DCOP FOR SMART HOMES: A CASE STUDY

FEDERICO PECORA AND AMEDEO CESTA

Institute for Cognitive Science and Technology, Italian National Research Council,
Via S. Martino della Battaglia, 44 – I-00185 Rome, Italy

The aim of this article is to bring forth the issue of integrating the services provided by intelligent artifacts in Ambient Intelligence applications. Specifically, we propose a Distributed Constraint Optimization procedure for achieving a functional integration of intelligent artifacts in a smart home. To this end, we employ ADOPT-N, a state-of-the-art algorithm for solving Distributed Constraint Optimization Problems (DCOP). This article attempts to state the smart home coordination problem in general terms, and provides the details of a DCOP-based approach by describing a case study taken from the ROBOCARE project. More specifically, we show how (1) DCOP is a convenient metaphor for casting smart home coordination problems, and (2) the specific features which distinguish ADOPT-N from other algorithms for DCOP represent a strong asset in the smart home domain.

Key words: distributed constraint optimization, multi-agent systems, ambient intelligence, integrated intelligent systems

1. INTRODUCTION

The concept of “smart” homes in future domestic technology is the central aspect of the current discussion in domotics. This debate has fostered the strongly inter-disciplinary research field of Ambient Intelligence (AmI). In recent years, the products of various research lines are converging into the domestic applicative scenario in the form of distinct intelligent artifacts which provide high added-value services.

A smart home is a system which is responsive to people’s needs and actions, a pervasive accessory to human cognitive and physical capabilities. The simple juxtaposition of sophisticated devices or services does not alone lead to a smart environment. Recent research in AmI has focused on techniques to compose the dedicated services provided by domotic devices so as to provide high added-value functionalities. Indeed, building a smart environment must necessarily deal with how the domotic services work together. Given also the strong diversity of domotic technology, the challenge today is to provide such methods of integration. For this reason, a strong priority in AmI research is the development of methodologies for service integration which can be re-used in diverse scenarios with potentially different domestic technology. While ad-hoc integration schemas have been put in place for the realization of AmI systems, the goal of building such systems by integrating off-the-shelf intelligent artifacts within a re-usable framework still lies beyond the current state-of-the-art. Specifically, the aim of this article is to demonstrate the need for an integration framework which provides the “functional cohesive” underlying the smart home.

In this article we show that integrating complex services is itself a problem which requires intelligent reasoning. As is the case in related fields such as web service composition, we believe that a key contribution to this problem can be provided by AI problem solving. This article proposes one possible solution to the service coordination problem, namely distributed constraint reasoning. These and other issues are presented through the ROBOCARE Domestic Environment (RDE), a prototypical smart home developed within the three-year ROBOCARE project at ISTC-CNR. Within this case study we present a coordination framework based on ADOPT-N a framework for solving Distributed Constraint Optimization Problems (DCOP) based on the ADOPT algorithm (Modi et al. 2005). We show how the domotic service integration problem can be cast to DCOP, and how the use of ADOPT-N allows to obtain a coherent, continuous, open and expandable coordination infrastructure for the domotic services which compose the smart home.

© 2007 Blackwell Publishing, 350 Main Street, Malden, MA 02148, USA, and 9600 Garsington Road, Oxford OX4 2DQ, UK.
The article is organized as follows. First, we describe the multi-agent coordination problem, and show how it can be reduced to DCOP for resolution with ADOPT-N. The DCOP problem and the ADOPT-N algorithm are also briefly described for completeness. Next, the ROBOCARE scenario is described, briefly detailing the domotic components (agents) which are to be integrated in the overall system. Finally, we show how the DCOP solving framework is deployed within the physically distributed domestic scenario, and conclude the article with some remarks on related and future work.

2. MULTI-AGENT COORDINATION IN THE SMART HOME

AmI addresses one of the open challenges in AI, namely that of integrating diversified intelligent capabilities to create a proactive assistant for everyday life in a domestic environment. The building blocks of such a system are intelligent artifacts (or domotic components), such as sensory systems, conversational assistants, health monitoring devices and so on. These components together provide the services necessary to observe the environment and the person(s) living in it, as well as the functionalities which are necessary to proactively participate in everyday life within the environment. In general, and for the purposes of this discussion, we qualify these components as applications which provide services.

One of the most crucial issues which arises when integrating diverse services is that of coordination. Multiple services can be activated concurrently, and the combination of different services can provide complex services whose added value is instrumental in creating useful support services for the user. For instance, suppose that two distinct applications provide, respectively, a person localization service (i.e., the location of a person in the smart home) and a posture recognition service. The services of these two sensory applications can be employed in a coordinated fashion to deduce symbolic information on the state of a person in the environment. For instance, it would be possible to recognize that the person is laying on the floor near the kitchen table (a situation which can be appropriately defined as “anomalous”), and this could be used to call into play some form of system intervention, such as calling an ambulance.

The idea we put forth in this article is that placing domotic applications within a coordination scheme allows us to leverage potentially sophisticated domestic services to obtain a smart home with more added value than the sum of its parts. An integration 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 an embedded agent (Coen 1998), i.e., a software agent to which tasks are dynamically allocated in function of the current state of the environment and of the assisted person. Thus, service coordination can be reduced to Multi-Agent Coordination (MAC). The general schema of MAC-based development is depicted in Figure 1, where applications in the physical environment (portion (a) of the figure) are wrapped by software agents whose participation in a distributed coordination infrastructure (portion (b)) defines the way their services are invoked.

2.1. Distributed Constraint Reasoning

Research in MAC has produced a number of results which can be instrumental to the smart home applicative scenario. In the case study presented in this article, the coordination of basic services provided by all applications is accomplished via a reduction of the MAC problem to a Distributed Constraint Optimization Problem (DCOP). DCOP can be defined as an extension
of classical constraint reasoning problems as follows. A constraint reasoning problem is a tuple \( (V, D, C) \), where \( v_i \in V \) is a variable whose values can be chosen in the domain \( D_i \in D \), and \( C \) is set of constraints. In constraint satisfaction (CSP), a constraint on a subset of variables \( \{v_i, \ldots, v_j\} \subseteq V \) is a function \( f_{\{v_i, \ldots, v_j\}} : D_i \times \ldots \times D_j \rightarrow \{\text{true, false}\} \), expressing the fact that certain assignments of values to variables are admissible (i.e., they evaluate to true) and that others are not admissible. In constraint optimization (COP), the constraints evaluate to costs rather than crisp values: a constraint in COP is a function \( f_{\{v_i, \ldots, v_j\}} : D_i \times \ldots \times D_j \rightarrow \mathbb{N} \), expressing the fact that assignments of values to variables evaluate to an integer representing the cost of the assignment. It is therefore possible in COP to express “soft” constraints, that is, assignments of values to variables evaluate to a level of admissibility rather than a strict Boolean value. A solution to a CSP is an assignment of values to variables which satisfies all constraints (i.e., the assignment is such that all constraints evaluate to true), while in COP, a solution is an assignment which minimizes a given objective function. Typically, the objective function is minimization, i.e., solving a COP equates to finding an assignment with minimum cost.

\[1\text{A complete description of constraint reasoning problems is outside the scope of this article. The reader is referred to Dechter (2003) for an overview of constraint satisfaction and optimization.}\]
In addition to CSP and COP, relatively recent research has also focused on grounding constraint reasoning on the additional requirements imposed by distributed systems. This has yielded a successful branch of constraint reasoning known as Distributed Constraint Reasoning (DCR). DCR focuses on algorithms for solving Distributed CSP and COP (DCSP/DCOP). These problem categories add to their centralized counterparts the following requirements: variables are controlled by physically distributed agents, and constraints exists among variables belonging to different agents; in addition, knowledge is distributed, as agents only know about the constraint that bind the variables they own. Therefore, the presence of inter-agent constraints entails the need for communication, because agents must communicate their decisions to agents with which they share a constraint, and agree upon assignments which satisfy (in DCSP) or minimize the cost of (in DCOP) constraints they are not aware of. Given its distributed nature, algorithms for solving the DCSP/DCOP problem leverage the capability of minimizing communication among agents. For instance, the ADOPT algorithm (Modi et al. 2005) strives to minimize the number of messages exchanged between agents (which is exponential in the worst case) while guaranteeing linear message size. Another well-known algorithm for DCOP is DYNOPT (Petcu and Faltings 2005), which guarantees a polynomial number of messages which are exponential in size in the worst case.

Our approach to solve the MAC problem is based on ADOPT-N, an asynchronous and optimal algorithm for distributed constraint optimization. ADOPT-N is an original extension of the ADOPT algorithm, its key features being the capability to solve DCOP problems with \( n \)-ary constraints and the possibility to include external constraint checking procedures in the cooperative solving process. As we will see, these two features are extremely useful in the context of our smart home case study.

DCOP is an established approach for MAC, and in recent years has been the focus of much research. Further details about algorithms for DCOP in general and ADOPT-N in particular are outside the scope of this article, and the interested reader is referred to (Pecora et al. 2006a) and (Modi et al. 2005) for further details.

### 2.2. DCOP for Smart Home Coordination

Figure 1(b) shows a high-level description of the DCOP reduction: each application is controlled by a software agent which participates in the distributed decision making through variables bound by constraints forming a DCOP. We call \( Ap_i \) the generic intelligent sub-system to be integrated in the smart home. Each application \( Ap_i \) provides a service \( S_i \). For instance, an application might be a sensor providing a person localization service, i.e., which contributes to the smart home the knowledge on the current position of the assisted person in the home. \( \{ v_1, \ldots, v_m \} \) is the set of variables in terms of which the coordination problems is defined. Specifically, each application manages one or more variables through an agent \( A_i \) which constitutes its interface with the underlying coordination mechanism. For instance, the application which provides the localization service will “write” through its agent a variable which represents the current location of the assisted person, and this variable will be “read” by one or more applications providing services which require the knowledge of the person’s current location. In general, some variables represent the input to an application, while others represent its output (in terms of “control signals” which trigger the services). Variables thus represent the interface through which coordination of the services \( S_i \) must occur.

The specific functional relations among the services are modeled as constraints among the variables. The overall constraint-based representation of these relationships encodes the desired behavior of the system. The set of constraints and variables thus constitutes a DCOP which is continuously reasoned upon by the agents through a constraint reasoning framework.
Specifically, the constraints are modeled so that minimum-cost assignments correspond to the desired concurrent behavior of the smart home. Thus, when the ADOPT-N algorithm is run, the agents cooperatively converge on a variable assignment which represents a suitable concurrent behavior of the system given the current situation.

The applications placed in the environment are a combination of sensors, actuators, and symbolic reasoning agents. The variables which constitute the DCOP can be grouped in two kinds, namely input variables ($V_{\text{in}}$) and output variables ($V_{\text{out}}$). Input variables represent the input to the coordination process: they model the external knowledge that is acquired by the sensory applications from the environment, such as the position or posture of the assisted person. Conversely, output variables represent the result of the distributed coordination process: their value, as it is determined through ADOPT-N, constitutes the input to the actuator applications.

In summary, our approach to solve the MAC problem which arises in the smart home domain consists in modeling and iteratively solving a DCOP problem. The constraints in the DCOP problem describe a cost function on the assignments of values to variables. These assignments in turn represent (1) the knowledge that domotic components can contribute to the overall coordination procedure (e.g., the symbolic representation of the state of the environment and of the assisted person obtained through sensory components), and (2) the consequent decisions deduced by the coordination procedure. The correspondence between input and output represents the desired behavior of the system as a whole. This behavior is modeled through the constraints in the DCOP problem: all together, the constraint in the DCOP problem describe a global cost function whose minima (assignments which minimize global cost) occur precisely in correspondence with those decisions which trigger the desired behaviors of the system. In the ROBOCARE case study proposed in the following sections, we will provide a complete example of this modeling strategy, through which we have achieved an advanced context-aware environment (Abowd et al. 2002) for supporting elderly people.

2.3. Cooperatively Solving the Coordination Problem

As depicted in Figure 1, a software agent is instantiated for each service provided by the components of the smart home. Let $S$ denote the current situation, i.e., the state of the services and of the assisted person. As we have said, given $S$, the agents are involved in a distributed computation which guarantees that services are activated according to the specifications modeled as constraints in the DCOP problem. In our case study this is achieved by means of the ADOPT-N algorithm, although in principle any distributed algorithm for DCOP could be employed.

Clearly, the state of the environment, of the assisted person, and of the services themselves changes in time. Let the situation at time $t$ be $S_t$. The DCOP formulation of the coordination problem represents the desired behavior of the system in function of the possible states of the environment and of the assisted person. Therefore, if $S_t \neq S_{t-1}$, the agents must trigger an “instance of coordination” so as to decide the assignment $A$ which represents the desired enactment of services.

One of the challenges for distributed coordination in the RDE case study described in the following sections (and in smart home scenarios in general) is the heterogeneity of the agents. The strong difference in nature between the various components of the RDE reflects heavily on the coordination mechanism because of the uncertainty connected to the time employed by services to update the symbolic information which is passed on to the agents. As we will see, the RDE contains devices which are driven by artificial stereo-vision algorithms, 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 because it is necessary to take into account also the noise introduced by the motion of the observer. This problem also affects other components of the smart home, such as the activity monitor (described later in this article as the ADL Monitor) which must propagate sensor-derived information on the temporal representation of the assisted person’s daily schedule. The aim of this component 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.

Algorithm 1 Synchronization schema followed by each ADOPT-N agent $a$ in the RDE.

```plaintext
iter ← 0
$S_{iter} ← \text{getSensoryInput}(V^a_{in})$
while true do
    $S_{iter-1} ← S_{iter}$
    /** Wait until (current input has changed) OR (someone else has started) **/
    while $S_{iter} = S_{iter-1}$ AND $\text{iter} \geq a', \text{iter}, \forall a' \neq a$ do
        $S_{iter} ← \text{getSensoryInput}(V^a_{in})$
    end while
    iter ← iter + 1
    for all $d_i ∈ D_{i∈V^a_{in}\cup V^a_{out}}$ do
        $lb(d_i) ← 0$ /** Reset lower and **/
        $ub(d_i) ← \infty$ /** upper bounds **/
    end for
    $A|_{V^a_{out}} ← \text{runAdept()}$ /** ADOPT-N termination **/
    triggerBehavior($A|_{V^a_{out}}$)
end while
```

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 is deployed in the RDE as described in algorithm 1, 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 getSensoryInput() method in the pseudo-code samples the state of the environment which is represented by agent $a$’s input variables $V^a_{in}$. All agents concurrently initiate the ADOPT-N algorithm whenever the state changes or another agent has initiated the solving iteration. Thus, when an agent senses a difference in its input variables, its decision to run the coordination algorithm is cascaded to all other agents in the priority tree (see the condition in the internal while loop above).

The values of input variables are constrained to remain fixed on the sensed value during the execution of the ADOPT-N decision process. As we will see, this occurs by means of a unary constraint checking procedure which prescribes that any value assignment which is different from the sensed value should evaluate to $\infty$, and is therefore never explored by the agent controlling the variable. Notice that it is also possible to restrict the values of these variables by modifying the problem before each iteration. The constraint checking 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. Specifically, re-initialization occurs in ADOPT-N by “resetting” the
global cost estimate of every agent, which in turn induces all agents to collectively start over
the distributed search for a cost-minimizing assignment.\(^2\)

Because ADOPT-N does not rely on synchronous communication between agents, it nat-
vively supports message transfer with random (but finite) delay. This made it possible to
employ ADOPT-N within the RDE scenario without modifying the algorithm internally. Fur-
thermore, while most distributed reasoning algorithms are employed in practice as concurrent
threads on a single machine (a situation in which network reliability is rather high), the asyn-
chronous quality of ADOPT-N strongly facilitated the step towards “real” distribution, where
delays in message passing increase in magnitude as well as randomness.

2.4. Specific Features of ADOPT-N

As noted earlier, ADOPT-N brings to DCOP two specific features which are instrumental
for smart-home coordination, namely:

- the capability of dealing with \(n\)-ary constraints;
- the possibility to include external constraint checking procedures in the distributed com-
  putation.

These features were conceived towards the aim of employing ADOPT-N to solve real-
world coordination problems, the nature of which often disallows the common assumptions
that (a) relations only occur among pairs of entities, and (b) all constraints can be explicitly
modeled in the DCOP formulation of the coordination problem. With respect to the former
point, we show in this case study that the ability to model relations among the behaviors of
\(n \geq 2\) services in the smart home is an important asset. In fact, it is intuitive to appreciate that
the added value of integrated service behavior increases with the number of inter-operating
services. As we show in the RDE specifically, the description of “interesting” concurrent
service enactment is often inherently \(n\)-ary.

It is also important to point out that \(n\)-ary relations can always be reduced to binary
ones. On the other hand, it is also true that this operation is often not advisable. From the
operational point of view, and especially in contexts such as smart home coordination (where
the specification of the coordination problem is mostly “hand-made”) the process of casting
the inter-relationships between \(n\) services in terms of binary relations (and the necessary
auxiliary variables) is simply too error-prone to be feasible. This operational advantage of
\(n\)-ary constraints will be clear in the following sections, where the complete DCOP modeling
the ROBOCARE smart home is described.

The latter assumption mentioned above, namely that the coordination problem as a whole
can be modeled in terms of a DCOP problem, is perhaps an even stronger limitation with
respect to certain categories of real-world coordination problems. For instance, it can be
shown (Pecora 2007) that modeling limited capacity resource constraints in “pure” DCOP
formulations necessarily leads to exponential-sized problem formulations. This is where the
second key feature of ADOPT-N comes into play, namely the possibility to subsume parts of
the constraint specification in an external constraint checking procedure. In domains such as
coordination subject to resource constraints this allows to avoid the exponential blow-up in
size of the DCOP specification. However, this key feature was developed with a more general
scope in mind, namely to provide a method for integrating specialized software components

\(^2\)Because ADOPT-based algorithms are grounded on bound-driven reasoning, this is achieved by re-setting the lower and
upper bounds of the domain values for each variable.
as constraint verifiers. In more simple terms, it is possible to delegate the verification of potentially very complex requirements (which for any reason we cannot extensionally represent as constraints in the DCOP formulation) to external computational procedures. More specifically:

**Definition 1.** Given a DCOP $\langle V, D, C \rangle$, a constraint checking procedure $\text{comp}$ takes as input a (partial) assignment $A$ of values in $D$ to variables in $V$, and outputs a cost evaluation of $A$.

In general, a coordination problem may require the definition of a set $\text{COMP}$ of constraint checking procedures. In practice, a computation (i.e., an input-output pair) of a procedure $\text{COMP}$ is similar to a constraint, and the set of all procedures $\text{COMP}$ is akin to the set of constraints $C$. Nonetheless, there is a key difference between constraint checking computations and constraints, namely that a constraint $f_{v_i \ldots v_j}$ is defined with a specific scope (i.e., the mathematical domain $D_i \times \cdots \times D_j$ of the function), while the scope of the procedure can be arbitrary. In other words, we can use constraints to express exactly the conditions to which variable assignments should adhere, while constraint checking procedures can be used only to return the cost of a given partial assignment. As we show in the ROBOCARE domain, external constraint checking plays a key role in integrating sensory components in the RDE. Most importantly though, this mechanism confers a strong quality of expandability to the system, allowing the incremental integration of further reasoning procedures such as machine learning, logical inference engines and so on.

### 3. THE ROBOCARE DOMESTIC ENVIRONMENT

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. It is with this aim that the final year of the project has focused on producing a demonstration exhibiting 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 cognitive support services for elderly people living at home. Such user services include non-intrusive monitoring of daily activities and activity management assistance.

In order to provide relevant services for the elderly user, a key capability of the RDE is the continuous maintenance of a high level of situation awareness. Our goal was to build a smart home which would be capable of knowing and acting upon the state of the environment and of the assisted person. This includes information of two types. First, the system needs to understand the current situation: for instance, what the assisted person is doing (in terms of household activities), if the assisted person is in need of assistance (i.e., contingent, unexpected states of the assisted person). Second, the system is required to be aware of the past and possible future situations. This is a key requirement for supporting different forms of cognitive impairment with services ranging from simple reminding to scenario projection. This feature is the principal source of pro-activity of the RDE: a system which is capable of performing complex temporal deductions on the past and current situations is in principal able to pro-actively suggest or warn the assisted person.

This leads to another key requirement of the RDE, namely its capability to interact with the user. The RDE needs to be provided with a means to communicate with the user, and the assisted elder should be able to summon the services provided by the RDE. For this reason,
the RDE is equipped with an “embodiment” of the system consisting in a context-aware domestic robot developed by the ROBOCARE team at the ISTC-CNR. The robot is aimed at demonstrating the feasibility of an embodied interface between the assisted elder and the smart home: it is the assisted person’s interface with the assistive environment.

Overall, the RDE can be viewed as a “robotically rich” environment composed of sensors and software agents whose overall purpose is to: (a) predict/prevent possibly hazardous behavior; (b) monitor the adherence to behavioral constraints defined by a caregiver; (c) provide simple dialogue services with the user. The system was partially re-created in the RoboCup@Home domestic environment during the RoboCup 2006 competition in Bremen\(^3\) where it was awarded third prize.

The RDE is a collection of service-providing components of various nature. Sensors contribute to building a symbolic representation of the current state of the environment and of the assisted person, and scheduling-based temporal reasoning allows the system to reason upon the evolution of the state and infer future possible contingencies. Based on this information, agents infer actions to be performed in the environment through distributed coordination. These actions are carried out 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, MAC is an important aspect of the RDE scenario. The following paragraphs briefly describe the fundamental components of the multi-agent system, while the rest of the article is dedicated to the description of the coordination mechanism, which occurs in the RDE’s current configuration by means of ADOPT-N.

3.1. The RDE Coordination Problem

The RDE coordination problem is defined so as to demonstrate instances in which the coordinated operation of multiple household agents can provide complex support services for the elder user. Some examples of such instances 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 if necessary sounds an alarm.

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.

\(^3\)Competition homepage: http://www.ai.rug.nl/robocupathome/.
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”).

The objective of the above scenarios is to show how a collection of service-providing and very diverse agents (namely, in our specific case, artificial reasoners, robots and smart sensors) can be integrated into one functionally coherent system which provides more added value than the sum of its parts. The type of elementary services deployed in the RDE mirrors the domotic components that will be available on the market in the near future. The integrated environment as a whole, i.e., the functionally coherent system demonstrated in the exhibit, represents a proof of concept of the potential benefits to independence that can be obtained with the domotic technology of the future.

3.2. Components of the Multi-Agent System

The RDE is equipped with the following agents:

- Two fixed stereo cameras providing a people localization and tracking (PLT) service, and a posture recognition (PR) service.
- 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.
- An activities of daily living (ADLs) 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 or the stereo-cameras to the PDA, and (4) stop the robot.

Because the description of the technology which provides the services in the RDE is outside the scope of this article, we here briefly describe the single components and the services they provide within the environment. The rest of this article is dedicated to the MAC issues which arise in the operational integration of the components.

The robotic mediator. While 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 other added-value services which require physical presence. Specifically, the robot is capable of carrying out topological path planning, and employs reactive navigation for obstacle avoidance and scan-matching for robust localization. The robot is also endowed with verbal user interaction skills: speech recognition is achieved with the Sonic speech recognition system,\(^4\) while speech synthesis occurs through Lucia, a talking head developed at ISTC-CNR-Padua (Cosi et al. 2003). Overall, the robotic mediator is composed of two agents: a user-interaction agent (UI), whose services include speech synthesis and recognition, as well as a simple dialogue manager; a mobility agent (Robot), which is responsible for managing all tasks involving path planning and obstacle avoidance.

Stereo-vision sensing. A Stereo-vision-based People Localization and Tracking service (PLT) provides the means to locate the assisted person. This environmental sensor was

\(^4\)See http://cslr.colorado.edu/beginweb/speech_recognition/sonic.html for details.
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. A height-map is used for planview segmentation and planview tracking, and a color-based person model keeps track of different people and distinguishes people from (moving) objects, e.g., the domestic robot. In addition, vision-based Posture Recognition (PR) can be cascaded to the PLT computation to provide further information on what the assisted person is doing. Further details on the PLT and PR services is given in (Iocchi and Bolles 2005; Iocchi and Pellegrini 2005).

**Activity monitoring.** 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 (O-OSCAR (Cesta et al. 2005; Pecora et al. 2006b) to follow the occurrence of Activities of Daily Living (ADLs). This agent, henceforth called ADL monitor, employs temporal reasoning to provide schedule management services, such as propagation of general precedence relations among activities and activity insertion and removal. These services are employed to assess the adherence of the daily activities of the assisted person (as they are perceived by the sensory components) to a set of pre-defined behavioral constraints. The behavioral constraints 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., the robot suggests “how about having lunch?,” or warns “don’t take your medication on an empty stomach!”). Given the high level of uncertainty on the expected behavior of the assisted person, domestic supervision represents a novel benchmark for complex schedule monitoring.

In addition, the temporal reasoning services of the ADL monitor also contribute to activity recognition through an external constraint checking procedure. Specifically, temporal propagation within the ADL monitor is useful to refine the sensory information regarding posture and location of the person.

### 4. AGENTS, VARIABLES, AND CONSTRAINTS IN THE RDE DCOP

In the previous sections we have stated the general coordination problem which arises in what can be recognized as a wide category of AmI applications. Specifically, the task of providing a functional integration of intelligent artifacts requires a methodologically sound and extensible approach to coordination, which is itself a problem which requires intelligent reasoning. For this reason, we have proposed the DCOP approach to coordination, which is grounded on a distributed constraint-based representation of the services provided the intelligent artifacts. Such a specification consists in agents, variables and constraints which bind the values of the variables. As described intuitively in Figure 1, MAC is cast as a DCOP which is cooperatively reasoned upon by the agents according to the (distributed) ADOPT-N algorithm. In this section, we ground these concepts on the RDE. The following paragraphs describe in detail this reduction.

In the specific case of the RDE, the constraints among variables model a cost function which reflects the desiderata of system behavior. Specifically, the domains of variables in $V_{\text{out}}$ model service invocations (i.e., what the system can provide), while those of the input variables ($V_{\text{in}}$) model 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. Figure 2 shows the agents, variables and constraints used to model the RDE’s components.

The variables depicted in shaded nodes represent input for the decision process, i.e., $V_{\text{in}}$, while those in clear nodes indicate instructions for controlling the enactment of the
services provided by the RDE ($V_{out}$). The figure also shows the domains of the variables. We dedicate the following sub-sections to the description of the fundamental elements of the DCOP. Specifically, we describe the variables and their domains, how input variables are assigned values which reflect sensory information, and finally two of the constraints depicted in the DCOP, namely those constraints which are responsible for robot mobility on one hand, and activity recognition on the other.

4.1. Semantics of the Variables

**PRState.** The values represent the postures which are recognizable by the PR service: SITTING, STANDING, and LAYING. The service does not take into account any uncertainty on the recognized posture, i.e., its output is always one of the above postures. We therefore cannot model an “unknown” value in the domain.

**PLTState.** This variable carries information on the location of the assisted person in the environment. The PLT service is capable of providing this information as 3D coordinates in space or in terms of pre-specified locations. We choose to employ the latter, higher level
information in the coordination schema, because there is no need (as far as coordination
goes) to reason upon the exact location of the person. Therefore, the values of the PLTState
variable are all the locations in which the assisted person can be in which are meaningful
with respect to the overall coordination schema. These values clearly depend not only on
the specific domestic environment, but also on which locations are meaningful. As we will
see shortly, PRState and PLTState are employed to deduce which activity is currently being
performed. In order to simplify the examples here, we assume the domain of this variable
is: BATHROOM, BEDROOM, KITCHEN, LIVINGROOM, COUCH. Unlike PRState, the PLT service
does take into account uncertainty, thus we also include in the domain the value UNKNOWN.

**Activity.** The Activity variable is an output of the coordination process. It indicates the current
activity in which the assisted person is engaged. Its value depends on the those of PLTState
and PRState through the ternary constraint connecting them (which we describe below).
Its value is the input for the ADL monitoring service, which maintains an updated temporal
representation of the activities performed by the assisted person. Its values reflect the possible
activities in which the assisted person can be engaged. As in the case of PLTState, we are
only interested in those activities which are meaningful for the ADL monitor. The domain
thus depends on the specific schedule that is being monitored: in this article we assume that
the meaningful activities are: ASPIRIN (taking an aspirin), BREAKFAST, DINNER, LUNCH, and
NAP. In addition, activity can take on the value EMERGENCY. This value is not relevant for the
ADL monitor, rather it is employed to trigger an emergency reaction of the robotic mediator
(explained later).

**ConstraintViolation.** Given the information carried in the activity variable, the ADL mon-
itor is responsible for propagating this information and assessing whether some behavioral
requirement which is being monitored has been violated. For example, if the assisted person
takes her aspirin too early with respect to lunch (while it is required that aspirin should be
taken at least ten minutes after lunch), then the ADL monitor will assess this situation as a
temporal constraint violation in its internal representation of the current schedule. When this
occurs, the situation must be flagged so as to provoke the coordination necessary to advise
the assisted person through the UI services. This flag is the ConstraintViolation variable,
whose values are thus YES and NO.

**Warning.** If the ADL monitor has assessed a violation, the two agents which constitute the
robotic mediator (UI and Robot) need to coordinate to relay a warning or suggestion to
the assisted person. Specifically, the verbalization of the corresponding message (which is
dynamically generated by the UI agent and delivered through the speech synthesis service) is
triggered by the Warning variable, whose values are YES (a warning must be verbalized) and
NO. Notice that the value of Warning must “follow” the value of ConstraintViolation. This is
ensured by an “all-equal” binary constraint between the two variables. ConstraintViolation
and Warning thus represent knowledge which needs to be directly transferred from the ADL
monitor application to the UI application. Notice that if all variables were bound by such
a constraint, there would be no need for coordination, as the actuator applications would
themselves infer what to do based on the input of the sensory applications. This is not the
case in the RDE, as the correspondence between what the system should do and the sensory
input is the result of a non-trivial distributed computation (e.g., the dependency of Activity
on sensory input).

**AskIfOkay.** This variable triggers a special case of user interaction. Specifically, the con-
straints in the problem are modeled so that AskIfOkay is YES when the UI agent must ask
the assisted person if she needs help. If no response is received, a contingency plan (which is handled internally by the UI agent) is enacted.

**GotoPlaceDestUI, GotoPlaceDestPDA, and GotoPlaceDestRobot.** These variables indicate the destination of the mobility task for the robot. Specifically, the first two are input variables which take on the value of the requested destination for the robot (which has been specified verbally by the assisted person in the case of GotoPlaceDestUI, or through the PDA in the case of GotoPlaceDestPDA). The GotoPlaceDestRobot variable is an output variable which depends on the other two variables. The domains of these variables are similar to that of PLTState, although the possible destinations of the robot do not necessarily need to match the places which are meaningful with respect to Activity deduction.

**PDAState.** The PDA offers additional services to facilitate interaction between the user and the robot as well as the environmental sensors. The possible values of the PDAState input variable reflect the possible requests the user can make through the PDA: REQUEST-VIDEO indicates that the user has requested a video stream (which is displayed on the PDA) from an environmental camera or from a camera (PLT agent) on board the robot (Robot agent). The PDA’s user interface offers the possibility to cycle through the different streams of the different cameras: when the next source is selected, the PDAState variable is set to SWITCH-VIDEO, and terminating the streaming application results in the STOP-VIDEO value, which induces the agent generating the video to close the stream. Also, the PDA allows to dispatch commands to the Robot: the robot can be stopped (value STOP), summoned (value COME-HERE), sent somewhere (PDAState is set to GOTO-PLACE and variable GotoPlaceDestPDA is set to the destination location). If the PDA user interface application is not running, then the PDAState variable is set to UNKNOWN, while the idle situation in no request is performed on the PDA’s user interface corresponds to value NULL.

**RobotCommand and GotoPlaceDestRobot.** This output variable represents commands for the robot. The state REACH-PERSON occurs when the robot must reach the assisted person. This is the case, for instance, when the robot should relay a warning or suggestion to the assisted person, or when the person summons the robot through the PDA. When this state is set, the Robot agent requests the person’s exact position in the environment from the PLT service (this point-to-point communication occurs outside the coordination mechanism). The robot may also be required to go to other places in the environment, which results in the RobotCommand value being GOTO-PLACE. This is the case for instance when the user verbally tells the robot to “go to the bedroom.” The possible destinations of the GOTO-PLACE command constitute the domain of variable GotoPlaceDestRobot.

**RobotState.** The input variable RobotState represents the current state of the robot. If the robot is done executing a command, its state is described by the value DONE; when the robot is idle, the state variable assumes the value INACTIVE; when the robot is path-planning or moving (the two processes are interleaved in the implementation of the mobility service), the robot’s state is COMPUTING; finally, if the robot has not managed to execute the last command successfully, the value of RobotState is FAILED.

### 4.2. Assignment of Input Variables

Before describing the constraints modeled to cast the coordination problem, we must first focus on how the input variables are assigned. As such, the values of the input variables are not decided by the coordinated solving process, rather their values correspond to the various “sensory” applications and are fixed during the solving process. Therefore, the only admissible value of a variable in $V_{in}$ is the value that reflects what is sensed by the corresponding application. For instance, if the PR service observes that the assisted person is sitting down, then the value of PRState must be fixed to SEATED during the entire solving process. One
A way to achieve this is to cast, at every solving iteration, a DCOP in which the domains of the input variables are limited to the observed state. Another option is to constrain the “choices” of the various input variables to select, during the solving iteration, the value which reflects the external information assessed by the sensors. This can be done by imposing a unary constraint which evaluates to $\infty$ for all assignments which do not match the sensed information. In order to achieve this, we exploit the constraint checking feature provided by ADOPT-N: each sensory agent $A_i$ is endowed with a constraint checking procedure $\text{comp}_{A_i}$ which evaluates to $\infty$ the assignment of its variable(s) to values which do not correspond to sensed values.

Figure 3 shows the computational procedure for variable PLTState. This allows us to maintain the DCOP static across all solving iterations, as sensory input is injected into unary constraint checking procedures which avoid the need to filter out domain values.

The two approaches were compared on preliminary versions of the RDE. The experimentation proved the two modeling strategies equivalent from the computational point of view. Given the feasibility of both approaches, the running instantiation of the coordination framework which is deployed in the lab was realized according to the latter approach.

The reason for employing the comp-agent based strategy is twofold. First, compiling the DCOP at every iteration also requires to communicate the relevant parts of the DCOP to every agent in the environment, the overhead of communication being summed to the time lost for compiling the DCOP. In addition to the cost of compilation and communication, this would also introduce an element of centralization in an approach which is purely distributed. The second reason for choosing the comp-agent approach is related to ease of implementation. These drawbacks tipped the scale in favor of maintaining a static DCOP formulation. Incidentally, working on the same DCOP across iterations also eases the exposition in the following sections.

4.3. Constraints

In the following, we refer to concurrent service enactments given a specific state of the environment and of the assisted person as states of the system. We call these states consistent or inconsistent depending on whether or not they represent a concurrent enactment of services which is desired.

The value functions which describe the constraints in the system describe a global cost function whose minima represent the desired system behavior. This cost function is defined so as to evaluate inconsistent states to a global cost of $\infty$. Dually, we model the constraints such that consistent states evaluate to a finite global cost. Consistent states establish a correspondence between observations from the sensors and the desired combination of behaviors of

---

5 Notice that this is not true in the general case, because reducing the size of domains can improve computational efficiency.
the services. In the simplest case, we can model the desired behavior of the RDE with crisp constraints, i.e., value functions evaluating to 0 or $\infty$. In a more refined model, we can employ finite-value constraints to model situations which are more acceptable than others. In this section, we describe how two of the RDE’s aspects are modeled as crisp constraints, namely the two $n$-ary constraints among the variables $V_1 = \{\text{Activity, PDAState, GotoPlaceDestUI, RobotCommand}\}$ and among the variables $V_2 = \{\text{PLTState, PRState, Activity}\}$. In the next section, we show how finite values can be used to refine the specification.

In the following, we describe constraints employing the positive notation, i.e., we model the situations of zero cost. In the actual formulation of the DCOP these value functions are complemented so as to describe assignment situations which entail high cost (as ADOPT-N’s objective function is cost minimization).

**Robot Mobility.** The constraint among the variables in $V_1$ describes the relation between the behavior of the robot and situations in which the assisted person needs and/or has requested the robot to perform a mobility task. These situations can arise as a consequence of an emergency or as a result of the user explicitly summoning or sending the robot somewhere through the UI or the PDA. In the former case, the Activity variable is set to EMERGENCY by the ADL monitor, while in the latter case the PDAState or the GotoPlaceDestUI variables are involved. One way to model these requirements is through three binary constraints stating that the cost of variable assignments which reflect desired situations is zero, e.g., the concurrent assignment of Activity = EMERGENCY and RobotCommand = REACH-PERSON should have cost $f_{\text{Activity,RobotCommand}} = 0$. These constraints are described by the tables below, containing the assignments of the three variable pairs which represent desired situations, and where $\star$ indicates that the variable can be assigned any value in its domain:

<table>
<thead>
<tr>
<th>Activity</th>
<th>RobotCommand</th>
<th>$f_{\text{Activity,RobotCommand}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMERGENCY</td>
<td>REACH-PERSON</td>
<td>0</td>
</tr>
<tr>
<td>PDAState</td>
<td>RobotCommand</td>
<td>$f_{\text{PDAState,RobotCommand}}$</td>
</tr>
<tr>
<td>COME-HERE</td>
<td>REACH-PERSON</td>
<td>0</td>
</tr>
<tr>
<td>GOTO-PLACE</td>
<td>GOTO-PLACE</td>
<td>0</td>
</tr>
<tr>
<td>GotoPlaceDestUI</td>
<td>RobotCommand</td>
<td>$f_{\text{GotoPlaceDestUI,RobotCommand}}$</td>
</tr>
<tr>
<td>$\star$</td>
<td>GOTO-PLACE</td>
<td>0</td>
</tr>
</tbody>
</table>

Notice that this specification alone does not define the behavior of the system in case of conflicting situations. For instance, supposing that the assisted person has told the robot to go to the kitchen but at the same time the Activity variable indicates that there is an emergency situation, we wish to guarantee that the robot reaches the person rather than going to the kitchen. In order to model such priorities, we clearly need to model dependencies among more than pairs of variables, and doing so with binary constraints requires to model auxiliary variables. It is much more intuitive to consider the four value assignments together, as they are indeed mutually related. To this end, we model the following quaternary constraint:

<table>
<thead>
<tr>
<th>Activity</th>
<th>PDAState</th>
<th>GotoPlaceDestUI</th>
<th>RobotCommand</th>
<th>$f_{V_1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMERGENCY</td>
<td>$\star$</td>
<td>$\star$</td>
<td>REACH-PERSON</td>
<td>0</td>
</tr>
<tr>
<td>$\neq$ EMERGENCY</td>
<td>COME-HERE</td>
<td>$\star$</td>
<td>REACH-PERSON</td>
<td>0</td>
</tr>
<tr>
<td>$\neq$ EMERGENCY</td>
<td>GOTO-PLACE</td>
<td>$\star$</td>
<td>GOTO-PLACE</td>
<td>0</td>
</tr>
<tr>
<td>$\neq$ EMERGENCY</td>
<td>NULL</td>
<td>$\neq$ UNKNOWN</td>
<td>GOTO-PLACE</td>
<td>0</td>
</tr>
<tr>
<td>$\neq$ EMERGENCY</td>
<td>$\neq$ GOTO-PLACE,</td>
<td>$\neq$ COME-HERE,</td>
<td>UNKNOWN</td>
<td>NONE</td>
</tr>
</tbody>
</table>
Activity Recognition. The constraint which binds the input variables PLTState and PRState to the dependant variable Activity is responsible for assessing which activity is currently being performed by the assisted person, or if the assisted person is in an emergency situation. A simple way of performing activity assessment is to assume that activities are recognized by the combination of position and posture information. For instance, assume that the activities which are monitored by the ADL service are COOKING, LUNCH, MEDICATION, and NAP, and that the locations in the home (all of which the PLT service can recognize) are BEDROOM, BATHROOM, and KITCHEN. If the assisted person is seated at the kitchen table we may deduce that she is having lunch. Under similar assumptions, we model the following constraint:

<table>
<thead>
<tr>
<th>PRState</th>
<th>PLTState</th>
<th>Activity</th>
<th>( f_{v} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>SITTING</td>
<td>KITCHEN</td>
<td>LUNCH</td>
<td>0</td>
</tr>
<tr>
<td>STANDING</td>
<td>KITCHEN</td>
<td>COOKING</td>
<td>0</td>
</tr>
<tr>
<td>LAYING</td>
<td>KITCHEN</td>
<td>EMERGENCY</td>
<td>0</td>
</tr>
<tr>
<td>STANDING</td>
<td>BATHROOM</td>
<td>MEDICATION</td>
<td>0</td>
</tr>
<tr>
<td>LAYING</td>
<td>BATHROOM</td>
<td>EMERGENCY</td>
<td>0</td>
</tr>
<tr>
<td>LAYING</td>
<td>BEDROOM</td>
<td>NAP</td>
<td>0</td>
</tr>
<tr>
<td>SITTING</td>
<td>\ne KITCHEN</td>
<td>UNKNOWN</td>
<td>0</td>
</tr>
<tr>
<td>STANDING</td>
<td>\ne KITCHEN, UNKNOWN</td>
<td>UNKNOWN</td>
<td>0</td>
</tr>
</tbody>
</table>

We thus model as an emergency all “anomalous” situations such as laying in the bathroom, and we assume that other meaningful activities can be inferred from posture and position.

5. EXTENDING THE RDE DCOP

As mentioned, the DCR-based coordination scheme underlying service integration in the RDE facilitates the task extending the intelligent environment with further functionality and services. In this section, we set out to illustrate the operational details of how such an extension can be implemented. Specifically, we show how to leverage two features of the ADOPT-N-based integration framework to the advantage of expandability, namely (a) the constraint checking procedure capability of the agents, and (b) the possibility of modeling soft (i.e., non-crisp) constraints in the coordination problem specification. We illustrate these extensions by refining the activity recognition capability of the RDE.

5.1. Refining Activity Recognition through Constraint Checking

So far, we have assumed that activity recognition depends only on position and posture. This entails that we cannot model as distinct two activities which take place in the same posture-position configuration. For instance, supposing that dinner is also a meaningful activity for the ADL monitor, we would have to introduce a further distinguishing factor, such as the time of day. This could be modeled as an extra input variable which contributes the current time to the coordination process. The natural source of this information is the ADL monitor, which maintains the current execution of the daily schedule and can deduce the adherence to temporal constraints provided by the caregiver. Indeed, the ADL monitor manages a great deal more information about activities. Observe though that lunch and dinner can be distinguished by the ADL monitor also based on other criteria, for instance the fact that lunch has already occurred, therefore it is reasonable to interpret the current meal situation
as dinner. In general, if there are a number of candidate activities, the ADL monitor could propagate each activity and infer that the most probable current activity is the candidate which does not give rise to temporal constraint violations.

Algorithm 2

**Algorithm 2** Procedure \( \text{comp}_{\text{ADL}}(\text{assignment}(\text{Activity}, \text{val})) : \text{integer} \)

1. \( \text{ADLMonitor}.\text{setActivityStartTime}(\text{val}, \text{timeNow}) \)
2. \( \text{violatedTemporalConstraints} \leftarrow \text{ADLMonitor}.\text{getConstraintViolations()} \)
3. \( \text{ADLMonitor}.\text{rollback()} \)
4. \( \text{return} [\text{violatedTemporalConstraints}] \)

In order to take leverage the ADL monitor’s deductive capabilities within the coordination framework we therefore provide a constraint checking procedure on variable Activity, shown in algorithm 2 and Figure 4. The procedure evaluates the alternative activity recognition assignments and injects a cost which reflects the probability of the assignments being realistic according to the results of temporal propagation. For instance, suppose that breakfast, lunch, and dinner all occur at the kitchen table. We model the constraint \( f_{V_2} \) as follows:

<table>
<thead>
<tr>
<th>PRState</th>
<th>PLTState</th>
<th>Activity</th>
<th>( f_{V_2} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>SITTING</td>
<td>KITCHEN</td>
<td>BREAKFAST</td>
<td>0</td>
</tr>
<tr>
<td>SITTING</td>
<td>KITCHEN</td>
<td>LUNCH</td>
<td>0</td>
</tr>
<tr>
<td>SITTING</td>
<td>KITCHEN</td>
<td>DINNER</td>
<td>0</td>
</tr>
<tr>
<td>STANDING</td>
<td>KITCHEN</td>
<td>COOKING</td>
<td>0</td>
</tr>
<tr>
<td>LAYING</td>
<td>KITCHEN</td>
<td>EMERGENCY</td>
<td>0</td>
</tr>
<tr>
<td>STANDING</td>
<td>BATHROOM</td>
<td>MEDICATION</td>
<td>0</td>
</tr>
<tr>
<td>LAYING</td>
<td>BATHROOM</td>
<td>EMERGENCY</td>
<td>0</td>
</tr>
<tr>
<td>LAYING</td>
<td>BEDROOM</td>
<td>NAP</td>
<td>0</td>
</tr>
<tr>
<td>SITTING</td>
<td>( \neq ) KITCHEN</td>
<td>UNKNOWN</td>
<td>0</td>
</tr>
<tr>
<td>STANDING</td>
<td>( \neq ) KITCHEN, UNKNOWN</td>
<td>UNKNOWN</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>( \neq ) BATHROOM</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The above constraint allows the Activity variable to assume any of the three states BREAKFAST, LUNCH, and DINNER. These assignments are propagated internally by the ADL monitor which returns an evaluation reflecting the soundness of the assignment, under the assumption that the candidate action which minimizes temporal constraint violations is the most probable. Thus, the total cost of the assignment of variables PRState, PLTState, and Activity is equal to \( f_{V_2}(\text{PRState}, \text{PLTState}, \text{Activity}) + \text{comp}_{\text{ADL}}(\text{Activity}) \). Even assuming \( f_{V_2} \) is a crisp constraint (evaluating to \( \{0, \infty\} \)), this cost is refined via the non-crisp evaluation.
performed by the ADL’s constraint checking procedure, which essentially counts constraint violations entailed by the current assignment to return an evaluation of “situation realism” in the interval \([0, N_{\text{max}}] \subset \mathbb{N}\), where \(N_{\text{max}}\) is the maximum number of constraints that can be violated in the ADL monitor’s schedule.

Observe that this approach constitutes a “two-pass” deduction on activity recognition: first, the ternary constraint avoids assignments to the Activity variable which are in contrast with basic assumptions such as “the assisted person never has a meal in bed.” This leaves a number of candidate assignments with zero cost, such as which meal is being consumed, or which medication is being taken, and so on. A further cost evaluation is then performed by the constraint checking procedure of the ADL monitor, which evaluates the assignments in the light of the past and possible future evolutions of the daily schedule.

5.2. Refining Activity Recognition through Service Addition

A further enhancement to the activity recognition task can be envisaged as follows. Suppose the RDE is equipped with RFID\(^6\) tags on selected household objects, and the assisted person is given a wearable RFID reader (e.g., in the form of a bracelet). The reader is activated when the user manipulates objects equipped with the RFID tag, which can include the television’s remote control, the person’s toothbrush, the knobs on the stove, the fridge handle, etc. The information related to which object is being handled by the assisted person can offer a further refinement on the possible activity which is being performed by the person. The information deriving from RFID readings can be included within the activity recognition sub-problem through a constraint. The values of this constraint can be employed to model different levels of confidence. For instance, while manipulating the fridge handle or the stove provide strong evidence that the person is in the kitchen, the fact that the person is handling a portable object such as a remote control may or may not indicate that the person is actually watching television.

Different levels of confidence can be taken into account in the distributed activity recognition computation by directly modeling given confidence levels in the DCOP specification. Specifically, the DCOP is extended to take into account the new RFID service by adding an ADOPT-N agent which controls a variable named Object. We can assume, for instance, that RFID tags are placed on the TV remote, in proximity of the knobs controlling the kitchen stove, on the coffee machine, on the bread knife, inside a cushion in the bedroom, and on the assisted person’s pill box. The activation of these tags is associated with the person, respectively, watching television, having dinner (the person usually has a hot meal only for dinner), having breakfast (the person only has coffee in the morning), having lunch (which usually involves the preparation of a sandwich), taking a nap, or taking medication. Accordingly, we include in the DCOP specification a new constraint \(f_{V_3}\) which binds the Object variable with the Activity variable. Within this constraint, we can directly model the confidence levels we associate to each specific variable assignment:

<table>
<thead>
<tr>
<th>Object</th>
<th>Activity</th>
<th>(f_{V_3})</th>
</tr>
</thead>
<tbody>
<tr>
<td>REMOTE</td>
<td>TELEVISION</td>
<td>20</td>
</tr>
<tr>
<td>STOVE</td>
<td>DINNER</td>
<td>10</td>
</tr>
<tr>
<td>COFFEE</td>
<td>BREAKFAST</td>
<td>10</td>
</tr>
<tr>
<td>BREADKNIFE</td>
<td>LUNCH</td>
<td>30</td>
</tr>
<tr>
<td>CUSHION</td>
<td>NAP</td>
<td>20</td>
</tr>
<tr>
<td>PILLBOX</td>
<td>MEDICATION</td>
<td>0</td>
</tr>
<tr>
<td>NONE</td>
<td>★</td>
<td>0</td>
</tr>
</tbody>
</table>

\(^6\)Radio Frequency Identification.
The overall effect of the added constraint in the evaluation of the current activity is an additional cost $f_{V_3}(\cdot) = c_3$. This cost interacts positively with the cost $c_2$ contributed by the $f_{V_2}$ constraint plus the cost determined by the ADL constraint checking procedure introduced in the previous section. The cost associated to each configuration in $f_{V_3}$ models the extent to which the RFID service should affect the “candidate” activity as it is determined by these other two predictors: the higher the confidence in the RFID information, the lower $c_3$, thus the more likely it is for the (cost minimizing) DCOP algorithm to converge on an Activity value that reflects the RFID readings.

5.3. Further Enhancements to Activity Recognition

The evaluation involved in activity recognition can be further biased by an $n$-ary constraint checking procedure over variables Activity, PRState, and PLTState which takes into account other external information. The criteria for evaluating the three concurrent assignments may consist in learned data for instance, or data obtained by means of probabilistic inference as described in (Philipose et al. 2004). In the former case, we can employ a machine learning algorithm to learn a function $f_{V_1}$ which can be used to refine the cost model given by constraint $f_{V_2}$. In this setting, it is easy to envisage that training sessions are initiated by the elder user, who can decide to “show” the system (through the robotic mediator) what she is doing. Clearly, if training occurs only during the RDE setup phase, then it makes sense to set $f_{V_2} = f_{V_2}'$, i.e., to employ the learned function as the ternary constraint directly. On the other hand, it is reasonable to assume that training can be initiated spontaneously, because the assisted person’s habits may change in time. This can be taken into account by employing a constraint checking procedure on the assignments of the three variables which evaluates the realism of the assignment in the light of learned data. Overall, we would achieve three levels of assignment evaluation: (1) a “baseline” assignment evaluation in $\{0, \infty\}$ occurs through the ternary constraint $f_{V_2}$, which excludes the physically impossible situations; (2) the evaluation of the previous point is refined with the cost (in $[0, N]$) injected by the ADL monitor's temporal constraint based evaluation; (3) lastly, the assignment is evaluated by further criteria, such as learned data, which results in a further refinement of the cost function.

6. DISCUSSION

During the first part of the project, work in ROBOCARE was primarily concerned with 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 Bahadori et al. (2004). This allowed to draw some interesting conclusions on the usefulness of robots, smart sensors, and pro-active domestic monitoring in general (see, e.g., Cesta and Pecora 2005).

In the final part of the project, and in part toward 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 MAC problem.

As demonstrated by the ROBOCARE case study, the use of the $(n$-ary) constraint metaphor 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. In addition to being intuitive, the constraint-based formulation 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 at the only cost of including constraints which bind the behavior of the added service to those already present in the DCOP. In addition to the general suitability of the constraint-based approach, there are also a number of other points of discussion which arise from the case study we have presented. We discuss them separately in the following paragraphs.

The added value of n-ary constraints and constraint checking procedures. 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 coordination 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 mentioned earlier is a strong added value as it allows intuitive constraint specification while minimizing the impact of constraint complexity.

Casting the RDE coordination problem has also benefited from the use of external constraint checking procedures. Their use as an alternative input method for the sensory agents avoids the need to taint the purely distributed nature of the RDE with a periodic update of domains (and consequently constraint functions among neighboring agents). Nonetheless, the most important added value of the comp-agent approach is the ease with which external computation can be employed to bias the coordination problem. In the RDE, this is significant in the case of Activity deduction, where costs are refined on the basis of temporal propagation carried out internally by the ADL monitor. As in other domains where this can be proved formally (Pecora et al. 2007), it is unfeasible to completely model the criteria for evaluating activity assignments. In fact, the deduction of the current activity in the RDE must take into account further knowledge which is deducible only by means of an external procedure. In addition, we have shown how constraint checking can forebode other incremental refinements to the DCOP model, such as integrated learning procedures.

High-level DCOP specification. 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, 12 variables, and over 1100 cost function definitions (corresponding to 12 constraints). Indeed, the definition of a complete behavioral model for the RDE cannot be given without the use of a high level specification formalism.

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. 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.
The specification interface represents a first step towards a tool for facilitating the process of modeling service coordination in DCOP. Interesting future developments should focus on providing more sophisticated representational capabilities. An interesting related work in this direction is (Frisch et al. 2005), which focuses on modeling constraint specifications through refinement and transformation.

**On system evaluation.** The ROBOCARE project was aimed at providing a set of proof-of-concept prototypes for demonstrating the use of robotic, sensory, and problem-solving technology in the domestic cognitive assistance scenario. The development of the DCOP coordination scheme described in this article made the realization of an integrated prototype demonstrating how these capabilities can work together possible.

In line with its proof-of-concept nature, a full experimental evaluation of the integration schema is outside the scope of this article. There are at least two meaningful measures for benchmarking the overall performance of the coordination system: (1) the *application responsiveness*, i.e., the time \( t_{Ap_i} \) between a physical change in the environment and the recognition of the change in the input variable(s) of application \( Ap_i \); (2) the *coordination responsiveness*, i.e., the time \( t_{iter} \) it takes for a complete iteration of the \( \langle DCOP-resolution; \ read-variable; \ application-decision; \ write-variable \rangle \) cycle shown in algorithm 1. The overall responsiveness of the system is therefore represented by \( t_{iter} + \max_{Ap_i} (t_{Ap_i}) \).

The former measure is affected by the uncertainty connected to the time employed by services to update the symbolic information which they are responsible for deducing. In the specific case of the ADL Monitor, this equates to propagating sensor-derived information on the temporal representation of the behavioral constraints. This requires a combination of simple temporal propagation, re-scheduling, as well as other procedures (e.g., deciding which of the violated constraints are meaningful with respect to verbal warnings and suggestions). A qualitative evaluation shows that the application responsiveness of the ADL monitor is negligible if compared to the coordination responsiveness (which depends on network load and number of variables involved in the coordination). In practice, the coordination responsiveness can vary between about 1 and 10 seconds. The current overall responsiveness of the system (e.g., the delay between an activity occurring and the system activating a verbalized warning) varies typically between about 5 and 30 seconds. A global view of the performance of the system can be seen in the lab videos available at http://robocare.istc.cnr.it.

### 7. RELATED WORK

The smart home described in this article is strongly oriented toward activity management. Temporal constraint violations in the on-line execution of pre-defined daily schedules are used to trigger system behavior (e.g., taking a pill too late triggers the robot to issue a suggestion). Also, we have shown how temporal information can be used to refine activity recognition through an external constraint checking procedure which invokes a CSP-based scheduling service. The use of temporal reasoning technology in the context of domestic cognitive assistance has been addressed in various research projects in the past (Pineau et al. 2003; Haigh et al. 2003; Pollack 2005). The expandable and general-purpose nature of the DCOP-based coordination framework we have described would allow to include some of the solutions already presented in the literature, such as the disjunctive temporal problem (DTP) based activity monitoring proposed in Pollack et al. (2003), into a more complex service providing context such as the RDE.
Increasing attention is being given in the literature to systems that support and enhance human cognition capabilities. As an example, the CALO (Myers 2006) project has as its primary goal the development of cognitive systems which are able to reason, learn from experience, be told what to do, explain what they are doing, reflect on their experience, and respond robustly to contingencies. Other projects connected to different aspects of this research topic are CMRADAR (Modi et al. 2004), whose aim is to develop an intelligent scheduler assistant, and CAMEO (Rybski et al. 2004) whose main objective is to build a physical awareness system to be used by an agent-based electronic assistant. The common denominator of the above results is the substantial effort involved in realizing a tightly-coupled integration of domestic services. All these projects have required the orchestration of different intelligent software technologies and have highlighted an open issue in intelligent system design, namely that of coordinating distributed, heterogeneous and intelligent components to achieve an overall service-providing system. The DCR-based infrastructure described in this article is an attempt to address this problem within the context of a smart home, where distributed and inherently asynchronous services provide a natural application scenario for asynchronous optimization algorithms such as ADOPT and ADOPT-N. Indeed, the inspiration for employing DCOP as a reduction for solving the smart-home coordination problem stems from other experiences in which distributed computation and knowledge are key features of the coordination problem, such as those reported in Barrett (1999), Kitano et al. (1999), Tambe (1997), Chalupsky et al. (2001), Calder et al. (1993), and Frei and Faltings (1999).

8. CONCLUSIONS

This article presents an approach to implementing a solution for the smart home coordination problem. Our approach is based on casting the smart home service coordination problem as DCOP to employ state-of-the-art distributed algorithms for continuous coordination. The system we have achieved is explained in the context of the RDE case study, which also allows to appreciate the importance of the \( n \)-ary constraint representational capabilities and of the external constraint checking feature offered by ADOPT-N. Indeed, ADOPT-N was conceived to facilitate the use of DCOP as a coordination strategy for real-world applications (Pecora 2007).

The feasibility of the DCOP approach in ROBOCARE points to an interesting new direction of research within the DCOP community, as it puts into practice all the “good features” of contemporary DCOP frameworks, such as asynchronicity and optimality, as well as more specific features of ADOPT-N, namely \( n \)-ary constraint reasoning and constraint checking. Also, our experience in applying DCOP in the RDE makes a case for constraint-based distributed coordination in AmI applications. The reason is twofold: on one hand, the process of realizing a coordination infrastructure for embedded agents is reduced to the specification in DCOP of the criteria to which service behavior should adhere, and the rest (namely minimization of message exchange, agreement among agents, synchronization, and so on) is left to the DCOP algorithm; on the other hand, the system which we achieve is scalable in the sense that adding a new service is reduced to including the criteria according to which it should operate to the pre-existing DCOP specification.

Although we have described the DCOP approach through the ROBOCARE case study, we believe it retains a rather general applicability. Specifically, there are two key features which distinguish the DCOP approach from numerous other successful approaches to integrating diverse intelligent artifacts. First, DCOP-based modeling of the coordination problem is valid in all scenarios where distributed artifacts concurrently provide intelligent services which need to be coordinated: the issues related to asynchronous communication are seamlessly
taken into account by the DCOP algorithm, which is designed to minimize the need for communication while guaranteeing an “optimal” enactment of services. Second, modeling the dependencies between services in terms of cost functions rather than crisp constraints allows to express “soft” relations among services. This is often a requirement in AmI scenarios, because service enactment is typically required as a consequence of multiple conditions. In ROBOCARE, this feature is used primarily to leverage the information of multiple sources for activity recognition. It is worth noting that this feature is also useful in other contexts, such as for compactly modeling complex user preferences which are learned over time.

The integration of learning as well as other intelligent procedures which can contribute to making domotic environments “smarter” is a key goal of AmI development. Accommodating these procedures within a unifying coordination scheme can yield systems which exhibit intelligence also in the way behaviors are concurrently enacted. For this reason, a flexible coordination methodology such as DCOP-based continuous resolution can significantly contribute to AmI by providing a reference framework in which to instantiate the numerous recent advancements in AmI technology.

ACKNOWLEDGMENTS

This research has been partially supported by MIUR (Italian Ministry of Education, University and Research) under project ROBOCARE: “A MultiAgent System with Intelligent Fixed and Mobile Robotic Components,” L. 449/97 (http://robocare.istc.cnr.it).

REFERENCES


MYERS, K. 2006. CALO: Building an intelligent personal assistant. Invited Talk. The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference (AAAI-06), Boston, MA.


