

A DISTRIBUTED MECHANISM FOR DEVICE COOPERATION IN SMART ENVIRONMENTS

Christiane Reisse¹ and Thomas Kirste²

Abstract

A key research issue in pervasive computing is how to make an ensemble of fixed and mobile devices cooperate spontaneously to best assist a user without requiring him to manually control the devices. We are investigating a distributed approach that relies on self-organization. It adapts a mechanism for action selection by Maes to enable an ensemble of devices to perform complex tasks without a central controlling component.

1. Introduction and Related Work

Meeting rooms are prime examples of places containing numerous devices. A central research question in pervasive computing is how to make such devices cooperate so that they assist a user, enabling him to focus on his core activities rather than forcing him to manually connect his notebook to the projector etc. In smart ad-hoc environments, devices can enter and leave the ensemble anytime. Nevertheless, users expect the ensemble to exhibit sensible behavior without requiring a training phase. Furthermore, a central controlling component should be avoided as ensembles consisting entirely of resource-constrained devices may not comprehend a device capable of managing all the others. In past smart environments projects, the device cooperation strategy was often hand-crafted by the system designer [2] or learned from the user [3], which is not feasible for ad-hoc environments. The EMBASSI project [5] used partial order planning to derive action sequences that would fulfill user goals – a more flexible approach, but not completely distributed. In this paper, we present our research on a well-known action selection mechanism [6] to deal with the above-mentioned challenges.

2. A Smart Meeting Room Scenario

Based on the current world state and the user's goals, our system tries to find an action sequence that will fulfill the goals. We assume that the world state (provided by sensors) and user goals are described as propositions such as *Document1 shown on Canvas1*. Typically, machine learning approaches are used for inferring user goals from sensor data and a priori knowledge about the user's behavior. In our group, we experiment with inferring the goals of a team of users based on a dynamic Bayesian model of team behavior [4]. A simple scenario: In a room with multiple projectors and several canvasses, the speaker wants to show two documents simultaneously. His notebook sends the documents to two projectors which show them on the canvasses next to the speaker. Based on the sensor data *speaker opened Document1 and Document2 on Notebook1* and the goals *Document1 shown on Canvas1* and *Document2 shown on Canvas3*, the system thus generates the action sequence *Notebook1 sends Document1 to Projector1 – Notebook1 sends Document2 to Projector3 – Canvas1 is lowered – Canvas3 is lowered – Projector1 displays Document1 on Canvas1 – Projector3 displays*

¹Supported by a grant of the German National Research Foundation (DFG), Graduate School 1424 (MuSAMA).

²{christiane.reisse,thomas.kirste}@uni-rostock.de, University of Rostock, A.-Einstein-Str. 21, 18059 Rostock, Germany.

Document2 on Canvas3. Note that our system can also handle more dynamic scenarios with changing sensor data, such as tracking a speaker who moves through the room with a movable camera.

3. The Key Idea

For generating action sequences, we adapt an algorithm by Maes [6]. Our algorithm is based on so-called *compmodinsts* (CMIs) representing possible actions which are described in terms of preconditions and effects. CMIs exchange energy to determine which action will be executed next. To this end, CMIs that share preconditions and effects are connected via logical links: predecessor and successor links between identical preconditions and effects, confliker links between a precondition and an opposite effect. CMIs send energy to CMIs they are linked to: positive energy to predecessors and successors and negative energy to conflicters. Figure 1 depicts three CMIs with preconditions and effects described in PDDL [7]. A CMI also receives a certain amount of energy if one of its preconditions corresponds to a percept (sensor data or effects of previous actions of the ensemble) or if one of its effects corresponds to a user goal (see Figure 1). This energy distribution runs in cycles. At the end of each cycle, one module may be executed (see the right part of Algorithm 1). This way, an action sequence emerges. For a detailed description of action selection we refer the reader to [8].

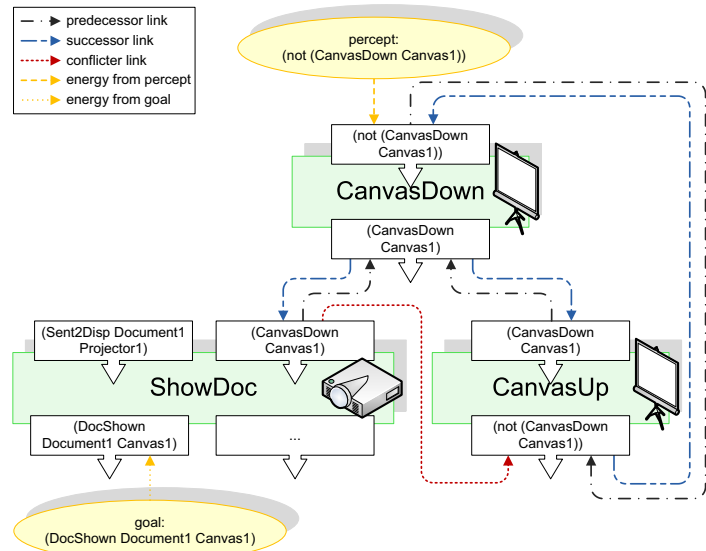


Figure 1. Three CMIs connected by links. Additionally, the flow of energy from a percept and a goal to a CMI is shown.

4. Extending the Basic Algorithm for Smart Ad-hoc Environments

Maes’ action selection process has been used in several domains [1]. Our research aims at investigating its suitability for smart ad-hoc environments and how it should be extended for our domain. We found that the following extensions are necessary for the algorithm to suit our needs:

- We distribute Maes’ algorithm among devices using software modules called *compmods* (CMs).
- To handle dynamic ensembles, variables in preconditions and effects should be allowed.
- CMs must build up a local world model at run-time based on communication.
- As the algorithm is distributed, CMs must handle synchronization themselves.

The following extensions are not mandatory, but allow to model preconditions and effects compactly:

- A small subset of first-order logic is allowed in effects.
- Persistent actions are supported.

Compmods: For Maes’ algorithm to work in a distributed environment, we assume that all devices are connected to a common network and have elementary computing capabilities. For every action a device can perform, there is a small software module called a *compmod* (CM) running on that device.

Algorithm 1 The algorithm executed by a CM: Construction and maintenance of the world model (left) and action selection (right) are carried out in parallel.

<pre> send message requesting context loop receive message if message is a request for context then broadcast own configuration else if message is information about devices/other CMs then update list of devices/templates of surrounding CMs update list of links create/destroy CMIs else if message is information about percepts/goals then update list of percepts/goals </pre>	<pre> while list of goals not empty do for each CMI do accumulate activation energy due to percepts, goals, and linked CMIs send fraction of own activation level to linked CMIs broadcast new activation level normalize activation level so overall activation level remains constant if CMI is executable and its activation level is above the activation threshold then if no other CMI has a higher activation level then execute CMI and broadcast effects else add random number in [0.0, 1.0] to activation level as a tie-break if no CMI was executed then lower activation threshold by 10 % else reset activation threshold </pre>
---	---

A canvas, for example, may host two CMs: *CanvasUp* and *CanvasDown*. CMs communicate via the network. We require each CM to describe its assigned action in terms of preconditions and effects. We call these action descriptions *templates*. They can be regarded as extended UPnP device descriptions. We assume them to be provided by the device vendors. Templates are precursors of CMIs: They may contain both variables and constants, whereas CMIs contain only constants.

Variables: As the structure of the device ensemble may change anytime, we need variables in preconditions and effects of templates. A variable is a placeholder for a device of a certain type. Templates are filled with names of devices in the ensemble by their CM at run-time: A CM creates an "instance" (*compmoinst*) of its template for every valid assignment of variables to device names. A CMI is thus a data structure inside a CM that represents a concrete instance of a possible action. Every CM takes part in the action selection process with all its CMIs, each representing a candidate action for execution. Therefore, each CMI is assigned its own activation level and its own links.³ The right part of Algorithm 1 shows the action selection process run by a CM for its CMIs.

Local world model: CMs introduce themselves to one another at run-time by broadcasting their templates. Every CM must constantly keep track of the current world state, user goals, devices in the ensemble, templates of surrounding CMs, and its CMIs (including links to predecessors, successors, and conflictors). This is carried out in parallel to action selection (see the left part of Algorithm 1).

Synchronization: CMs manage all transitions between the phases of the action selection process themselves. Any CM that has not received any message for a certain fixed timespan will announce the beginning of the next phase. The other CMs receive this message and proceed to the next phase.

First-order logic and persistent actions: In the "real world", there are laws like *If a projector displays a document on a canvas and another device sends it another document, the first one is not displayed anymore*. These are hard to express in propositional logic, making it difficult to maintain the correct world state. Of course, one could simply avoid such situations by introducing locks such as *Busy Projector1* and forbidding any device to send documents to the projector when it is busy. However, this requires extra conditions (the locks) and extra modules that must be executed to unlock e.g. *Projector1* first if another device wants to send it a new document. So we decided to handle situations

³Instantiation of links works accordingly: CMs maintain a list of templates of their *surrounding CMs* (predecessor, successor and conflictor CMs) and instantiate a link for every possible device combination.

like the one described above by allowing for certain first-order logic expressions in effects (see Figure 2 for an example) and by supporting persistent actions: Once activated during action selection, their effects remain true as long as their preconditions are fulfilled, which must be checked by the respective CM. For instance, if the CMI described in Figure 2 has become active, its CM constantly checks if any incoming percept states that Canvas1 has been raised or another document has been sent to Projector1. In that case, it becomes inactive and its CM broadcasts that the effects are not valid anymore.

```
(:action ShowDoc
:parameters (?Doc - Document)
:precondition (and (SentToDisp Document1 Projector1) (CanvasDown Canvas1))
:effect (and (DocShown Document1 Canvas1)
(forall (?Doc) (when (not (= ?Doc Document1)) (not (DocShown (?Doc Canvas1))))))
)
```

Figure 2. A description of a projector CMI in PDDL [7]. The effects state that Document1 is shown on Canvas1, and if there was another document on Canvas1 before, it is not visible anymore.

5. Conclusions and Future Work

In this paper, we have presented extensions to Maes' action selection mechanism for smart ad-hoc environments. To our knowledge, it is the first attempt at distributing Maes' algorithm over several devices, using it for device cooperation in smart environments. Our system is implemented as a distributed Java environment. We have tested several scenarios with up to 22 devices and 35 CMs. In most cases a sequence with the minimal number of actions was found. In rare cases the sequence was slightly longer. Our next steps will be to investigate what causes suboptimal action sequences and to deploy our system in our prototype smart environment to test it under real-world conditions.

References

- [1] Vincent Decugis and Jacques Ferber. An extension of Maes' Action Selection Mechanism for Animats. In *SAB'98*, pages 153–158, Cambridge, MA, USA, 1998. MIT Press.
- [2] Michael H. Coen et al. Meeting the Computational Needs of Intelligent Environments: The Metaglug System. In *Proceedings of MANSE'99*, Dublin, Ireland, 1999.
- [3] Sajal K. Das et al. The Role of Prediction Algorithms in the MavHome Smart Home Architecture. *IEEE Wireless Communications*, 9(6):77–84, December 2002.
- [4] Martin Giersich and Thomas Kirste. Effects of Agendas on Model-based Intention Inference of Cooperative Teams. In *Proceedings of CollaborateCom*, 2007.
- [5] Thomas Heider and Thomas Kirste. Supporting Goal-Based Interaction with Dynamic Intelligent Environments. In *Proceedings of ECAI'2002*, pages 596–600, 2002.
- [6] Pattie Maes. Situated Agents Can Have Goals. In Pattie Maes, editor, *Designing Autonomous Agents*, pages 49–70. MIT Press, 1990.
- [7] Drew McDermott. PDDL. <http://citeseer.ist.psu.edu/mcdermott97pddl.html>, 1998.
- [8] Christiane Reisse and Thomas Kirste. Distributed Device Cooperation in Smart Environments. Technical Report, University of Rostock, <http://www.informatik.uni-rostock.de/mmis/paper/reisse08-01.pdf>, 2008.