Multimodal Appliance Cooperation based on Explicit Goals: Concepts & Potentials

Thomas Heider and Thomas Kirste

Mobile Multimedia Information Systems Group
Rostock University, Germany
{th,tk}@informatik.uni-rostock.de

Abstract

Smart Environments are increasingly composed from individual components (smart appliances) that have to assemble themselves into a coherently acting ensemble. This requires software technologies that enable appliances to cooperate spontaneously on behalf of the users needs.

In this paper we will illustrate why a goal based approach is necessary and how explicit goals can be used to find system comprehensive strategies and how goals can be used as a benchmark to evaluate the system design.

1. Introduction

In the future we will have intelligent systems, where people are surrounded by intelligent intuitive interfaces that are embedded in all kinds of objects. These environments will be capable of recognising and responding to the presence of different individuals in a seamless, unobtrusive and often invisible way. Such kind of systems are described for example in the visions of “Ubiquitous Computing” [1] or “Ambient Intelligence” [2]. They share the belief of a smart, personal environment which characterizes a new paradigm for the interaction between a person and his everyday surroundings: Smart Environments are aware of the user and his surroundings and are equipped with computing and communication capabilities to make intelligent decisions in automated and situation-aware fashion.

However, Smart Environments will have to be composed from individual components (smart appliances) that have to assemble themselves into a coherently acting ensemble. This requires software technologies that enable appliances to cooperate spontaneously on behalf of the users needs.

A rather popular scenario illustrating this application area is the smart conference room (or smart living room, for consumer-oriented projects) that automatically adapts to the activities of its current occupants. Such a room might, for instance, automatically switch the projector to the current speakers presentation as she approaches the lectern, and subdue the room lights - turning them up again for the discussion.

Such a scenario doesn’t sound too difficult, it can readily be constructed from common hardware available today. Todays smart environments in the various research labs are usually built from devices and components whose functionality is known to the developer. So, all possible interactions between devices can be considered in advance and suitable adaptation strategies for coping with changing ensembles can be defined. When looking at the underlying software infrastructure, we see that the interaction between the different devices, the intelligence, has been carefully handcrafted by the software engineers, which have built this scenario. This means: significant changes of the ensemble require a manual modification of the smart environments control application.

This is obviously out of the question for real world applications, where people continuously buy new devices for embellishing their home. Enabling the devices to configure themselves into a coherently acting ensemble, requires more than setting up a control application in advance. Here, we need software infrastructures that allow a true self-organization of ad-hoc appliance ensembles, with the ability to afford non-trivial changes to the ensemble.

Things will become even more complicated when looking at the visions of “The Invisible Computer” from Don Norman [3] or “The Disappearing Computing” from the FET-IST [4]. This raises the following questions:

- How do you interact with smart things you are not aware of?
- How do you control devices you do not percieve?
- How to do this in a dynamic environment?

When dealing with these challenges indicated above, we can distinguish two different aspects here: Architectonic Integration and Operational Integration. Architectonic Integration refers to the integration of the device into the communication patterns of the ensemble. Operational Integration describes the aspect of making (new) functionality provided by the device (or emerging from the extended ensemble) available to the user. Obviously, both aspects eventually have to be accounted for by a Smart Environment Software Architecture. The aspects of architectonic integration we have discussed here [5]. In this paper we will look at potential concepts for addressing the challenges of Operational Integration.

We will show that to cope with the problems of *invisible computer* and *dynamic infrastructures* we have to rely on explicit goals to allow the smart ensemble to cooperate spontaneously on behalf of the users needs.

The remainder of this paper is structured as follows: In Section 2 we review how present-day Smart Environment projects have addressed the raised questions. In Section 3, we outline the concepts of our approach. In Section 4 we present different realizations of our approach. Section 5 will give a conclusion and an outlook.

2. Present-day Smart Environment Projects

A Smart Environments must identify the user’s intention and, in response, it needs to be able to generate multi-appliance strategies for a coherent ensemble reaction. It is now interesting to look at the means current projects employ for performing these obligations. Typical examples are for instance Microsoft’s EasyLiving [6] and MavHome from UTA [7].

The intelligent agents of MavHome for example predict the inhabitants next action in order to automate selected repetitive tasks for the inhabitant. This prediction is based only on previously-seen inhabitant interaction with various devices. In order to do this prediction the researcher of MavHome have saturated the house with sensors and characterize inhabitant-device interaction as a Markov chain of events and utilize an *Active-LeZi* algorithm to do the prediction. To learn strategies they use a reinforcement learning agent.

Table 1: *Smart Environment Projects*

Project	Intention Analysis	Strategy Planning?	Strategy Source
MavHome, UTA	Learning and Prediction, ALZ	Learned Procedures	Learned from User
The Adaptive House, Boulder	Learning and Prediction, NN	Learned Procedures	Learned from User
The Aware Home, GaTech	Context Widgets; MySQL	Rule Set (manually eng.)	System Designer
Easy Living, Microsoft	Geometry Model	Rule Set (manually eng.)	System Designer
AIRE, MIT Oxygen	Rule-based Programming	Rule Set (manually eng.)	System Designer
Intelligent Classroom, NWU	Plan Recognition	Rule Set (manually eng.)	System Designer

The EasyLiving Geometric Model (EZLGM) provides a general geometric service for ubiquitous computing, focusing on in-home or in-office tasks in which there are input/output, perception and computing devices supporting multiple users. EZLGM provides a mechanism for both determining the devices that can be used for a user interaction and aiding in the selection of appropriate devices. The EasyLiving system has behavior rules that cause things to happen automatically when certain relationships are satisfied in the world model.

Table 1 summarizes the intention analysis and strategy generation mechanisms for a variety of well-known Smart Environments projects. As can bee seen, there are two basic approaches to strategy generation: (a) learn from user – by observing the user's interaction with the infrastructure, as is done by MavHome (b) learn from system designer – by receiving a set of behavioral rules, as has been done for EasyLiving.

Unfortunately, both approaches are not viable any more, as soon as we look at dynamic ensembles.

Why the system designer can't provide the strategies:
Consider the example outlined in Figure 1, which shows the built-in infrastructure of two hypothetical conference rooms is displayed (greenish boxes). The room at left provides two beamers and a video crossbar, enabling a rather straightforward way for swapping two presentations. At right, the conference room just contains a single beamer; the second one has been presumably provided by an attendee. (In both sketches, the reddish boxes denote components that have been added dynamically.)

Clearly, both conference rooms require two significantly different strategies for realizing the user's goal of swapping two presentations. And, while in the built-in case one maybe could expect the room designer to provide a suitable macro, this is not realistic for the ad-hoc situation: No designer of a smart room can be expected to anticipate every possible ad-hoc extension of the built in infrastructure and to provide control strategies for every possible activity that could be performed with the thusly extended ensemble.Therefore, approaches such as EasyLiving are not viable for the case, where the environment's capabilities are provided by a dynamic ensemble.

Why the system can't learn the strategies from the user:
The approach taken by MavHome, to learn strategies from the user, is not an option either: If a substantial set of devices is *invisible* to the user, they can obviously not become part of a control strategy the user might develop. Therefore, a system can not learn from the user how and when to use these devices.

Since in dynamic ensembles neither system designer, nor system user have an overview over the complete ensemble and its potential, there is no human being that could provide strategies to this ensemble.

Either, the user has to be made aware of the available devices and their potential (pushing the responsibility back to the user), or the ensemble *itself* must become able to develop strategies on its own, based on the user's objectives. With respect to this, it should be noted that the systems developed in the above projects have *no* explicit notion of the user's objectives: They learn *procedures* from the user (or receive them from the system designer), but they have no concept of the *effect* of these procedures with respect to the user's objectives.
The consequences of the Ubicomp Visions are:

- *Disappearing Computer:* The system can't learn the strategies from the user!
- *Computers are everywhere - dynamic ensembles:* The system designer can't predict the ensemble and therefor can't provide predefined system strategies!
- *So:* Strategies have to be generated dynamically by the ensemble!

3. Goals

The consequense of disappearing computer and dynamic ensembles is that we need appliances that cooperate spontaneously and are able to generate strategies that accomplish the goal of the user. To make that possible we rely on goal based interaction.

3.1 Goal based Interaction

When people are using their technical infrastructure they have certain goals they want to achieve; a certain satisfaction they want to experience. This goal-based nature of users is agreed in the field of cognitive psychology. But todays engineered environments force us to think of interaction in terms of the individual functions that the numerous devices provide: functions such as "on", "off", "play", "record", etc.. When interacting with devices, we select, parameterize, and then execute functions these devices provide. Upon execution, they cause an effect: a broadcast is recorded on videotape, the light is turned brighter, and so on.

But then, a user is not really interested in the function he needs to execute on a device - it is rather the functions effect which is important.

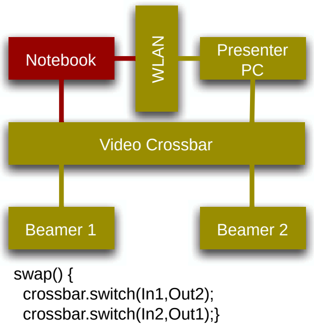
This observation immediately leads to the basic idea of goal-based interaction. Rather than requiring the user to invent a sequence of actions that will produce a desired effect (goal) based on the given devices and their capabilities, we should allow the user to specify just the goal (I want to see Star Wars now!) and have the ensemble fill in the sequence of actions leading to this goal. Goals allow services to be named by their semantics - i.e., by the effect they have on the users environment - thereby evading the problems of syntactical service addressing.

Goal-based interaction requires two functionalities: **Intention Analysis**[1], translating user interactions and context information into concrete *goals*, and **Strategy Planning**, which maps goals to (sequences of) device operations (see Figure 2).

3.2 Explicit Goals

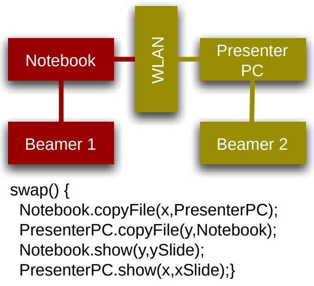
In order for a system to *autonomously* generate strategies for achieving certain goals, we need a mechanism to *explicitly* represent goals, which allows the system to reason about goals and different ways for achieving them. Specifically, we need a *declarative* representation of goals. The application area Smart

[1]In the project *EMBASSI*[8] we used for example speech recognition to translate user interaction into system goals.



```
swap() {
  crossbar.switch(In1,Out2);
  crossbar.switch(In2,Out1);}
```

(a) built-in



```
swap() {
  Notebook.copyFile(x,PresenterPC);
  PresenterPC.copyFile(y,Notebook);
  Notebook.show(y,ySlide);
  PresenterPC.show(x,xSlide);}
```

(b) ad-hoc

(c) Example room

Figure 1: Achieving the same effect with different ensembles

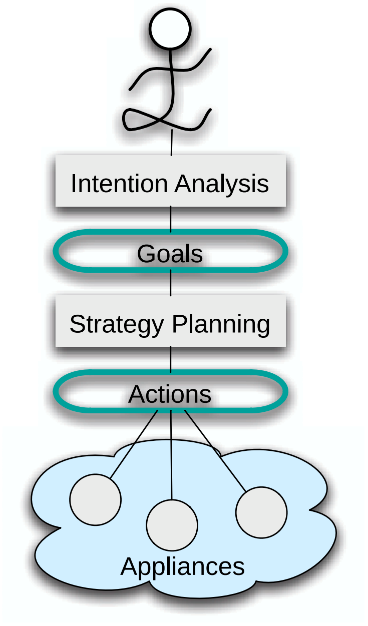


Figure 2: Principle of goal based interaction

Environment is concerned with achieving effects of interest to the user in his current environment – therefore, an explicit representation of user goals is basically given by a suitable state model for the environment. Goals are then represented by state vectors that are to be made true in the given environment.

Once the concept of explicit declarative goals is made available, it serves *two* purposes: (a) it allows goal-based interaction for the user (b) it is the foundation for the autonomous computation of control strategies by the ensemble.

So, we are no longer restricted to strategies that are either learned from the user or from the system designer – the ensemble itself is leveraged to unsupervised spontaneous cooperation.

Following, we give two examples for the explicit definition of goals and the accompanying strategy computation mechanism.

4. Exploiting Goals

4.1 Achieving Effects

Once an explicit declarative representation of the user goal is available, it becomes possible to exploit partial-order planning mechanisms. This requires to describe the operations provided by the available devices as *precondition/effect rules*, where the preconditions and effects are based on the environment state model. These rules then can be used by a planning system for deriving strategies for reaching user goals, which consider the capabilities of all currently available devices. The planning system receives the goal identified by the Intention Analysis. It must then find a strategy that changes the environment from its current state to the goal state. This can be understood as a classical planning problem:

- The goal is given as a set of positive and negative literals in the propositional calculus.
- The initial state of the world (resp. the state of the system and the environment-condition which is known to the system) is also expressed as a set of literals.
- The actions provided by the available devices ("operators") have to be characterized using a suitable definition language. It describes the action's relation to the environment: it contains a set of preconditions that must be true before the action can be executed and a set of changes, or effects, that the action will have on the world. Both the preconditions and effects can be positive or negative literals.

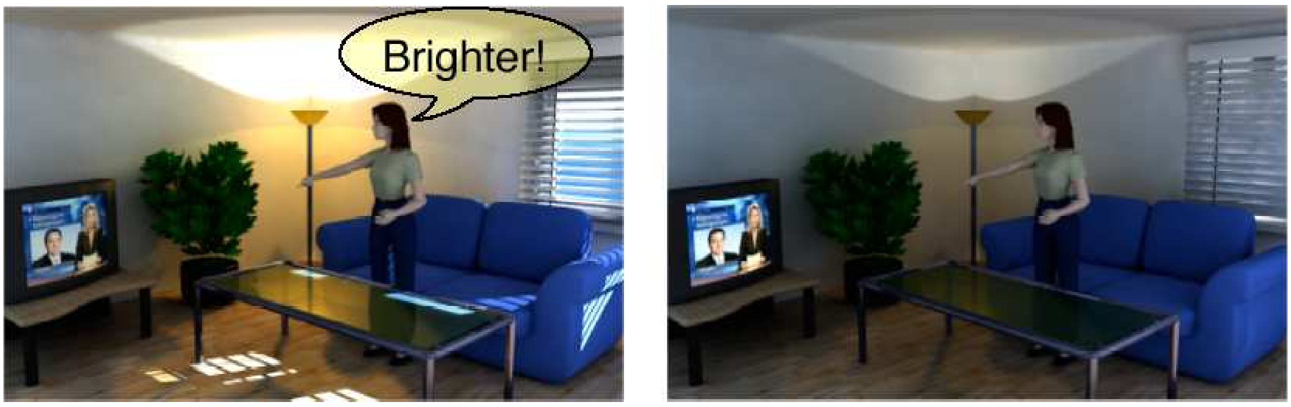


Figure 3: Goal-based ensemble control: Example

The critical aspect here is the expressive power of the model used for describing device operators, which needs to be strong enough to capture at least the operational semantics of todays consumer appliances.

4.1.1 *Concrete example*

As example, consider the situation outlined in Figure 3, left, where a user would like to increase the brightness of his TV set. Assuming the TV is already set to maximum brightness, the sensible reaction of the ensemble would be the one given at right: reduce ambient light. In order for an ad hoc ensemble to arrive at this conclusion, TV set, lamp, and shutter must provide a description of their capabilities, similar to the one given below[2]:

The Lamp's impact on the Environmentstate:
Action: *dim-down*(?x)
  Precond: *luminosity*(?x) = **high**
  Effect: *luminosity*(?x) = **low**

The Shutter's impact on the Environmentstate:
Action: *closeShutter*(?x)
  Precond: *open*(?x)
  Effect: ¬*open*(?x) ∧ *luminosity*(?x) = **low**

The TV's dependance of the Environmentstate:
Axiom: *ambBrightness-low*
  Context: ∀?x ∈ dom *luminosity* : *luminosity*(?x) = **low**
  Implies: *ambientBrightness* = **low**

Axiom: *increaseTVBrightness*(?x)
  Context: *brightness*(?x) = **max** ∧ *ambientBrightness* = **low**
  Implies: *brighter*(?x)

Then, based on a specific situation given by
Inits: (*brightness*(**TV**) = **max** ∧ *luminosity*(**Lamp**)= **high** ∧ *dimmable*(**Lamp**) ∧ *open*(**Shutter**))
a suitable plan for the Goal: *brighter*(**TV**) could then be computed as Plan: [*dim-down*(**Lamp**), *closeShutter*(**Shutter**)].

4.1.2 *Requirements for Planning Tools*

Providing a suitably expressive operator definition language is not a completely trivial requirement when looking at the host of features included in modern technical environments. But with expressiveness comes computational intractability – the more expressive a language is, the more computation is required to reason about sentences in that language. On the other hand, the solution capability of the planning system determines the space of the possible functionality of the device components. For example, the choice of discrete operators obviously excludes devices that provide continuous functions. So finding the right balance between expressiveness and computational tractability is very important for our application domain.

The experience from the modelling of our domain has shown that we need a planning environment that supports conditional effects and disjunction in the preconditions – this allows a compact representation of device operator sets. Furthermore it is mandatory to have universal quantification in the preconditions and the effects. This for instance allows to define operators that apply to an arbitrary number of objects – which is extremely important in an environment that is dynamically extensible.

For our first running prototype we used the UCPOP planner. But the experience has shown, that the expressiveness of this systems operator definition language was not well suited for modeling various problems of our application domain. Especially the feasibility to modeling temporal and continuous processes was missing, what is necessary to provide a reasonable time and resource management. Important is also to be able to representing mixed discrete/continuous domains. Amongst other approaches from different researchers the PDDL2.1 language [9] goes in this direction. PDDL2.1 was the input language for the 3rd International Planning Competition. This competition is a biennual challenge for the planning community, inviting planning systems to participate in a large scale evaluation. Consequently there are now quite a lot planning systems which can process domains modelled in PDDL. Moreover it gives an overview about the performance of the available systems. In the current implementation of our planning component we use the systems Metric-FF , LPG and MIPS. Our default system is Metric-FF. Should it find no solution for the given problem, automatically one of the other systems gets the task to find a solution. See [10] for details.

4.1.3 *The ontology*

The description of the component functions and capabilities as operators for the planning domain is essential. In order to support the interoperability of devices provided by different vendors, we need a shared understanding of the common environment domain they operate upon – a uniform ontology. Standardized environment ontology concepts such as *ambientBrightness*, *luminosity*, etc. make it possible to develop the components operator definitions independently from each other. Different vendors have to adhere to these ontology concepts as an explicit specification of the environment

[2]For sake of brevity, this capability definition has been very much simplified.

$$q_{\max} = \max_{\substack{ym \in YM \\ dm \in DM}} \left(\sum_{\substack{u \in User \\ d \in Document}} i(d,u) * \max_{y \in dm\ d} v(ym\ y, u) * p(y, ym\ y) \right) / \left(\sum_{\substack{u \in User \\ d \in Document}} i(d,u) \right)$$

Figure 4: Maximum Quality Function for a Multi-User Multi-Display Environment

aspects for their specific planning subdomains. If different components use common concepts for the same features, *e.g. ambientBrightness* for the capability of a lamp and of a venetian blind (by daylight), a cooperation is feasible. The vendor of a component characterizes its products in accordance with the specification of the ontology and the potentialities of the chosen problem-specification language. The planning operators will reasonably abstract from the device's concrete internal state and use a simplified state model that is tailored towards attaching the operators' environmental effects.

4.2 Resource Scheduling/Optimization

Another important application area is the definition of an optimal ensemble behavior regarding the mapping of (sub)-tasks to available resources. Here, a metric has to be defined that describes, how "good" a certain mapping ("schedule") of tasks to the available resources is and which allows to compare different schedules with respect to their optimality. So, these types of goals provide an explicit statement of a system designer's idea of optimal ensemble behavior[3]. This can be regarded as a theory of optimal ensemble behavior. The ensembles responsibility w.r.t. unsupervised spontaneous cooperation is then to jointly approximate this global optimum as good as possible. These implicit goals will be triggered by the situation through the intention analysis and must then be achieved by the appliance ensemble. In this section we present an example scenario of a multi-display environment.

In meeting scenarios, where many people come together, many documents have to be visible for the participants. The accessing of digital information in environments like meeting rooms is increasingly supported by new available technologies. For example the concept of projecting various content on different surfaces in a real environment including walls, floors and desks has been proposed recently. Pinhanez for example introduced an interesting method for a ubiquitous display called Everywhere Display [11] which uses a projector with a pan-tilt mirror for covering a large area with a single or a few projectors. What the available technologies are lacking of, is a concept of what information (document) should be displayed on which display ("Display Mapping").

In our scenario we have a meeting room with an ad-hoc ensemble of different projectors, steerable projectors, electric screens, notebooks and a number of users (see Figure 5 to get an idea). Now, it is interesting to provide an explicit representation of such an ensemble's optimal behavior with respect to the Display Mapping. This representation is given by Fig. 4: $q_{\max}$ provides a *global* and *generic* definition of an ensemble's optimal behavior. Once $q_{\max}$ is given, it allows *any* ensemble to optimize its Display Mapping.

This specific formulation of $q_{\max}$ is based on the concepts of *Visibility / Projectability* and *Importance*. Obviously, this definition of $q_{\max}$ represents the idea that all participants of the meeting can see the documents that are important for the respective participant in the best possible way.
The Visibility value results from the angle between the view direction of the user and the respective screens:

[3]Interestingly, in contrast to the user goals discussed in the previous section, scheduling objectives tend to be *predefined*, implicit goals.

Visibility $v : Surface \times User \rightarrow [0;1]$, *e.g.* simplified as:

$$v(s,u) = \max\left\{0, \frac{\langle \vec{n}_s, \vec{u} - \vec{s} \rangle}{\|\vec{u} - \vec{s}\|}\right\}$$

The Projectability value results from the angle between the projection direction and the respective (electric) screens:
Projectability $p : Display \times Surface \rightarrow [0;1]$
The Importance value depends on the agenda of the meeting and the role of the user in the different situations:
Importance $i : Document \times User \rightarrow [0;1]$
The goal is the optimization of the function $q_{\max}$ in Figure 4 which results in a Display Map, which associates each display (*e.g.* a steerable projector) with a display surface (*e.g.* a projector screen) and a Document Map, which associates each document with a display:
Display Maps: $YM = Display \rightarrow Surface$
Document Maps: $DM = Document \rightarrow \mathbb{P}Display$
Note that the definition of q_{max} in Fig. 4 is obviously not complete – for instance, it lacks a notion of "history", that would keep a document from confusingly hopping from display to display as the user slightly shifts position. However, the objective here has not been to develop the *optimal* definition for $q_{\max}$, but rather to argue that *explicit* global optimization goals are required in order to allow dynamic ensembles to cooperate. Only goal definitions that completely abstract from the concrete ensemble composition – such as $q_{\max}$ – enable *arbitrary* ensembles to optimize their behavior.

4.3 Experimental settings

The two pictures in Fig. 5 show the behavior of a typical ensemble controlled by $q_{\max}$. In the left image of Fig. 5 we have a scenario with one steerable projector, one screen and two users, whereby the right user gives a presentation. After adding a second beamer and two notebooks the system automatically calculates a remapping of the document display assignment (based on the maximum quality function of Fig. 4), which you can see in the right image.

We have tested this scenario in our Environment Simulation System, a visual simulation tool for Smart / Instrumented Environments that provides a simple rendering & physics simulation server, to which device and strategy planning agents may connect via sockets. The behavior of sensors, devices, simulated users, environment geometry, etc. can be controlled and changed dynamically.

Our test settings up to this point seems to justify the assumption that our intuitive formulation of a globally optimal ensemble behavior w.r.t. display mapping is heading in the right direction. Now it makes sense to look at the actual aspect: unsupervised spontaneous cooperation regarding joint approximation of this goal. Specifically, this goal-based approach now allows us to evaluate *different* distributed ad-hoc cooperation strategies w.r.t. their fidelity in approximating the global optimum. Currently we testing distributed versions of the well known algorithms "local beam search", "hill climbing" and "simulated annealing". We consider market-based strategies for distributed cooperation as a promising approach.

5. Conclusion and Outlook

This paper describes how we can deal with the problems of *invisible computer* and *dynamic infrastructures* if we rely on

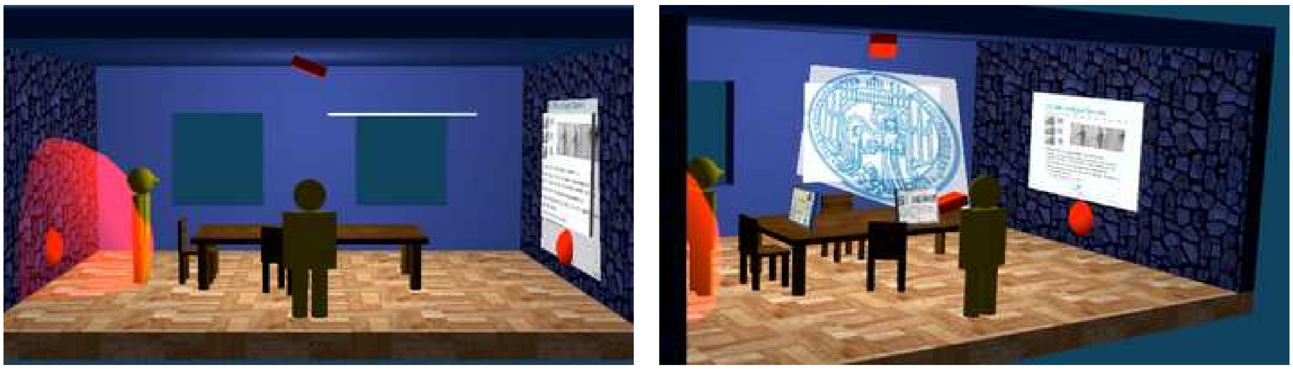


Figure 5: Environment Simulation System

explicit goals to allow the smart ensemble to cooperate spontaneously on behalf of the user's needs. To make goal based interaction possible we need to have an explicit State Model of the environment. This is the foundation for the formulation of explicitly uttered goals by the user and for the formulation of implicit persistent goals.

Currently, we are working on an unified environment ontology, which can be used to formulate both described kinds of goals.

Goal function as Benchmark: As different software infrastructures emerge, criteria are required by which the potential and efficiency of different solutions can be compared. Clearly, user trials - the ubiquitous evaluation strategy for pervasive computing applications - are not a viable approach in this case: the users of system software are application designers. Doing extensive user trial with highly trained experts is prohibitively expensive.

Therefore, it seems desirable to identify comparison criteria that can be evaluated at a formal level, using a standardized set of example problems - a set of Benchmarks.

It may be interesting to use explicit definitions of optimal behavior as a means for creating such benchmarks. So, as already outlined above, different approaches to computing display mapping could be compared with respect to their ability to approximate $q_{\max}$ (or a more refined "theory of an optimal display mapping").

To summarize: Smart environments promise to enable ubiquitous computing technology to provide a new level of assistance and support to the user in his daily activities. An ever growing proportion of the physical infrastructure of our everyday life will consist of smart appliances. In our opinion, an effective realization of smart environments therefore inherently requires to address the challenge of unsupervised spontaneous cooperation for ad-hoc ensembles of smart appliances.

We argue that a possible solution should be based on the fundamental concept of goal based interaction, because this enables an ad-hoc ensemble to generate strategies, instead to be dependent on predefined or learned strategies.

6. References

[1] M. Weiser. The computer for the 21st century. *Scientific American*, 3(265):94–104, 1991.

[2] E. Aarts. Ambient intelligence: A multimedia perspective. *IEEE Multimedia*, 11(1):12–19, 2004.

[3] D.A. Norman. The Invisible Computer. MIT Press, 1998.

[4] The Disappearing Computer Initiative. IST- Workprogramme 2003-2004-FET.

[5] T. Heider and T. Kirste. Architecture Considerations for Interoperable Multi-modal Assistent Systems. In *International Workshop on Design, Specification, and Verification*, pages 571–575, Rostock, Germany, 2002.

[6] B. Brummitt, B. Meyers, J. Krumm, A. Kern, and S. Shafer. Technologies for Intelligent Environments. In *Proc. Handheld and Ubiquitous Computing.*, Springer, 2000.

[7] D. Cook, M. Huber, K. Gopalratnam, and M. Youngblood. Learning to Control a Smart Home Environment. In *Innovative Applications of Artificial Intelligence*, 2003.

[8] T. Herfet, T. Kirste, and M. Schnaider. EMBASSI: multimodal assistance for infotainment and service infrastructures. *Computers & Graphics*, 25(4):581–592, 2001.

[9] M. Fox and D. Long. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. Technical report, University of Durham, UK, October 2001.

[10] T. Heider and T. Kirste. Supporting goal-based interaction with dynamic intelligent environments. In *Proceedings of the 15th European Conference on Artificial Intelligence*, pages 596–600, Lyon, France, July 2002.

[11] C. Pinhanez. The Everywhere Displays Projector: A Device to Create Ubiquitous Graphical Interfaces. In *Proc. of Ubiquitous Computing 2001 (Ubicomp'01)*, Atlanta, USA, September 2001.