Knowledge Modelling in Multiagent Systems: The Case of the Management of a National Network

Martin Molina [1] and Sascha Ossowski [2]

[1] Department of Artificial Intelligence, Technical University of Madrid,
Campus de Montegancedo s/n, 28660 Boadilla del Monte (Madrid), Spain
mmolina@dia.fi.upm.es

[2] School of Engineering, Rey Juan Carlos University,
Campus de Móstoles, Calle Tulipán s/n, 28933 Móstoles (Madrid), Spain
S.Ossowski@escet.urjc.es

Abstract. This paper presents the knowledge model of a distributed decision support system, that has been designed for the management of a national network in Ukraine. It shows how advanced Artificial Intelligence techniques (multiagent systems and knowledge modelling) have been applied to solve this real-world decision support problem: on the one hand its distributed nature, implied by different loci of decision-making at the network nodes, suggested to apply a multiagent solution; on the other, due to the complexity of problem-solving for local network administration, it was useful to apply knowledge modelling techniques, in order to structure the different knowledge types and reasoning processes involved. The paper sets out from a description of our particular management problem. Subsequently, our agent model is described, pointing out the local problem-solving and coordination knowledge models. Finally, the dynamics of the approach is illustrated by an example.

1. Introduction

The problem of coherent distributed decision-making is intrinsic to many complex real-world situations, where the behaviour of a complex dynamic system is to be regulated by a group of operators, each performing particular control actions on different parts of the system. In the management of a road traffic network, for instance, different control centres are responsible for making certain types of control decisions (e.g. private vehicles, public transport, etc.). Another example is the management of a computer network, where different local administrators reconfigure sub-networks to improve network performance. In general, local decisions made by a particular operator may affect the decisions of other control personnel. So, besides making good local control decisions, it is necessary to achieve certain level of coordination in order to obtain a coherent and mutually acceptable set of local control decisions.

A *distributed decision support system* is a software tool that assists operators in their decision-making within the frame of a distributed organisation, by automatically monitoring a dynamic system, warning about present or future undesired situations and suggesting appropriate control actions to operators [9]. In order to develop a

In "Intelligence in Services and Networks". H. Zuidweg et al. (Eds.). Proc. 6th International Conference on Intelligence and Services in Networks, IS&N 99, LNCS 1597, pp. 501-513. Barcelona, Spain, April, 1999.

software architecture that implements such a system, a variety of technical problems have to be considered: efficient communication between management centres, a powerful interface to the dynamic system to be controlled etc. However, there are two key issues that are crucial for the successful implantation of this kind of system in an existing organisation:

- Operators are responsible for the effects of the control decisions taken. As bad decisions may have catastrophic effects, the support system must be able to provide *explanations* respecting its recommendations, which need to be logically and conceptually *understandable* by the operators
- The operators' responsibility goes in line with certain *autonomy* in their decision making. In such a distributed organisation a decision support system will not be accepted, if it tries to achieve coordination by centrally imposed decisions, without taking into account local preferences and autonomy

In order to cope with these characteristic features, two advanced techniques from the field of Artificial Intelligence appear adequate:

- *Knowledge-based systems* technology is popular for its capability of providing explanations. By using recent knowledge modelling techniques, the knowledge of local decision-makers can be organised and structured, so as to provide explanations at a reasonable level of complexity.
- *Multiagent* techniques provide a natural context to solve the coordination problem between local decisions: unlike centralised traditional approaches that rely on a top-level authority to solve conflicts, each agent is given the autonomy to make its own decisions. Coordination emerges by means of peer communication in a "bottom-up" fashion. Its result is biased by existing conventions and organisational structures.

Still, when trying to build real-world systems in a principled manner on the basis of the above insights, a problem arises. Although there is a variety of methodologies and tools that support design and implementation of "monolithic" knowledge-based systems (e.g. [25, 8, 20, 15]), similar approaches for multiagent system design (e.g. [16, 12, 4]) are still in quite initial stages. A convincing integration of knowledge modelling and multiagent design methodologies is still to come.

In this paper we take a pragmatic tack towards the problem, showing how existing knowledge modelling approaches can be transferred to solve a real-world distributed decision support problem. We first describe our particular problem: the management of a national Wide Area Network in Ukraine. Local problem-solving and coordination knowledge models are described next within a task-oriented knowledge modelling framework, giving rise to the implementation. We illustrate the dynamics of the system by an example and discuss the lessons learnt from this enterprise.

2. The Problem: The Management of a National Network

The government of Ukraine recently initiated the programme *Informatization of Ukraine*, directed towards the development of telecommunication infrastructures and information technologies at a national level, during the years 1996-2000. Together with partners from Greece, Spain, Ukraine and Hungary we contributed to this

enterprise in the frame of the project *ExperNet*, which is financed through the European Union's Inco-Copernicus Programme. The goal of the *ExperNet* project was to develop a distributed expert system that supports the management of an Ukrainian national data network.

Within this context, we considered that most conventional network management systems (e.g. [11]), based on algorithmic techniques, are usually not able to provide the required level of assistance for the operators. Such a level of support requires conceptually understandable recommendations together with explanations, which can be offered by knowledge-based technology. Recently, agent-based approaches have been applied to network management [13,23], but they attack mainly complexity issues and provide rather limited support to the different node operators within the framework of a distributed organisation. In the sequel we are concerned with our contribution to the project: the definition of the conceptual model of the system, and the development of a computational version.

Relcom, one of the main national networks of Ukraine, was chosen as the experimental zone of the *ExperNet* project. *Relcom* mainly provides Internet services (email, ftp, news, telnet, Cyrillic-oriented version of Lynx for the access to www-servers, etc.). The connectivity between the nodes is based on UUCP using switched telephone lines and permanent/leased lines using TCP/IP. Besides the usual dial-up modem connection, *Relcom* makes use of different radio, fibre and satellite channels. As illustrated by figure 1, the network includes several hundreds of network nodes provided by independent organisations. For the access to international networks channels provide speeds up to 512 K.

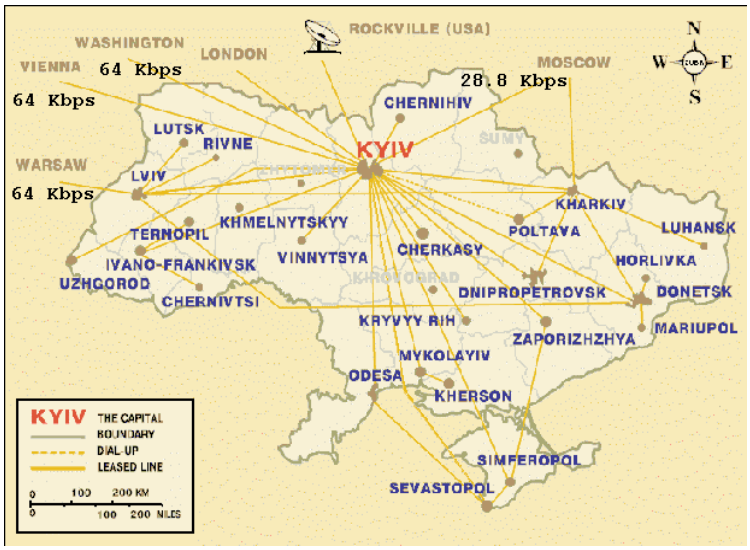

Figure 1. The Relcom Network in Ukraine (Courtesy of Relcom).

The infrastructure of a typical network node is shown in figure 2. The node is endowed with an external link of 256 K, and maintains connections to four other network nodes: Technosoft (256K), UACOM (33.6K), the Institute of Physics and Semiconductors (28.8K) and to the Donetsk University of Physics and Technology (64K). Routing is supported by two Cisco 2511 ("goodwin" and "gal") and one Cisco 2501 ("skiff") hardware routers. A Sun UltraSparc1 and 8 Pentium computers are configured to provide a variety of different services.

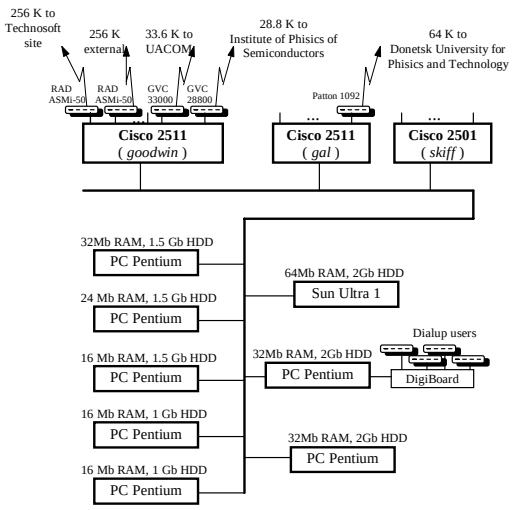

Figure 2. Typical node infrastructure (Courtesy of Technosoft).

At each node of the Relcom network, there is a node administrator, responsible of maintaining a convenient quality of service. The administrator obtains information about the state of certain part of the network by means of a whole bunch of different *observables*, including current and historic data on link and host utilisation, routing tables, message buffer queues, average message delays between nodes, service delays, etc.. In the course of the ExperNet project several tools have been installed (such as HNMS+, an adapted version of the Hierarchical Network Management System [11], and the DEVICE system [2], which includes an object-oriented database), by means of which most of the above data is supplied on-line. Administrators can also access additional variables as the result of *exploratory actions*, such as the output of executing operating system commands.

3. Knowledge Acquisition

During the knowledge acquisition phase it became clear that node administrators perform essentially three tasks for local problem-solving. (1) *fault detection*: in case of communication malfunctions, a repair plan is to be configured that besides the primary actions (e.g. repair equipment) also comprises additional secondary actions that isolate a fault so as to minimise its impact (e.g. change routing); (2) *performance optimisation*: in order to assure a convenient network performance, resource utilisation is to be balanced, by deferring services (e.g. news), changing services to different hosts, changing routing tables etc.; (3) *network re-configuration*: while maintaining a certain quality of service, the economic efficiency of the network is to be assured through actions such as increasing/decreasing the capacity of channels, leasing or cancelling new lines, installing new equipment etc.

It soon turned out that coordination problems play an essential role in the tasks of administrators. They arise in at least four situations: (1) *Information acquisition problems*: a node lacks observables which are available to (or can be acquired by) another node. (2) *Responsibility problems*: nodes with overlapping problem-solving capacities have to agree upon which one of them is to attack a task. (3) *Interest problems:* a node's plan to overcome a problem concerns another node, because it affects the utilisation of resources (hosts, links etc.) that "belong" to the latter (who paid for them). Still, the affected node does not agree with this. (4) *Resource conflicts*: nodes develop complementary plans (in order to cope with different problems), but these plans try to employ the same, scarce resource in different, incompatible ways.

In order to solve these problems, the internal organisation of the *Relcom* network has evolved towards a hierarchical structure. Nodes may correspond to any of three levels (national, regional and district), associated with different authority and responsibility patterns. Within this framework, nodes communicate in accordance with certain conversation patterns so as to overcome coordination problems. Information acquisition problems are tackled by observable acquisition conversations, where a node simply asks an acquaintance to provide a desired information. Responsibility problems are coped with by delegation to higher level nodes. The solution to interest problems relies on a conversation scheme, in the course of which a higher level node successively generates alternative management plans, until all affected nodes agree with it. The occurrence of resource conflicts are mostly avoided by a hierarchical link placements, but if they occur their solution involves peer negotiation.[1]

4. The System Architecture

Our objective was to design a distributed decision support system, which assists the *Relcom* node administrators in coping with their network management tasks. The inherent distribution of the problem suggested to use a multiagent system architecture, where each management node in the network is endowed with one *agent*, specialised in managing the network area that the node is responsible for. Each decision support agent communicates the results of its reasoning processes to its human administrator, which is in charge of settling the management actions to be taken. In this section we describe an agents' "individual" knowledge endowment for local problem-solving as well as its "social" knowledge that enables it to render support to inter-node coordination.

4.1 The knowledge modelling approach

In order to characterise the knowledge model of each agent we have applied the concept of *model-based* expert system development, which has become a popular approach to the development of complex knowledge-based systems. This modelling approach considers the existence of an abstract *knowledge level* [18] at which the

[1] Resource conflicts are out of the scope of the current version.

knowledge can functionally described on the basis of its role, independently on the particular symbolic representation. Some recent methodologies for system development (such as KADS [23], KSM [8], Protégé-II [19] and KREST [15]) follow this model-based approach. Any of these methodologies organise the knowledge according to certain structuring principles. In particular, one organisation followed by most of the methodologies is the task-oriented perspective.

According to this view, a *task* is an abstract description that identifies a goal to be achieved (for instance, mineral classification or the design of the machinery of an elevator). Tasks are usually characterised by the classes of premises that they receive as input and the classes of conclusions that they produce as output. On the other hand, *problem-solving methods* (or *methods* in short) are used to cope with the tasks. A method indicates *how* a task is achieved, by describing the different reasoning steps by which its inputs are transformed into outputs. Simple tasks (called *primitive inferences* following the KADS methodology [23]) can be attained directly by means of *basic methods*. They rely on a *knowledge base*, modelling the declarative domain knowledge used by basic methods. The complexity of real-world tasks requires compound methods that decompose tasks into subtasks. These subtasks may again be decomposed by a method and so on, developing a task-method-subtask hierarchy, whose leaves are given by basic methods that use simple knowledge bases. Thus, the resulting model for an agent can be viewed as a collection of types of knowledge bases (each one with its own symbolic representation) together with a hierarchically structured set of reasoning strategies that make use of such knowledge bases.

4.2 Agent Local Knowledge Model

We have modelled the problem-solving competence of each decision support agent within this task oriented perspective (figure 3). It reflects the fact that a node administrator's reasoning comprises three main tasks, which follow a three step sequence: (1) *symptom detection*, where administrators watch out for symptoms of undesired network states and behaviours (e.g. a certain service –ftp, www, etc.– does not respond, a host is unreachable, over/under-utilisation of links or equipment, etc.), (2) *diagnosis*, which is done by discriminating hypothesis of different degrees of precision on the basis of network data and the result of exploratory actions to find the causes of symptoms (e.g. inadequate capacity for some resource, unbalance of workload and resources, resource malfunctions, etc.) and (3) *repair*, where a sequence of repair actions is proposed to solve the problem. Figure 3 shows this abstract reasoning structure corresponding to an agent. The top-level method *network diagnosis & repair* performs the reasoning cycle outlined above: first, the *detect* method generates symptoms, whose causes are determined during the invocation of the *diagnose* method, which deduces a set of current problems. Finally, the *repair* method generates repair plans by means of which these problems shall be overcome.

In order to achieve these sub-tasks we have adapted three well-known problem-solving methods: heuristic classification for symptom detection, establish-and-refine for diagnosis, and hierarchical planning for repair.

The *heuristic classification* problem-solving method [6] identifies a typical reasoning structure for classification problems, such as symptom detection. It follows

three steps (abstraction, matching and refinement) that are performed in our model by three subtasks using two types of knowledge bases: one about the network model for abstraction and refinement, that includes a declarative representation of the network structure, and another one that uses a set of problem scenarios relating symptoms and observables.

For diagnosis, the *establish and refine* method is used [5]. This method can be conceived as an abstract reasoning pattern based on a heuristic search in a taxonomy of hypotheses of problems. The particular adaptation of the *establish and refine* method in the network management domain makes use of three primitive inferences: (1) *refine problem hypotheses* uses a knowledge base represented by a taxonomy of hypothesis classes using the *is-a* relation; (2) *select best hypothesis* makes use of knowledge about the validity of hypotheses (represented using frames) to establish whether any of the input hypothesis can be proved; (3) *acquire additional observables* determines the sequence of exploratory actions to get additional observables by using a knowledge base about acquisition methods (represented by rules).

Finally, the *hierarchical planning* method is used for the repair task. This method is based on a search in a hierarchy of specialists that are knowledgeable about partial abstract plans, which are dynamically composed during the reasoning [3]. The particular instance of the *hierarchical planning* method that we use in the network management domain makes use of four specialists (top level, fault detection, performance management and configuration) and uses five primitive inferences supported by four types of knowledge bases.

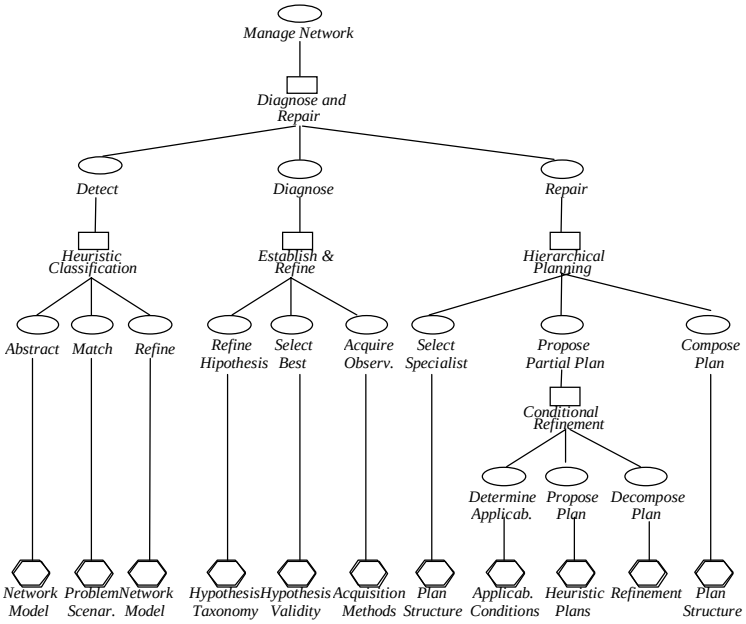

Figure 3. Knowledge model of an agent.

4.3 Agent Social Knowledge Model

In this section we show how social abilities can be "wrapped" around the local problem-solving capabilities of our stand-alone decision support agents within the task-oriented knowledge modelling framework. For this purpose we see the different conversations that resulted from the knowledge acquisition process (see section 3) as logically coherent sequences of agent interactions [1]. Interactions within a conversation are based on a message passing model. Every message that is exchanged during such interactions conveys a *speech act*, as by emitting it the sender wants to influence the behaviour of the receiver [17]. In consequence, we identified additional methods that generate such speech acts as part of their execution, given that the agent is to *act* within a conversation. In much the same way, methods have been built, which are activated when agents need to *react* to a speech act within the conversation, i.e. which are executed when certain messages are received.

The first column of table 1 shows the most important methods that comprise speech acts as well as the message types that they generate during their execution (column 2). In column three you see the agents' reaction upon the reception of messages, again in terms of methods to be activated. Note that for the problem at hand just three speech acts were sufficient: *DO* directive acts, where the sender wants the receiver to perform a certain problem-solving activity; *ASK FOR* directive acts, where the sender wants the receiver to perform a certain activity, and to inform it about its outcome; and *ANSWER WITH* assertive acts, by means of which the sender provides the receiver of a message with a certain information.

Sender's active method	*Message types*	*Receiver's method activation*
query agents for observable	ASK FOR observable	reply with observable
query agents for acceptance	ASK FOR plan acceptance	reply with acceptance
query agents for refinements	ASK FOR plan refinements	reply with refinements
notify agents to diagnose	DO diagnosis and repair	perform diagnosis and repair
notify agents to isolate problem	DO isolation	perform problem isolation
notify agents to repair	DO repair	perform repair
reply with observable	ANSWER WITH observable	<continue with suspended method>
reply with acceptance	ANSWER WITH plan acceptance	<continue with suspended method>
reply with refinements	ANSWER WITH plan refinements	<continue with suspended method>

Table 1. Methods and Messages

Within conversations there are various degrees of freedom for the involved agents, as they usually may choose from several behaviour options (in the simplest case to accept or to reject a request). This accounts for the autonomy of the network administrator within the frame of the organisation. The behaviour of an administrator in a conversation (i.e. his/her choice among the different options) is not just determined by information respecting its local situation, but also by its knowledge and experience with other nodes in the network. This knowledge is represented in the *agent models* (this type of knowledge is also referred to as "acquaintance model" [7] or "external description" [22]).

An agent maintains such local agent model of all acquaintances that it interacts with in a frame-based knowledge base. Frames consist of three parts: the modelled agent's interest, its capability and its social relation. The *interest* part of a local agent model describes the local reasoning results that the modelled agent may be interested in. It contains slots for the problems and plan patterns of interest. The *capability* part describes the types of results of local reasoning that the modelled agent is capable of. It contains slots for the observables, problems, and plans that others may be asked to obtain or perform. Finally the *social relation* part describes the relation between the maintainer of the model and the modelled agent. For instance, it might contain an expression that determines the relative importance (or authority) of the latter. Note that every agent is also endowed with (reflective) knowledge about itself, which it maintains in a frame of similar structure.

Agent model knowledge provides basic functionalities that are needed by the methods of table 1 in order to cope with the "non-determinism" within conversations. These functionalities are expressed by the following basic inference methods: (1) *problem interest:* checks whether the modelled agent is believed to be interested in being notified about a problem (e.g. because it is indirectly affected by that problem and wants to isolate it in order to keep its effects as local as possible); (2) *plan interest:* checks whether the modelled agent is believed to be interested in being notified about a given plan (either because it is involved in it or because its side-effects concern the modelled agent); (3)*plan rights:* checks whether there is a need to obtain the agreement of the modelled agent for enacting a given plan; (4) *observation capability*, checks whether the modelled agent is believed to be capable of acquiring the value of a given observable; (5) *diagnosis capability*, checks whether the modelled agent is believed to be able to perform diagnosis for a given symptom; (6) *plan repair capability*, checks whether the modelled agent is believed to be able to elaborate a plan for a given problem; (7) *plan refinement capability*, checks whether the modelled agent is believed to be capable of refining a given abstract plan for a given problem.

It remains to be shown how the "social" methods described in this section are finally integrated with the local problem-solving methods described previously. For this purpose, the "symptom detection – diagnosis – repair" cycle that coped with the *manage network* task (see figure 3) is extended. The new top-level method for the social decision support agents is given by the following steps:

```
1. Detect symptoms.
2. Inform agents interested in the symptoms, in order to diagnose them.
3. Diagnose problem (if the agent is responsible).
   If there are missing observables, ask agents for acquiring the
   corresponding value.
4. Inform agents interested in problems, in order to isolate them.
5. Inform agents interested in problems, in order to repair them.
6. Generate a repair plan (if the agent is responsible).
   If necessary, asks agents for plan acceptance
```

4.4 Implementation

The above system has been implemented on top of the KSM knowledge modelling tool [8], developed at the Artificial Intelligence Department of the Technical University of Madrid, and has been applied to different test cases. The final version of this system, using efficient software components, has been installed at the end of 1998 at a particular test site that included several network nodes in Ukraine [24]. The initial experience showed the validity of the present technical solution, although further trials with a more extensive installation still need to be performed.

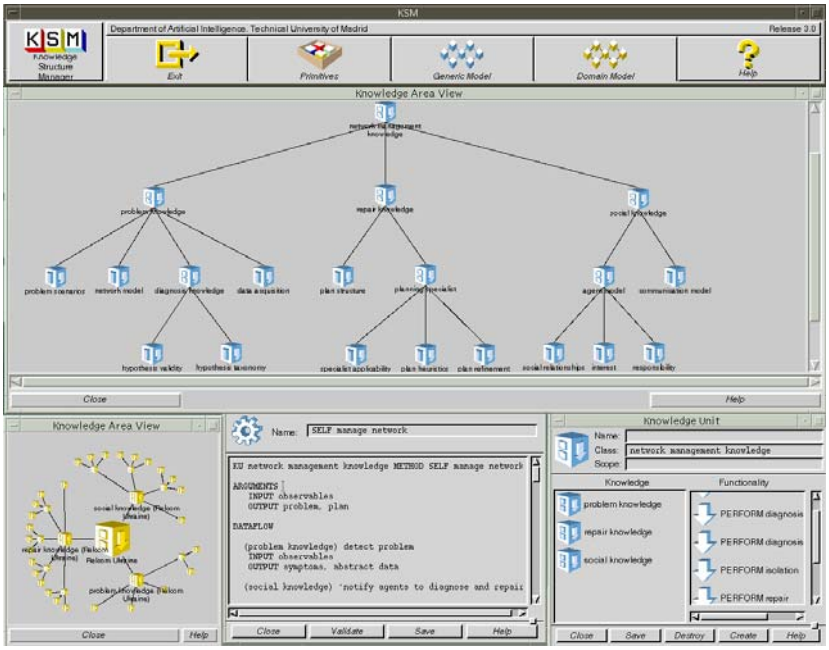

Figure 4. KSM Interface to an *ExperNet* agent model

5. An Example

This section illustrates the previous model on the basis of an example case. The case comprises three agents – *Relcom Ukraine*, *Technosoft* and *UACOM*– corresponding to three network nodes, and presents the interactions that arise from a certain situation in the frame of a reconfiguration task. Suppose that the administrator at the *Technosoft* node is notified by an external observer that the FTP service of the node *UACOM* is slow. This is notified manually by the administrator to the local agent at this node and triggers the task for diagnosis and repair of this agent. Internally, this agent activates specific subtasks and concludes that an additional observable is needed to discriminate between the hypotheses (state of carrier detect indicator of Relcom

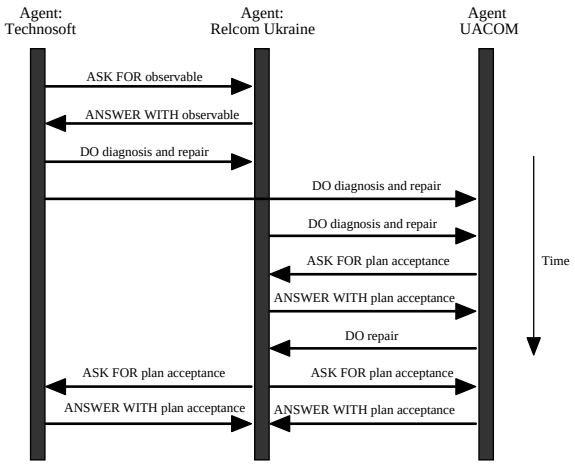

Figure 5. Agent Interaction in the example.

Modem on Technosoft channel). The agent uses its own social knowledge about the other agents, and determines that only *Relcom Ukraine* is capable acquiring the value. So, *Technosoft* sends the message *ASK FOR observable* to agent *Relcom Ukraine* and waits until the answer from that agent arrives. Upon arrival of the *Technosoft* message, the agent *Relcom Ukraine* replies with the message *ANSWER WITH observable* after acquiring the value of such a variable.

Once this message arrives at *Technosoft*, the diagnosis process continues and concludes with a partial description of the problem (WAN problem between Relcom Ukraine and UACOM), stating that this type of problem is under foreign responsibility. The agent *Technosoft* determines which agent might continue the diagnosis concluding that both agents, *UACOM* and *Relcom Ukraine*, are responsible for this problem. So, *Technosoft* sends the corresponding *DO diagnosis & repair* messages and finishes its reasoning cycle. Upon receiving the above message, the agent *Relcom Ukraine* continues with the diagnosis and detects that only *UACOM* is responsible for this problem. So, *Relcom Ukraine* also sends the message *DO diagnosis & repair* to *UACOM* and terminates its reasoning cycle.

When the agent *UACOM* receives the above messages, it concludes the "final cause" of the detected problem (link problem between *Relcom Ukraine* and *UACOM*). *UACOM* tries to select the agents to be notified about the result of diagnosis in order to isolate it. For this, *UACOM* determines that no agents need to be informed for the sake of problem isolation. The next step consists in determining the agents that will perform problem repair. *UACOM* determines that itself will perform the repair and thus sends no messages. It performs hierarchical planning to compose a repair plan, and checks whether it can be enacted directly or whether the acceptance of any acquaintance is necessary. By looking up its social knowledge, it determines that this plan affects agent *Relcom Ukraine*, so this agent needs to be asked for plan acceptance. In consequence, it sends the message *ASK FOR plan acceptance* to *Relcom Ukraine*. Still, in its answer *Relcom Ukraine*, rejects the plan.

As a consequence, problem-solving actions in *UACOM* are "redone" successively in order to generate an alternative plan. However, this procedure does not find a valid plan and *UACOM* determines that problem repair needs to be delegated. Using the social knowledge *UACOM* determines that agent *Relcom Ukraine* should do this job, so it sends the message *Do repair* and finishes its reasoning cycle. Upon receiving the above message, the agent *Relcom Ukraine* proposes a new repair plan and checks whether the acceptance of any acquaintance is necessary. By looking up its social

knowledge, it detects that this plan affects agent *UACOM* as well as agent *Technosoft*. In consequence, *Relcom Ukraine* sends *ASK FOR plan acceptance* messages to both agents. Finally, both agents accept the proposed plan and return *ANSWER WITH plan acceptance* messages to the *UACOM* agent, indicating that they accept the plan.

6. Conclusions

This paper has reported our experience in the development of a real-world distributed decision support system. We have argued that a combination of knowledge based systems technology and multiagent techniques is appropriate for this task, showed how both approaches can be expressed homogeneously in a task-oriented knowledge modelling framework, and sketched the implementation and operation of the system.

This paper claims to provide a pragmatic contribution in the area of multiagent system design rather than a theoretical approach. We built our system around the knowledge obtained during the knowledge elicitation phase of the project, whose structure implied the "shape" of our agent model. By this, we have shown that, at the time being, knowledge-based multiagent systems *can* be built in a principled fashion on the basis of today's knowledge engineering tools. Setting out from our experience in the *ExperNet* project we argue that without our multiagent problem analysis and knowledge-centred structuring approach the construction of the system would have been much more difficult (or maybe even impossible given the available resources). The key point for the success of our approach in the present case study lies in its "parsimony" and "economic" efficiency.

Acknowledgements

This work has been supported by EU's Inco-Copernicus project no. 960114 (*ExperNet*). The partners from Ukraine, Technosoft and the Glushkov Cybernetics Institute, provided expertise about network management. The partner Link S.A. from Greece provided also expertise in network management and participate in the knowledge elicitation process. The partner Aristotle University of Thessaloniki from Greece was the coordinator of the project and, among others activities, was responsible for the development software tools for knowledge representation and the final implementation of the system (together with Link S.A). The partner from Hungary, ML Consulting and Computing Ltd., provided a distributed logic programming environment for the implementation of the final system.

References

1. Barbuceanu M., Fox S.: "COOL: A Language for Describing Coordination in Multi Agent Systems". Proc. ICMAS, 1995

2. Bassiliades N.; Vlahavas I.: "Processing Production Rules in DEVICE, an Active Knowledge Base System", *Data & Knowledge Engineering 24(2)*, pp. 117-155, 1997
3. Brown, D.; Chandrasekaran, B.: *Design Problem-solving: Knowledge Structures and Control Strategies*, Morgan Kaufman, 1989
4. Burmeister, B.: "Models and methodology for agent-oriented analysis and design". *Proc. KI-96 Workshop on Agent-oriented Programming and Distributed Systems*, DFKI, 1996
5. Chandrasekaran, B.; Johnson, T.; Smith, J.: "Task-Structure Analysis for Knowledge Modelling". *Communications of the ACM 35 (9)*, 1992
6. Clancey W.: "Heuristic Classification". *Artificial Intelligence 27*, 1985
7. Cockburn, D.; Jennings, N.: "ARCHON: A Distributed Artificial Intelligence System for Industrial Applications". In: *Foundations of DAI* (O'Hare & Jennings, eds.), Wiley, 1996
8. Cuena J., Molina M.: "KSM: An Environment for Design of Structured Knowledge Models". In: *Knowledge-based Systems: Advanced Concepts, Techniques and Applications* (Tzafestas, ed.), World Scientific, 1997.
9. Cuena J., Ossowski S.: "Distributed Models for Decision Support". To appear in: *Introduction to Distributed Artificial Intelligence* (Weiß & Sen, eds.), MIT Press, 1998
10. Cuena, J.: "Los Sistemas multiagente basados en el conocimiento: Una posible alternativa para la ingeniería del Software". To appear in a special issue on DAI of *Inteligencia Artificial - Revista Iberoamericana sobre I.A.* (García-Serrano & Ossowski, eds.), 1998
11. George J.; Schecht L.: "The NAS Hierarhical Network Management System". In: *Integrated Network management III*, (Hegering & Yemini, eds.), Elsevier, 1993
12. Glaser, N.: *Contribution to Knowledge Modelling in a Multi-Agent Framework*. Ph.D. thesis, L'Université Henri Poincaré, Nacy Y, 1996
13. Gyires, T.: Intelligent Routing Agents in Wide-Area Networks. *Intelligent Agents for Telecommunications Applications* (Albayrak, ed.), IOS Press, 1998
14. Matov A.: "The development of Internet-like networks in Ukraine". *Networks and Telecommunications,, no.2*, 1997, pp.4-11
15. McIntyre A.: *KREST User Manual 2.5*. Vrije Universiteit Brussel, AI lab, Brussels, 1993
16. Moulin B., Brassard M.: "A scenario-based design method and an environment for the development of multiagent systems". In: *DAI Architectures and Modelling* (Zhang & Lukose, eds.), Springer, 1995
17. Müller, H.-J.: "Negotiation Principles". In: *Foundations of DAI* (O'Hare & Jennings, eds.), Wiley, 1996
18. Newell A.: "The Knowledge Level". *Artificial Intelligence 18*, 1982, pp. 87-127
19. Ossowski S.: *Social Structure in Artificial Agent Societies*. LNAI 1535, Springer, 1998
20. Puerta A.R., Tu S.W., Musen M.A.: "Modelling Tasks with Mechanisms". *International Journal of Intelligent Systems, Vol. 8*, 1993.
21. Rao, A.; Georgeff, M.: "BDI Agents: From Theory to Practice". *Proc ICMAS-95*, 1995
22. Sichman J., Demazeau Y.: "Exploiting Social Reasoning to deal with Agency Level Inconsistencies". *Proc. ECAI-94*, 1994
23. Somers, F.: HYBRID: Intelligent Agents for Distributed ATM Network Management. *Intelligent Agents for Telecommunications Applications* (Albayrak, ed.), IOS Press, 1998
24. Vlahavas, I. et al.: "System Architecture of a Distributed Expert System for the Management of a National Data Network". *Proc. Int. Conf. on Artificial Intelligence – Methodology, Systems, Applications (AIMSA)*, Springer, 1998
25. Wielinga B.J., Schreiber A.T., Breuker J.A.: "KADS: A Modelling Approach to Knowledge Engineering". *Knowledge Acquisition*, 1992.