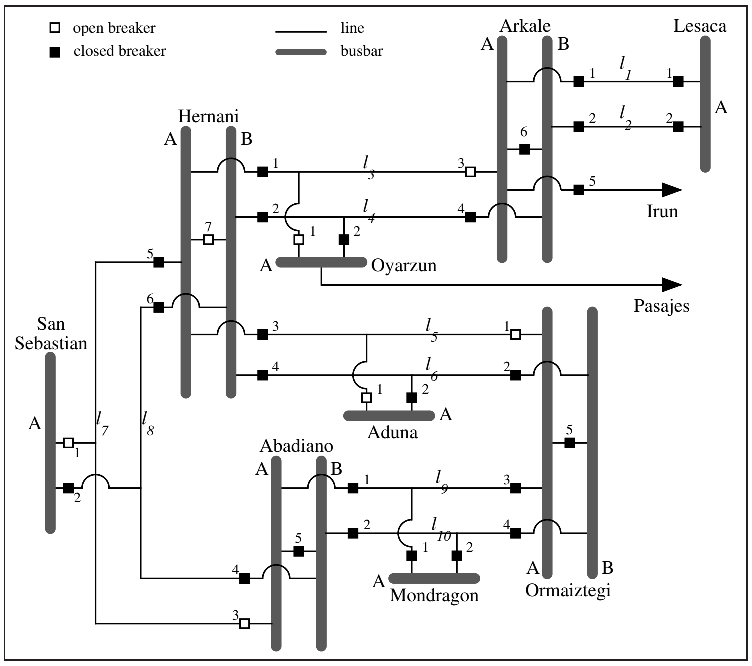

Figure 11.7 A snapshot of a high-voltage transport network in the Basque Country (adapted from [3]).

by establishing alternative routes (by operating breakers and switches) that skip the faulty devices. More specifically, the tasks comprise:

- to identify malfunctioning in switches and protective relays;
- to generate a fault diagnosis that explains all received alarm messages;
- to generate an efficient and *safe* switching plan, i.e. to restore supply for a maximum number of customers under the condition that the network is always in a consistent state (in all intermediate states the load on equipment is within acceptable ranges etc.).

The available information sources for this enterprise are also twofold. While *snapshots* provide a comprehensive picture of the current state of all components in the network, the *alarm messages* indicate how the state of the components changes over time, i.e. they describe the network's transition from one state to another. The former can be produced quickly, whereas it may take minutes until the latter arrive at the control room.

The Multiagent Architecture

In the frame of the ARCHON project [20] the distributed power transportation DSS has been built for parts of electricity distribution network as described above. The preexisting management infrastructure of Iberdrola had a crucial influence in the final shape of the multiagent system that has been developed. The assumptions made in this section are similar to those encountered by the ARCHON project. There are essentially two design constraints [20]:

- The supplier company was using stand-alone DS tools to ease the workload of the control engineers. The functionality of DS should be augmented on the basis of these *preexisting systems*, but without modifying them, as they have evolved to work efficiently and reliably in the Iberdrola network. Instead, preexisting applications should interact with new functionality in the frame of a distribution intelligent system.
- the current *configuration of the data transmission network* had to be maintained. In particular, this configuration assigns a low priority to messages containing temporal information, giving rise to two different classes of alarm messages:
  - *Non-chronological alarm messages (NAM)* contain abstracted information about the state changes in the network and are timestamped at the time of arrival at the control center. Thus, NAMs arrive at the control center rather fast, but important information respecting the sequence of events is lost.
  - By contrast, *chronological alarm messages (CAM)* receive their timestamp at the substations. So, they contain precise chronological information at the cost of a greater delay in their availability at the control center and a higher error probability due to synchronization problems between the substations' clocks.

The design of an agent society for the electricity transport network management DSS sets out from the definition of the constituting agents. In the sequel, the six agents that have been identified will be described, but with simplified knowledge and services for illustrative purposes.

The preexisting applications have been "wrapped" into two agents:

- *Alarm Analysis Agent (AAA)*
  On the basis of NAMs, the *AAA* detects the occurrence of a disturbance, determines the type of fault and establishes hypotheses respecting the malfunctioning equipment. The *AAA* to provides the following basic methods:
  - on the basis of the NAMs the basic method *simple classify* of the *AAA* detects new disturbances and classifies them into categories such as short circuit, overload etc. The conceptual vocabulary on which this process is based will minimally contain a list of possible NAMs as well as of classes of situations (e.g. short circuit, overload and manoeuvre). The knowledge can be given in terms of rules that directly associate sets of NAMs with these classes.
  - *cover NAMs* takes NAMs and their classification in order to output a set of hypotheses about faults that "explain" them. As conceptual vocabulary the domain's NAMs, network elements and fault types need to be given. A rule base uses these concepts for a shallow hypothesis generation.

The only compound method is *cover and differentiate by NAMs*. It specifies that first a fault hypothesis needs to be generated that explains all symptoms. Then, the *determine BOA* subtask is set up, that determines what would have been the detected symptoms if the hypothesis were true. The fault hypothesis is refined until real and simulated symptoms are consistent.

- *Control System Interface (CSI)*
  The *CSI* constitutes the application's front end to the control system computers. Besides serving as an interface to the conventional management system application programs, its main objective is to acquire and distribute network data to other agents. It offers two basic methods:
  - *acquire data* detects, preprocesses and formats NAMs and CAMs for their future use within the system; this is done by a fast hard-wired formatting algorithm.
  - *simulate effects* calculates the power distribution in the system given a certain state of devices (breakers etc.). For this, a numerical simulator applies knowledge about the physics of load distribution in electrical installations to the distribution network which is described on the basis of a conceptual vocabulary defining the network lines, busbars, breakers and their characteristics.

  The compound method *classify situation* specifies that in order to cope with the *disturbance detection* task, it is necessary first to perform *alarm detection* and then to do *alarm classification*. In addition, it offers the social method *coordinate classification*, which determines where the results of the CAM and NAM classifications should be sent to, and that no integration of the different classifications is necessary.
  The CSI's motivation detects situations in which logical coherent sequences of alarm messages have arrived (probably all messages that have been caused by one network event). In this case, it sets up a *disturbance detection* task.

In addition to the above preexisting applications, four new agents have been developed to cover all tasks that the system specification requires.

- *Blackout Area Identifier (BAI)*
  The *BAI* uses NAM in order to identify which network elements are initially out of service. It offers just one basic method: *determine BOA*. Given a network state and faults, a rule base that constitutes a causal model of the network determines what the results of such a scenario would be. The conceptual vocabulary necessary for this will minimally contain a list of faults and alarms.
- *Breakers and Relays Supervisor (BRS)*
  The *BRS* is a new alarm analysis expert system that works on the basis of CAMs. It detects the occurrence of a disturbance, determines the type of fault and establishes hypotheses respecting the malfunctioning equipment. The following basic methods are provided:

- on the basis of CAMs the basic method *simply classify* of the *BRS* detects new disturbances and classifies them into categories such as short circuit, overload, manoeuvre etc. As in the case of the *AAA*, the conceptual vocabulary on which this process is based will minimally contain a list of possible CAMs as well as of classes of situations. The declarative domain model can be given in terms of rules that directly associate sets of CAMs with these classes.
- *cover CAMs* takes CAMs and their classification uses a rule base in order to provide a shallow fault hypothesis that explains all received CAMs.

The only compound methods is *cover and differentiate by CAMs*, which specifies that first a fault hypothesis needs to be generated that explains all symptoms and then it is simulated which symptoms would have been produced if the hypothesis were true. The fault hypothesis is refined until real and simulated symptoms are consistent.

- *Service Restauration Agent (SRA)*
  The objective of the SRA is the elaboration of a safe service restauration plan after a blackout has taken place. On the basis a snapshot of the current network state and information about malfunctioning equipment, it offers the following basic methods:
  - *propose switching plan* devises an initial service restauration plan given the alarm messages and the results of the diagnosis process. On the basis of network model as described by the conceptual vocabulary, heuristic knowledge is used to construct a sequence of operations that finally make up a switching plan.
  - *modify switching plan* uses rules that, depending on plan critique, indicate how a certain plan is to be modified. Again, this is done on the basis of a vocabulary that expresses the network model.
- *User Interface Agent (UIA)*
  The *UIA* serves as an interface between the users of the multiagent system and the agent society for presenting data (e.g. to browse through the list of alarm messages), the result of the different tasks that are resolved within the system (e.g. to display the results of diagnosis together with an explanation) and the effect of control actions (e.g. the simulated effect of the enactment of a certain service restauration plan). In addition, the *UIA* provides compound methods *propose and revise* for service restauration, by means of which the user controls how long the cycle of proposal generation, critique and revision should be run. In addition, the social method *coordinate diagnosis* sets up guidelines for the coordination of the *diagnose by CAM* and *diagnose by NAM* tasks.
  The motivation of the *UIA* specifies that no disturbance classified as "short circuit" should be kept untreated and generates a *diagnosis* task in consequence. In much the same way, it requires that a diagnose fault should be overcome, so that a *service restauration* task is set up.

Agent	Simple Methods	Compound Methods	Social Methods
CSI	acquire data	classify situation	coordinate classification
	simulate effects		
BRS	classify CAMs	cover & differentiate	
	cover CAMs		
AAA	classify NAMs	cover & differentiate	
	cover NAMs		
BAI	determine BOA		
SRA	propose switching plan		
	revise switching plan		
UIA		propose & revise	coordinate diagnosis

Table 11.1 Overview of Methods for Energy Management

The methods of the agents are summarized in table 11.1. For the purpose of this example, it is assumed that each agent is endowed with a model of any other acquaintance (although this is not necessary for the model to work well). So, in each agent this model comprises six frames, that contain the methods shown in table 11.1. Besides the method *type* (simple, compound or social), the acquaintance model contains information about the competence with which the method can be applied. For instance, the *cover and differentiate* method of the *BRS* can diagnose breaker, busbar and clock faults, while the *AAA*'s *cover and differentiate* method is just competent to diagnose breaker and busbar failures. Control knowledge is not specified in this example, as the characteristics of tasks and agents imply a certain determinism in the problem-solving process.

An Example Operation

Consider the situation shown in Figure 11.7. The following CAMs and NAMs have occurred.

```
10:10:10 BRK OYA 1        10:10:09.34 PRT ADU 1
10:10:10 PRT OYA 1        10:10:09.36 BRK ADU 1
10:10:10 BRK ADU 1        10:10:09.41 PRT ORM 1
10:10:10 PRT ADU 1        10:10:09.43 BRK ORM 1
10:10:10 BRK ORM 1        10:10:09.43 PRT HER 7
10:10:10 PRT ORM 1        10:10:09.45 BRK HER 7
10:10:10 BRK ARK 3        10:10:09.46 PRT OYA 1
10:10:10 PRT ARK 3        10:10:09.46 BRK OYA 1
10:10:10 BRK HER 7        10:10:09.49 PRT ARK 3
10:10:10 PRT HER 7        10:10:09.51 BRK ARK 3
10:10:12 BRK ABA 3        10:10:09.54 PRT SSE 1
10:10:12 PRT ABA 3        10:10:09.56 BRK SSE 1
10:10:12 BRK SSE 1        10:10:10.04 PRT ABA 3
10:10:12 PRT SSE 1        10:10:10.07 BRK ABA 3
```

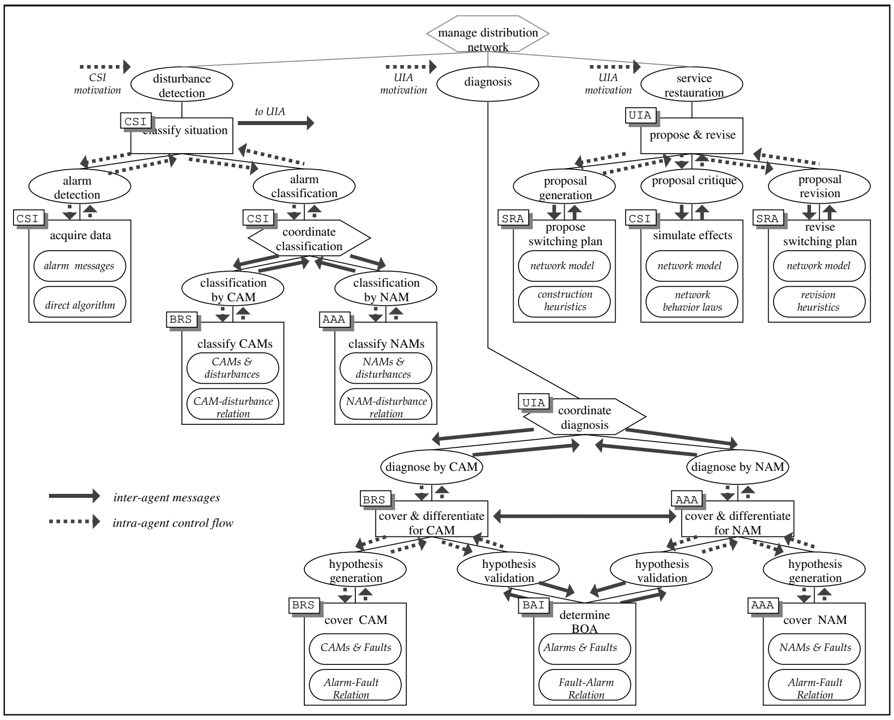

Figure 11.8 Task structure generated by the society of energy management agents.

Figure 11.8 shows the TMST tree that is generated when these alarms arrive at the *CSI*. In the sequel, we illustrate how this tree is constructed following the general control method.

The *CSI*'s motivation frames match the representation of these messages in the information model, and a *disturbance detection* task is generated and put on the agenda. The agent notices that it is endowed with the *classify situation* method, and puts the resulting subtasks on its agenda. In a next cycle, it notices that the *alarm detection* task can be coped with by the basic method *acquire data*, which formats alarm messages. The resulting alarm strings are passed upwards, until the *classify situation* method specifies that they serve as input to the *alarm classification* task. The *alarm classification* task is coped with by the social method *coordinate classification* of the *CSI*. It requires direct task assignment and, according to the agent's capabilities contained in the acquaintance models, messages are sent to *BRS* and *AAA*, assigning the tasks *classification by CAM* to the former and *classification by NAM* to the latter. Once any of these tasks have finished, the results of the classification of alarm messages are passed upwards to the *classify situation* task. A message with the resulting classification (a short circuit occurred) is sent to the *UIA*.

As a result, the UIA's information model is modified in such a way that it matches a motivation frame. So, a *diagnose* task is activated and put on the *UIA*'s agenda. The agent copes with it by activating the social *coordinate diagnosis* method, which directly assigns the tasks *diagnose by CAM* and *diagnose by NAM* to the *BRS* and the *AAA* respectively. Furthermore it specifies that in case of differing resulting diagnosis, the "more knowledgeable" diagnosis should be given preference. The *AAA* chooses this task and assigns the compound *cover and differentiate by NAM* method to it, giving rise to the subtasks *hypothesis creation* and *hypothesis validation*. First, the *AAA* generates its hypothesis on the basis of the NAM. The rule base assigned to the simple *cover CAM* method, makes it initially suspect that each firing of a protection has been caused by a fault in the adjacent lines and busbars, which leads to the following list of hypotheses respecting faulty devices:

$$\text{fault}(\{\text{l}_7,\ \text{Her}_A,\ \text{l}_3,\ \text{l}_5\})$$

As the *AAA* is not capable of resolving the hypothesis validation task, the control cycle looks for an acquaintance frame with a method slot that matches that task. The BAI does and so it asssigns this task to the *BAI*.

The *BAI* tackles this task by the *determine BOA* method, which infers what the effects of the occurrence of the hypothesis would have been: given the initial situation, it concludes that the following events would have occurred in case of the faults:

$$\begin{aligned} \text{l}_7 &\Rightarrow (\{\text{SSe}_1,\ \text{Aba}_3,\ \text{Her}_5\}),\\ \text{l}_3 &\Rightarrow (\{\text{Her}_1, \text{Ark}_3,\ \text{Oya}_1\}),\\ \text{l}_5 &\Rightarrow (\{\text{Orm}_1,\ \text{Adu}_1,\ \text{Her}_3\})\\ \text{Her}_A &\Rightarrow (\text{Her}_7,\ \text{Oya}_1,\ \text{Adu}_1,\ \text{Ark}_3,\ \text{Orm}_1,\ \text{SSe}_1,\ \text{Aba}_3) \end{aligned}$$

The *AAA* receives this answer and reconsiders its hypothesis. By default, the *AAA* suspects that breaker failures are more likely than simultaneous faults in different devices, so it refines its initial hypothesis, giving rise to the following list which is ordered by decreasing confidence in them

$$\text{fault}(\{\text{Her}_A\}),\ \text{fault}(\{\text{l}_5,\ \text{Her}_3\},\ \text{fault}(\{\text{l}_3,\ \text{Her}_1\}),\ \text{fault}(\{\text{l}_7,\ \text{Her}_5\}).$$

Again, each of these fault hypotheses covers all alarm mesasages. It sends these hypotheses to the *BAI* and receives the answer that each of them explains the received NAM.

Suppose that the *BRS* generates its initial hypothesis in the same way as the *AAA*, but on the basis of the CAMs, so it arrives at the same fault hypothesis. A *hypothesis validation* task is created and passed to the *BAI*. As in the case of the *AAA*, the *BRS* receives the differentiating information from the *BAI* and starts to reconsider its hypothesis. As clock severe synchronization failures are rather rare, the *BRS* reorders its list of validated hypothesis in the following way:

$$\text{fault}(\{\text{l}_5,\ \text{Her}_3\}),\ \text{fault}(\{\text{Her}_A,\ \text{Adu}_{clock},\ \text{Orm}_{clock}\}),$$

The *BAI* confirms that each of these faults covers all received alarms. Still, it has also received the diagnosis from the *AAA* and notices a difference of opinions. As indicated by the social *coordinate diagnosis* method, the *BRS* does its part in the solution integration searching its acquaintance models and discovering that the *AAA* does not know about clock failures, which means that it is not competent to reject the *BRS*'s diagnosis. So, it maintains its diagnosis and sends it to the *AAA* and the *UIA*. The *AAA*, after receiving that diagnosis, performs a similar reasoning with its acquaintance models and, as a result, adapts its own by integrating the clock fault hypothesis. Finally, it also sends this adapted diagnosis (which is now consistent with the diagnosis of the *BRS*) to the *UIA*.

Again, these messages modify the *UIA*'s information model, a motivation frame applies and a *service restauration* task is created. The *UIA* assigns the *propose and revise* method to cope with it, which results in the *proposal generation*, *proposal critique* and *proposal revision* subtasks. The *UIA* has no method for any of them, so after consulting its acquaintance model, the first task that the method requires to be executed, *proposal generation*, is assigned to the *SRA*. The *SRA* generates the following switching plan

plan(open(Her_3), close(Ark_3), close(Oya_1), close(Her_7) , close(SSe_1), close(Aba_3))

which it returns to the *UIA*. The *proposal critique* subtask is delegated to the CSI, which applies the *simulate effects* methods, returning characterizations of all intermediate states of the potential switching plan to the *UIA*. As the last intermediate state is not considered to be safe, the *UIA* delegates the *revise switching pla*n task to the *SRA*, which revises it by exchanging the last two actions:

plan(open(Her_3), close(Ark_3), close(Oya_1), close(Her_7) , close(Aba_3), close(SSe_1))

Again, the *UIA* sends the revised plan to the *CSI*. Its simulation proves that the modified switching plan is acceptable, and the *propose and revise* method returns it as a result.

The above example describes a complete reasoning cycle of the energy management multiagent DSS. The functionality of a fictitious top-level task *manage distribution network* has been generated "bottom-up" by communication among agents and their adequate motivations. Other types of events may lead to other task distributions between agents. Note that every agent decision may be explained to the responsible engineers using the trace of the reasoning methods.

11.4.3 Road Traffic Management

The increasing popularity of road transport and the incessant rise of the number of vehicles have caused a tremendous growth of the magnitude of traffic flows on public roads. Especially in urban areas, where the road network is dense and the traffic volume in peak situations is enormous, significant economic losses are produced by enduring and recurrent congestions. Still, it is precisely in these urban areas where there are severe obstacles to the expansion of traditional infrastructure, due to the scarcity of space and resources as well as for environmental reasons. As a consequence, urban road traffic management has become an increasingly important task: strategies to guide traffic flows are essential in order to avoid collapses of individual transport and the corresponding losses for the local economy.

In big cities, traffic control centres (TCC) are in charge of managing urban transport. A TCC's responsibilities cover a wide range of different tasks. In particular, traffic engineers within a TCC are to supervise the current road traffic situation, detect problems and take actions to overcome them, so as to maintain and restore flows of vehicles adequate for the network capacity. Information about the current traffic state is obtained from many different sources, the most important of which include:

- messages transmitted from human *observers*, which constitute a classical source of information for traffic control centres. In most cases, such information is provided by urban police or members of related public organisms;
- visual control of certain problematic areas is possible by means of *TV cameras*. They are especially useful to assess unusual and emergency situations, such as the gravity of accidents etc.;
- *sensors*, which are installed in strategic parts of the network and generate a continuous flow of numerical data about traffic conditions at a certain point. There are different types of sensors with different costs and capabilities. One of the classical sensors are loop detectors that usually provide information about speed (mean velocity of the vehicles detected by the sensor), flow (average number of vehicles that pass through a certain road section per time unit) and occupancy (the average time that vehicles are spotted by the sensor).

On the basis of such heterogeneous traffic information (informal, visual and numeric data) a traffic control engineer identifies potential problems and decides upon signal plans to overcome them. Such signal plans comprise a coherent set of uses of control devices that the TCC can act upon. The most popular devices for urban road traffic management are the following:

- *Variable Message Signs (VMS)*, that allow to influence traffic behavior by dynamically setting, modifying or deleting traffic signals. The most advanced VMS are panels which are installed above the road. They allow to display arbitrary messages that inform drivers about the network situation downstream. In addition to this, they can display pictograms and traffic signs, thereby announcing warnings, speed limits, prohibitions to overtake etc. Older types of VMS support just a small collection of different traffic signs or constrain the set of messages that can be displayed due to technical limitations;
- *traffic lights*, by means of which the access of vehicles to certain parts of a road network, typically junctions, is controlled. Different features of traffic light signaling can be modified: the relative amount of green time can be increased or decreased, the overall length of a cycle may be changed and the time offset between different cycles is modified. Ramp meters are special traffic lights that are positioned on the entry ramps of motorways. They enable the traffic controller to regulate the amount of vehicles entering the motorway.

Figure 11.9 depicts a typical traffic management installation. The task of traffic management performed in the TCC is to generate *signal plans* for control devices in order to alleviate traffic problems that have been identified on the basis of the collected traffic information.

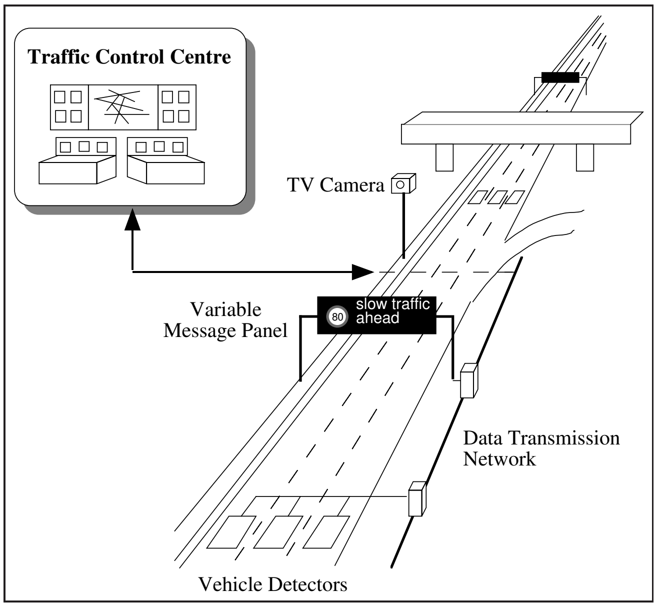

Figure 11.9 A typical traffic control infrastructure: traffic information from vehicle detectors and TV cameras is transmitted to the Traffic Control Center, which acts upon road traffic by displaying messages on Variable Message Panels.

The Problem: Urban Highway Traffic Control

This section is concerned with road traffic management of the urban highway network around Madrid. It consists of two beltways and 8 adjacent highways is endowed with about 100 VMS panels, 50 cameras and over 300 loop detectors. These devices are connected to the Madrid traffic control center via fiber optics communication links. Certain parts of the network support peak load of over 6.000 vehicles per hour.

Figure 11.10 depicts part of the highway network of eastern Madrid. It shows the eastern part of the two beltways *M30* and *M40* as well as the three adjacent highways *NII*, *NIII* and *NIV*. Data sources such as loop detectors are not shown, but the figure indicates the location of the nine VMS that are installed in this area.

The objective of traffic management is to ensure the "smooth flow of traffic" which can be expressed along measures such as travel times or "length" of congestions etc. It is pursued by enacting signal plans, which—in the example of Figure 11.10—coincide with coherent sets of messages to be displayed on the different VMS. These messages inform drivers about the traffic situation downstream. This guidance influences the "size" of traffic flows in the network (thereby reducing the traffic load in congested areas), as some drivers will choose a different "route" from their origin to their destination. More specifically, traffic management comprises the following tasks:

- to identify problematic situations, classifying their type (incident congestion,

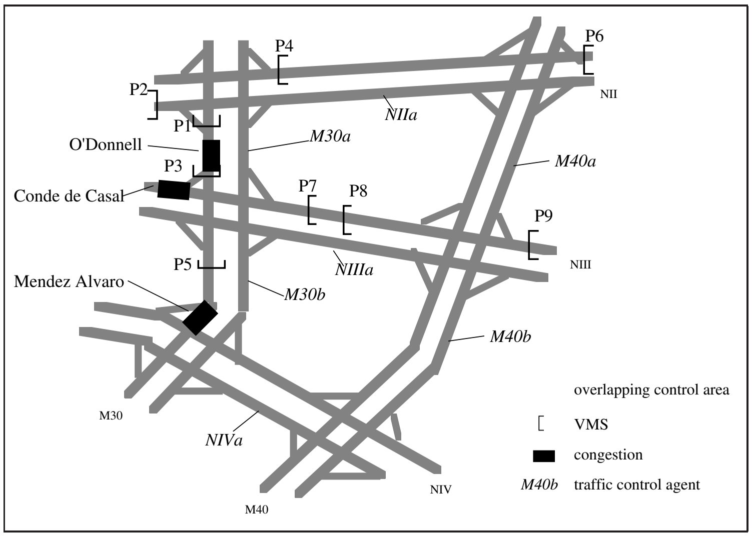

Figure 11.10 Part of the motorway network of eastern Madrid (adapted from [12].)

overload congestion etc.) and location;

- to diagnose the *causes* of the problems in terms of the traffic flows that contribute to it;
- to generate a legal signal plan proposal on such a way that the causes of problems be eliminated or alleviated enough to improve the traffic conditions.

The Multiagent Architecture

In the frame of the European KITS [12] and the Spanish TRYS projects [13] [14] agent-based traffic management systems have been built for the road network of different towns. The structure of the systems was crucially determined by the fact that the human operators, from which traffic management knowledge had to be elicited, analyzed traffic flows in terms of *problem areas.* Problem areas usually reflect special characteristics of the network topology, but also comprises empirical behavior rules for certain parts of the town. In consequence, one traffic control agent was built for each problem area. Agents are homogeneous, in the sense that they share the architecture and the reasoning structure, but the knowledge they are endowed with is pertinent to the specific problem area they are responsible for (e.g. empiric knowledge about area behavior rules).

In the example of Figure 11.10 there are seven traffic control agents—*M30a*, *M40a*, *M30b*, *M40b*, *NIIa*, *NIIIa* and *NIVa* —whose problem areas comprise the highway section between two highway junctions, including all entries and exits that connect this section with the ordinary road network. Each of them is to generate local signal plan proposals that help to overcome traffic problems in their area. For this. it is endowed with the following basic methods:

- *data abstraction* takes as input raw sensor data, filters out noisy data, calculates aggregate values such as temporal and spatial gradients and finally determines qualitative measures for different system variables (e.g. the average speed may be high, medium, or low).
- *problem type identification* takes all the data generated through data abstraction in order to identify problems, such as incident congestions, overload congestions etc. This is done by matching the data against problem scenario frames. Each such frame characterizes a specific problem situation by relating the type of problem to patterns of abstracted traffic data.
- *solution refinement* specifies the identified problem in further details.
- *demand estimation* determines the traffic demand between some arbitrary origin and some arbitrary destination in the network. Obviously, this varies temporally (peak hours etc), so that this basic method uses a frame base that associates temporal patterns (hour, day of the week etc) with demands for the different origin/destination pairs.
- *effect estimation* determines the effect that certain traffic flows have on a problem. This is done by means of distribution scenario frames that relate three types of information: traffic problems, the state of control devices and the contribution of certain routes to the problem. By matching the current state of control devices and the traffic problem against these frames, the "final causes" of the problem are found in the traffic flows that contribute to it.
- *signal plan selection* aims to generate a proposal of how a problem can be overcome. It also uses the distribution scenario frames, but in another way. All scenarios that match the traffic problem are potential signal plans.
- *short term prediction* estimates the result of a change in traffic flows within the network. It uses a simple assignment algorithm to do this.

These are used by the following compound methods:

- *heuristic classification* is a classical problem-solving method [8], which comprises three subtasks, which are executed sequentially: *abstract state* determines relevant aggregate information for the classification, which is done by the *match problem* subtask. The result is refined in the following *refine problem* subtask.
- *contributor differentiation* determines how much each of a set of potential causes contributes to a problem. In order do assure this, it first generates the subtask *find contributors* and then *estimate contribution*. This is repeated until all contributors are evaluated.
- *generate and test* first sets up a generate proposal subtask, which produces a set of tentative signal plans. These are judged subsequently within the test proposal subtask. If a plan is not considered to be sufficiently adequate, a now proposal is generated.
- *local management* just indicates that traffic control of the local area consists of first identifying traffic problems, then diagnosing its causes in order to finally

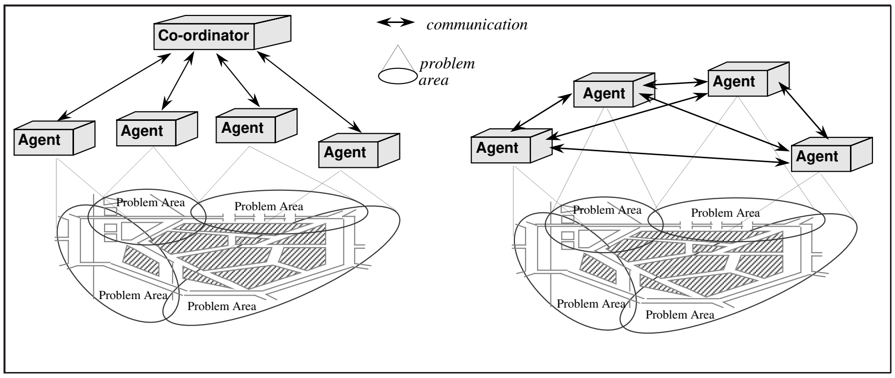

Figure 11.11 Centralized and decentralized multiagent traffic control architectures: in the former there is a special coordinator agent in charge of harmonizing local control proposals of the traffic control agents, while in the latter this functionality emerges among traffic agent interaction in the frame of a social method.

configure signal plans to overcome them.

Still, the subdivision of the road network is made on the basis of empirical (logical) criteria, so the spatial division of the network is not perfect, i.e. the areas are not disjointed, but rather constitute a set of overlapping zones. So, local signal plan proposals use to be interdependent. There may be either *physical* conflicts between them, when the proposals of two agents require to display different messages on the same panel, or *logical* conflicts when one agent's signal plan proposal hinders or even cancels the effectiveness of another. In consequence, the agents' proposals need to be coordinated. As Figure 11.11 depicts this can be either done by means of a dedicated coordinator agent, or laterally through peer-to-peer communication. It has shown to be difficult to elicit explicit coordination knowledge, so in the sequel the second option will be discussed.

In this domain, the *acquaintance models* are also given by frames. Still, such a frame does directly represent information concerning methods that another agent can perform or tasks that it can cope with (as the agents are, in principle, self-sufficient for the management of their local area). Rather it describes the actions that the acquaintance can take, specifying which resources they require (causing potential physical conflicts), and which effects they may have (causing potential logical conflicts).

An agent's *motivation* matches the information model once complete sequences of new data have arrived. In consequence, it generates *manage 'local area'* task. This task is similar to those generated by a social method. Although no synchronization with the acquaintance's tasks is necessary, it is required that local signal plans be sent to those agents with which there might be a conflict (e.g. those with overlapping problem areas). In case of conflicts, the agent with the most *severe* problem will be allowed to maintain its signal plan.

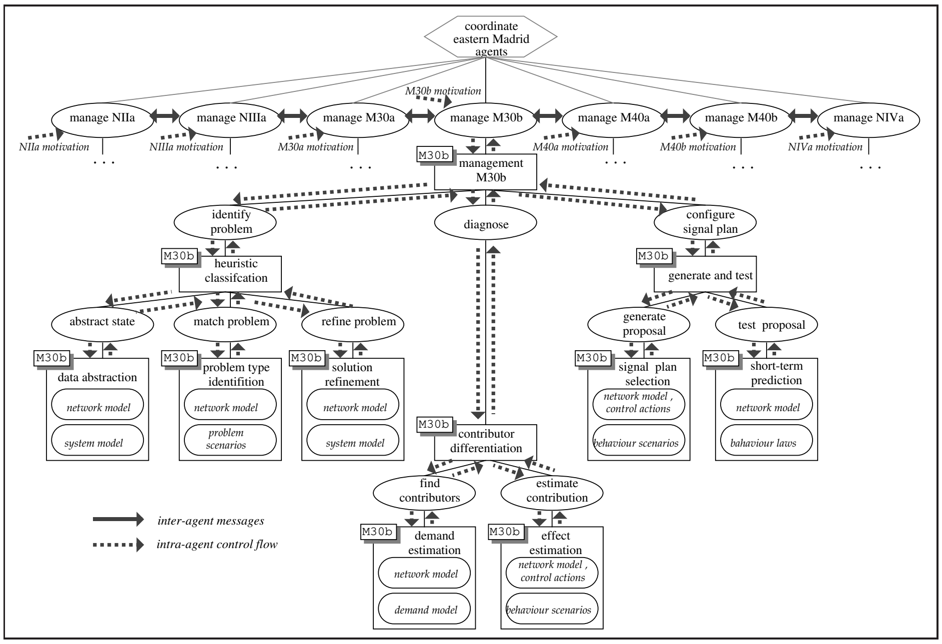

Figure 11.12 Task structure generated by the society of traffic management agents.

An Example

Consider the traffic situation shown in Figure 11.10. There are three problems: in the area of *M30a* at O'Donell, in the area of *M30b* at Mendez Alvaro and in the *NIII* area at Conde de Casal. Figure 11.12 shows the TMST tree that the society of traffic control agents generates from this situation.

The reasoning process of the *M30b* agent will be described as an example. The arrival of new data makes its motivation create the *manage M30b* task. The *management M30b* method is found to be an adequate means to cope with it. Its first subtask, *identify problem*, is solved by *heuristic classificatio*n, which first performs *data abstraction* by completing possibly erroneous sensor data and calculating temporal and spatial gradients etc. The *M30b* then agent matches this information against its frames, representing typical problem scenarios, and identifies that there is an overload problem at Mendez Alvaro. In a *solution refinement* step, it detects that the overload problem manifests itself in a saturated off ramp.[3]

The *management* method then requires a *diagnosis* task to be solved, which is done by the *contributor differentiation* method. First, the current traffic demand between entry and exit points in the network is determined (*demand estimation* method), in order to estimate then by means of pattern matching (*effect estimation* method) how much each flow between each origin and destination contributes to

3. Note that this problem identification already comprises certain aspects of diagnosis.

VMS	M30a	M30b	NIII
P1	C. at O'Donell	C. at Mendez Alvaro	
P2	C. in M30 at O'Donell		
P3		C. at Mendez Alvaro	
P4	C. in M30 at O'Donell		
P5		C. at Mendez Alvaro	
P7			C. at Conde de Casal
P8			C. at Conde de Casal

Table 11.2 Local signal plan proposals ("C." stands for "Congestion").

the problem.

Once this has been done the *configure signal plan* task is set up, which is coped with by a generate and test method. First, a proposal is generated by matching distribution scenario frames against the problem situation, deducing a potential signal plan and the corresponding change in traffic flows. The subtask *test proposal* gives rise to the execution of a short term prediction method, which estimates the effectiveness of the potential signal plan. It is selected or rejected accordingly.

Other agents perform similar local reasoning. As a result of this reasoning process, just the agents *M30a*, *M30b* and *NIII* have identified problems and generated signal plan proposals shown in table 11.2.

The agent *M30b* detects a physical conflict with *M30a*, because the latter proposes to display a message on the panel $P1$, which *M30b* already wants to access with a different message. The agent *NIII* detects a logical conflict with *M30a*, as the message shown on *P3* will make drivers on *M30* leave this beltway and turn into Madrid by *NIII*, aggravating the problem at Conde de Casal. In much the same way, agent *M30a* detects the physical conflict with *M30b* as well as a logical conflict with *NIII*, as the message on *P7* induces drivers on *NIII* heading towards downtown Madrid to use alternative entrances, thereby worsening the problem at Mendez Alvaro.

The agents notice that the physical conflict at *P1* can be solved by merging the proposals to a "Congestion from O'Donnell to Mendez Alvaro" message. In addition, the agents *M30a* and *NIII* interchange information about the gravity of their problems. As a result, and according to the distributed *coordinate* method, *M30a* gets preference, as its problem is more severe: Panel *P3* will show as proposed "Congestion at Mendez Alvaro," while *P7* will be switched off. This final proposal is presented to the control personnel.

As in the electricity management domain, the agents' motivations have been designed in a way that is equivalent to a hypothetic social method *coordinate eastern Madrid agents*. Still, the way in which solution integration is performed relies on knowledge about a "preference ordering" (the more severe problem is given precedence) which all agents share. But this is precisely that kind of knowledge that a fictitious central coordinator agent (see Figure 11.11) would be endowed with; it is just replicated among many agents.

Other, more decentralized modes of coordination can be achieved by augmenting the *degree of autonomy* of the traffic control agents. For instance, the designer can set out from the notion of self-interested agents, whose autonomy to manage the traffic in their area is only bounded by the existence of its acquaintances. In such a design, agents behave selfish and, in principle, are only interested in the traffic state of their problem area. Still, the interactions between their local signal plans makes it rational from them to coordinate with others.

However, although the agents may benefit from coordinating their signal plans, they have a conflict of interest respecting which global plan should be agreed upon. The compromise that is reached in such situations of mutual dependence will certainly depend on the agents' "social position," i.e. as to how far an agent can help or harm others, and up to which degree others may influence in the effectivity of an agent's local signal plan proposals.

One way of modeling decentralized coordination in such a scenario is to see the compromise reached (or: the social equilibrium) as being biased towards the less dependent agent: the "stronger" an agent is, the more "weight" will have its individual preferences in a potential compromise. The designer can influence this social equilibrium by means of prescriptions, which prohibit or permit certain agents to use certain control panels, thereby increasing or decreasing their degree of dependence [24], [22].

The formal representation and operationalization of such models will most probably be based on some model of distributed rational decision making, which are discussed in detail in chapter 5 of this book. Cooperative bargaining models, for instance, are strong candidates for this enterprise [23]. An example of the application of these ideas is given by the decentrally coordinated traffic management system which has been built on top of the ProsA_2 architecture described in [24]. Such approaches, however, are still an area of research and their potentials and drawbacks in different real-world applications have to be tested.

11.5 Conclusions

This chapter has outlined the potential of DAI models for decision support. The DS problem has been described by means of a semi-formal model and the knowledge-based approach to DSS design explained. The application of DAI techniques to DS problems has been motivated, and the design and operation of such distributed DSSs has been illustrated by presenting an abstract agent architecture and by simulating the reasoning and interaction processes within problem-solving societies of such DS agents in different domains.

From an abstract point of view, the concept of an agent has been used as modularization principle for the DSSs' software and knowledge. The principle by which agents were formed reflected an *a priori* distribution in the domain. Emergency management agents were built in line with preestablished organizational entities. In the electricity distribution example, agents were created around preexisting expert systems. In the road traffic management example, the agentification was implied by the control engineer's understanding of a town's traffic behavior in terms of problem areas which, to a certain degree, reflect a logical distribution of the domain.

It has been argued, that such a non-functional distribution implies a need for additional coordination that standard problem-solving methods cannot cope with in a natural way. The notion of a social method has been introduced in order to cope with coordination problems.

The intersection of the areas of Knowledge Modeling and DAI is a busy and exciting research area. Once the concept as well as structuring and enactment principles of social knowledge have further evolutioned, libraries of distributed interaction methods might be developed that will greatly ease the development of distributed DSSs.

11.6 Exercises

In this section two example scenarios for decision support are presented. Each such scenario constitutes the basis for several exercises. Do not be driven back by the large introductory texts; they will help you to cope with the subsequent exercises.

1. Consider the following emergency management scenario for flood prevention. There is an area with a high risk of flooding, whose structure is shown in Figure 11.13. Six watershed basins collect rainfall and produce a runoff to four major water streams, which lead into two reservoirs created by dams located downstream the rainfall collection areas. The dams are endowed with spill gates and discharge pipes by means of which the water retention capacity of the dams can be controlled. Consider the relation between discharge capacity, water level and the state of spill gates to be given in a table.
   Downstream the dams there are two villages within floodable areas, where damage can potentially be done to people. In every such area civil protection works can be performed (e.g. to reinforce flood walls), in order to improve the security against floods. Three levels of civil protection can be planned: "low," "medium," and "high." For every level of protection different amounts of human resources are required, which are available either locally or from the next main town which is located within a secure area. Furthermore, the availability of resources changes with the time of the year.
   According to the situation, it is possible to infer a relation between the acceptable outflow from the reservoirs and the level of protection works to be performed in every village. The following two extreme emergency management strategies are possible:
   - minimum flow limitation in dams (i.e. wide opening of spill gates) together with maximum protection works;

- maximum flow limitation in dams with minimum protection works;

Eventually, these plans need to go in line with evacuation actions, which are to be provided by the village transport resources and additional central services from the main town (e.g. given a situation where defenses have a limited capacity it may happen that some subset of population requires evacuation).
Local emergency management may be supported through the help of the following organisms:

- the water management authority, which determines the amount of flow to be discharged from dams within the next hours, given the current remaining capacity of the reservoir and the predicted inflows.
- the fire brigade which manages manpower to provide capacity for protection works. Limited resources will be defined in terms of available vehicles of known capacity and available human resources.
- the sanitary and ambulance authority that is responsible for population evacuation and medical treatment where also a limited number of ambulance vehicles and persons are available.

Any of the above organisms receives requests for support from the endangered areas and determines as to how far to grant help on the basis of a predefined policy of preferences.
The aim of the exercise is to design an agent-based DSS for this domain. It is recommended to define the following agents:

- two LEMA agents, which perform local emergency management in the endangered areas, i.e. LEMA-P for area P and LEMA-Q for area Q,
- one DMA agent, responsible for dam control on reservoirs A and B;
- one FBMA and one TAMA agent for central resource management concerning protection and evacuation works respectively.

In the sequel we will provide descriptions of two types of these agents, leaving the design of the remaining agents to the reader. The following knowledge components of a LEMA agent can be identified:

- The motivation knowledge must be applied to react to the meteorology conditions. A classification method will be applied supported by a domain model, which is based on predefined patterns which consist of two parts. The first part is defined by slots representing the current state and the recent trend in the variables. For instance, for area P the slots will include the measures of the state of the rainfall basins 1,2 and 5 as well as the situation in reservoir A; for area Q the patterns include slots for all six rainfall basins and both reservoirs. Either on the basis of available expertise or through simulation, different emergency situations are characterized by assigning characteristic values to the slots. The second part includes slots that characterize the estimated impact in floodable areas in terms of waterlevel excess and overflows entering the area. So, the patterns relate meteorology situations with their potential impacts. In consequence, a pattern matching

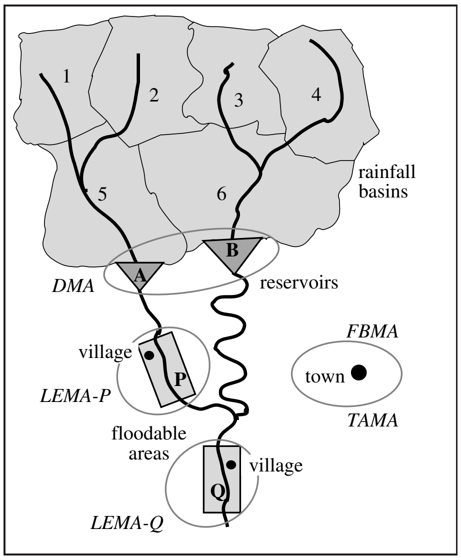

Figure 11.13 A Flood Management Scenario.

method can be used to generate hypotheses about potential impacts, whose negative effects need to be avoided.

- The local problem-solving methods are used to repair or to alleviate the potential damages predicted previously. Essentially two local problem-solving methods are enacted sequentially:
  - on the basis of water-level excess and overflows entering the area the impact specification deduces the affected area and the number of affected persons using a rule based classification.
  - the impact alleviation knowledge estimates the protection actions in terms of dam control actions, the level of protection works or the magnitude of population evacuation. First, a sequence of actions leading to impact alleviation is constructed. This is a planning problem, which may be coped with by a hierarchical classificative planner as proposed by Brown and Chandrasekaran [6] for routine design. Local resources are assigned to the plan on the basis of a rule base. External resources are assigned using social methods.
- The acquaintance models for DMA, FBMA and TAMA contain slots describing the kind of help that any of these agents can supply together with an estimation of its maximal amount.
- The social methods used by the LEMA are of quite simple structure. Depending on the type of resource required, the task of determining the help to be granted to a LEMA is assigned to DMA, FBMA or TAMA. After accomplishing the task its results are sent to the LEMA.

- The norms of coexistence determine the order in which the LEMA's acquaintances are asked for help. They can be modelled by rules, e.g.:
```
IF rainfall in basins 1,2,5 = very high AND
   rainfall in basins 3,4,6 = medium THEN
   preference = (DMA, LEMA, FBMA)
```

In consequence, the following sequence of conversations may arise:

- the LEMA asks the DMA to reduce water level excess totally;
- the DMA answers with an alternative, expressed by a range of admissible reductions;
- the LEMA estimates the civil protection works necessary for any of these options. If external help is required it asks the FBMA for additional manpower;
- the FBMA responds with the range of possibilities to satisfy the LEMA's help request, specifying the amount of manpower and the required waiting time;
- finally, in accordance with the previous answer, the LEMA infers evacuation needs and asks the TAMA to provide the necessary resources;

The knowledge contents of the remaining agents determines their behavior within these conversations. Concerning the DMA agent the following observations can be made:

- As in the case of the LEMA agent, the DMA's motivation knowledge is expressed in terms of a frame base of patterns, which characterize storms that act on rainfall areas and relate them to possible undesirable impacts that have to be avoided.
- The problem-solving knowledge aims to find reasonable values for the opening of spill gates and discharge pipes at the dams so as to avoid the aforementioned undesirable impacts of meteorology conditions, taking into account the limiting criteria on water storage and the maximum waterflow downstream that the reservoir's outlets may produce. To meet this goal a generate-and-test method is applied:
    - a rule knowledge base generates reasonable values for openings on the basis of the current data and the default impact estimation that the motivation frame base provided;
        - a simulator determines the impact of this decision, using an artificially generated rainfall series and the structure of the drainage network (knowledge for the operationalization of a simulator can be derived from flow routing methods in hydrology engineering);
        - in an evaluation step it is determined whether the short-term effects of dam management actions as estimated by the simulator satisfy the safety constraints mentioned above. If this is not the case, another action proposal is generated and the generate-and-test cycle reinitiated.

- The behavior of the DMA in social interactions with a LEMA agent is based on its current situation and a priority evaluation of the different LEMAs. It determines the amount of help that can be granted in the current situation (negative impacts of rainfall as determined by the motivation knowledge in relation to security constraints). The amount of help that is granted to each LEMA agent in response to its request is determined by a priority rule base.

The reader may proceed by the following steps:

(a) *[Level 2]* Specify knowledge models—analogous to those presented for LEMA agents—for DMA, TAMA and FBMA.

(b) *[Level 3]* Propose a detailed contents of the knowledge bases for the four types of agents.

(c) *[Level 1]* Perform a "manual" simulation of agent reasoning and agent interaction.

2. Consider the urban traffic management domain as described in section 11.4.3. A fictitious road-network is given, consisting of one beltway and three adjacent highways as shown in Figure 11.14. Highways hit the beltway at junctions J1, J2 and J3 respectively, and meet each other in the center of the town at junction J4. The entry and exit points to the network are labelled by E1, E2 and E3. There are three traffic management agents, A-I to A-III, each responsible for the swift flow of traffic in a different beltway section. The agents act upon the traffic by means of messages to be displayed on any of the nine VMS P1 to P9.

The different VMS messages and their effects are as follows:

- switch off
  all drivers take the shortest path to their destination. For instance, if the amount of vehicles going from E1 to E2 increases by 500 veh/h, the beltway area of A-I will support an additional load of 500 veh/h.
- "to <exit> on beltway by <junction>"
  80% of the drivers follow the recommended route, while 20% still try the shortest path. In the above example, the message "to E2 on beltway by J3" will result in an additional load of 100 veh/h for the A-I and 400 veh/h for the A-II and A-III areas.
- "to <exit> by center"
  just half of the drivers will obey the message and pass through the center, while the rest still tries the direct path. In the above example, the message "to E2 by center" will result in an additional load of 250 veh/h in the A-I area and another 250 veh/h in the center.

If one agent wants to display some messages M on a VMS while another would like the same panel to be switched off, message M will be displayed. Messages "to X by Z" and "to Y by Z" can be merged into a message "to X and Y by Z." All other combinations of messages on the same panel are incompatible.

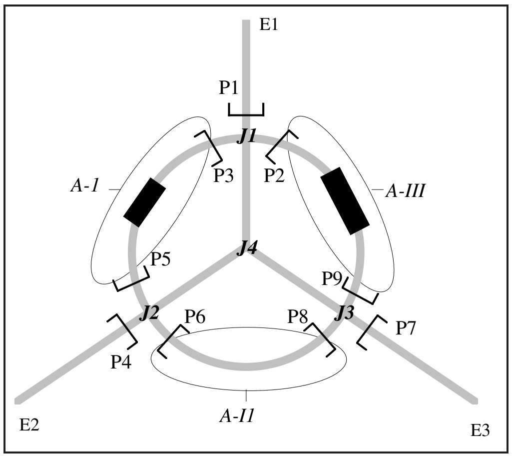

Figure 11.14 Traffic scenario.

Consider a scenario with a medium size congestion in the area of A-I and a severe congestion in the area of A-III as shown in Figure 11.14. In the downtown and A-II areas there is only light traffic; these areas can absorb further traffic load without generating problems.

(a) *[Level 1]* Which are the different alternative local signal plans generated by the agents A-I and A-III? They should be ordered by their effectivity in the eyes of A-I and A-III.

(b) *[Level 3]* Develop knowledge models for the A-I and A-III agents based on the agent architecture presented in this chapter. Enacting this knowledge agents should come up with the local signal plans mentioned above.

(c) *[Level 1]* Simulate the social method *coordination* performing "severity-based" integration of local signal plans as in section 11.4.3. What is the resulting global signal plan?

(d) *[Level 2]* What "compromise solutions" are possible as global signal plans. What criteria could a social method use so as to ensure that its outcome is such a compromise solution?

(e) *[Level 4]* Develop a formal framework by means of which you specify the precise meaning of your notion of a "compromise solution." How can such solutions be computed in a distributed fashion? You may want to borrow from part one of this book (especially the chapters 4 and 5) to accomplish this.

11.7 References

1. Alonso, M.; Cuena, J.; Molina, M.: SIRAH: An Architecture for a Professional Intelligence. In *9th Europ. Conf. on Artificial Intelligence (ECAI-90)*. Pitman, 1990.
2. Avouris, N.: Cooperating Knowledge-based Systems for Environmental Decision Support. *Knowledge-based Systems* 8 (1): 39-54, 1995.
3. Barandiaran, J.; Laresgoiti, I; Perez, J.; Echavarri, J.; Corera, J.: Solucion de Problemas Complejos utilizando diversos puntos de vista. In *Proc. Spanish Conf. on Artificial Intelligence(CAEPIA-91)*, pages 413-421, 1991.
4. Barbuceanu, M.; Fox, S.: COOL: A Language for Describing Coordination in Multi Agent Systems. In *Proc. 1st Int. Conf. on Multiagent Systems (ICMAS-95)*, pages 17-24. AAAI/MIT Press, 1995.
5. Breuker, J., van de Velde, W.: *CommonKADS Library for Expertise Modelling*. IOS Press, 1994.
6. Brown, D.; Chandrasekaran, B.: *Design Problem Solving*. Morgan Kaufmann, 1989.
7. Chandrasekaran, B.; Johnson, T.; Smith, J.: Task-Structure Analysis for Knowledge Modelling. *Communications of the ACM 35 (9)*. 1992.
8. Clancey, W.: Heuristic Classification. *Artificial Intelligence 27*. 1985.
9. Correra, J.; Laresgoiti, I.; Cockburn, D.; Cross, A.: A Co-operative Approach Towards the Solution of Complex Decision Problems in Energy Management and Electricity Networks. In *Proc. Int. Conf. on Electricity Distribution*, pages 4.19.1-4.19.6, 1994.
10. Cuena, J.: The Use of Simulation Models and Human Advice to Build an Expert System for the Defense and Control of River Floods. In *Int. Joint Conf. on Artificial Intelligence (IJCAI-83)*. Morgan Kaufmann, 1983.
11. Cuena, J.; Molina, M.; Garrote, L.: An Architecture for Cooperation of Knowledge Bases and Quantitative Models: The CYRAH Environment. In *11th Int. Workshop on Expert Systems. Special Conference on Second Generation Expert Systems (Avignon-91)*. EC2, 1991.
12. Cuena, J.; Hernandez, J.; Molina, M.: Case Presentation for the Use of Knowledge-based Models for Traffic Management – Madrid. In *Proc. 1st World Congress on Applications of Transport Telematics and Intelligent Vehicle-Highway Systems*, pages 564-571, 1994.
13. Cuena, J.; Hernandez, J.; Molina, M.: Knowledge-based Models for Adaptive Traffic Management. *Transportation Research* 3 (5): 311-337, 1995.
14. Cuena, J.; Hernandez, J.; Molina, M.: Knowledge Oriented Design of an Application for Real Time Traffic Management. In *Proc. 12th Europ. Conf. on Artificial Intelligence (ECAI-96)*, pages 308-312. Wiley, 1996.
15. Cuena, J., Hernandez, J.: An Exercise of Knowledge Oriented Design: Architecture for Real Time Decision Support Systems. In S. Tzafestas, editor, *Knowledge-Based Systems. Advanced Concepts, Techniques and Applications.* World Scientific, 1997.
16. Emaldi, M.; Fernandez, J.; Laresgoiti, I; Perez, J.; Amantegui, J.; Echavarri, J.: Design and Operation of an Expert System Prototype for Fault Analysis in Electrical Nets. In J. Campbell and J. Cuena, editors, *Perspectives in Artificial Intelligence Vol. 1*, pages 145-152. Ellis Horwood, 1989.

17. European Commission: Telematics Applications Programme (1994-1998). In *Guide to the 1995-1996 Telematics Projects*. European Commission, Directorate-General XIII, Telecommunications, Information Market and Exploitation of Research, 1996.
18. Haugeneder, H.; Steiner, D.: A Multi-agent Approach to Co-operation in Urban Traffic. In *Proc. Co-operative Knowledge-based Systems SIG Workshop*, pages 83-99, 1993.
19. Hewitt, C.: Viewing Control Structures as Patterns of Message Passing. *Artificial Intelligence* 8: 323-364, 1977.
20. Jennings, N.; Corera, J., Laresgoiti, Y.; Mamdani, E.; Perriollat, F.; Skarek, P.; Varga, L.: Using ARCHON to Develop Real-world DAI Applications for Electricity Transportation Management and Particle Accelerator Control. In *IEEE Expert Special Issue on Real World Applications of DAI systems*, 1996.
21. Newell, A.: The Knowledge Level. *Artificial Intelligence* 18: 87-127, 1982.
22. Ossowski, S.; Garcia-Serrano, A.; Cuena, J.: Emergent Co-ordination of Flow Control Actions Through Functional Co-operation of Social Agents. In *Proc. 12th Europ. Conf. on Artificial Intelligence (ECAI-96)*, pages 539-543. Wiley, 1996.
23. Ossowski, S.; Garcia-Serrano, A.: Social Co-ordination Among Autonomous Problem-solving Agents. In *Proc. 3rd Australian Workshop on Distributed Artificial Intelligence (DAI-97)*, 1997.
24. Ossowski, S.: *On the Functionality of Social Structure in Artificial Agent Societies—Emergent Co-ordination of Autonomous Problem-solving Agents*. Ph.D. Thesis, TU Madrid, School of Computer Science, AI Dpt., 1997.
25. Parunak, H.: Visualizing Agent Conversations: Using Enhanced Dooley Graphs for Agent Design and Analysis. In *Proc. 2nd Int. Conf. on Multiagent Systems (ICMAS-96)*, pages 275-282. AAAI/MIT Press, 1996.
26. Puppe, F.: *Systematic Introduction to Expert Systems*. Springer, 1993.
27. Russell, S.; Norvig, P.: *Artificial Intelligence—A Modern Approach*. Prentice Hall, 1995.
28. Sichman, J.; Demazeau, Y.; Conte, R.; Castelfranchi, C.: A Social Reasoning Mechanism Based on Dependence Networks. In *Proc. 11th Europ. Conf. on Artificial Intelligence (ECAI-94)*, pages 188-192. Wiley,1994.
29. van de Velde, W.: Issues in Knowledge-level Modelling. In J. David, J. Krivine and R. Simmons, editors, *Second Generation Expert Systems*, pages 211-231. Springer, 1993.
30. Varga, L.; Jennings, N.; Cockburn, D.: Integrating Intelligent Systems into a Cooperating Community for Electricity Distribution Management. In *Proc. Int. Journal of Expert Systems* 7 (4): 168-179, 1994.
31. Winograd, T.; Flores, F.: *Understanding Computers and Cognition*. Ablex, 1986.
32. Zhang, M.; Zhang, C.: Analysis and Methodologies of Synthesis of Solutions in Distributed Expert Systems. In *Proc. 2nd Int. Conf. on Multiagent Systems (ICMAS-96)*, pages 417-424. AAAI/MIT Press, 1996.