Interacting with a Robot Ecology using Task Templates

Mathias Broxvall, Amy Loutfi, Alessandro Saffiotti
AASS Mobile Robotics Lab
Örebro University, Örebro, Sweden
{mbl,ali,asaffio}@aass.oru.se

***Abstract*— Robot ecologies provide a new paradigm for assistive, service, industrial, and entertainment robotics which is quickly gaining popularity. These ecologies contain a large number of robotic components pervasively embedded in the environment and interacting with each other. Human users of such systems need to be able to interface with both the system as a whole and, if desired, which each individual component. The humans should be able to transmit, in a natural way, commands that range from basic ones, such as "turn on the lights in the bedroom", to abstract ones, such as "bring me a cup of coffee". Human users may also need to interact with task execution, especially at decision points. In this paper, we introduce an approach to interface a human user to a specific type of robot ecology, called an ecology of Physically Embedded Intelligent Systems, or PEIS-Ecology. The ecology includes simple sensors and actuators and more complicated devices such as mobile robots. The proposed interface satisfies two requirements: 1) to easily and automatically generate component interfaces, and 2) to provide a simple mechanism by which to request and monitor the execution of tasks in the ecology.**

I. INTRODUCTION

There is a marked tendency today toward the embedding of many intelligent, networked robotic devices in our homes and offices. A particularly interesting case is the recent emergence of a paradigm in which many robotic devices, pervasively embedded in everyday environments, cooperate in the performance of possibly complex tasks. Instances of this paradigm include the so called network robot systems [11], intelligent space [7], sensor-actuator networks [2], ubiquitous robotics [6], and PEIS-Ecologies [14]. Common to these systems is the fact that the term "robotic device" is taken in a wide sense, including both mobile robots, static sensors or actuators, and automated home appliances. In this paper, we generically refer to a system of this type as an "ecology of robots".

The problem of having a human user interact with an ecology of robots is different from, and more complex than, the conventional human-robot interaction problem [15]. A robot ecology may provide a large number of possibilities to perform tasks of varying levels of complexity and abstraction. Tasks may range from simple ones, such as "turn on the lights in the bedroom", which change the state of a single component; to abstract ones, such as "bring me a cup of coffee", which may result in the coordinated operation of many, possibly complex functionalities like localizing the human and generating robot plans. Despite this inherent heterogeneity, the user should have an intuitive and uniform way to request any task at any moment. This implies that a robot ecology should include a mechanism for inputing relevant tasks to the system, and to present the user with the available options and status.

Many current approaches to these problems rely on the use of speech and dialogue to interact with the user [12], or on the use of sensors to observe the activities of the users and infer their possible goals, desires and intentions [16], [4], [9]. While these approaches are very promising in a long term perspective, they suffer from the brittleness of current speech and vision systems, and from our limited understanding of dialogue modeling and of intention elicitation.

In this paper, we propose a more practical approach to interacting with a robot ecology based on the notion of *task templates*. Task templates allow us to dynamically generate a list of relevant high level options from which the user can select tasks to be performed. Templates provide a simple but effective mechanism to encode the contextual conditions under which a task should be offered to the user. For instance, in the context in which the user sits on the sofa after dinner, the options to bring a drink, bring the phone, or play music should be presented. When the user leaves the house, the options to patrol the house or clean the floor should be offered instead. Template-based approaches have traditionally been used in expert systems and in reactive robot control. Recently, their use has also been introduced in the field of interface generation [10]. However, to the best of our knowledge, this is the first attempt to apply template-based approaches to the field of human-robot interaction.

In the rest of this paper, we describe our template-based approach in the context of a specific type of robot ecology called Ecology of Physically Embedded Intelligent System, or PEIS-Ecology. The next section provides a reminder of this concept. Following is a general schema of our approach to interface users with PEIS-Ecology including the main functional blocks and discussion of the presentation of information to the user. We end the paper by presenting a simple scenario that illustrates the above concepts.

II. THE PEIS-ECOLOGY APPROACH

The concept of PEIS-Ecology, originally proposed by Saffiotti and Broxvall [14], combines insights from the fields of ambient intelligence and autonomous robotics to generate a new approach to the inclusion of robotic technology into smart environments. In this approach, advanced robotic functionalities are not achieved through the development of extremely advanced robots, but rather through the coopera-

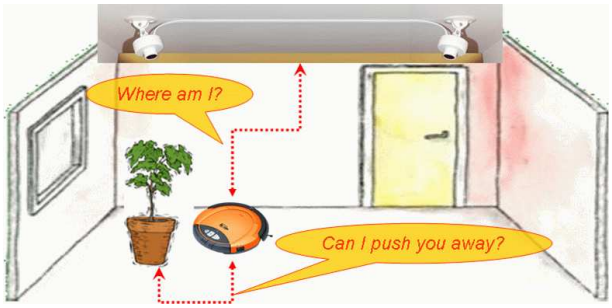

Fig. 1. A simple PEIS-Ecology consisting of a vacuum cleaner, an overhead tracking system, and a plant.

tion of many simple robotic components. The concept of a PEIS-Ecology builds upon the following ingredients:

First, any robot in the environment is abstracted by the *uniform notion* of a PEIS (Physically Embedded Intelligent System), which is any device incorporating some computational and communication resources, and possibly able to interact with the environment via sensors and/or actuators. A PEIS can be as simple as a toaster and as complex as a humanoid robot. In general, we define a PEIS to be a set of inter-connected software components, called PEIS-components, residing in one physical entity. Each component may include links to sensors and actuators, as well as input and output ports that connect it to other components in the same or another PEIS.

Second, all PEIS are connected by a *uniform communication model*, which allows the exchange of information among PEIS, and can cope with them joining and leaving the ecology dynamically.

Third, all PEIS can cooperate using a *uniform cooperation model*, based on the notion of linking functional components: each participating PEIS can use functionalities from other PEIS in the ecology in order to compensate or to complement its own. We define a PEIS-*Ecology* to be a collection of inter-connected PEIS, all embedded in the same physical environment.

As an illustration, consider the autonomous vacuum cleaner (PEIS) in Figure 1. By itself, the simple device can only use basic reactive cleaning strategies, because it does not have enough sensing and reasoning resources to assess its own position in the home. But suppose that the home is equipped with an overhead tracking system, itself another PEIS. Then, we can combine these two PEIS into a simple PEIS-Ecology, in which the tracking system provides a global localization functionality to the vacuum cleaner. Suppose then that the cleaner encounters a plant, and that the plant is equipped with a micro-PEIS (e.g., a mote) able to communicate its properties — e.g, size, humidity, and temperature and type of support. Then, the vacuum cleaner can use these properties to decide whether it can push the plant away and clean under it.

In our realization of a PEIS-Ecology, the PEIS rely on a distributed middleware to communicate and cooperate, called the PEIS-middleware. The PEIS-middleware implements a distributed tuple-space on a P2P network: PEIS exchange information by publishing tuples and subscribing to tuples, which are transparently distributed by the middleware. By stipulation, each PEIS also provides a set of standard tuples, e.g., to announce its physical appearance or the functionalities that it can provide. More details on the PEIS-middleware can be found in [1] and [13].

III. Overall Schema

The approach proposed in this paper considers the human as another PEIS in the PEIS-Ecology. As any other PEIS, the human can be made aware of which functionalities can be provided by the PEIS-Ecology, and decide to use any one of those. In the reverse direction, the PEIS-Ecology may rely on the human user to provide certain functionalities, like choosing between alternative courses of action. In practice, this intuitively appealing viewpoint is complicated by the following four aspects.

First, the set of tasks that the whole PEIS-Ecology could perform is, in general, very large. These tasks can be at very different levels of abstraction and complexity, ranging from basic tasks which are performed by a single PEIS, like dimming the lights in the bedroom; to tasks which involve the coordinated activity of several PEIS, like collecting all the expired items from the fridge and putting them in the garbage. In addition, the set of possible tasks is dynamic: PEIS may continuously be added to or removed from a PEIS-Ecology, and the capabilities of the PEIS-Ecology change accordingly.

Second, since the set of possible tasks is so large, the human user cannot be presented with the whole set at all times. Instead, an effective interface should filter the tasks to be presented to the user according to their relevance to the current context. As discussed in the Introduction, different tasks would be relevant in the context in which the user is relaxing after dinner, and in the one in which she is leaving the house.

Third, the human user should not only be given the possibility to start a task, but she may also be interactively involved in its execution. In PEIS-Ecology terms, the ecology may ask the user to provide some functionalities which are needed for the performance of the task. Typically, these will be cognitive decision making functionalities, like deciding which ones of the expired items in the fridge should be re-ordered. But sometimes the PEIS-Ecology might rely on the user to provide physical functionalities, e.g., to empty the dust-bag of the vacuum cleaner.

Fourth, the user should interact with the PEIS-Ecology via interfaces which all have the same “look and feel”, independently on the specific task and on the specific device which is being used to realize the interface. These interfaces should be dynamically generated to take into account the type of task, user, and device.

Considering these problems, we propose to organize the interface between the PEIS-Ecology and the human user according to the schema shown in Figure 2. The different tasks that the PEIS-Ecology can perform are encoded in a set of *Task Templates*. Each template describes how a given task

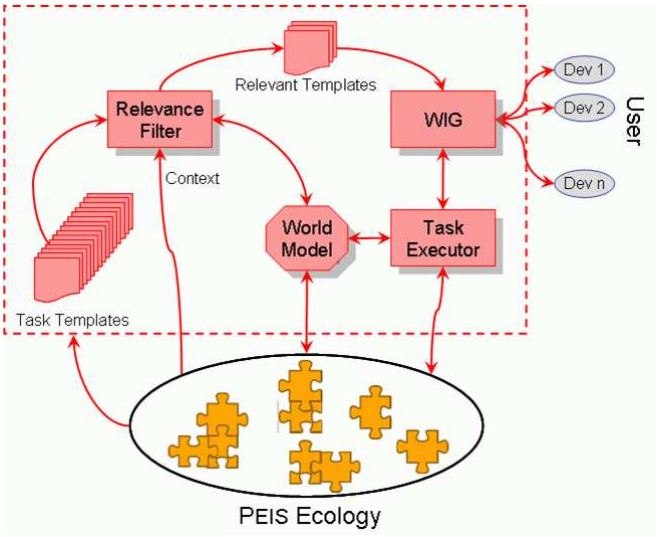

Fig. 2. Overall schema of our PEIS-Ecology user interface.

can be achieved, what are the contextual preconditions under which the task can be executed, and what is the information that should be presented to the user. The *Relevance Filter* selects relevant task templates by evaluating their preconditions against the current PEIS-Ecology context. The selected templates are presented to the user: if the user selects one of these tasks, the corresponding template is loaded into the *Task Executor*, which initiates and monitors the activity of the PEIS-Ecology according to the body of the template, and possibly interacts with the user. All interactions with the user are mediated by the *Web Interface GUI* (WIG), which dynamically generates interfaces specific to the given interaction that can be accessed by web-enabled devices.

The next two sections will discuss the above functional blocks in more detail. It is important to notice that these blocks do not need to reside in a single PEIS, e.g., in a centralized interface device. These blocks can be distributed across different PEIS in the PEIS-Ecology, and there might be multiple instances of them.

IV. TASK TEMPLATES

We now detail the central ingredients of the mechanism sketched above, intended to assess the situation and generate a list of relevant high level options from which the user can select tasks. For this purpose we use a template based approach which uses some ideas from classical expert systems. These templates serve three purposes:

- To decide which tasks are relevant in a given situation. For instance, a template for serving food should not appear as an option unless the user is at home and there is food available.
- Give a user friendly presentation of a specific task to be achieved. This is done by presenting preformatted messages suitable for the different interface modalities available, as detailed in the next section.
- Provide the necessary information for *how* a task is to be achieved by eg. providing a specific goal to a planning system, starting specific components and setting specific tuples.

The information contained in a task template corresponds closely to these three purposes and each template is an XML description containing three sub-parts: PRECONDITION, PRESENTATION and EFFECT.

These task templates and the information they contain reside in the distributed tuplespace and can originate from any components. Typically, the templates for providing specific tasks involving eg. a lamp PEIS is supposed to be delivered together with that PEIS but can also be delivered by, for instance, downloading the template into a completely different PEIS such as a television based multimedia system.

A. World model

Before we can explain the workings of the task template processing components, the world model checking component needs to be introduced. This is a component which uses the distributed tuplespace to find all execution constraints that have been registered by any components in the ecology. This is done by subscribing to all `world-constraint` tuples. These constraints can express, for instance, that the lights in the bedroom must be off or that a specific resource is occupied.

Whenever a component starts or is configured to use a resource it registers constraints with the model checker(s) by creating a tuple expressing that constraint in predicate logics with a predetermined shared semantics. Also, when a component is about to perform an action or setup a configuration it sends a request to the world model checker which can verify that the actions are legal or, if not, generate a description of which constraint it breaks. Typically there exists only one model checker in the ecology but multiple model checkers can also be used.

B. Filtering task templates

In order to provide a list of relevant tasks that can be achieved by various components available in the PEIS-ecology one or more filtering components are used. These task template filters work by subscribing to all available raw task templates and reading the PRECONDITION part of these tuples. This filtering constraint is built up from a simple XML based language that can perform:

- Basic boolean logical operations and conditionals.
- Primitive operations checking existence of and values of tuples, components and PEIS. Checking if a component can execute a given functionality in general and testing if it can be performed under the current situation in particular.
- Unify variables to satisfy the primitive operations.

By using unification the template can have a number of variables which are shared between the precondition, presentation and effects parts and which are used to match the template to any specific available component. When the filtering requires a component to test if its functionality can be executed that component will both perform its internal

```
<template>
  <precondition>
    <user var="?U"/>
    <or>
      <tuple> ?U location = livingroom </tuple>
      <tuple> ?U location = kitchen </tuple>
    </or>
    <peis var="C"/><peis var="?F"/><component var="?P"/>
    <class> ?C drinkable </class>
    <class> ?F refrigerator </class>
    <tuple> ?C location = ?F </tuple>
    <provides> ?P planning </provides>
    <tuple> ?C name = ?T </tuple>
    <test>
      <request><set-tuple> ?P goal.test = (at ?C ?U)
        </set-tuple></request>
      <monitor> ?P </monitor>
    </test>
  </precondition>
  <presentation user="?U">
    <id> (serve ?T ?F) </id>
    <priority> 1 </priority>
    <category> serve </category>
    <short> Would you like a cold ?T ? </short>
    <long> A robot will bring you a ?T from ?F. This
    operation may take a few minutes and you will be
    notified if it fails. </long>
  </presentation>
  <effect>
    <execute> <set-tuple> ?P goal = (at ?C ?U)
      </set-tuple></execute>
    <monitor> ?P </monitor>
  </effect>
</template>
```

Fig. 3. Example template for providing the user with drinks. For conciseness a slightly simplified XML representation and template is used.

checks as well as checks against the global world model component (if available). An example of a complete task template can be seen in Figure 3.

C. Presenting task templates

The PRESENTATION part of a task template contains an XML representation of a descriptive explanation of the templates. After determining which templates are *active* these templates are received by the modality specific interface which formats and uses the presentation part of the task to the user, as discussed in the next section.

The presentation part of the tuple can use the variables instantiated in the precondition part to generate one or more concrete tasks from one task template. Although the `id` part of a task is never presented to the user it serves an important function in filtering away multiple instances of identical tasks - only one task with a specific identifier is presented. As an example of this consider a scenario where we have two different planners and three drinkable items in the house, two bottles of water and one bottle of juice. The task template in Figure 3 would in this case be instantiated six times since we have two choices for the `?P` variable and three choices for the `?C` variable. However, only two unique identifiers, `(serve water fridge>)` and `(serve juice fridge)`, would exist and hence the user would be presented with only these two tasks.

D. The task executor

Once a user has selected one or more task templates for execution it is the role of the task executor to implement the instructions given in the EFFECT part of a template.

When a selected task is given to the template executor a task execution tuple is created which shows the state of the currently executing task. Next, each referenced component is launched as necessary, and goals are given to the corresponding individual components (for simple tasks) or to configurator and planning components (for complex tasks) as described in the EXECUTE part of the template. The configurators and planners serve an important role in the PEIS-Ecology and we refer the reader to [8], [3] for details.

The template executor continues to monitor the state of each component given in the MONITOR part and updates the task execution tuple accordingly. If any component fails, the failure as well as the description of the failure (as given by the component) is reported in the task execution tuple accessible to the interface component. Note that if a component tries to perform an action violating constraints set up by other components this will be noticed by the world model checker and a failure together with an explanation will be sent to the component. It is this failure that the task executor will see in this case.

V. WEB INTERFACE GUI

The Web Interface GUI (WIG) is a web based service that uses an XML protocol to interface the ecology and its components to an end user. The advantage of a web-based system is the facilitation to access the ecology from a number of heterogeneous devices and operating systems (mobile phones, media centers, laptops). From any WIG, users can monitor current tasks, request additional tasks to be performed via task templates, and change specific components. Since much of the interaction with the ecology should normally occur at the task level, the primary interface points of the WIG list both the current active tasks in the ecology, their respective status (or satisfaction) as well as the possible (sub)tasks that can be requested by a user.

The display of tasks and sub-tasks occur in the following manner:

- The WIG subscribes to task execution tuples and generates a list of currently executed tasks in the ecology. Adjacent to each task is the task status, also provided in this tuple as outlined in Section IV-D.
- New tasks to execute can be called from a list given by the filter component. This list of new tasks is ordered hierarchically and can be searched using special search strings using corresponding fields within the templates (e.g. category, involved components).
- Available tasks to be executed are displayed to the user using the presentation field defined in the template.

In special cases, it may be desired to interact directly with a specific component and change its status. This is also possible via the WIG not only for advanced users but also by novice users whose intervention may be necessary to assist high level tasks with the continuation of a failed execution. The display of components occur in the following manner:

- Each component contains an xml-schema either stored directly on the component or located in an online

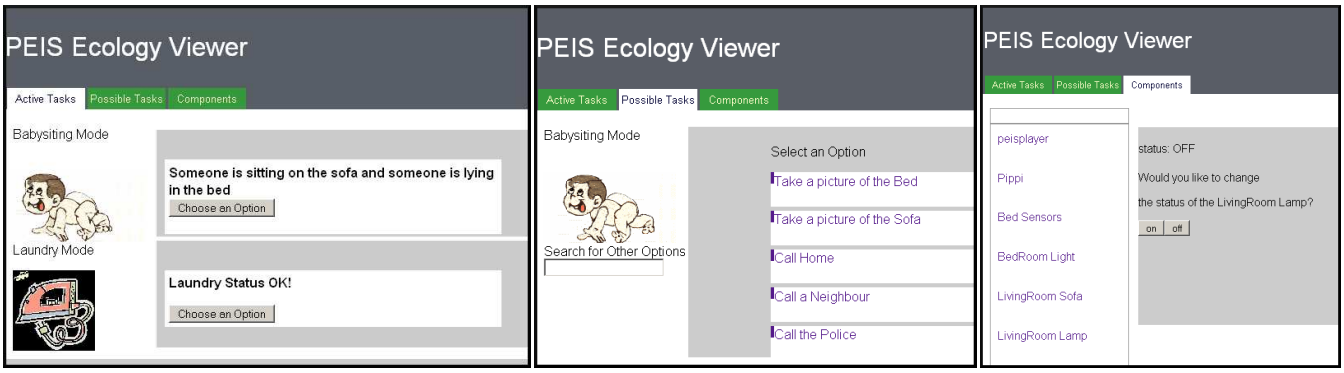

Fig. 4. *(Left)* A WIG has been started and displays the list of currently running tasks and their respective satisfaction. Here the satisfaction for the babysitting component is low. *(Center)* Options relevant to the this task are displayed. *(Right)* The interface also provides the option to access specific components directly. Here, a light switch interface is generated using the schema guidelines.

database. The schema defines the semantic for a component's status and possible field values. For example, a light switch has one field called "status" whose value can be "on/off", or camera mounted on the ceiling may contain an image data and position information.

- XML is generated for each active component in the ecology and its respective fields. The value of these fields and properties are populated using the current values of the components given from the tuplespace. Here, the xml-schema is used to verify the validity of the data format.
- Using the xml-schema, a HTML snippet is generated that is integrated with a WIG upon request. For example, the afore mentioned light switch will translate into an HTML snippet with two input buttons whose values are "on" and "off", and the camera will display the image file and perhaps have a text box displaying the position coordinates.
- The request to change the status of a component is made through the web-service which calls the respective functions in the tuplespace (e.g., setTuple) with the appropriate parameters. At the next refresh cycle, the XML is regenerated and updated with the new status information.

Figure 4 illustrates an example of a WIG that interacts both at the task and the component level.

VI. Scenario

Here we present a conceptual scenario that describes a intention of a PEIS ecology interface. Referring to Figure 2, both the PEIS ecology and a working shell of the WIG have been implemented, whereas the other components are simulated:

Par and Madelaine, parents to a 13 year old girl named Malin, are going out for the evening to attend a dinner and a movie. Being concerned over their daughter's well-being in their absence, she has received a number of instructions which she is expected to follow while at home (e.g. go to bed at 21:00, do not answer the door etc.). Before leaving home, Par and Madelaine activate a babysitting component which activates a number of sub-components at pre-defined times to assist in the babysitting task (e.g. front door is locked, exterior lights illuminate, turn off lights in Malin's bedroom at 21:00). Madelaine also activates an sms service which subscribes to the satisfaction level of the babysitting component through a task executor, such that she is able to receive warnings if a satisfaction value of a component decreases by a certain threshold.

At some point later in the evening, the home has detected an unusual event — that someone is sitting in the sofa while someone is lying in the bed (in other words, the pressure sensors in both furniture have been triggered). The occurrence of this event causes the satisfaction of the babysitting component to decrease. This decrease is detected by the sms service and an sms is sent to Madelaine's mobile informing her of these circumstances. Being in the middle of the film, Par and Madelaine discretely leave the room and while outside Madelaine uses the web service on her phone to connect to the ecology (Figure **??**). Upon doing so, the web service subscribes to the task execution tuples retrieving a list of all relevant tasks being performed and their corresponding satisfaction level. Seeing that the satisfaction of the babysitting component is low, the parents decide to begin an interface session with the ecology and enter the babysitting component in an attempt to troubleshoot the problem (Figure 4, left).

Once inside the babysitting component interface, a number of options are presented to the parents. These include: taking a picture of Malin's bed, taking a picture of the sofa, alerting a neighbour, alerting the police etc (Figure 4, center). Par and Madelaine decide that it is best to start by taking a picture of Malin's bed. They select that option, the command is sent to the task executor which then requests a plan to be determined and executed. In this instance, Pippi a nearby mobile robot is sent to the bedroom to take a picture. However, upon entering the room Pippi detects that the lighting is insufficient to take a picture. Normally, the ecology would automatically configure a light switch to be enabled, but in this case, the babysitting component insists that the lights in Malin's bedroom remain off after 21:00. The conflict thereby causes a violation and the lights are not switched on and a failure to execute the task and its cause is communicated on the interface. At this point, Par and Madelaine could go directly to the bedroom light component and activate the light switch, resuming the task of Pippi (Figure 4, right).

Fig. 5. (Left) Astrid, a mobile robot approaches the living room sofa to take a picture. (Right) The result of the picture taken on the sofa shows that the dog has triggered the pressure sensors.

Instead, Par and Madelaine having a suspicion as to the cause of the triggered pressure sensor in the living room, decide to take a picture of the living room sofa. Again a command is sent to the configurator via the task executor which determines that a camera located on Astrid, another mobile robot in the vicinity, is available to do the job. Unfortunately, there is still not enough illumination in the living room to take a picture. The ecology generates a new plan requiring the lights to be turned on. This configuration like all previous ones, is cross-checked against the world model. Here no conflicts occur, the lights are turned on in the room, a picture is taken and transmitted to Madelaine's mobile (Figure 5). The culprit is revealed to be the pet dog sitting on the sofa. Par and Madelaine feeling relieved, return to the film and enjoy the rest of the evening.

In this short scenario, the key points of the ecology interface are illustrated. Here we see that a mobile device is able to access the components in the ecology if necessary, request additional tasks to be performed and provide discrete warnings of the status of the ecology to a user. The task templates combined with the filter provide a seamless interchange between limiting the choice of the user to the relevant tasks at hand whilst still providing the user with the capability to impact the current state of the ecology and its components. One should note that only a shell of the WIG has been tested, and so far issues dealing with security and protection have not been taken into account.

VII. Conclusions

The idea of integrating robots and smart environments inside our homes and working places is quickly gaining popularity. Accordingly, there is a urgent need to develop viable interfaces for users of such systems. Current methods in human-robot interaction are not adequate for an environment as complex as an intelligent home that contains distributed sensing components, a large number of possible tasks to perform, and context restrictions given by the current situation. The main contribution of this work has been to highlight this fact and propose a possible solution to interacting with robot ecologies based on the use of templates.

The presented interface uses the current context and knowledge of the available resources (components) to determine and present the relevant tasks. It also allows the user to monitor the execution and be notified in case of failures, and to intervene with new tasks as necessary. The system can easily be expanded to cover new tasks by adding the corresponding template descriptors: these may be built-in inside new PEIS added to the ecology, or they may be downloaded from the Internet.These templates can be compared to procedural systems, where the possible tasks depend both on the available components and on the context [5].Furthermore, we have deliberately avoided the use of traditional text and dialogue generation systems which otherwise have received much attention by the scientific community. The major reason for this is the possibility to tailor the interaction for the PEIS-Ecology problem domain and the focus on *tasks* which should be performed.

Acknowledgment

The authors would like to express their gratitude to Alberto Garcia Sola for his implementation work on aspects of the interface. This work has been supported by: ETRI (Electronics and Telecommunications Research Institute, Korea) through the project "Embedded Component Technology and Standardization for URC(2004-2008)" and Vetenskapsrådet.

References

[1] M. Broxvall, M. Gritti, A. Saffiotti, B.S. Seo, and Y.J. Cho. PEIS ecology: Integrating robots into smart environments. In *Proc of the IEEE Int Conf on Robotics and Automation*, pages 212–218, Orlando, FL, 2006.

[2] F. Dressler. Self-organization in autonomous sensor/actuator networks. In *Proc of the 19th IEEE Int Conf on Architecture of Computing Systems*, 2006.

[3] M. Gritti, M. Broxvall, and A. Saffiotti. Reactive self-configuration of an ecology of robots. In *Proc of the ICRA Workshop on Networked Robot Systems*, Rome, Italy, 2007.

[4] S. Iba, C.J.J Paredis, and P.K. Khosla. Intention aware interactive multi-modal robot programming. In *Proc of the IEEE/RSJ Int Con on Intelligent Robots and Systems*, 2003.

[5] F. Ingrand, M.P Georgeff, and A.S Rao. An architecture for real-time reasoning and system control. *IEEE Expert*, 7(6):34–44, 1992.

[6] J.H. Kim, Y.D. Kim, and K.H. Lee. The third generation of robotics: Ubiquitous robot. In *Proc of the 2nd Int Conf on Autonomous Robots and Agents*, Palmerston North, New Zealand, 2004.

[7] J.H. Lee and H. Hashimoto. Intelligent space – concept and contents. *Advanced Robotics*, 16(3):265–280, 2002.

[8] R. Lundh, L. Karlsson, and A. Saffiotti. Plan-based configuration of an ecology of robots. In *Proc of the IEEE Int Conf on Robotics and Automation*, Rome, Italy, 2007.

[9] Y. Nakauchi, K. Noguchi, P. Somwong, T. Matsubara, and A. Namatame. Vivid room: human intention detection and activity support environment for ubiquitous autonomy. In *Proc of the IEEE/RSJ Int Con on Intelligent Robots and Systems*, 2003.

[10] J. Nichols, B.A. Myers, and K. Litwack. Improving automatic interface generation with smart templates. In *Proc of the 9th Int Conf on Intelligent User Interfaces*, pages 286–288, 2004.

[11] Network Robot Forum. www.scat.or.jp/nrf/English/.

[12] The Role of Dialogue in Human Robot Interaction. C.l. sidner and c. lee and n. lesh. In *Proc of the 1st Int Workshop on Language Understanding and Agents for Real World Interaction*, 2003.

[13] The PEIS ecology project. Official web site. www.aass.oru.se/˜peis/.

[14] A. Saffiotti and M. Broxvall. PEIS ecologies: Ambient intelligence meets autonomous robotics. In *Proc of the Int Conf on Smart Objects and Ambient Intelligence (sOc-EUSAI)*, pages 275–280, Grenoble, France, 2005.

[15] J. Scholtz. Theory and evaluation of human robot interactions. In *Proc of the 36th Hawaii Int Conf on System Sciences*, 2002.

[16] H. Takeda, N. Kobayashi, Y. Matsubara, and T. Nishida. Towards ubiquitous human-robot interaction. In *Proc of IJCAI-97 Workshop on Intelligent Multimodal Systems*, pages 1–8, 1997.