Progetto RoboCare*: sistema multi-agenti*
con componenti fisse e robotiche mobili intelligenti

Settore "Piattaforme ITC abilitanti complesse ad oggetti distribuiti"
MIUR legge 449/97 per l'anno 2000

Robotic, Sensory and Problem Solving Ingredients for the Future Home

A. Cesta[2]*, L. Iocchi*[1]*, G.R. Leone*[1]*,*
D. Nardi[1]*, F. Pecora*[2]*, R. Rasconi*[2]

[1]Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
{iocchi,leone,nardi}@dis.uniroma1.it

[2]Istituto di Scienze e Tecnologie della Cognizione
Consiglio Nazionale delle Ricerche
{name.surname}@istc.cnr.it

The RoboCare Technical Reports

RC-TR-0806-5

Robotic, Sensory and Problem Solving Ingredients for the Future Home
A. Cesta, L. Iocchi, G.R. Leone, D. Nardi, F. Pecora, R. Rasconi

Robocare Technical Report N. 5

June 2006

Corresponding Author:
Amedeo Cesta
Istituto di Scienze e Tecnologie della Cognizione - CNR (Italy)
amedeo.cesta@istc.cnr.it

"RoboCare: A Multi-Agent System with Intelligent Fixed and Mobile Robotic Components"
A project funded by the Italian Ministery for Education, University and Research (MIUR) under Law 449/97 (Funds 2000).

http://robocare.istc.cnr.it

ISTC-CNR
Consiglio Nazionale delle Ricerche
Istituto di Scienze e Tecnologie della Cognizione
Via S. Martino della Battaglia, 44 - 00185 ROMA

Robotic, Sensory and Problem-Solving Ingredients for the Future Home

Amedeo Cesta[2], Luca Iocchi[1], G.Riccardo Leone[1 2], Daniele Nardi[1], Federico Pecora[2] and Riccardo Rasconi[2]

[1] Dipartimento di Informatica e Sistemistica
University of Rome "La Sapienza", Italy
<surname>@dis.uniroma1.it

[2] Institute for Cognitive Science and Technology
National Research Council of Italy
<name.surname>@istc.cnr.it

Summary. RoboCare is a three-year Italian research project aimed at assessing the extent to which different state-of-the-art technologies can benefit the creation of an assistive environment for elder care. In its final year, the project focused on producing a demonstration exhibiting an integration of robotic, sensory and problem-solving software agents. This article describes the RoboCare Domestic Environment, an experimental three-room flat in which a number of heterogeneous robotic, domotic, and intelligent software agents provide domestic cognitive support services for elderly people. The RDE is a deployed multi-agent system in which agents coordinate their behavior to create user services such as non-intrusive monitoring of daily activities and activity management assistance. This article provides a summary of the system's key features, focusing on the integrated prototypical environment which was deployed in the RoboCare lab in Rome and exhibited at the RoboCup 2006 competition.

1 Introduction

RoboCare is a three year research project[3] which aims at developing multi agent systems for the care of the aging population. The principal aim of RoboCare is to assess the extent to which different state-of-the-art technologies can benefit the creation of an assistive environment for elder care, one of its main driving forces being the increasing abundance of "intelligent" domestic devices and affordable pervasive computing technology. It is with this aim that the final year of the project focused on producing a demonstration exhibiting

[3] This research is sponsored by MIUR (Italian Ministry of Education, University and Research) under project RoboCare (A Multi-Agent System with Intelligent Fixed and Mobile Robotic Components), L. 449/97.

an integration of robotic, sensory and problem-solving software agents. To this end, an experimental setup which recreates a three-room flat was set up at the ISTC-CNR in Rome, named The RoboCare Domestic Environment (RDE). The RDE is intended as a testbed environment in which to test the ability of heterogeneous robotic, domotic, and intelligent software agents to provide cognitive support services for elderly people at home. Specifically, the RDE is a deployed multi-agent system in which agents coordinate their behavior to create user services such as non-intrusive monitoring of daily activities and activity management assistance.

A key feature of the RDE is Lucia, a context-aware domestic robot developed by the RoboCare team at the ISTC-CNR[4]. The robot is aimed at demonstrating the feasibility of an embodied interface between the assisted elder and the smart home. Thus, the RDE can be viewed as a "robotically rich" environment composed of sensors and software agents whose overall purpose is to:

- predict/prevent possibly hazardous behavior;
- monitor the adherence to behavioral constraints defined by a caregiver;
- provide basic services for user interaction.

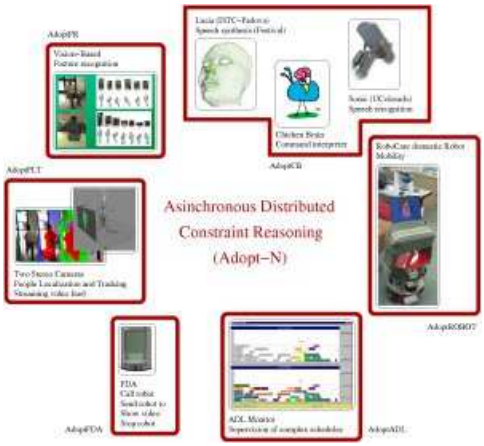

Fig. 1. Overall view of the RoboCare Domestic Environment (RDE).

The system was partially recreated in the RoboCup@Home domestic environment during the RoboCup 2006 competition in Bremen[5] where it was awarded third prize.

1.1 Components of the Multi-Agent System

The RDE is equipped with the following agents, which provide services of various nature:

- Two fixed stereo cameras providing a People Localization and Tracking (PLT) service, and a Posture Recognition (PR) service.

[4] The development team is the result of a combined development effort stemming from two partners of the RoboCare project, namely the Planning and Scheduling Team at ISTC-CNR and SPQR at the University of Rome "La Sapienza".

[5] Competition homepage: `http://www.ai.rug.nl/robocupathome/`.

- A domestic service robot which is capable of navigating in the environment, processing simple commands through an on-board speech recognition system, as well as speaking to the assisted person through a voice synthesizer. Synthesized speech is verbalized through a 3D representation of a woman's face, named Lucia.
- An ADL (Activities of Daily Living) monitor, a scheduling and execution monitoring system which is responsible for monitoring the assisted person's daily activities and assessing the adherence to behavioral constraints defined by a caregiver.
- One personal data assistant (PDA) on which a very simple four-button interface is deployed. The interface allows to (1) summon the robot, (2) send the robot to a specific location, (3) relay streaming video form the robot to the PDA, and (4) stop the robot.

Lucia the robotic mediator was built to explore the added value of an embodied companion in an intelligent home. Its mobility also provides the basis for developing a number of added-value services which require physical presence. Overall, Lucia is composed of two agents: a mobility subsystem and a human-robot interface. These components are described in section 2.

A Stereo-vision based People Localization and Tracking service (PLT) provides the means to locate the assisted person. This environmental sensor was deployed in Bremen in the form of an "intelligent coat-hanger", demonstrating easy setup and general applicability of vision-based systems for in-door applications. The system is scalable as multiple cameras can be used to improve area coverage and precision. In addition, vision-based Posture Recognition (PR) can be cascaded to the PLT computation in order to provide further information on what the assisted person is doing. The sensory subsystem is described in section 3.

Continuous feedback from the sensors allows to build a symbolic representation of the state of the environment and of the assisted elder. This information is employed by a CSP-based schedule execution monitoring tool (T-Rex [1, 2, 3]) to follow the occurrence of Activities of Daily Living (ADLs). Aspects of daily life to be monitored are specified by a caregiver in the form of complex temporal constraints among activities. Constraint violations lead to system intervention (e.g., Lucia suggests "how about having lunch?", or warns "don't take your medication on an empty stomach!"). The details of the ADL monitor are shown in section 4.

Overall, the RDE is a collection of service-providing components of various nature. Sensors contribute to building a symbolic representation of the state of the environment and of the assisted person. Based on this information, automated reasoning agents infer actions to be performed in the environment, principally through the robotic mediator. Both enactment and sensing requires the synergistic operation of multiple agents, such as robot mobility, speech synthesis and recognition, and so on. For this reason, multi-agent coordination is an important aspect of the RDE scenario. Section 5 is dedicated to the

description of the coordination mechanism, which occurs in the RDE's current configuration by means of ADOPT-N [4], a distributed constraint reasoning algorithm.

2 Lucia the Robotic Mediator

The mobile robotic platform, called Lucia, was built to explore the added value of an "embodied" companion in an intelligent home. The robot's mobility also provides the basis for developing added-value services which require physical presence.

Fig. 2. Lucia the Robotic Mediator.

The robot hardware has been built on top of a Pioneer platform, by adding additional sensors: a laser range finder, a stereo camera, an omni-directional camera, as well as computational resources: 2 laptops, one for sensor processing and navigation, one for human-robot interaction. In the following sections we first describe the basic functionalities of the robot, specifically navigation, path planning, mapping ang localization; then the component for human-robot interaction, i.e. speech recognition and speech synthesis.

Navigation. Navigation of the robot is based on two components that are integrated togheter: a topological path planning and a reactive obstacle avoidance module.

Mapping and localization. In our project we assume that the robot acting in the domestic environment can map the environment before operating in it and that only minor changes in the environment occur during operations (e.g., pieces of furniture can be moved). Therefore, our approach to mapping and localization is divided in two phases: in the first one, the robot is guided in the environment in order to build a map (this is usually done only once when the robot should operate in a new environment); in the second phase, the robot localizes itself using the previously acquired map. Mapping and localization services make use of a laser range finder that provides for accurate range measures. Mapping is implemented through a scan matching approach (since domestic environments are usually small enough not to require more sophisticated SLAM approaches) and localization is obtained through a particle filter based approach.

Speech recognition and synthesis. Lucia is also endowed with verbal user interaction skills: speech recognition is achieved with the Sonic speech recognition system (University of Colorado)[6], while speech synthesis occurs through

[6] See `cslr.colorado.edu/beginweb/speech_recognition/sonic.html` for details.

the Lucia talking head [5] developed at ISTC-CNR-Padua[7] (Lucia the robotic mediator takes its name from the talking head).

3 Environmental Sensor for People Tracking

A major objective of the RoboCare project was the integration of different intelligent components, that are deployed not only on board of a mobile robot, but also as "intelligent" sensors in the environment. In particular, we have developed a People Localization and Tracking service[8] (PLT) based on a stereo vision sensor, which provides the means to locate the assisted person and other people in the environment.

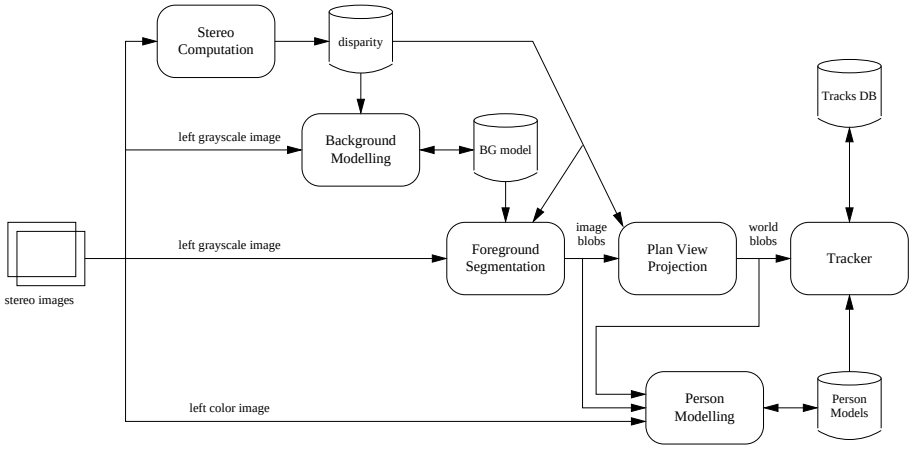

Fig. 3. PLT System Architecture

The general architecture of the PLT System is depicted in Figure 3. The input is a stream of stereo images captured by a calibrated stereo-vision device and processed by the *Stereo Computation* module, extracting disparity and 3-D information about the scene (a description of the stereo vision system can be found in [6]).

Subsequently, the *Background Modelling* maintains an updated model of the background, composed by three components: intensities, disparities, and edges. Then *Foreground Segmentation* extracts foreground pixels and *image blobs* from the current image, by a type of background subtraction that combines intensity and disparity information. Points in the foreground are then projectd in the plan-view within the *Plan View Projection* module, by using

[7] See also `www.pd.istc.cnr.it/LUCIA/` for details.
[8] www.dis.uniroma1.it/∼iocchi/PLT

3-D information computed by the stereo algorithm. This module computes *world blobs* that identify moving objects in the environment. World blobs are also used to refine image segmentation detecting situations of partial occlusions between people.

Once we have a set of segmented figures for each person, the *People Modelling* module creates and maintain *appearance models* for the people being tracked; these information are then passed to the *Tracker* module, which maintains and tracks a set of *tracked objects*, associated with *world blobs* extracted by previous modules. Tracking is performed by using a Kalman Filter on a state that is formed by the location and velocity of the person in the world and by his/her appearance model. The state for each person is updated by using a constant velocity model for person location and by also considering a similarity measure between appearance models and current segmented figures of each tracked person. The tracker mechanism is able to reliably track multiple people in situations of partial mutual occlusions.

3.1 Image Segmentation

When using a static camera for object detection and tracking, background model maintenance and consequent background subtraction is a common technique for image segmentation and for identifying foreground objects. In order to account for variations in illuminations, shadows, reflections, and background changes, it is useful to integrate information about intensity and range and to dynamically update the background model. Moreover, such an update must be different in the parts of the image where there are moving objects [7, 8, 9, 10].

The implementation of the image segmentation module thus requires the computation of a background model and a subsequent foreground extraction procedure.

Dynamic Background Modelling

In our work we maintain a background model including information of intensity, disparity, and edges, as a unimodal probability distribution represented with a single Gaussian. Although more sophisticated representations can be used (e.g. Gaussian mixture [7, 9, 10]), we decided to use a simple model for efficiency reasons. We also decided not to use color information in the model, since intensity and range usually provide a good segmentation, while reducing computational time.

The model of the background is represented at every time t and for every pixel i by a vector $X_{t,i}$, including information about intensity, disparity, and edges computed with a Sobel operator. In order to take into account the uncertainty in these measures, we use a Gaussian distribution over $X_{t,i}$, denoted by mean $\mu_{X_{t,i}}$ and variance $\sigma^2_{X_{t,i}}$. Moreover, we assume the values for intensity, disparity, and edges to be independent from each other.

This model is dynamically updated at every cycle (i.e., for each new stereo image every 100 ms) and is controlled by a *learning factor* $\alpha_{t,i}$ that changes over time t and is different for every pixel i.

$$\mu_{X_{t,i}} = (1 - \alpha_{t,i})\, \mu_{X_{t-1,i}} + \alpha_{t,i}\, X_{t,i}$$
$$\sigma^2_{X_{t,i}} = (1 - \alpha_{t,i})\, \sigma^2_{X_{t-1,i}} + \alpha_{t,i}\, (X_{t,i} - \mu_{X_{t-1,i}})^2$$

The learning factor $\alpha_{t,i}$ is set to a higher value (e.g. 0.25) in the first few frames (e.g. 5 seconds) after the application is started, in order to quickly acquire a model of the background. In this phase we assume the scene contains only background objects. After this training phase $\alpha_{t,i}$ is set to a lower nominal value (e.g. 0.1) and modified depending on the status of pixel i as explained below. Notice that the training phase can be completely removed and the system is able to build a background model even in presence of foreground moving objects since the beginning of the application run. Of course it will require a longer time to compute this model.

After the training phase the learning factor $\alpha_{t,i}$ is increased (e.g. 0.15) when there are no moving objects in the scene (speeding up model updating), and is decreased (or set to zero) in the regions of the image where there are moving objects. In this way we are able to quickly update the background model in those parts of the image that contain stationary objects and avoid including people (and, in general, foreground objects) in the background.

In order to determine regions of the images in which background should not be updated, the work in [10] proposes to compute *activities* of pixels based on intensity difference with respect to the previous frame. In our work, instead, we have computed *activities* of pixels as their difference between the edges in the current image and the background edge model. The motivation behind this choice is that people produce variations in their edges over time even if they are standing still (due to breathing, small variations of pose, etc.), while static objects, such as chairs and tables, do not. However, note that edge variations correctly determine only the contour of a person or moving foreground object, and not all the pixels inside this contour; therefore, if we consider as *active* only those pixels that have high edge variation, we may not be able to correctly identify the internal pixels of a person. For example, if a person with uniform color clothes is standing still in a scene, there is high probability that the internal pixels of his/her body have constant intensity over time, and a method for background update based only on intensity differences (e.g., [10]) will eventually integrate these internal pixels into the background.

To overcome this problem we have implemented a procedure that computes *activities* of pixels included in a contour with high edge variation. This computation is based on first determining *horizontal and vertical activities* $H_t(u)$ and $V_t(u)$, as the sum over the pixels (u, v) in the image, of the variation between current edge E and edge component of the background model μ_E, for each row/column of the image.

$$H_t(v) = \sum_u |E_{t,(u,v)} - \mu_{E,t,(u,v)}| \qquad V_t(u) = \sum_v |E_{t,(u,v)} - \mu_{E,t,(u,v)}|$$

Then, these values are combined in order to assign higher activity values to those pixels that belong to both a column and a row with high horizontal and vertical activity.

$$A_t(u,v) = (1-\lambda)\, A_{t-1}(u,v) + \lambda\, H_t(v) V_t(u)$$

In this way, the pixels inside a contour determined by edge variations will be assigned a high activity level. Note also that, since the term $H_t(v)V_t(u)$ takes into account internal pixels for people with uniformly colored clothes, the learning factor λ can be set to a high value to quickly respond to changes. The value $A_t(u,v)$ is then used for determining the learning factor of the background model: the higher the activity $A_t(u,v)$ at each pixel $i = (u,v)$ the lower the learning factor $\alpha_{t,i}$.

Foreground segmentation

Foreground segmentation is then performed by background subtraction from the current intensity and disparity images. By taking into account both intensity and disparity information, we are able to correctly deal with shadows, detected as intensity changes, but not disparity changes, and foreground objects that have the same color as the background, but different disparities. Therefore, by combining intensity and disparity information in this way, we are able to avoid false positives due to shadows, and false negatives due to similar colors, which typically affect systems based only on intensity background modeling.

The final steps of the foreground segmentation module are to compute connected components (i.e. *image blobs*) and characterize the foreground objects in the image space. These objects are then passed to the Plan View Segmentation module.

3.2 Plan View Segmentation

In many applications it is important to know the 3-D world locations of the tracked objects. We do this by employing a *plan view* [11]. This representation also makes it easier to detect partial occlusions between people.

Our approach projects all foreground points into the plan view reference system, by using the stereo calibration information to map disparities into the sensor's 3-D coordinate system and then the external calibration information to map these points from the sensor's 3-D coordinate system to the world's 3-D coordinate system.

For plan view segmentation, we compute a *height map*, that is a discrete map relative to the ground plane in the scene, where each cell of the height map is filled with the maximum height of all the 3-D points whose projection

lies in that cell, in such a way that higher objects (e.g., people) will have a high score. The *height map* is smoothed with a Gaussian filter to remove noise, and then it is searched to detect connected components that we call *world blobs*.

Since we are interested in person detection, *world blobs* are filtered on the basis of their size in the plan view and their height, thus removing blobs with sizes and heights inconsistent with people. Notice that even in situations of partial occlusions, the world blob of the occluded person has generally a size and a height similar to the case in which this person is not occluded. This is due to the position of the camera and to the use of plan view information. In fact, the upper part of a person is typically visible even when she/he is occluded, and our method for computing the *height map* is not sensible to the lack of information in the lower parts of a person. Therefore, the tuning of the filter is not very critical and we prefer to have a filter that is not very selective on such world blobs, since possible false detections are then solved by the tracked module (for example, a noise producing an incorrect world blob isolated in time will be discarded by the tracker since the observation is not confirmed over time). However, it is necessary to observe that the method proposed in this paper does not use any technique for people recognition, and therefore if a moving object with similar dimensions of a person (e.g. a high mobile robot) moves in the environment, then this object will be tracked by the system.

Fig. 4. The different phases of the image segmentation

3.3 Tracking

We have chosen a probabilistic approach for tracking multiple people, based on a mono-modal probability distribution, like in [12, 13, 14]. Tracking is performed by maintaining a set of *tracked objects* x_t, updated with the measurements of *world blobs* and *appearance models* z_t, extracted in the previous segmentation phase. As a difference with previous approaches, the state of each tracked object has two components: the spatial parameters, and the appearance.

3.4 People Modelling

In order to track people over time in the presence of occlusions, or when they leave and re-enter the scene, it is necessary to have a model of the tracked people. Several models for people tracking have been developed (see for example [15, 16, 17, 18]), but color histograms and color templates (as presented in [16]) are not sufficient for capturing complete appearance models, because they do not take into account the actual position of the colors on the person.

Following [15, 17], we have defined *temporal color-based appearance models* of a fixed resolution, represented as a set of unimodal probability distributions in the RGB space (i.e. 3-D Gaussian), for each pixel of the model. Computation of such models is performed by first scaling the portion of the image characterized by a *person blob* to a fixed resolution and then updating the probability distribution for each pixel in the model.

Appearance models computed at this stage are used during the tracking step for improving reliability of data association process (see Section 3.3).

3.5 Probabilistic tracking and data association

Tracking is performed with a probabilistic formulation. The uncertainty about the i^{th} tracked object at the time t is represented by a multi-variate normal distribution, $N(\mu_{i,t}, \sigma_{i,t})$, and a weighting factor $w_{i,t}$.

The probability distribution $p(x_t)$ for the tracked people is represented as a collection of Gaussians and weights. Each Gaussian models the information about a single person, and the weight models the confidence about the person estimate. Therefore $p(x_t)$ is represented as a set $P_t = \{N(\mu_{i,t}, \sigma_{i,t}), w_{i,t} \,|\, i = 1..n\}$ where $N(\mu_{i,t}, \sigma_{i,t})$ is a Gaussian in a multi-dimensional space representing the i^{th} person that is tracked at time t and $w_{i,t}$ a weighting factor. Similarly, observations in z_t are represented as a set of Gaussian distributions $Z_t = \{N(\mu'_{j,t}, \sigma'_{j,t}) \,|\, j = 1..m\}$ denoting position and appearance information of detected people in the current frame.

The probability distribution $p(x_t|z_t)$ is computed by using a set of Kalman Filters. The system model used for predicting the people position is the constant velocity model, while their appearance is updated with a constant model.

This model is adequate for many normal situations in which people walk in an environment. It provides a clean way to smooth the trajectories and to hold onto a person that is partially occluded for a few frames. However, multi-modal representations (e.g. [19]) should be used in more complex situations.

Data association is an important issue to deal with. In general, at every step, the tracker must make an association between m observations (*world blobs*) and n people (*tracked objects*).

Association is solved by computing the Mahalanobis distance $d_{i,j}$ between the i^{th} tracked object $N(\mu_{i,t|t-1}, \sigma_{i,t|t-1})$ and the j^{th} observation $N(\mu'_{j,t}, \sigma'_{j,t})$. Here the Gaussian $N(\mu_{i,t|t-1}, \sigma_{i,t|t-1})$ is the predicted estimate of the i^{th} person.

An association between the predicted state of the system $P_{t|t-1}$ and the current observations Z_t is denoted with a function f, that associates each tracked person i to an observation j, with $i = 1..n$, $j = 1..m$, and $f(i) \neq f(j)$, $\forall i \neq j$. The special value $\perp$ is used for denoting that the person is not associated to any observation (i.e. $f(i) = \perp$). Let $\mathcal{F}$ be the set of all possible associations of the current tracked people with current observations. Data association is then computed by solving the following minimization problem

$$\underset{f \in \mathcal{F}}{\operatorname{argmin}} \sum_i d_{i,f(i)}$$

where a fixed maximum value is used for $d_{i,f(i)}$ when $f(i) = \perp$.

Although this is a combinatorial problem, the size of the sets P_t and Z_t on which this is applied are very limited (not greater than 4), so $|\mathcal{F}|$ is small and this problem can be effectively solved.

The association f^*, that is the solution of this problem, is chosen and used for updating $p(x_t)$, i.e. for computing the new status of the system P_t. During the update step the weights $w_{i,t}$ are computed from $w_{i,t-1}$ and $d_{i,f(i)}$, and if a weight goes below a given threshold, the person is considered *lost*. Moreover, for observations in Z_t that are not associated to any person by f^* a *new* Gaussian in entered in P_t.

The main difference with previous approaches [12, 13, 14] is that we integrate both plan-view and appearance information in the status of the system, and by solving the above optimization problem we find the best matching between observations and tracker status by considering in an integrated way the information about the position of the people in the environment and their appearance.

Finally, in order to increase robustness of the system to tracking errors, a finite state machine is associated to any person being tracked. Each track can be in one of these status: *new*, *candidate*, *tracked*, *lost*, *terminated*. The transition from one status to another depends on the current association f^* as well as on the track history. For example, after a *new* observation of a person, the corresponding track becomes a *candidate*, if it is confirmed for a certain number of frames it is moved to the *tracked* status; then if the track

is not observed it gets the *lost* status that is maintained for some frames. In case the track is observed again it goes back to the *tracked* status, thus correctly dealing with partial occlusions (e.g., one person in front of another) and bridging short-term breaks in a person's path (i.e. short-term re-acquisition). Otherwise, after some time, the track is declared *terminated*.

4 Monitoring Activities of Daily Living

This section focuses on the Execution Monitoring System, or Activity of Daily Living (ADL) monitor. The ability to detect and follow the actions of the assisted person in the environment is one of the central features of the system. The main goal is infact to cognitively and phisically support the assisted person while continously guaranteeing his/her independence, well-being and safety. The desired behaviour the assisted person should adhere to in order to achieve the previous goal, is initially decided by a caregiver (a doctor, or a family member): the tool we devised encourages easy interaction even for people who are not acquainted with the technology underlying the system[9], and allows to specify the desired behavior in terms of activities which are particularly significant for the person's health and/or safety. Of course, not all the daily activities are supposed to be monitored; furthermore, the system should allow for a configurable degree of tolerance with respect to the action timings. Timing's strictness of the schedule will be decided by the caregiver and will depend on the criticality of the activities at hand.

Activity recognition and management plays a significant role in advice and warning synthesis, as it will be shown shortly. The desired behavior is initially synthesized in terms of a set of activities to be monitored (schedule), bound one another through complex temporal relationships. These temporal links are of great importance: infact, not only does the schedule need to be constantly monitored in order to know *which* activities are indeed being executed; also, the time at which the activities are performed is essential, as delays or anticipations on temporally related tasks might trigger some initiative on behalf of the monitoring system. Through temporal constraint analysis, the ADL monitor decides which pieces of information to store and make available to the other agents, in order to ensure a correct global reaction. Some intervention might even be directly triggered by the ADL monitor analysis itself in a more reactive fashion, depending on the gravity of the occurred circumstance. In general, the system is able to assess the situation by querying all available agents, which are designed to act independently and asynchronously.

[9] The description of the modeling tool is outside the scope of this paper — see [1] for further details.

4.1 The Schedule Representation

The scheduling technology underlying the whole system is based on Constraint Satisfaction Problem (CSP) solving techniques. More specifically, the baseline schedule defined by the caregiver is represented in a temporal CSP, usually called Temporal Constraint Network (TCN) [20]. The variables in a temporal CSP represent the time points, which can be constrained one another by binding the distance between any two variables. Every activity in the schedule is associated with two time points (the start and the end time); by imposing distance constraints among the time points in the TCN, it is possible to define complex temporal relations among the activities, task durations as well as general separation constraints. TCN's consistency on insertion of new time points and/or new constraints among existing time points is checked through proper propagation algorithms ([20]).

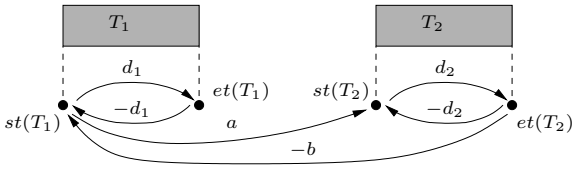

Fig. 5. A simple schedule, represented as a temporal CSP: notice the temporal constraints insisting on the time points.

Figure 5 depicts a simple but explanatory example: the shown schedule is composed of two activities T_1 and T_2. Each activity is characterized by a start point and an end point: activity T_1 has a duration equal to d_1 while T_2 has a duration equal to d_2 temporal units; the activities of the schedule are supposed to be bound by specific temporal constraints imposed by the caregiver: T_2 should start at least a time units after activity T_1 has started, and T_2 must end within b time units after the beginning of T_1.

Under the hypothesis that the schedule represents the activities that have to be monitored, each imposed constraint helps to specify the desired behavior we would like the assisted person to adhere to. For instance, T_1 might represent the activity of having breakfast and T_2 the activity of taking a medicine: in this case, the constraints would model the circumstance that the medicines should not be taken neither too soon nor too late after eating; the values associated with the temporal constraints quantitatively specify the extent of such bounds. According to this representation it is possible to describe behavioral patterns which can be very complex, either in terms of the number of activities involved, and in terms of temporal constraints which may insist among them.

Note that this modeling paradigm implicitly allows for temporal flexibility in the synthesis of the desired behavioral pattern: in fact, the possibility to

introduce miminum and maximum temporal constraints permits to specify temporal slacks in order to allow for some tolerance before a constraint is deemed violated. It is straightforward how this is the only viable solution in the context of execution monitoring of human behaviors, as it avoids to put the assisted person (and the caregiver!) against unacceptably strict, and thus unmanageable, action sequences.

4.2 Execution Monitoring vs. Execution Control

The problem of execution monitoring of activities belonging to a pre-defined schedule represents a delicate issue, the main reason being that the words "control" and "monitoring" are often interpreted as synonyms[10]. For the remaining of this document, by "control" we mean the deployment of a corrective action aiming at altering the state of the world; by "monitoring" we mean the simple action of observing reality, giving up any volition of interference;

Normally, given a number of activities that have to be scheduled, the main goal of every scheduling procedure is to find a temporal allocation for all the start times so as to meet some desired conditions such as makespan optimality or temporal flexibility maximization, to mention but two. After an initial schedule has been computed, the next step is to put it into execution in a real working environment, and constantly "follow" its execution through proper algorithms. In this case, the challenge is to use intelligent repair strategies to counteract the occurrence of possible exogenous events that may occur at execution time, minimizing all possible disruptions, therefore maintaining as much as possible the schedule's initial characteristics. The activities can thus be interpreted as commands to be dispatched at scheduled times, while the rescheduling action is seen as a necessary reshuffling of the activities, performed according to conservative criteria. In this respect, rescheduling *is a control action* in that it commits itself to changing the future actions which are going to influence real world's evolution (for instance, deciding to swap the execution of two commands).

On the other hand, in the RoboCare context, we obviously have no control whatsoever in the actions the assisted person is going to perform, despite the caregiver's prescriptions. Therefore, the task of following the person's behavior falls exclusively in the *monitoring* category. The system limits itself in taking note of the evolution of the environment, continuosly keeping an updated internal representation of the latter, and possibly reacting to some significant events, if deemed necessary. The monitoring efforts will therefore focus upon: (1) keeping the internal representation of the real world consistent with the behavioral decisions of the assisted person at all times, and (2) performing the necessary rescheduling actions so as to keep at a maximum the number of temporal constraints originally imposed by the caregiver. This

[10] In the italian language, for instance, "to control" and "to monitor" are translated with the same term "*controllare*".

second point is of great importance as the maintenance of temporal information in terms of constraints is essential in order to perform correct situation assessment and/or future-consistent *what-if* analysis.

4.3 Constraint Management in the RoboCare Context

An extrememly important role in the execution monitoring problem within the RoboCare context, is played by the management of all the temporal constraints present in the schedule. As the environment sensing cycle commences, the system periodically checks the state of the monitored area, trying to detect and recognize the execution state of all the activities.

Regardless of the prescribed behavior described in the baseline schedule, the assisted person is obviously free to act as she likes: this basicly means that at each detection cyle, the system is called to precisely assess the possible differences between the actual and desired state. Assessing such differences does not necessarily entails the necessity for a system reaction, as the schedule is in general synthesized according to flexibility criteria: only when a true constraint violation occurs, shall reaction be triggered.

To be more concrete, let us consider the monitoring of a behavioral pattern described by a schedule composed of activities $A = \{a_1, a_2, \ldots, a_n\}$ be the set of activities involved, and $C = \{c_1, c_2, \ldots, c_m\}$ the set of temporal constraints insisting among the activities. In order to represent an *executable* schedule, $\langle A, C \rangle$ must be both temporally and resource consistent. It is responsibility of the caregiver to synthesize a *semantically* correct plan, while the system is able to detect possible temporal and resource inconsistencies, after the problem loading phase. In case of resource inconsistencies (i.e., should the assisted person be wrongly scheduled to perform two activities at the same time), the system automatically proposes an alternative plan and waits for the caregiver's acceptance; instead, temporal inconsistencies require immediate corrective intervention on behalf of the user. The alternative plan is computed exploiting the scheduling capabilities of the most recent version of O-OSCAR Scheduling Architecture [3]: in fact, the ADL Monitor largely bases upon O-OSCAR's scheduling and modelling features.

Algorithm 1 shows the execution monitoring algorithm employed in the RoboCare context. As shown in the algorithm, an "environment sensing" action is periodically performed (line 2). This occurs by accessing the symbolic representation of the current situation (S_t). As we show in section 5, this information is obtained by means of a cooperative multi-agent deduction process. The details of how deduction occurs starting from the symbolic information deriving from the sensors is the object of section 5. As a result, the set $Events_t$ of the occurred events is periodically acquired. By *event* we mean any mismatch between the expected situation, according to the caregiver's prescriptions, and the actual situation (i.e., a planned action which fails to be executed, is considered as an event).

Algorithm 1 The Execution Monitoring Algorithm.

```
1.  while true do
2.    Events_t ← S_t
3.    if Events_t ≠ ∅ then
4.      C_{r,t} ← removeConstraints()
5.      insertContingencies(Events_t)
6.      K_t ← ∅
7.      while C_{r,t} ≠ ∅ do
8.        c_j ← chooseConstraint(C_{r,t})
9.        if ¬ re-insertConstraint(c_j) then
10.         K_t ← K_t ∪ c_j
11.       end if
12.     end while
13.   end if
14. end while
```

If events are detected, the first action is to remove all the active constraints present in the schedule (line 4). By *active* constraints, we mean those which do not completely belong to the past, with respect to the actual time of execution t_E. More formally, given an execution instant t_E and a constraint c_k binding two time points t_a and t_b, c_k is considered *idle* if and only if $(t_a < t_E) \wedge (t_b < t_E)$. All constraints that are not idle are active. Obviously, idle constraints do not take part in the analysis because they will not play any role in the evolution of the future states of the world.

In the next step (line 5) all the detected contingencies, properly modeled as further constraints, are inserted in the plan. This is the step where the system updates the internal representation of the schedule in order to preserve consistency with the world's true state.

Lines 7–12 implement the constraint re-insertion cycle, where the algorithm tries to restore as many caregiver requirements as possible given the current situation. Notice in fact that it is probable that not all the original constraints will be accepted at this point: the occurrence of the contingencies might in fact have changed the temporal network constrainedness, so as to make impossible the complete re-insertion of the constraints removed at the previous step. During the cycle, all the constraints which are rejected are stored in the set K_t.

Constraints insertion (and rejection) is an extremely delicate issue, for many reasons:

- System reaction may consist in verbal suggestions or warning. The information conveyed by these messages strongly depends on the contents of the set K_t. The analysis of all the rejected constraints quantitatively and qualitatively determines the system's response. Given a temporal network TN underlying the current schedule, the set $K_t = \{k_{t,1}, k_{t,2}, ..., k_{t,r}\}$ must be such that: (1) the insertion of each $k_{t,j}$ in TN causes a propagation

failure; (2) the cardinality of K_t is maximum. Condition (1) ensures that every constraint in K_t plays a role in determining system's reaction, ruling out false-positive situations; condition (2) ensures that no contingency escapes system's attention.

- The acceptance of each constraint c_j (and complementarily, the contents of K_t), is generally dependent on the particular order chosen for re-insertion. In general, a number of different choice heuristics (chooseConstraint() method) can be envisaged, leading to different approaches for contingency management. To clarify this issue, let us consider a temporal network TN and two constraints c_1 and c_2 such that the attempt of posting both of them in TN would determine an inconsistency: in this case, if the insertion order is $\{c_1, c_2\}$, then c_2 is going to be rejected; if the opposite order is used, c_1 is rejected. Since in the RoboCare context it is essential that the reaction be related to the closest contingency with respect to execution time t_E, the particular heuristic employed for re-insertion is backward-chronological. The result of this choice is that the rejected constraints will be the ones which are temporally closer to the actual instant of execution, therefore meeting the condition of reaction urgency. In other terms, the RoboCare monitoring system is oriented towards synthesizing a suggestion regarding the primary cause of a violation, rather than forming one based on a distant effect of the assisted person's behavior. The constraints are chronologically ordered taking into account the values of the time point pairs they are connected to. More formally, given a set of constraints $\{c_1(t_{1,s}, t_{1,e}), c_2(t_{2,s}, t_{2,e}), \ldots, c_n(t_{n,s}, t_{n,e})\}$, where each $c_i(t_{i,s}, t_{i,e})$ connects the time points $t_{i,s}$ and $t_{i,e}$, the constraint $c_i(t_{i,s}, t_{i,e})$ chronologically precedes the constraint $c_j(t_{j,s}, t_{j,e})$, if $min(t_{i,s}, t_{i,e}) < min(t_{j,s}, t_{j,e})$.
- The importance of maximizing the number of accepted constraints is directly linked to the need to maintain a schedule's representation which is at all times as close as possible to the original specifications, despite the assisted person's actions. The reason is twofold:
  1. the system should at all times be able to give correct answers to question related to future allocations of the activities, as well as to the temporal bounds insisting among them: Questions like: "At what time do I have to take my medication?" or "How much time have I got between lunch and dinner?" should always be answered correctly (according to the original caregiver's plan);
  2. the system should retain the ability to perform correct *what-if* analysis, in order to deliver reliable information in case of requests like: "If I go for a walk at four o'clock, will I come back in time to watch my favourite TV show?". It is straightforward how the reliability of the answer is strictly related to the quantity of original temporal information that the system is able to retain during the monitoring.

Figure 6 depicts a snapshot during the execution monitoring of a schedule composed of three activities. Activity A_1 is currently under execution, as

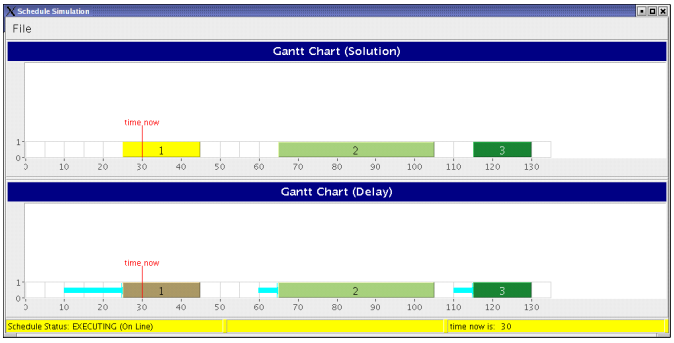

Fig. 6. A simple schedule being monitored

the t_E *time now* mark shows, while activities A_2 and A_3 have yet to be executed. The thick arrows before the activities show that A_1 did not begin at the originally scheduled time, but underwent a certain delay. Note that as a consequence of A_1's postponement, also A_2 and A_3 have been delayed. This is caused by the existence of minimum time constraints between the end of A_1 and the beginning of A_2 and A_3. This example shows the system's capabilities to maintain temporal information in terms of constraints among the activities, despite the occurrence of disturbing events.

4.4 From Constraint Violations to Verbal Interaction

After each detection cycle, every constraint in the set K_t undergoes a thorough analysis. All possible semantic associations between the type of violated constraint and some particular execution conditions are explored in order to trigger meaningful and intelligent system reaction.

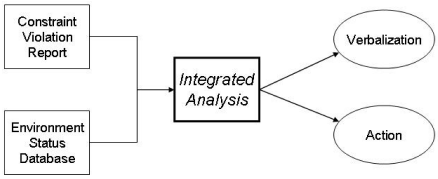

Fig. 7. Situation assessment during execution monitoring.

User interaction with the robot is ensured through the coupling of a voice recognition system callled *Sonic*, developed at the University of Colorado, and a speech synthesis system called *Lucia*, developed at the Institute of Cognitive Sciences and Technologies of Padua. Lucia is a software talking head, able to verbalize and mime phrases in text form. Verbalization synthesis and user feedback interpretation is demanded to a particular agent, called *ChickenBrain* on account of its so far limited capabilities.

Information about timings of past, present and future actions is continously retained: coupling this information with proper constraint violation analysis turns out to be extremely useful. For instance, let us suppose that

two scheduled activities are being monitored, in particular the activity of *cooking* and *having lunch*; let us also suppose that these two activities are temporally constrained so that having lunch should follow cooking but no later than two hours after cooking. If the assisted person starts having lunch too soon (i.e., violating the minimum constraint between the two activities), the system should issue a warning. Obviously, constraint violation alone is not enough to synthesize a meaningful warning verbalization. In fact, it is very useful to integrate the information about the state of the *cooking* activity (see figure 7): for instance, depending whether the assisted person has performed the cooking or not, the system might suggest to delay the lunch or to prepare something warm to eat. This simplified semantic analysis is performed by the *ChickenBrain* agent, which is also responsible for the coordination and management of information synthesis and exchange to/from the robot, through the talking face *Lucia*. As figure 7 shows, system reaction may involve a simple verbalization, a physical action (the robot goes to the assisted person), or a combination of the two, depending on the nature of the triggering event.

5 Multi-Agent Coordination Infrastructure

Coordination of multiple services is achieved by solving a Multi-Agent Coordination (MAC) problem. The MAC is cast as a Distributed Constraint Optimization Problem (DCOP), and solved by ADOPT-N [4], an extension of the ADOPT (Asynchronous Distributed Optimization) algorithm [21] for dealing with n-ary constraints.

The agents placed in the environment are a combination of sensors and actuators. In addition, the ADL monitor has a somewhat double nature. This is because the ADL monitor can operate according to two possible modalities: the assisted person may want to ask "have I taken my afternoon medication?", thus the more sensor-like role of the ADL; on the other hand, upon ascertaining that the assisted person has not taken an important item of medication, the ADL can itself intervene pro-actively, e.g., issuing a suggestion or warning.

One of the most crucial issues which arises when integrating diverse agents is that of coordination. Specifically, the combination of basic services provided by all these agents is accomplished by a distributed constraint reasoning infrastructure. The coordination scheme provides a "functional cohesive" for the elementary services, as it defines the rules according to which the services are triggered. Each service corresponds to a software agent to which tasks are dynamically allocated in function of the current state of the environment and of the assisted person. For instance, if the PLT and PR services recognize that the assisted person is lying on the floor in the kitchen (a situation which is defined as "anomalous" in the overall rule set), then the coordination mechanism will trigger the robot to navigate towards the assisted person's location and ask whether everything is all right.

The coordination of the above mentioned elementary services is defined so as to demonstrate complex added value services which require the cooperation of multiple elementary services. Some examples of global behaviors are the following:

Scenario 1 *The assisted person is in an abnormal posture-location state (e.g., lying down in the kitchen).* ***System behavior:*** *the robot navigates to the person's location, asks if all is well, and enacts a pre-defined contingency plan, such as placing an emergency phone call.*

Scenario 2 *The ADL monitor detects that the time bounds within which to take a medication are jeopardized by an unusual activity pattern (e.g., the assisted person starts to have lunch very late in the afternoon).* ***System behavior (option 1):*** *the robot will reach the person and verbally alert him/her of the possible future inconsistency.* ***System behavior (option 2):*** *the inconsistency is signaled through the PDA.*

Scenario 3 *The assisted person asks the robot, through the PDA or verbally, to go and "see if the window is open".* ***System behavior:*** *the robot will navigate to the designated window (upon obtaining its location from the fixed stereo cameras) and* ***(option 1)*** *relay a streaming video or snapshot of the window on the PDA, or* ***(option 2)*** *take a video/snapshot of the window, return to the assisted person and display the information on its screen.*

Scenario 4 *The assisted person asks the intelligent environment (through the PDA or verbally to the robot) whether he/she should take a walk now or wait till after dinner.* ***System behavior****: the request is forwarded to the ADL monitor, which in turn propagates the two scenarios (walk now or walk after dinner) in its temporal representation of the daily schedule. The result of this deduction is relayed to the assisted person through the PDA or verbally (e.g., "if you take a walk now, you will not be able to start dinner before 10:00 pm, and this is in contrast with a medication constraint").*

5.1 Casting the MAC Problem

As mentioned, multi-agent coordination is cast as a distributed constraint optimization problem and solved by the agents according to the (distributed) ADOPT-N algorithm. Specifically, a distributed constraint optimization problem is a tuple $\langle \mathcal{V}, \mathcal{D}, \mathcal{C} \rangle$ where $\mathcal{V} = \{v_1, \ldots, v_n\}$ are variables with values in the domains $\{D_1, \ldots, D_n\} = \mathcal{D}$, and $\mathcal{C}$ is a set of constraints among variables. Constraints may involve an arbitrary subset of the variables (n-ary constraints): a constraint among the set $C \subset \mathcal{V}$ of k variables is expressed as a value function in the form $f_C : D_1 \times \ldots \times D_k \rightarrow \mathbb{N}$. For instance, a constraint involving the three variables $\{v_1, v_3, v_7\}$ may prescribe that the cost of a particular assignment of values to these variables amounts to c, e.g., $f_{v_1,v_3,v_7}(0, 3, 1) = c$. The objective of a constraint optimization algorithm is

to calculate an assignment $\mathcal{A}$ of values to variables while minimizing the cost of the assignment $\sum_{C \in \mathcal{C}} f_C(\mathcal{A})$, where each f_C is of arity $|C|$.

In the specific case of the RDE, the cost function is modeled so as to reflect the desiderata of system behavior. Specifically, the domains of the variables model the states of the services (i.e., what the system can provide) as well as the possible states of the environment and of the assisted person (i.e., what can occur). Constraints bind these variables to model relations among services, that is, the overall behavior of the smart home and how knowledge is shared among the agents.

In general, the variables represent input for the decision process and/or instructions for controlling the enactment of the services provided by the RDE. For instance, the PLTState variable represents the position of the assisted person in the environment (whose domain is $D_{\text{PLTState}} = \{KITCHEN, LIVINGROOM, BATHROOM, BEDROOM, UNKNOWN\}$), while the PRState variable carries the information on the person's posture ($D_{\text{PRState}} = \{STANDING, SEATED, LYING, UNKNOWN\}$). These two variables are purely "sensory", as their value is determined by the sensory input obtained from the PLT and PR services. An example of "enactment" variable is RobotCommand, which is set autonomously by its agent according to the decisions taken during the execution of the ADOPT-N cooperative solving algorithm. Moreover, agents can have more than one variable. This is the case of the agent representing the robot, which also has the variable RobotState representing the current task in wihch the robot is engaged ($D_{\text{RobotState}} = \{DONE, COMPUTING, FAILED, INACTIVE\}$).

The value functions which model the constraints in the system describe a global cost function whose minima represent the desired system behavior. All consistent states evaluate to a global cost of 0, while inconsistent situations evaluate to ∞. Consistent states establish a correspondence between observations from the sensors and the desired combination of behaviors of the services. For reasons of space we cannot describe the full set of constraints which models the behavior of the RDE as it is instantiated in the ROBO-CARE lab. One meaningful example of such constraints is the following: when the PLT and PR sensors assess that there is an emergency situation (e.g., the assisted person is lying on the floor in the kitchen), the PLTState and PRState variables are set to $KITCHEN$ and $LYING$ respectively; we wish to model the fact that the variable representing the assisted person's current activity (Activity) should be set to $EMERGENCY$ in the event of anomalous situations such as this one; therefore, we model the ternary constraint $C = \{\text{PLTState}, \text{PRState}, \text{Activity}\}$, which prescribes that

$$f_C(KITCHEN, LYING, EMERGENCY) = 0$$
$$f_C(BATHROOM, LYING, EMERGENCY) = 0$$
$$\dots$$

and that all assignments such that the situation does not appear to be an emergency and variable Activity $= EMERGENCY$ have infinite cost:

$f_C(KITCHEN, SEATED, EMERGENCY) = \infty$
$f_C(BEDROOM, STANDING, EMERGENCY) = \infty$
...

To continue this example, we wish now to model the proper enactment of Lucia the robotic mediator in case of an emergency (i.e., when the Activity variable indicates that the assisted person is in a state of emergency): we thus need to model the fact that the robot should navigate towards the assisted person and ask if assistance is needed (an AskIfOkay variable is modeled on the agent representing Lucia's verbal capabilities); in terms of the relevant variables this equates to modeling (Activity = $EMERGENCY$) $\wedge$ (RobotState = $DONE$) $\Rightarrow$ (AskIfOkay = YES); thus, we specify the ternary constraint $C' = \{\text{Activity}, \text{RobotState}, \text{AskIfOkay}\}$, which prescribes that

$f_{C'}(EMERGENCY, DONE, YES) = 0$
$f_{C'}(EMERGENCY, FAILED, YES) = 0$
$f_{C'}(EMERGENCY, COMPUTING, YES) = 0$
$f_{C'}(EMERGENCY, INACTIVE, YES) = 0$

and that all other assignments such that $\langle \text{Activity}, \text{RobotState}, \text{AskIfOkay} \rangle \neq \langle EMERGENCY, DONE, YES \rangle$ have infinite cost:

$f_{C'}(LUNCH, COMPUTING, YES) = \infty$
$f_{C'}(BREAKFAST, STATE\_INACTIVE, YES) = \infty$
...

Clearly, the constraint optimization paradigm, as opposed to constraint satisfaction, allows us to employ also soft constraints, i.e., constraints described by cost functions which evaluate to any $\mathbb{N}$ and not just 0 or ∞. This is useful in case we wish to specify degrees of acceptance of the system's overall behavior. For instance, it may be useful to model the fact that it is preferable for the robot to reach the assisted person to make a suggestion, but that a sub-optimal solution in which the robot does not reach the person before issuing advice is also acceptable. Modeling this kind of situation can facilitate the distributed reasoning algorithm because it allows for the existence of sub-optimal solution assignments which could avoid significant computational overhead. This is the case, for instance, in ADOPT and its variant ADOPT-N, which can perform bounded-error approximation to quickly find approximate solutions while maintaining a theoretical guarantee on solution quality.

5.2 High-Level Specification of System Behavior

The behavior of the RDE agents as they are set up in the ROBOCARE lab in Rome is modeled through a DCOP comprising 6 agents, 11 variables and over 1100 cost function definitions (corresponding to 11 n-ary constraints). Indeed, the definition of a complete behavioral model for the RDE cannot be given without the use of a high level specification formalism.

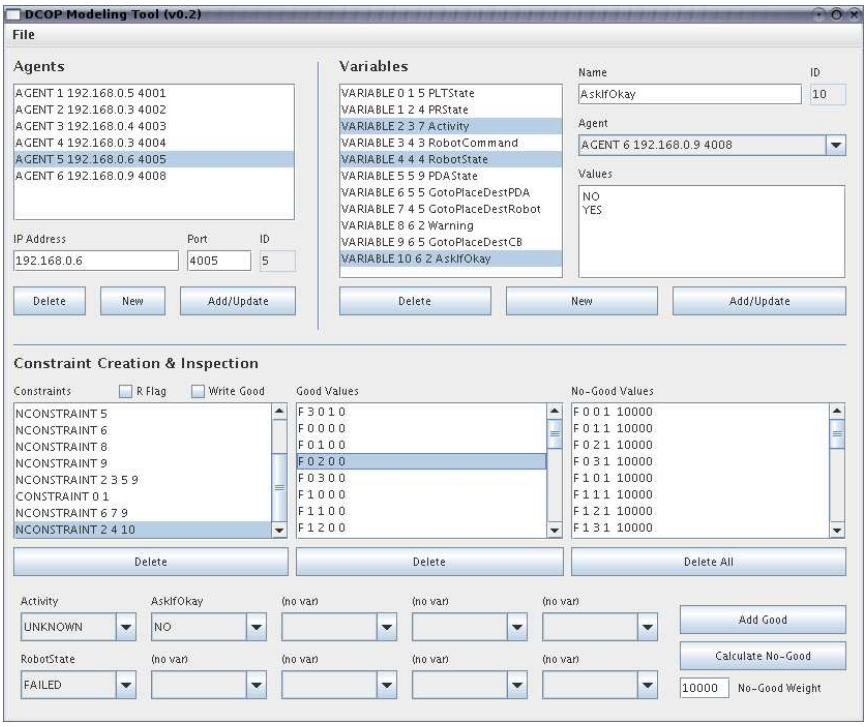

Fig. 8. The DCOP specification interface employed for defining the RDE's overall behavior.

While the study of an efficient specification language has not been the focus of our work during the project, an initial interface for facilitating DCOP formulation has been developed (figure 8). Specifically, the interface allows to create agents (specifying also the relevant parameters for distribution), variables and their domains, and constraints among the variables. In addition to allowing the use of user-intelligible names for domain values, the system allows to visually establish constraints among groups of variables. Also, the user can specify constraints which model the desired behavior of the agents (i.e., describing situations with zero cost) and automatically obtain value function specifications which model the resulting undesired assignments (i.e., describing situations with high cost), and vice-versa. This form of automatic no-good specification is extremely handy in the RDE scenario. In fact, it is often the case that there are very few assignments which are admissible (thus modeling them is trivial) while the transitive closure consists of hundreds of no-goods, the manual definition of which is tedious and error-prone.

In conclusion, it is interesting to notice how the ability of ADOPT-N to handle n-ary constraints is an important advantage in the RDE scenario. While it would be possible to specify the RDE MAC problem in terms of only binary

constraints, the possibility to employ n-ary constraints is a strong advantage because it is well suited for modeling complex situations which involve many different components of the smart home. This renders problem specification more intuitive as well as more compact (in terms of number of constraints). Notice, though, that this comes at the price of complex value functions as mentioned above. In this respect, the DCOP specification interface is a strong added value as it allows intuitive constraint specification while minimizing the impact of constraint complexity.

5.3 Cooperatively Solving the MAC Problem

As noted, an ADOPT-N agent is instantiated for each service provided by the components of the RDE. Given the current situation S, these agents communicate to each other messages which allow them to trigger the appropriate behavior. Clearly, the state of the environment, of the assisted person, and of the services themselves changes in time: let the situation (i.e., the state of the environment, of the assisted person and of the services) at time t be S_t. The DCOP formulation of the MAC described earlier represents the desired behavior of the system in function of the possible states of the RDE. Therefore, if $S_t \neq S_{t-1}$, the ADOPT-N agents must trigger an "instance of coordination" so as to decide the assignment $\mathcal{A}$ which represents the desired enactment of services.

One of the challenges of the RDE scenario with respect to distributed coordination is the heterogeneity of the agents. The strong difference in nature between the various components of the RDE reflects heavily on the coordination mechanism. This is because of the uncertainty connected to the time employed by services to update the symbolic information which is passed on to the agents. For instance, the PLT service is realized through an artificial stereo-vision algorithm the convergence of which is strongly affected by a variety of factors: first of all, object tracking is difficult when many similar objects move in the same space; second, object recognition is generally difficult when the environment and the agents cannot be adequately structured; third, when moving observers are used for monitoring large environments the problem becomes harder since it is necessary to take into account also the noise introduced by the motion of the observer. A similar problem affects also the ADL monitor, which must propagate sensor-derived information on the temporal representation of the behavioral constraints. The aim of the ADL monitor is to constantly maintain a schedule which is as adherent as possible to the assisted person's behavior in the environment. This may require a combination of simple temporal propagation, re-scheduling, as well as other complex procedures (e.g., deciding which of the violated constraints are meaningful with respect to verbal warnings and suggestions).

As a consequence, it is in general impossible to have strict guarantees on the responsiveness of the agents. For this reason the albeit asynchronous solving procedure needs to be iterated synchronously. More specifically, ADOPT-N

Algorithm 2 Synchronization schema followed by each ADOPT-N agent a in the RDE.

```
1.  t ← 0
2.  S_t ← getSensoryInput(V_a)
3.  while true do
4.    S_{t−1} ← S_t
5.    while (S_t = S_{t−1}) ∧ (t ≥ a′.t, ∀a′ ≠ a) do
6.      S_t ← getSensoryInput(V_a)
7.    end while
8.      t ← t + 1
9.    forall d_i ∈ D_{v∈V_a} do
10.     lb(d_i) ← 0          /** Reset lower and **/
11.     ub(d_i) ← ∞          /** upper bounds   **/
12.   end for
13.   A|_{V_a} ← runAdopt()  /** Iteration terminates on ADOPT-N termination **/
14.   triggerBehavior(A|_{V_a})
15. end while
```

is deployed in the RDE as described in algorithm 2, according to which the agents continuously monitor the current situation, and execute the ADOPT-N algorithm whenever a difference with the previous situation is found. The getSensotyInput() method in the pseudo-code samples the state of the environment which is represented by agent a's variables V_a (what we have informally called "sensory" variables). Specifically, the values of these variables are constrained to remain fixed on the sensed value during the execution of the ADOPT-N decision process. In practice, this occurs by posting a unary constraint which prescribes that any value assignment which is different from the sensed value should evaluate to ∞, and is therefore never explored by the agent controlling the variable. This constraint posting mechanism is a feature of ADOPT-N. Clearly, it is also possible to restrict the values of these variables by modifying the problem before each iteration. The constraint posting strategy was employed to facilitate representation and re-use of code. In fact, the DCOP problem never needs to change between iterations, and this allows to minimize the re-initialization phase between iterations (which can be reduced to resetting the lower and upper bounds of the domain values for each variable as shown in the algorithm — see [21, 4] for details on the ADOPT and ADOPT-N algorithms). Moreover, posting a unary constraint on a variable for the entire duration of the solving process does not affect the computational complexity of the algorithm.

Notice, though, that ADOPT and its variant ADOPT-N do not rely on synchronous communication between agents, thus natively supporting message transfer with random (but finite) delay. This made it possible to employ ADOPT-N within the RDE scenario without modifying the algorithm internally. Furthermore, while most distributed reasoning algorithms (like ADOPT

itself) are employed in practice as concurrent threads on a single machine (a situation in which network reliability is rather high), the asynchronous quality of ADOPT-N strongly facilitated the step towards "real" distribution, where delays in message passing increases in magnitude as well as randomness.

6 Conclusions

During the first two years of project development, efforts were concentrated on developing the technology to realize the individual components (or services) of the RDE. The services provided by this technology were deployed in the environment according to a service-oriented infrastructure, which is described in [22]. This allowed to draw some interesting conclusions on the usefulness of robots, smart sensors, and pro-active domestic monitoring in general (see, e.g., [23]).

In the final year of the project, and in part towards the goal of participating in the RoboCup@Home competition, the attention shifted from single component development to the functional integration of a continuous and context-aware environment. The issue was to establish a convenient way to describe how the services should be interleaved in function of the feedback obtained from the sensory sub-system and the user. The strategy we chose was to cast this problem, which can also be seen as a service-composition problem, in the form of a multi-agent coordination (MAC) problem.

It is interesting to notice that the specific constraint-based formulation of the MAC problem is strongly facilitated by the possibility to encode n-ary constraints. As discussed, this is convenient for modeling the functional relationship among multiple services as it allows to precisely indicate the relationships between sensed input and the resulting enactment. Another advantage of the constraint-based formulation is that the system is easily scalable. In fact, adding another sensor, service or intelligent functionality requires adding an ADOPT-N agent and its variables to the problem, and system behavior can be specified incrementally.

Finally, as noted earlier, an interesting area for future research is the development of more powerful formalisms for specifying service interaction and invocation in terms of a DCOP problem. One of the goals of ROBOCARE has been to develop technology which is at least to a certain degree useable by non-experts[11]. The knowledge acquired in three years of ROBOCARE can certainly contribute to building systems which are close to becoming market-level products.

[11] See, e.g., the behavioral constraint specification formalism used by caregivers described in [1].

References

1. F. Pecora, R. Rasconi, G. Cortellessa, and A. Cesta, "User-Oriented Problem Abstractions in Scheduling, Customization and Reuse in Scheduling Software Architectures," *Innovations in Systems and Software Engineering*, vol. 2, no. 1, pp. 1–16, 2006.
2. A. Cesta, G. Cortellessa, A. Oddi, N. Policella, and A. Susi, "A constraint-based architecture for flexible support to activity scheduling," ser. Lecture Notes on Artificial Intelligence (LNAI), vol. 2175. F. Esposito (Ed.), 2001, pp. 369–381.
3. A. Cesta, R. Rasconi, G. Cortellessa, A. Oddi, F. Pecora, and N. Policella, "O-OSCAR 2.0: a Flexible Scheduling Tool for Total Plan Life-Cycle Support," ISTC-CNR, Institute for Cognitive Science and Technology, Italian National Research Council, Tech. Rep., October 2004.
4. F. Pecora, P. Modi, and P. Scerri, "Reasoning About and Dynamically Posting n-ary Constraints in ADOPT," in *Proceedings of 7th Int. Workshop on Distributed Constraint Reasoning, at AAMAS'06*, 2006.
5. P. Cosi, A. Fusaro, and G. Tisato, "LUCIA a New Italian Talking-Head Based on a Modified Cohen-Massaros Labial Coarticulation Model," in *Proceedings of Eurospeech 2003, Geneva, Switzerland, 2003.*, 2003.
6. K. Konolige, "Small vision systems: Hardware and implementation," in *Proc. of 8th International Symposium on Robotics Research*, 1997.
7. N. Friedman and S. Russell, "Image segmentation in video sequences: a probabilistic approach," in *Proc. of 13th Conf. on Uncertainty in Artificial Intelligence*, 1997.
8. C. R. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 780–785, 1997.
9. C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR'99)*, 1999, pp. 246–252.
10. M. Harville, G. Gordon, and J. Woodfill, "Foreground segmentation using adaptive mixture models in color and depth." in *Proc. of IEEE Workshop on Detection and Recognition of Events in Video*, 2001, pp. 3–11.
11. T. Darrell, D. Demirdjian, N. Checka, and P. F. Felzenszwalb, "Plan-view trajectory estimation with dense stereo background models," in *Proc. of 8th Int. Conf. On Computer Vision (ICCV'01)*, 2001, pp. 628–635.
12. D. Beymer and K. Konolige, "Real-time tracking of multiple people using stereo," in *Proc. of IEEE Frame Rate Workshop*, 1999.
13. J. Krumm, S. Harris, B. Meyers, B. Brumitt, M. Hale, and S. Shafer, "Multi-camera multi-person tracking for easyliving," in *Proc. of Int. Workshop on Visual Surveillance*, 2000.
14. A. Mittal and L. S. Davis, "M2Tracker: A multi-view approach to segmenting and tracking people in a cluttered scene using region-based stereo," in *Proc. of the 7th European Conf. on Computer Vision (ECCV'02)*. Springer-Verlag, 2002, pp. 18–36.
15. I. Haritaoglu, D. Harwood, and L. S. Davis, "W4: Real-time surveillance of people and their activities," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, 2000.

16. K. Roh, S. Kang, and S. W. Lee, "Multiple people tracking using an appearance model based on temporal color," in *Proc. of 15th Int. Conf. on Pattern Recognition (ICPR'00)*, 2000.
17. J. Li, C. S. Chua, and Y. K. Ho, "Color based multiple people tracking," in *Proc. of 7th Int. Conf. on Control, Automation, Robotics and Vision*, 2002.
18. J. Kang, I. Cohen, and G. Medioni, "Object reacquisition using invariant appearance model," in *Proc. of 17th Int. Conf. on Pattern Recognition (ICPR'04)*, 2004.
19. M. Isard and A. Blake, "Condensation – conditional density propagation for visual tracking," *International Journal of Computer Vision*, vol. 29, no. 1, pp. 5–28, 1998.
20. R. Dechter, I. Meiri, and J. Pearl, "Temporal constraint networks," *Artificial Intelligence*, vol. 49, pp. 61–95, 1991.
21. P. Modi, W. Shen, M. Tambe, and M. Yokoo, "Adopt: Asynchronous distributed constraint optimization with quality guarantees," *Artificial Intelligence*, vol. 161, pp. 149–180, 2005.
22. S. Bahadori, A. Cesta, L. Iocchi, G. Leone, D. Nardi, F. Pecora, R. Rasconi, and L. Scozzafava, "Towards Ambient Intelligence for the Domestic Care of the Elderly," in *Ambient Intelligence: A Novel Paradigm*, P. Remagnino, G. Foresti, and T. Ellis, Eds. Springer, 2004.
23. A. Cesta and F. Pecora, "Integrating Intelligent Systems For Elder Care In RoboCare," in *Promoting Independence for Older Persons with Disabilities*, W. Mann and A. Helal, Eds. IOS Press.