# T-REX: a Tool for schedule Representation and EXecution

**A. Cesta, G. Cortellessa, F. Pecora, R. Rasconi and R. Rosato**
ISTC-CNR
Institute for Cognitive Science and Technology
Italian National Research Council
I-00137 Rome, Italy

**Abstract**

The T-REX software is a tool for the representation and execution of scheduling problems. On top of a constrained-based core representation, a schedule execution system guarantees a complete support to monitoring the completion of predefined action sequences (schedules). Based on the representation features of the underlying scheduling technology, the system offers the means to easily synthesize behavioral patterns in terms of activity sequences and to follow their execution in real working environments, to users with no or little scheduling expertise.

## 1 Introduction

This paper describes a tool which integrates the schedule execution monitoring technology described in (Cesta & Rasconi 2003) within the scheduling problem modeling framework presented in (Cesta *et al.* 2005c; 2005b; 2005a). The design of the supervision system we present in this article exploits and extends the scheduling capabilities of an existing software architecture, namely O-OSCAR (Object-Oriented Scheduling ARchitecture) (Cesta *et al.* 2001). The O-OSCAR tool has been enriched with a Schedule Execution Monitoring Module (Cesta & Rasconi 2003), which allows to follow the real time execution of a set of scheduled activities (namely, the output of a previous scheduling process), so as to continuously check the adherence of what is being monitored to a predefined set of constraints and/or preferences.

In the ROBOCARE context, we have employed the scheduling technology in quite an unorthodox manner, as the main focus is given to the *representation* features of the technonology itself, while giving less emphasys to the capabilities of the automatic solver. The main contribution of the present work consists in the realization of a domain and problem modeling layer (see (Cesta *et al.* 2005c) for further details) which, by capturing the real user specific domain knowledge, allows to represent scheduling domains and/or problems through a domain-specific terminology with which the end user is familiar with.



T-REX (Tool for Representation and EXecution) attempts to comply with the need of integrating the above mentioned representation and execution monitoring capabilities within a real applicative context such as the ROBOCARE project.

Automatically following the execution of a prescribed flow of actions to be performed by a patient through the use of state-of-the-art scheduling and sensor-based monitoring technology, implies an amount of specific knowledge which normally goes far beyond the competences of anyone who is not directly involved in the field of information technology. In the ROBOCARE context, the actors who are mostly deputed to providing assistance to elderly and/or impaired people (holding thus the biggest responsibility), are normally the doctors and the patient's family members. Their only concern is the assurance that the assisted person adheres as much as possible to a desired pattern and, dually, the prompt recognition and assessment of any behavior which could endanger the patient's health. It becomes therefore extremely important to provide the caregivers with user-oriented tools able to relieve them from the heavy burden of learning the technicalities which are too far from their area of expertise.

T-REX aims at offering a valid help by building a bridge between the end user and the underlying planning and scheduling technology. The rest of the paper is devoted to the T-REX system description. The following description will be based on the analysis of a reference scheduling problem instance, from the ROBOCARE domain.

## 2 The Representation Tool

The system starts by loading the GUI shown in fig. 1.

The left part of the interface is dynamically created depending on the characteristics of the domain of interest: a list of buttons is presented, each referring to a particular construct. In first approximation, a *construct* is a metaphor which captures a certain piece of domain knowledge and maps this high level concept to a "piece" of a low level scheduling problem description, which is normally beyond the caregiver's competence. The shown constructs are sufficient to describe all the possible behaviors the caregiver is interested in monitor-

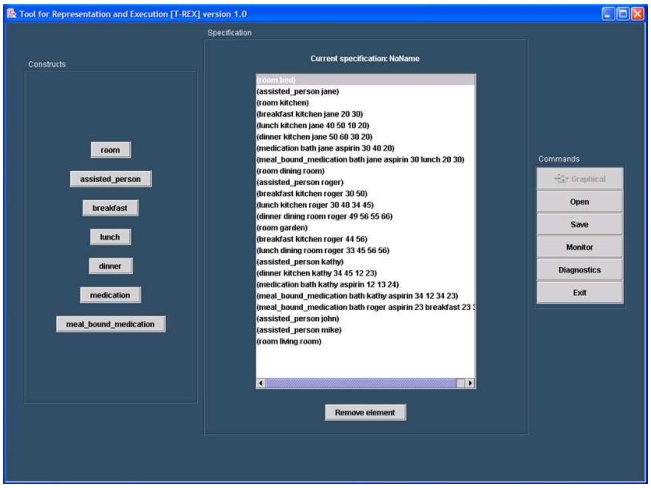


Figure 1: The graphical environment: a problem is loaded.

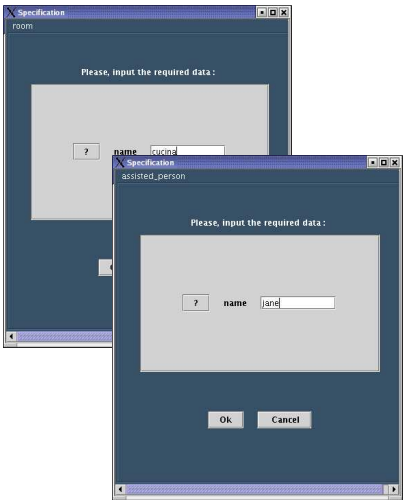


Figure 2: The *room* and *assisted_person* constructs.

ing. All necessary behavioral patterns concerning the domain of interest will therefore be composed through the correct instantiation of the presented contructs.

The domain used in the RoboCare project (shown in fig. 1), entails the presence of the following contructs:

- *room*: used to define each room of the environment (see fig. 2);
- *assisted_person*: used to define the names of the people who will have to be monitored (see fig. 2);
- *breakfast*: used to define the breakfast task. Each task must be executed in a specific room, by a specific person, and is characterized by a start and an end time;
- *lunch*: used to define the lunch task (see fig. 3);
- *dinner*: used to define the dinner task;
- *medication*: used to define a task related to medicine taking;
- *meal_bound_medication*: same as the previous construct, with the difference (see fig. 3) that these tasks are temporally bound to meals (e.g., aspirin cannot be taken before eating);

What should be noted is that by instantiating the available constructs, the caregiver is given the opportunity to easily specify complex temporal relationships among the activities of the plan. For instance, the minimum and maximum time lag between breakfast and lunch or minimum and maximum intervals of execution for some tasks.

By clicking on each button, the proper data input window pops up, and the caregiver is enabled to give all necessary data for proper construct instantiation.

As the constructs are added to the current behavioral pattern, they are shown in the central panel of the interface, giving the user an immediate and clear textual view of the plan which is being synthesized (see the "Specification" panel in fig. 1). Instantiated constructs can also be removed from the plan by clicking the "Remove element" button.

Once the user is satisfied with the plan specification, he or she can decide to save it on disk ("Save" button) or alternatively, decide to directly start its execution monitoring ("Monitor" button).

The execution monitoring phase is devoted to assessing the adherence of the actual execution of the activities which model the pattern prescribed by the caregiver as they are performed by the assisted person; all deviations from the nominal schedule are recorded, and can be analyzed through the interface in caregiver-specific terms — as such, the recorded execution log constitutes diagnostic view of the patient's actual behavior. This information is accessed by the caregiver by selecting the "Diagnostics" command, as a result of which the constructs which are involved in any constraint violation are highlighted (see fig.4).

## 3 The Execution Monitoring Tool

By clicking on the "Monitor" button, the problem is sent to the Scheduling and Execution Monitoring module. Once the problem is loaded (see fig.5), this module is initially responsible for the detection of possible inconsistencies.

In fact, during the preparation of the behavioral patterns on behalf of the caregiver, not all the tasks which will have to be monitored must necessarily follow a prespecified causal order; on the other hand, the caregiver may make some mistakes in the production of the plan and produce a schedule with two or more overlapping tasks. In both cases, as soon as the plan is loaded, possible inconsistencies can be detected. The scheduling capabilities of the system are therefore called into play, and the tool proceeds to find a valid daily schedule which satisfies the constraints modeled by the care-

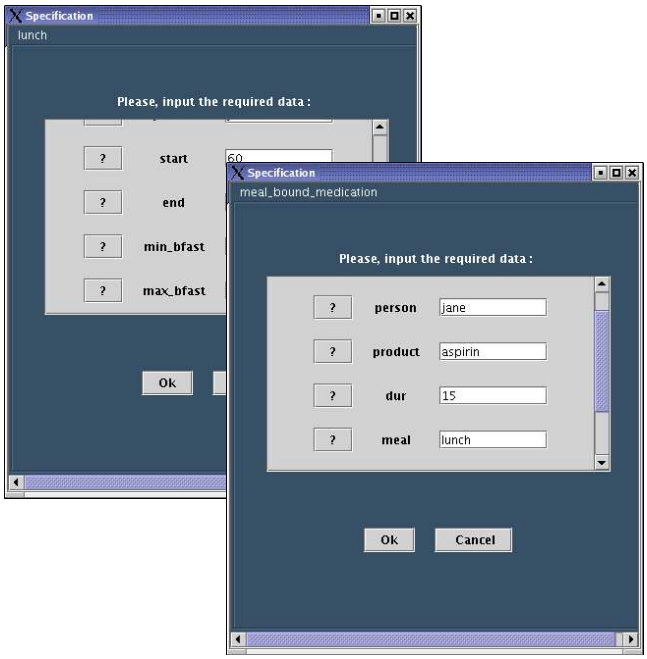


Figure 3: The *lunch* and *meal_bound_medication* constructs.

giver by adequately sequencing the activities: if the resulting schedule is satisfactory for the caregiver, execution monitoring may directly commence, otherwise the preparation of a different pattern will be necessary.

Execution monitoring can be started at any time by the caregiver. Once the monitoring is started, the tool enters a loop where data concerning the assisted person's behavior are periodically received from the environmental sensors and analyzed. The details of the sensor-monitor loop are outside the scope of this paper, and are detailed in (Bahadori *et al.* 2005; Cesta *et al.* 2005d). Given the signals generated by the sensors, the status of each activity part of the behavioral pattern is recognized and the system goes on propagating the instants in which all tasks initiate and terminate, according to the person's actual behavior.

The results of the signal analysis are continously reported on the screen (see fig.6): this dynamic Gantt chart reflects at all times the execution status of the person's tasks.

Let us suppose that the caregiver has synthesized a plan composed of three tasks which should be executed by a patient within a determined monitoring period:

- *Breakfast* (activity 1): this task should not be started before $t = 10$;
- *Lunch* (activity 2): this task should be commenced within 20 and 50 time units after the end of activity 1 (the lunch cannot be too close to breakfast and cannot be eaten too late);
- *Aspirin taking* (activity 3): this task is directly bound to the end of the lunch. Independently from lunch termination, the aspirin should be taken no later than 20 time units after the meal is finished;


Figure 4: A constraint violation is being highlighted.

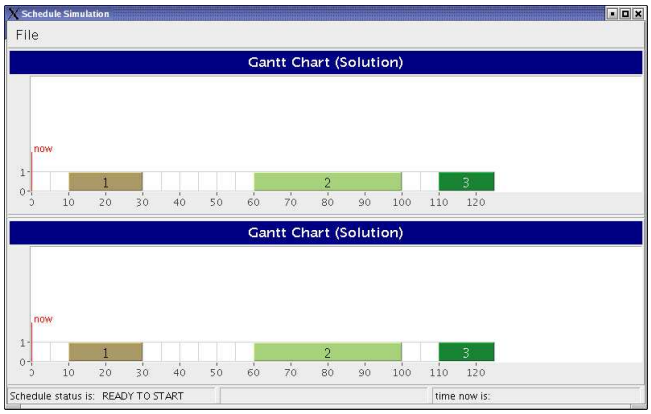


Figure 5: Initial schedule.

Supposing that execution monitoring has started, figure 7 depicts a situation where activity 1 has been commenced by the patient after an initial delay. Activity 1 is currently under execution.

What should be noted, is that due to the current temporal relationships among all the activities in the schedule, the delay on activity 1 is propagated to the activities 2 and 3. The difference in the extent of the delays is due to the fact that, as mentioned above, the system allows for the definition of "elastic" time bounds: in fact, this representation hides a great deal of temporal information, as the schedule activities may be constrained among each other in quite complex ways.

The delay detected on activity 1 entails the violation of a first temporal constraint. In other words, the patient has started the activity *later than originally prescribed* by the caregiver. Not only does the system keep

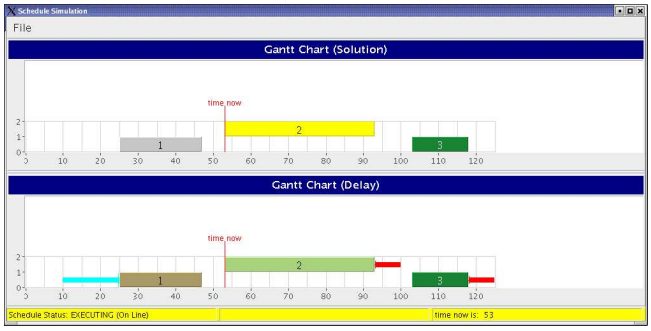


Figure 6: Schedule under execution.

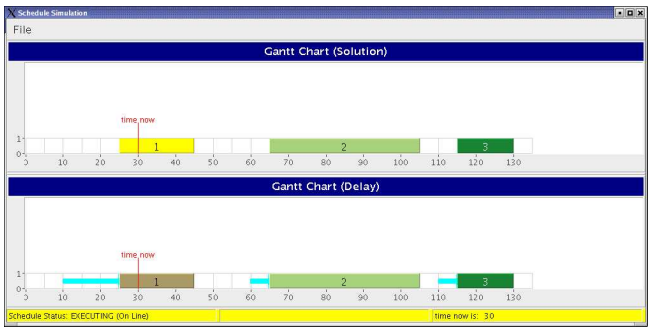


Figure 7: Activity 2 has been delayed.

track of what and when is being executed, it also stores inforamtion related to the constraints which have been violated during the whole monitoring period. This log can be visualized in the GUI and can be automatically processed in order to issue alarms and/or warnings in the environment, or to provide a cognitive support for the assisted person.

Fig. 6 reports the sudden anticipation of activity 2 against the caregiver's prescription which suggested a minimum time lag of 20 time units between activities 1 and 2 (see fig. 5). Again, this implies the violation of another important constraint (recorded in the log) and therefore the possible issuing of another warning. What should be noted at this point is that the detected anticipation is also "inherited" by activity 3, as activities 2 and 3 must not exceed a maximum mutual temporal distance.

The same effect would be appreciated if for any reason the patient decided to finish activity 2 before its nominal duration, as depicted in fig. 8 (note the entailed anticipation of activity 3).

If the execution of activity 3 should be postponed so as to violate the maximum time lag between activities 2 and 3 (see fig. 9), the system would still correctly assess the situation, as previously described, and record this last violation in the diagnostics log file.

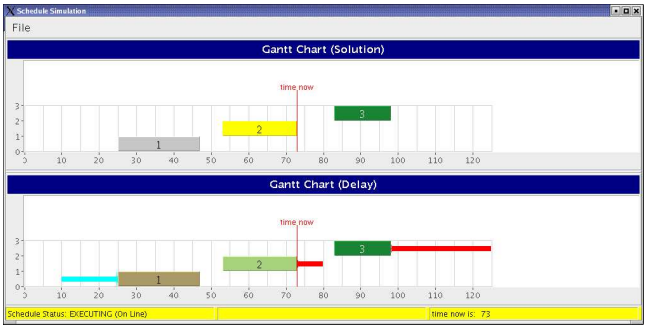


Figure 8: Activity 2 has terminated in advance.

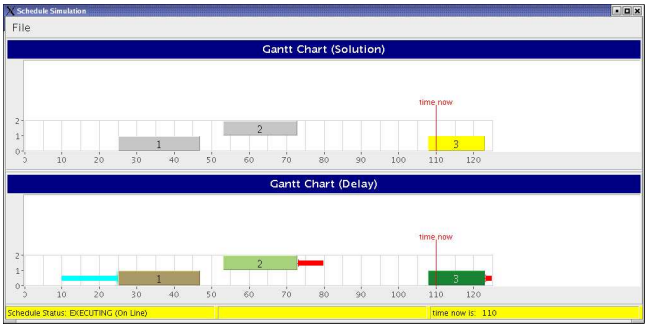


Figure 9: Activity 3 has been delayed.

## 4 Conclusions

In this paper we have briefly described the main functionalities of T-REX, a tool for modeling, solving, and following the execution of user-specified scheduling problems. The framework integrates functionalities drawn from various scheduling components developed over the past years by our group. Specifically, the main components of the system are based on a modeling framework which encapsulates technology specific domain knowledge within domain-specific terms (or constructs) (Cesta *et al.* 2005c; 2005b; 2005a), and on a general purpose schedule monitoring tool developed as a part of the O-OSCAR suite (Cesta & Rasconi 2003).

The functionalities of T-REX have been illustrated within the ROBOCARE Domestic Environemnt scenario, which represents a somewhat unorthodox example of scheduler customization. Indeed, the core technology is mainly employed for its constraint-based representational characteristics: while the problem solving capabilites of T-REX are quite secondary, its flexible disturbance propagation functions are heavily used. In fact, while in a project scheduling or job-shop scenario the expected disturbances at run-time are generally few and far between, in the ROBOCARE application external disturbances are the norm, since the initial schedule represents an expected "nominal" behavioral pattern. Moreover, there is also a strong difference in the ef-

fects of contingencies: while in a more classical domain of application even a small disturbance potentially entails a strong invalidation of the current schedule, the schedules which characterize the RoboCare scenario are generally quite elastic, absorbing most contingencies without the need for re-scheduling.

Overall, we can say that the RoboCare applicative context is interesting for two reasons: first, it has stimulated us to design a modeling and representation framework which can be easily customized for potentially very different domains; second, it allows us to perform intensive tests on functionalities which are less used in other applicative contexts, such as frequent propagation as opposed to re-scheduling.

## References


Bahadori, S.; Cesta, A.; Iocchi, L.; Leone, G.; Nardi, D.; Pecora, F.; Rasconi, R.; and Scozzafava, L. 2005. Towards Ambient Intelligence for the Domestic Care of the Elderly. In Remagnino, P.; Foresti, G.; and Ellis, T., eds., *Ambient Intelligence: A Novel Paradigm.* Springer.

Cesta, A., and Rasconi, R. 2003. Execution Monitoring and Schedule Revision for O-OSCAR: a Preliminary Report. In *Proceedings of the Workshop on Online Constraint Solving at CP-03, Kinsale Co. Cork, September.*

Cesta, A.; Cortellessa, G.; Oddi, A.; Policella, N.; and Susi, A. 2001. A Constraint-Based Architecture for Flexible Support to Activity Scheduling. In *Proceedings of Italian Conference of Artificial Intelligence, AI*IA 01.*

Cesta, A.; Cortellessa, G.; Pecora, F.; and Rasconi, R. 2005a. Exploitation of Scheduling Techniques to Monitor the Execution of Domestic Activities. *Intelligenza Artificiale (Accepted for publication).*

Cesta, A.; Cortellessa, G.; Pecora, F.; and Rasconi, R. 2005b. Mediating the Knowledge of End-Users and Technologists: a Problem in the Development of Scheduling Technology. In *Proceedings of The IASTED International Conference on Artificial Intelligence and Applications (AIA 2005), Innsbruck (Austria), 14–16 February.*

Cesta, A.; Cortellessa, G.; Pecora, F.; and Rasconi, R. 2005c. Monitoring Domestic Activities with Scheduling Techniques. In *Proceedings of the 2nd* RoboCare *Workshop, Rome, Italy.*

Cesta, A.; Farinelli, A.; Iocchi, L.; Leone, G.; Nardi, D.; Pecora, F.; and Rasconi, R. 2005d. Robotically Rich Environments for Supporting Elderly People at Home: The RoboCare Experience. In *Proceedings of the AISB'05 Workshop on Robot Companions, 12-15 April, Hatfield, England.*