

Knowledge Acquisition in TRACE*

*Gennady Agre**, *Roberto Paggio***, *Leon Batachia****

** Institute of Information Technologies, 1113 Sofia*

*** Italsoft – Ingegneria di Sistemi, Roma*

**** Research Institute for Informatics, Bucharest*

1. Introduction

TRACE (TRAnsfering CHARADE technology in Central and Eastern Europe) is an ongoing research project whose objective is to provide a software platform enabling the development of applications for the management of environmental emergencies. The TRACE system falls within the wide category of Environment Management Systems (D e n z e r et al. [2]), or more precisely in the one of Environmental Decision Support Systems (EDSS) (G u a r i s o et al. [3]), which is gaining an ever increasing consideration by the public administrations and the scientific community. Indeed, the protection of the environment and the principle of sustainable development are high priority issues in most countries; on the other side, advances in ICT research made possible the provision of cost-effective and reliable solutions.

EDSSs are characterised by a complex approach involving several technologies, tools and devices, and dealing with different aspects of environment-related activities, and therefore addressing user needs which are differentiated according to the specific application domain, the operational context, the tasks to be supported, the organisational structure, the applying regulations.

Activities may be grouped in five general functional areas [10]:

- prediction (risk analysis, forecasts);
- medium/long-term planning;
- monitoring and surveillance;
- crisis/emergency management;
- post-incident damage evaluation and reclamation.

Another important classification criterion takes into account the context in which an EDSS must operate. Three kinds of contexts are usually identified: real-time management, training and simulation. TRACE is conceived as a system for real-time emergency management, whose main objective is to support the user (a decision-maker operating in a co-ordination centre) in assessing a crisis situation and building an effective intervention plan. The main benefits to be expected from such approach are:

* This research is done under the support of TRACE INCO-COPERNICUS Project CP-960138, funded by the European Union within the ESPRIT programme.

- decrease the human error rate (time and risk pressure are key factors in emergency management, which result in error-prone decisions);
- diminish the time required for performing a full assessment of the situation, by exploiting the Geographical Information System (GIS) visual capabilities and providing the ability to filter and display context-driven information;
- enable concurrent management of multiple situations (a common case for instance in wildfire domain);
- support optimised and safe usage of intervention resources (means, personnel, equipment, natural resources), which are usually limited in number and capabilities, and therefore must be exploited at best;
- facilitate the training of operators.

Although these features can be found also in other EDSS, the innovative character of TRACE relies on the attempt to provide them in a cost-effective, fully programmable environment.

The present paper is devoted mainly to the Task Editor – a component of the EDDS being developed within TRACE project. It is a tool for knowledge acquisition and documentation within the *Task analysis* (TA) and *task modelling* methodology developed within CHARADE project [1]. TE is intended to support the task modelling activities for two pilot applications – fire prevention and fighting in Bulgarian hard-to-reach massifs and wind-throw management in Romanian spruce forests

The paper is organised in six sections. The second section following the introduction briefly describes the TRACE architecture. The third and fourth sections consider the conceptual framework of the TRACE system, including user and application modelling methodologies, tools and techniques. The fifth section provides an example of a TRACE application which is currently under development. The last section is a conclusion.

2. TRACE architecture

The organisation of the various TRACE components shown in Fig. 1 reflects the typical architecture of most EDSS systems. The HCI part is in charge of driving the overall dialogue between the user and the system. It exploits a run-time task model, i.e. a structured model of the activities of the user, and a session controller, which enables concurrent handling of multiple emergency situations. The task model is effective in driving user action throughout a session, by providing continuous feedback about the situation being monitored, showing the realisation status of each task, enabling or disabling commands according to the current progress, displaying information which is relevant to the current task context (Marti [8]).

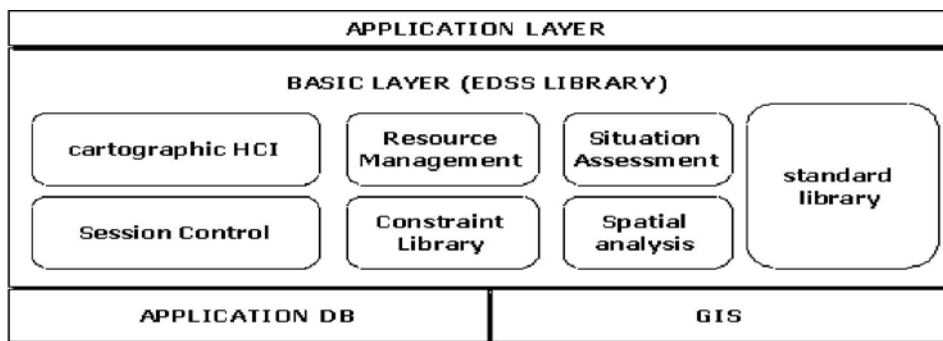


Fig. 1. TRACE components

The *Situation Assessment* module provides support for a quick analysis of an emergency situation, including basic spatial analysis and computational models based on geographically referenced data (e.g.: spread model). It exploits a general mechanism for handling dependencies and propagating events among objects, thus enabling automatic re-calculation of relevant information.

The *Intervention Planning* module enables the user to define missions and automatically retrieve an appropriate set of resources to accomplish missions goals. Requests for resources may be stated either in specific terms (e.g. a fixed number of resources belonging to a certain type), or as general criteria (e.g. minimisation of intervention costs). User requests for resources can be expressed with a simple language by the user, and are then translated into constraint satisfaction problem, which is handled by the constraint manager.

3. Methodological Framework

TRACE is a generic software platform providing support for application development at two different levels. As a software library, it consists of a set of programmable objects implementing basic functions for the management of environmental emergencies. As a methodological framework, it includes guidelines and tools for designing applications through task modelling methods.

3.1. Task analysis

Task analysis and *task modelling* is a general methodology for modelling *activities* which can be observed in a large variety of domains from human factors (ergonomics and cognitive psychology) to artificial intelligence or software system design. The notion of an activity may have very different scopes depending on the focus of interest. TA provides a simplifying approach to activity analysis; it relies on the identification and representation of the *task structure* underlying the activity. Generally speaking, a task is “what is to be done”, while an activity is “what is effectively done”. An activity is structured as the processing of a series of tasks performed by the operator and the system.

TA aims at capturing and representing task structures underlying user activities – mainly routine and operative activities. The result of TA is a *task model* – a model of the task structure of an activity that needs to be validated through a confrontation between its predictions and the observed traces of the effective activity. In this context we wish to emphasise the variety of the knowledge to be captured – *Declarative knowledge* (the goal and condition of a task and the object involved) and *procedural knowledge* (the process component of a task) (J o h n s o n, N i c o l o s i [6]).

TA distinguishes between design-time and run-time exploitation of a task model. Design-time exploitation (predesign task model) specifies the requirements for the design of the user interface, data and functionality of a system, while run-time exploitation (design task model) refers to the insertion of a task model as a particular software component of a system (K i r w a n A i n s w o r t h [7]). A task model is specified as:

- the description of *roles* as sets of tasks (a role is a set of tasks that an individual is responsible for performing);
- the description of *tasks*, along with additional information regarding representatives and typicality, interruptability of tasks, and temporal relations among tasks;
- the description of *objects*, described as a list of features.

The task-modelling notation used in TRACE includes a graphical notation for representing tasks (which is the main kind of notation used in the task model) and a notation for representing objects. The graphical notation is partially derived from TKS notation (J o h n s o n

Johnson [5]). The examples in Fig. 2, Fig. 3 and Fig. 4 illustrate some of the means for representing hierarchical task-subtask structure, as well as logical and temporal relations (Marti Normand [8]).

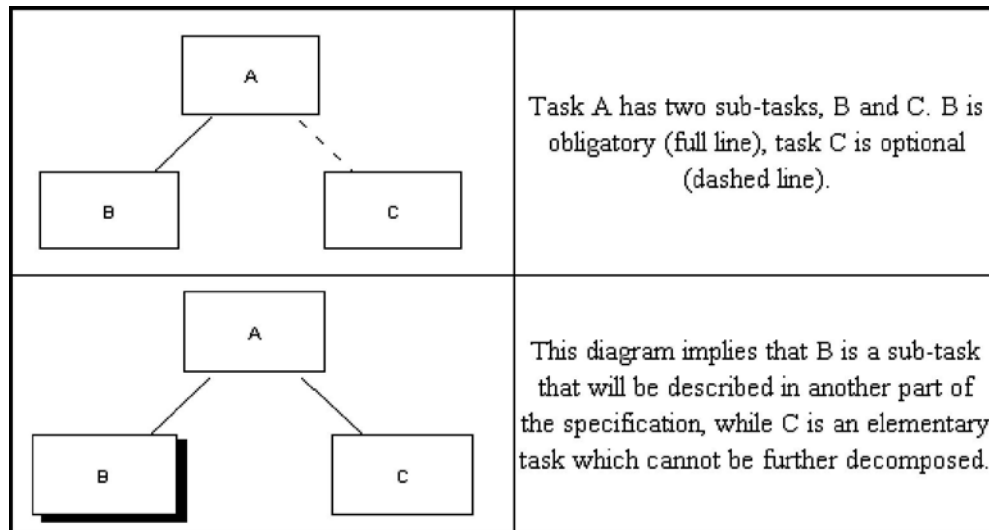


Fig. 2. Representing hierarchical sub-task structure

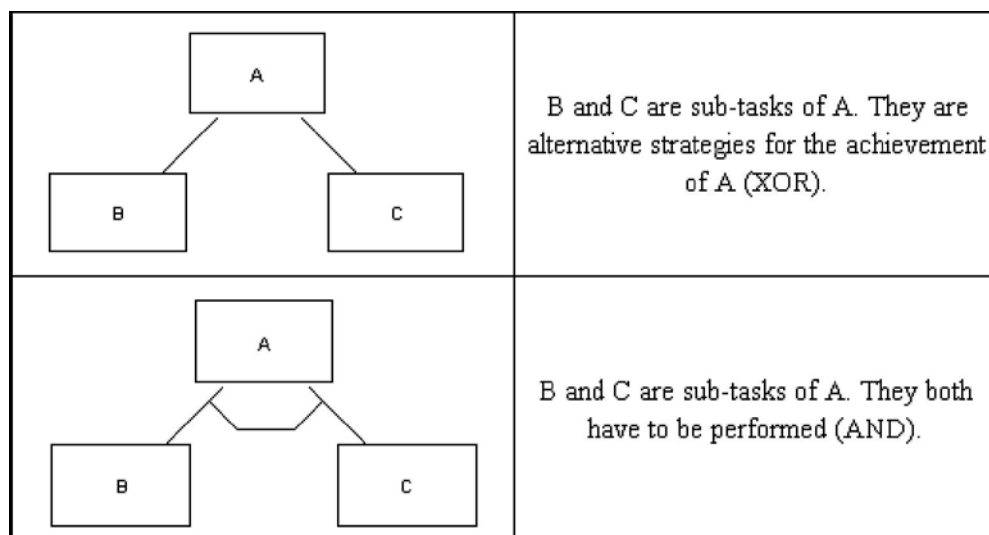


Fig. 3. Representing logical relations between tasks




	<p>Strict precedence: sub-task B precedes sub-task C: B must be performed before C.</p>
	<p>Strict interleaving: sub-tasks B and C are interleaved.</p>
	<p>Indicative interleaving: sub-tasks B and C may be performed one after the other (no strict constraint), but they are usually interleaved.</p>

Fig. 4. Representing temporal relations

3.2. Representing an emergency situation

As claimed in the previous section, task models provide a convenient way of representing the decision-making process of emergency management. Such information may be exploited in a computerised system to guide the user in the actual decision-making process of each emergency management session.

To this end, we need to bridge system functionality and user task model, by specifying for each task the set of results which must be achieved to consider it fully accomplished. In other words, we may replace the abstract notion of task *goal* with a set of concrete objects the value of which must be computed or updated by the system (or even provided by the user) during task execution.

By establishing a propagation mechanism which automatically updates the realisation status of each task – according to the status of computation of each object in the task’s goal - we obtain a way of controlling the flow of activity for each TRACE session, and thus supporting the user in performing the right action at the right moment. Furthermore, this mechanism may also be applied to handle dependencies among computable objects. Actually all environmental emergencies share a number of general tasks which should be accomplished to manage them, such as: characterisation and classification of the specific emergency, assessment of the potential damage, control of the phenomenon, organisation of the damage reclamation.

All tasks rely on information which is dynamic in nature, since it is collected from the external world, or it is obtained through computational models the results of which evolve with the time elapsed, or are in turn depending on other parameters which need to be updated according to a certain rate. For instance, the assessment of a situation relies on heterogeneous information that is either collected on the emergency site, or retrieved from territorial databases. This information may be exploited by simulation models to forecast how the situation will evolve, as well as by heuristic rules providing a synthetic, easy to read characterisation of the situation. In general, each feature of a situation is called in TRACE an *index*. Examples may be: global severity, set of values at risk, spread forecast, suggested intervention strategy, suggested intervention time, suitable operational resources and so forth. Each index has a value the computation of which may depend on the values of other indexes.

An emergency situation may therefore be represented as a network of tasks and situation indexes organised in a network of dependencies, which is ruled by a publish&subscribe mechanism (see (P a g g i o et al. [11]) for more details). This mechanism ensures that the status of each item – either a task or a situation index – is automatically updated during each phase of a session execution, preserving the global network consistency and enabling the user to monitor the current status of operations.

4. Task Editor

Task Editor (TE) is a graphical knowledge acquisition and documentation tool developed to support designers in creating and maintaining task models. The basic principles of TE design are as follows:

- *Supporting conceptual level operation* by focusing the user on the concept of the task model besides its graphical representation. The interaction between the user and the editor must be carried out mainly in terms of task analysis language thus avoiding details irrelevant to the knowledge acquisition process.
- *Building syntactically consistent and complete models* by applying context-sensitive syntactical checks at each step of the process of creation of the task model and by maintaining strict correspondence between its textual and graphical counterparts.
- *Developing special means* making it easier for the user to localise the incomplete fragments of the constructed model.
- *System driven interaction with the user* preventing the user from possible errors and allowing her/him to concentrate on the relevant part of the knowledge to be acquired.
- *Direct manipulation of the graphic structure* supporting graphical, palette-based editing capabilities for direct manipulation of task structures. This is an exploitation of the graphical tree metaphor, and a prerequisite for future extensions of the tool towards generality (configurable palette of task symbols and task connectors).
- *Unified processing of predesign and design task models facilitating operation with the editor.*

4.1. Architecture

The TE architecture consists of four main modules (see Fig. 5). *Data structures* represent the current stage of a constructed domain model. The *Logical Data Structures* (LDS) are related to the object-oriented internal representation of the model in the TA terms while *Graphical Data Structures* (GDS) contain information used for visualising the model (i.e. task symbol co-ordinates etc.). The LDS are maintained by a set of functions forming the *General Functionality* module (GF). The *Human Machine Interface* module (HMI) supports the communication between the user and the current domain model using a “client-server”

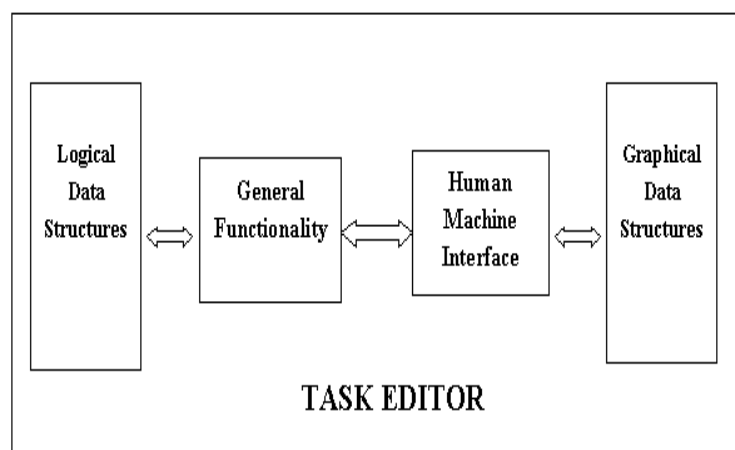


Fig. 5. Task Editor architecture

interaction with GF. It also supports some configurable graphical structures used for module visualisation (graphical symbols used for representing different types of task and relations etc.).

4.2. Logical Data Structures

A domain model is represented as a document containing one pre-design and several design task models. Each design model is generated from the same pre-design model and reflects different aspects related to concrete software implementation. Each model contains three main parts – a set of task pages showing the task diagrams, the set of task model objects the tasks are referred to and the set of task pages currently marked for deletion. The latter denotes those tasks that are not currently included in the domain model but may be used in the future. All task pages are organised into hierarchical tree-like structures via hyper links used for splitting descriptions of complex tasks into several manageable and completed chunks aimed at documentation. Each task page contains a task diagram representing a tree the nodes of which are tasks connected via temporal, logical, task–subtask or pre-/post-requisite task relations. Each task is in turn a complex object containing a number of attributes used for coding various types of declarative task knowledge.

4.3. General Functionality

According to the main purpose of the GF its functions are grouped into the following four clusters:

- *Browsing* functions are intended to access any arbitrary element of the model. Browsing the task diagram and using shortcut keys for navigation allows a context-sensitive access to the task. Direct access to a given task page is provided by browsing a task page structure. In both cases the browsing process is facilitated by task and page completion status reflecting in a graphical form the state of the syntactical completeness of the current task or task page.
- *Create/Edit* functions are intended for maintaining the task model. They allow a direct graphical manipulation on the task structure as well as automatic creating and updating of the task page structure. All Create/Edit operations are automatically checked for consistency and accompanied with automatic updating of the task and task page completion status.
- *Visualisation* functions are intended to maintain the image of the task model on the screen. These functions preserve the task model integrity and provide zooming, grid and alignment facilities.
- *Documentation* functions are intended for storing the model on a “hard medium” (files, paper documents etc.). These functions create new and edit existing documents. They also print the task model as a collection of specially designed task and object description forms, preview task pages and export task specifications to a text file

4.4. Human machine interface

The design of the HMI module which is part of the editor is guided by the requirements of TE target environment – MS Windows 95/NT. The main TE screen is shown in Fig. 6. The workspace is composed of a menubar, a message (status) bar, a graphical palette containing some graphical symbols used in the model and two scrollable panels: the *Structure Panel* and the *Task Diagram Panel*. The Structure Panel provides a synthetic view on the different task pages included in a task model, and typical tree control facilities for browsing and selecting

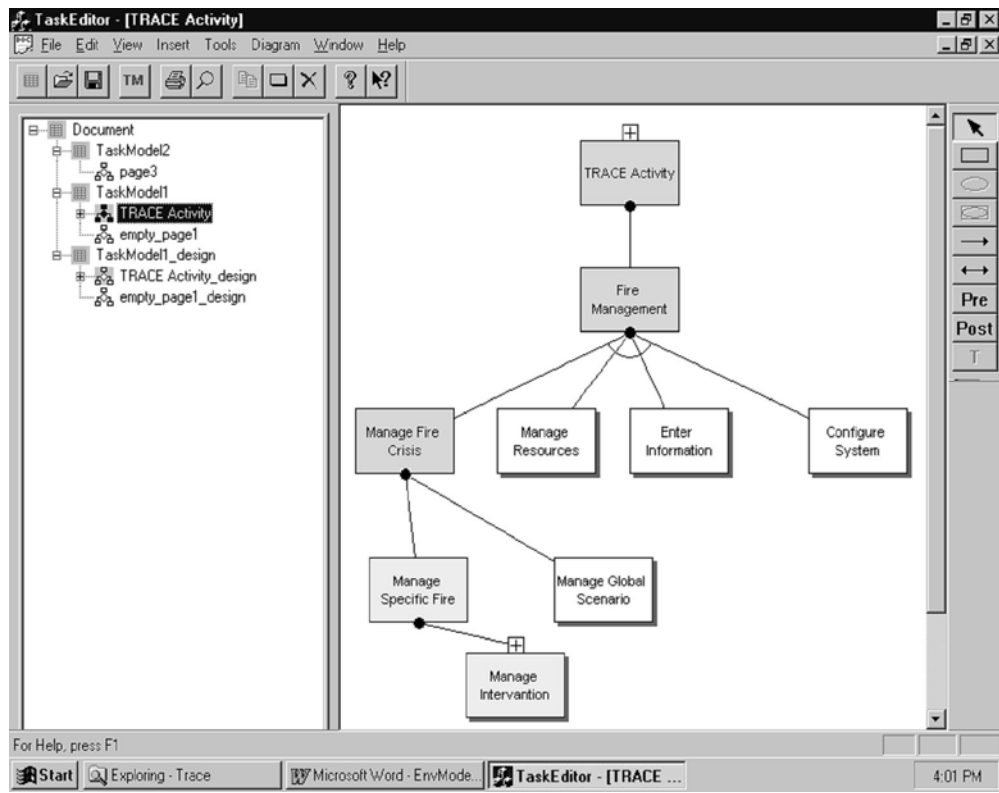


Fig. 6. TE screen layout

task pages. On the diagram panel the active task page is presented. Task diagrams may be edited with direct manipulation from within such panel. The textual task attributes are created/edited by specialised dialogue boxes.

5. Application Scenario

In order to demonstrate the TRACE approach in developing EDSSs, two pilot applications were considered. The first EDSS application, denoted *Demonstrator 1* within the project, is intended to support the decision making in the case of wind-throw management in Romanian forests. The second one, denoted *Demonstrator 2* is intended to support the decision making in the case of fire prevention and fighting in Bulgarian hard-to-reach massifs. The applications address two types of emergencies, frequently encountered in the European forest sector. The wind-throw is a typical event mainly in the Norway spruce. The forest fires are quite frequent in the Mediterranean countries. The approaches in the two cases are different in many respects and only few features are common. While a fire should be controlled in real time and the most effort is done during the fire, in the case of a wind-throw the main logistic problems occur after the catastrophic event has been finished. The differences concerning the type of decisions to be supported in the two cases and the intervention plans defined in both situations were taken into account in elaborating the specifications for the two EDSS applications. In the paper only the development process for *Demonstrator 1* will be considered.

The development of *Demonstrator 1* is based on the guidelines of the methodological framework proposed in the TRACE project. The process of knowledge acquisition and documentation has been following the task analysis methodology and has been implemented by the TRACE Task Editor. This is the first real application of that tool in practice.

Starting from the pre-design task model (see Fig. 7) that reflects the user view concerning the flow of activities in post wind-throw management, we created the design task model (see Figure 8). It was obtained by evolving the predesign task model taking into account what the TRACE system can do (the basic functions of the components of the generic software platform developed within the project) and iterating either of the following:

- add new tasks or subtasks;
- modify existing task (by replacing simple tasks with more complex sub-trees) in order to separate system and user pieces of activity;

After allocating the tasks between the user and the system, the work was concentrated on the system tasks in order to identify the objects involved in the task execution. In addition to the domain task analysis mentioned above, the user data files available for running the application, the available cartographic data and in general all the data the users usually manipulate in case of a wind-throw event were taken into account.

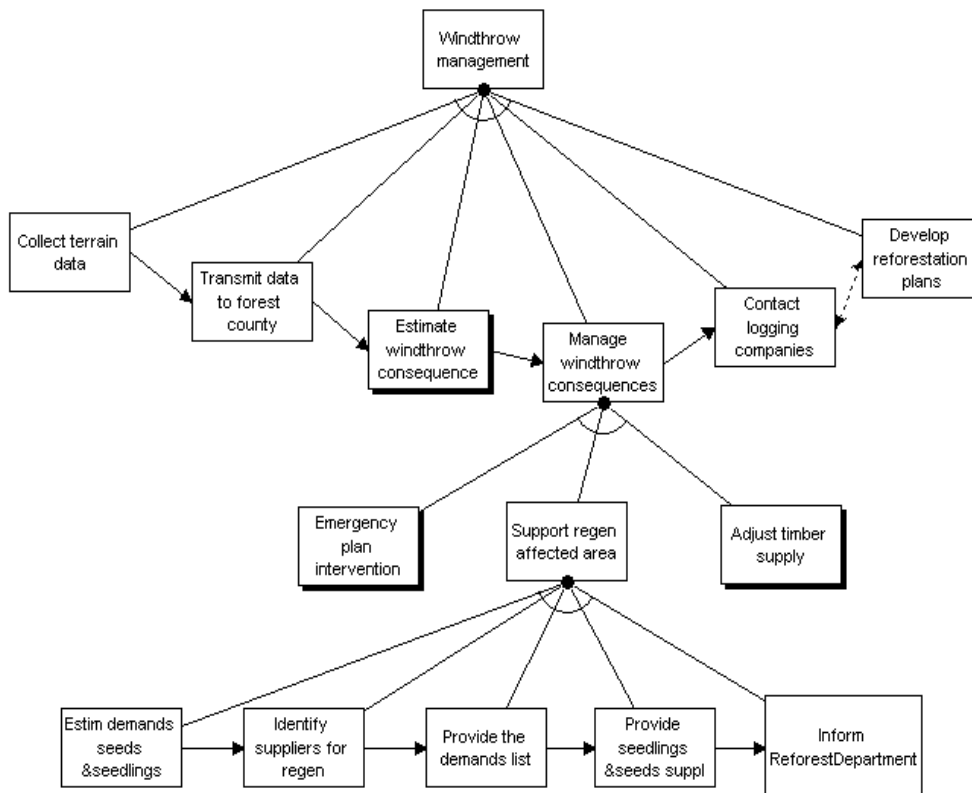


Fig. 7. Wind-throw management pre-design task model (partial view)

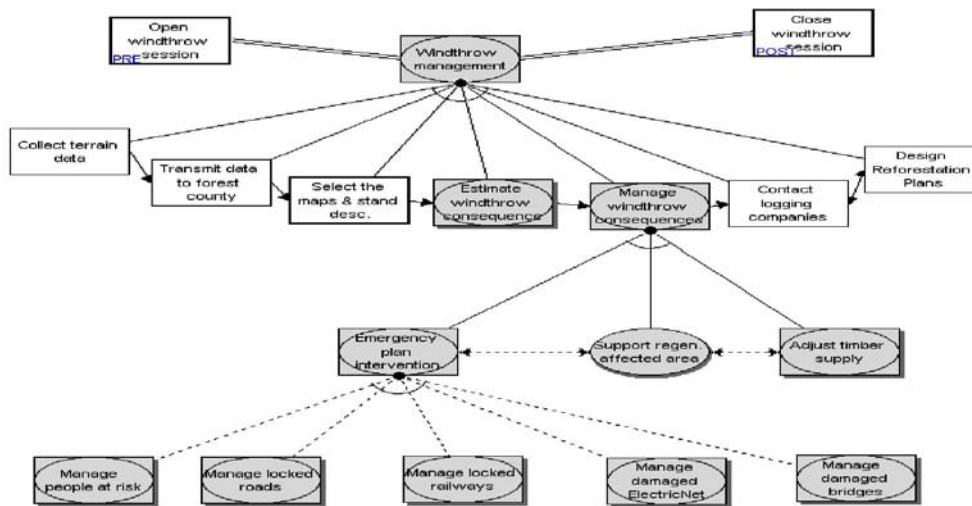


Fig. 8. Wind-throw management design task model (partial view)

Apart from defining the application specific data, the types of the resources and their typical capabilities used in post wind-throw emergency actions have been identified. In order to exploit the framework of defining actual missions in the TRACE system, a set of specific mission stereotypes to be integrated in the mission types library has been defined.

We identified also geo-referenced data specific to the application (data describing forest stands, forest roads, electric networks crossing the forestland, etc.). These geo-referenced data are manipulated using the GIS tool integrated in the TRACE system (Map Objects). Demonstrator 1 mainly aims at supporting the decision-making in:

- Post wind-throw *emergency actions* like estimate wind-throw consequences, rescuing people in danger, opening locked roads, opening locked railways, repairing damaged electric networks, repairing damaged bridges, etc.
- Post wind-throw *medium/long run actions* like estimating the timber volume in the affected area, designing reforestation plans for the affected area, modifying the cutting plans at the level of subordinated production units, etc.

Facing with an emergency situation is a current activity in post-wind-throw management. Developing applications for supporting the decision-maker in this kind of activity was a primary objective of the TRACE technology. The application under development exploits the dependency management mechanism by representing an emergency situation as a network of tasks and situation indexes organised in a network of dependencies. From the implementation point of view, the interface of the application is linked to the dependency management module. This connection means that the interface provides support for mapping user commands onto task requests. The overall dialogue between the user and the system is guided through underlying task structure, in terms of tasks realisation status, filtering of commands which are not enabled at a given stage of a situation handling, suggestions on the following steps to be performed.

The user is supported in rapidly assessing the emergency situation and creating effective intervention plans that exploits at best the available resources. Intervention plans are defined by the user on the basis of plan and mission stereotypes identified in the domain analysis stage. A simple example of mission stereotype with the goal of facing the emergency situation

of locked road by fallen trees is presented below:

mission type name: open_road

a) resource types which might be involved in this kind of mission: **dozer, trailer, tractor, loading_tractor, sawing_machine, jeep, fuel_tank, sleeping_wagon.**

b) capability list includes: volume_to_remove, dozer_transport, trailer_transport, load/unload_logs, cuttings_logs, persons_carried, fuel_tank_transport, fuel_volume, workers_accommodation.

(**dozer** - *volume_to_remove*)

(**trailer** - *dozer_transport*)

(**tractor**- *trailer_transport, fuel_tank_transport*)

(**loading_tractor** - *load/unload logs*)

(**jeep** - *persons_carried*)

(**fuel_tank** - *fuel_volume*)

(**sleeping_wagon** - *workers_accommodation*)

(**sawing_machine** - *cutting_logs*)

In Fig. 9 is presented the example of design task model for the management of this kind of emergency. The creating of the intervention plan is not detailed in the figure, but when creating actual missions for the plan the user may start from the exemplified above mission stereotype.

The library of available mission stereotypes can be easily extended with new ones if the user consider necessary.

In the wind-throw management session the overall allocation process starts from building mobilisation table (MT), and is activated by the user by creating at least one mission, defining some requirements for it and launching the automatic allocation. In more detail, it may be decomposed in the following steps:

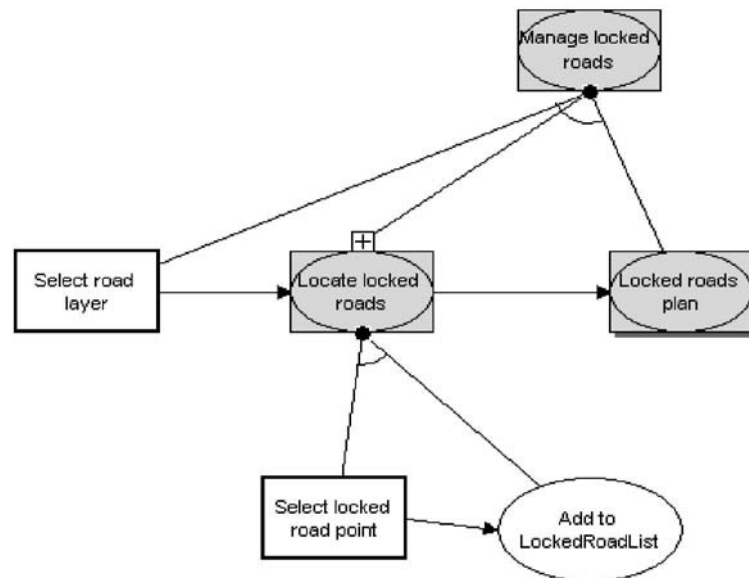


Fig. 9. Manage_locked_roads design task model (partial view)

- a. Build the mobilisation table;
- b. Define a user request for resources
- c. Apply the constraint reasoning techniques for selecting a suitable set of resources to be allocated;
- d. Commit the solution (after the user confirmation).

The first step, automatically performed by the system after the access points in the emergency area have been identified, assures the computation of dynamic data for each available vehicle in the resource bases. These dynamic data contain for each vehicle the shortest paths (or fastest paths, computed taking into account the average speed of every vehicles on different road types) from the owner base to the intervention points and the estimated travelling times (specific preparation time for each vehicle may be added).

In the second step the user defines its request following several stages:

- i) Defining a plan as a set of missions. The missions are defined from scratch or starting from mission stereotypes.
- ii) Declaring the appropriate number of requested persons, resources for each type and the amount required of each capability for all missions.

At the level of interface this kind of declarations represents a simple way for formulating requirement statements which in natural language could be expressed like:

- “I want X amount of personnel”;
- “I want X resources of type Y”;
- “I want enough resources able to perform the X amount of the Y capability”

At the level of actual mission the user must also specify the deadline and the operational point (it must be one of the intervention points processed in the mobilisation table). Optionally the user has the possibility to specify general conditions for selecting the resources that may be used in mission. The conditions are specified using interface elements that enable the user to build and apply atomic filters or more complex logical AND-OR filters exploiting the filter generic classes provided at the level of Resource Management module. Examples of statements which can be managed by this simple formal language are:

- “I want resources being on emergency area before given deadline”;
- “I want resources belonging to certain bases”;
- “I want terrestrial resources from certain bases being on emergency area before given deadline”, etc.

Applying the filters has the effect of removing from MT all those values not meeting the user defined statements.

At the level of plan the user may specify the global allocation criteria defining the allocation strategy. All the missions in a plan share the same global constraints. Appropriate interface elements enable the user to formulate in a simple way user preferences that in natural language could be expressed in statements like:

- “I want to minimise intervention costs”;
- “I want to minimise the number of resources employed”, etc.

The step c, performed by the system, assures the translation of allocation problem into constraint satisfaction problem terms thus making it possible to exploit the capabilities of the constraint management module in solving it. If no solution is found the user may repeat the process from step b, specifying a new, relaxed request.

Usually, for a given set of requirements, many possible solutions can be found. The user has the possibility to browse the solutions list and to select the preferred one. The commitment of the selected solution in the step d must not be done until the resource dispatching task is

accomplished because resources that are marked “available” in the system repository might actually be not available, or not ready to be employed.

The decision-makers are interested in being supported not only in the emergency activities but also in the medium and long-term activities. The main medium/long-term activities are those concerning the regeneration of the affected area and timber supply adjustment.

Reforestation plans must be designed in the early stages of post wind-throw management. The application provides useful data related to the species in the affected area and the genetic resources demand for each affected forest district. At the same time the application provides data concerning the seed and seedlings suppliers and their offer for the species in the affected area. Based on the information provided by the system, the users can decide on the appropriate policy for the regeneration of the affected area.

Adjustment of the timber supply is necessary in order to modify the current cutting plans at the level of production units so that they takes the timber volume in the affected area into account. This modification is necessary in accordance with the *sustained yield principle* (J o b s t l, [4]) which states that annual or decennial yields should be as even as possible. Cutting plans are modified depending on the volume of timber in the affected area and the implemented policy at the forest county level of decision. The EDSS application provides support for an appropriate analysis of the possibilities of propagating the economic consequences of the wind-throw in time or/and in space. Propagating the economic consequences *in time* means adjusting only the cutting budget of the production units belonging to the affected forest districts for one year or more, depending on the size of the wind-throw. The cutting budgets of the neighbouring forest districts not affected by the wind-throw, are not adjusted in this case. *In space* propagation means adjusting the cutting budget of both the affected forest districts and the neighbouring ones.

6. Conclusion

The present paper describes the ongoing INCO-COPERNICUS Project CP-960138 - TRACE. The main objective of the TRACE project is the transfer of the knowledge-based software engineering technology developed within CHARADE both in a new operational context and in another application domain. The results of the project are twofold: on one side, a library of basic, fully programmable EDSS software objects; on the other, a methodological framework for designing and building emergency management applications.

As a methodological framework, TRACE includes guidelines and tools for designing applications through task modelling methods and design patterns. A *task model* is a hierarchical structure representing the flow of activity of the user with the system. In TRACE, it is possible to build the application first by modelling the user of a given domain, then by translating it in a general model of system-supported activities, and finally by associating system functions and objects to each task. The execution of a TRACE-supported session thus closely reflects that observed in current practice, and the user is smoothly guided throughout the entire process by the underlying task model. The TRACE methodology provides tools and guidelines for all phases of the application development: task models are built through the Task Editor; a pattern-based mechanism enables declarative definition of task objects (e.g.: situation indexes) and their computation algorithms. Run-time creation and management of such objects is ensured by specific parts of the library.

The TRACE project is still in progress, nevertheless most parts of the software platform are available and currently being validated. Feedback gathered so far among potential end users is very encouraging, but more significant results are expected when applications will be developed. In particular, the project is presently working at two pilot applications, which aim at demonstrating the features of the system in real contexts. One addresses the management of wildfires in Bulgarian massifs; the other deals with post-wind-throw management in Romanian spruce forests.

References

1. CHARADE Consortium. Final Report, CEC DGIII – ESPRIT, 1996, p. 6095.
2. Denzer, R., D.A. Swayne, G. Schimak. Environmental Software Systems. London, Chapman & Hall, 1997.
3. Guariso, A., H. Werthner. Environmental Decision Support Systems. Chichester, Ellis Horwood Limited, 1989.
4. Jobschl, H. Dynamic translation model. A concept and tool for forestry planning and validation. – In: Proceedings of the IV International Symposium on Operational Research, Lubliana, 1997, 273-278.
5. Johnson, P., H. Johnson. Designers identified requirements for tools to support task analysis. – In: Proceedings of INTERACT'90, 1990, 256-264.
6. Johnson, P., E. Nicolosi. Task-based user interface development tools. – In: Proceedings of INTERACT'90, 1990, 383-387.
7. Kirwan, B., L. K. Ainsworth. A guide to Task Analysis. London, Taylor and Francis, 1992.
8. Marti, P., V. Norman. Bridging software design and usability analysis through task modelling. – In: Human Comfort and Security. K. Varghese and S. Pfliegel eds. Berlin, Springer-Verlag, 1995, 39-50.
9. Marti, P. The interface design of an integrated system for handling environmental emergencies. – In: Proceedings of International Workshop on Cognitive Ergonomics, Padua, Italy, 1995.
10. NOW Consortium. Final Report, CEC DGIII – ESPRIT, 1998, p. 2674.
11. Paggio, R., G. Agre, C. Dichev, G. Umann, T. Rozman, L. Batachia, M. Stocchero. A cost-effective programmable environment for developing environmental decision support systems. Environmental Modelling & Software Journal (forthcoming), 1998.

Приобретение знаний в TRACE

Геннадий Агре, ** Роберто Паджо, Леон Батачия*

**Институт информационных технологий, 1113 София*

***Italsoft - Ingegneria di Sistemi, Roma*

(Резюме)

В работе описан подход к проблеме приобретения и представления знаний в системе TRACE, предназначенной для оказания помощи при планировании действий при природных бедствиях. TRACE представляет собой набор средств и указаний по разработке конкретных приложений на основе методов моделирования задач. Использование редактора знаний системы – Task Editor проиллюстрировано на примере планирования действий, по отстранению последствий ураганов в горных районах Румынии.