Using Cooperative Artefacts as Basis for Activity Recognition

Martin Strohbach, Gerd Kortuem, Hans-Werner Gellersen, Christian Kray

Computing Department, Lancaster University, Lancaster LA1 4YR
{strohbach, kortuem, hwg, kray}@comp.lancs.ac.uk

Abstract. Ambient intelligent applications require applications to recognise user activity calmly in the background, typically by instrumentation of environments. In contrast, we propose the concept of *Cooperative Artefacts* (CAs) to instrument single artefacts that cooperate with each other to acquire knowledge about their situation in the world. CAs do not rely on external infrastructure as they implement their architectural components, i.e. perceptual intelligence, domain knowledge and a rule-based inference engine, on embedded devices. We describe the design and implementation of the CA concept on an embedded systems platform and present a case study that demonstrates the potential of the CA approach for activity recognition. In the case study we track surface-based activity of users by augmenting a table and household goods.

1 Introduction

Ambient Intelligence research aims to create environments that support the needs of people with 'calm' technology. [4, 31]. To this effect many ambient intelligence systems make use of knowledge about activities occurring in the physical environment to adapt their behaviour. Hence, one of the central research challenges of ambient intelligence research is how such systems acquire, maintain, and use models of their changing environment. Approaches to address this challenge are generally based on instrumentation of devices, physical artefacts or entire environments. Specifically, instrumentation of otherwise non-computational artefacts has been shown to play an important role, e.g. for tracking of valuable goods [9, 18, 26], detecting safety critical situations [28], or supporting human memory [17]. In most augmented artefact applications artefacts are instrumented to support their identification [18, 21, 26] while perception, reasoning and decision-making is allocated in backend infrastructures [1, 6] or user devices [23, 27]. This approach, however, makes artefacts reliant on supporting infrastructure, and ties applications to instrumented environments.

In this paper we introduce the notion of *Cooperative Artefacts* (CAs) that combine sensing, perception, reasoning and communication. Such artefacts are able to cooperatively identify, track and interpret activities in dynamically changing environments without relying on external infrastructure. Cooperative artefacts model their situation on the basis of domain knowledge, observation of the world, and

sharing of knowledge with other artefacts. Thus, knowledge about the real world becomes integral with the artefact itself.

In section 2 we introduce the notion of Cooperative Artefacts and describe the structure and mechanisms by which CAs cooperate to acquire knowledge and reason about the world. In section 3 we describe an implementation of the CA concept on an embedded systems platform based on a case study to demonstrate the potential of our approach for activity recognition applications. In the case study an augmented table and household goods are used to track surface-based user activities. In section 4 we describe the interaction between involved artefacts. Finally, we discuss related work in section 5 and conclude our paper in section 6.

2 Cooperative Artefacts

Cooperative Artefacts (CAs) are self-contained physical entities that are able to autonomously observe their environment and reason about these observations, thus acquiring knowledge about their world. Most notably they cooperate by sharing their knowledge which allows them to acquire more knowledge collectively than each of them could acquire individually. It is a defining property of our approach that world knowledge associated with artefacts is stored and processed within the artefact itself. Although this structure is independent of any particular hardware platform, all components are intended to be implemented on low-powered embedded devices with inherent resource limitations. Figure 1 depicts the architecture of a Cooperative Artefact.

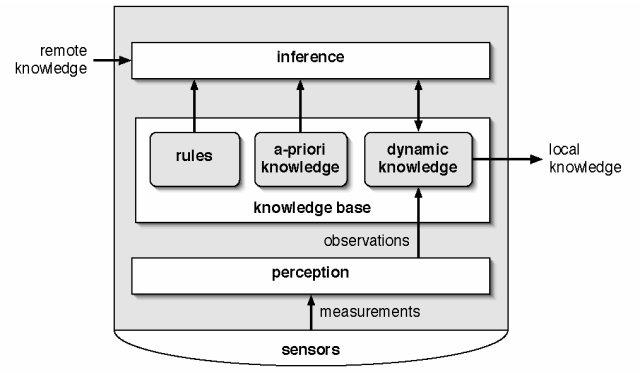

Fig. 1. Architecture of a Cooperative Artefact

- **Sensors.** Cooperative Artefacts include sensors which provide measurements of phenomena in the physical world.
- **Perception.** The perception component associates sensor measurements with meaning, producing observations that are meaningful in terms of the application domain.

- **Knowledge base.** The knowledge base stores the acquired knowledge of the artefact and externalises the artefact knowledge.
- **Inference.** The inference component processes the knowledge of an artefact as well as knowledge provided by other artefacts to infer further knowledge.

2.1 Structure of the Artefact Knowledge Base

The artefact knowledge base is structured into facts and rules. Facts are the foundation for any decision-making and action-taking within the artefact, and rules allow inferring further knowledge based on facts and other rules (table 1).

Table 1. Knowledge managed by a Cooperative Artefact.

Domain Knowledge	Domain knowledge built into the artefact, e.g. facts describing the physical nature of the artefact or general world knowledge.
Observational Knowledge	Knowledge describing the situation of an artefact in the world. It is based on facts that result from sensor-based observations.
Inferred Knowledge	Rules are used to infer further knowledge based on previously established facts, which may be based on domain knowledge, observation, previous inference, or knowledge made available by cooperating artefacts.

2.2 Cooperation between Artefacts

Activity recognition applications will rely on rich knowledge about users and their environment. It is therefore a desirable feature that artefacts cooperate to maximise the knowledge available about the physical world. Our model for cooperation is that artefacts share knowledge. More specifically, knowledge stored in an artefact's knowledge base is made available to other artefacts where they feed into the inference process generating additional knowledge. Effectively, the artefact knowledge bases taken together form a distributed knowledge base on which the inference processes in the individual artefacts can operate.

Although different artefacts may be able to observe the same physical phenomenon, the acquired knowledge is likely to be incomplete and different between observing artefacts. This is due to the nature and diversity of the used sensors and perception algorithms. However, the artefacts can use the distributed knowledge to exchange and reason about their knowledge. This leads to a synergetic effect: cooperating artefacts are able to acquire more knowledge collectively than each of them could acquire individually. This principle is illustrated in figure 2.

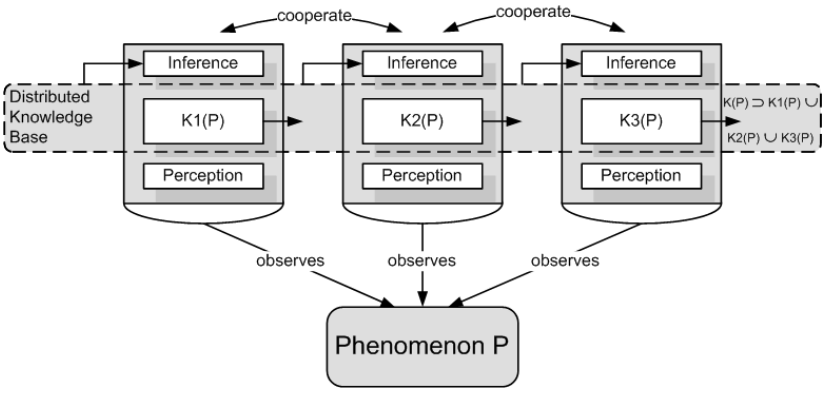

Fig. 2. Cooperation of artefacts is based on sharing of knowledge

3 CA Case Study: Tracking Surface-based User Activities

Recent research has identified activity centres as areas in domestic settings where human activity is focused on [8]. This research suggests that the identified activity centres, such as surfaces provided by kitchen tables, should be considered as prime sites for technological augmentation. Other research has shown that tagged artefacts may reveal valuable information about users' activity [17]. Based on this research we chose to track surface-based user activities as a case study for Cooperative Artefacts. The basic application idea is to track artefacts on and across tables to infer the user activity.

Fig. 3. CA demonstrator

The CA demonstrator (Fig. 3) was developed to show the potential of Cooperative Artefacts for activity recognition applications. Glasses and jugs were built as artefacts that are aware whether they are on surface or not. They cooperate with a load sensing table to infer their location on the table and, as a further synergetic effect of cooperation, they can infer their filling state. The picture in Fig. 3 shows the table,

glasses and jugs as it was demonstrated at SIGGRAPH [13]. This setup also included chairs to measure the weight distribution of people sitting on them. A graphical representation of the position of the glasses and the jug on the table was projected on a screen in order to display the recognised interactions of visitors with the artefacts. This section describes the implementation of the CA architecture on an embedded device platform.

3.1 Artefact Sensors and Perception

As a first step, the artefacts involved in the demonstrator require some basic form of intelligence, i.e. they need sensors and computing power to implement perception. Fort this purposes we used the DIY Smart-its platform, an easy to use and highly customizable hardware platform for wireless sensing applications [11]. The modular design of the DIY Smart-its allows using a range of different sensors by plugging add-on boards on a basic processing and communication board.

We used an add-on board to interface four industrial load cells that were put under each leg of the table. With this technology we can easily augment most tables in an unobtrusive way. We implemented a perception algorithm on the microcontroller of the DIY Smart-it to calculate events based on the weight changes on the table. Thus, we are able to recognise when an artefact has been put or removed from the table. In a second step, we use the load distribution on the four load cells to calculate the position of the artefact on the table surface. Additionally we obtain the weight of the artefact that has just been put or removed from the table. Further details about the implementation of context acquisition based on load sensing and its potential applications have been published in [24] and [25].

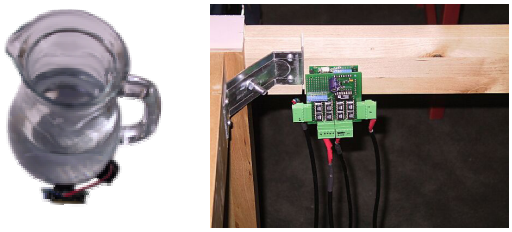

Fig. 4. Left: A mini Smart-its with battery attached to a water jug. The FSR sensor is mounted on the bottom of the jug. The glass was augmented in a similar way. Right: A DIY Smart-it with load add-on board attached to the frame of the table. The black cables connect to the four load cells.

A smaller version of the DIY Smart-its has been used to augment jugs and glasses with force sensitive resistance sensors (FSR sensors). Thus the perception algorithm of glasses and jugs can observe if the artefact is put down on a surface or lifted up. Figure 4 shows the physical augmentation of the artefacts.

Thus, each artefact is able to make individual observations about its world:

- The **load table** observes the position and weight of artefacts on its top but does not know about their identity, i.e. is it a jug, glass or something else.

- **Jugs and glasses** observe whether they have been put on a surface or lifted up, but they know nothing about their location.

3.2 Artefact Knowledge Bases

These observations are stored and managed in the knowledge bases embedded in the individual artefacts. They contain facts to represent the knowledge and rules to infer further knowledge. Each of the artefacts implements an inference engine similar to a simple Prolog interpreter that operates on these facts and rules expressed in a subset of Horn logic [14]. The inference engine uses backward-chaining with depth first search as inference algorithm. Compromises in terms of expressiveness and generality were necessary to facilitate the implementation on a micro-controller platform. The data structures for the predicate arguments provide information whether the argument refers to an external artefact which allows the inference engine to acquire knowledge from other artefacts using a query/reply protocol.

Table 2. Knowledge base of a load table

Domain Knowledge	`concurrent(<time>,<time>)`
Observational Knowledge	`location_and_weight(me, x, y, _, w, <added/removed>, <time>)`
Rules	

(R1)	`location_and_weight(me, X, Y, A, W, added, T2):-`
	`  location_and_weight(me, X, Y, _, W, added, T1),`
	`  location_and_weight(_, _, _, A, _, added, T2),`
	`  concurrent(T1,T2).`

Table 3. Knowledge base of a jug/glass

Domain Knowledge	`concurrent(<time>,<time>)`
	`above_weight_threshold(<weight>)`
	`below_weight_threshold(<weight>)`

Observational Knowledge	`location_and_weight(_, _, _, me, _, <added/removed>, <time>)`
Rules	

(R2)	`filling_state(me, full, T2):-`
	`  location_and_weight(TABLE, X, Y, _, W, EVENT1, T1),`
	`  location_and_weight(_, _, _, me, _, EVENT2, T2),`
	`  concurrent(T1, T2), above_weight_threshold(W), EVENT1 == EVENT2.`

(R3)	`filling_state(me, empty, T2):-`
	`  location_and_weight(TABLE, X, Y, _, W, EVENT1, T1),`
	`  location_and_weight(_, _, _, me, _, EVENT2, T2),`
	`  concurrent(T1, T2), below_weight_threshold(W), EVENT1 == EVENT2.`

Rules and some facts are specified by the developer. Other facts such as `location_and_weight(<table>, <x>, <y>, <artefact>, <weight>, <added/removed>, <time>)` model the observational knowledge acquired by the artefacts. This observation indicates the position and weight on a table on which an artefact has been added or removed at a certain time. Both kinds of artefacts, the load

table and glasses/jugs, model their observation with the same fact; however with different levels of information according to their perception capabilities as described in the previous subsection. Table 2 lists the knowledge base of an load table while Table 3 lists the knowledge base of an jug or glass. Lowercase letters are constants and model a specific value of an argument, while we use the special character "_" to indicate an unknown or irrelevant value of an argument. Uppercase letters indicate variables. The special constant `me` always refers to the artefact that stores the fact or rule. Arguments put in brackets are to be replaced with the concrete values for each observation. Rule R1 can be verbalised as follows:

R1: A table knows location, weight and identity of an artefact on its top if both, the table and the artefact, observe the event of putting the artefact on the table at approximately the same time.

This rule relies on synchronised time between artefacts as each observation is time-stamped to determine time relationships between two observations. `concurrent(<time>, <time>)` is semantically equivalent to the expression `|T1-T2|<time_th` with an appropriate threshold `time_th`. The table perception algorithm for the `location_and_weight` observation takes a few milliseconds longer than the algorithms of the glasses and jugs. Consequently we take T2, the time of the glass (or jug respectively) observation as a timestamp for the inferred location and weight. Rules R2 and R3 can be verbalised as follows:

R2 and R3: A glass or jug knows its filling state when it observes the same event at approximately the same time as the table.

Rules R2 and R3 are a by-product of our initial goal to obtain location information about artefacts that are put on the table. This is due to the fact that we measured the weight distribution to calculate the position of artefacts. These rules also make use of additional domain knowledge: `above_weight_threshold(<weight>)` and `below_weight_threshold(<weight>)` model for each individual artefact a weight threshold to determine if a glass or jug is full or empty.

4 Tracking User Activities with the CA Demonstrator

In this section we describe a set of actions as they have occurred during the demo setup. We use an initially empty table with which users interact by putting and removing a conventional bottle and the augmented glass on and from the table. The glass is initially empty and located on a conventional table. We perform the following actions:

Action 1. Put a conventional bottle in the middle of the table
Action 2. Put an empty, augmented glass on top left corner of the table
Action 3. Remove glass from table and fill it with water
Action 4. Put the glass back on the bottom right corner of the table

Initially the artefact knowledge bases only contain their respective domain knowledge entries (cf. Table 2 and Table 3). The knowledge base of the glass also reflects that it

is put on the conventional table. After putting the conventional bottle on the table the perception component of the table adds the corresponding observation to the knowledge base and we obtain the observations as detailed in Table 4.

Table 4. Observations after action 1

Glass	Table
`location_and_weight(_, _, _, me, _, added, 30)`	`location_and_weight(me, 25, 25, _, 400, added, 768)`

In order to detect the changes in the environment and update the screen display, a query for `location_and_weight` must be sent to the table. The query contains only one value, the table identifier to which the query should be sent. All other arguments contain variables representing the values we are interested in:

```
location_and_weight(table, X, Y, A, W, EVENT, TIME)
```

When the table receives this message it tries to unify the message with the entries in the knowledge base. The inference engine always finds the most specific answer and tries to evaluate rule R1. It checks the premises of the rule and unifies the table observation with the entry in the fact base. The external observation `location_and_weight(_, _, _, A, _, added, T2)` requires cooperation with an unknown artefact `A`. Therefore, the inference engine issues a broadcast query for this observation. The glass replies with its observation as detailed in table 3. However, rule R1 cannot be applied as the timestamps are not close in time. Therefore, the table replies with

```
location_and_weight(me, 25, 25, _, 400, added, 768)
```

not being able to provide information about the actual identity of the bottle. The visualization projected on the screen is now updated showing a question mark for the bottle.

After the second action, both the table and the glass add new observations to their knowledge bases as shown in Table 5.

Table 5. Observations after action 2

Glass	Table
	`location_and_weight(me, 25, 25, _, 400, added, 768)`
`location_and_weight(_, _, _, me, _, added, 2103)`	`location and weight(me, 25, 25, _, 131, added, 2105)`

Note, that the perception component of the glass always updates the knowledge base to reflect the latest observed state, i.e. there was an intermediate observation when the glass was lifted from the conventional table that is not displayed in the table. Queries to the table will now result in replies that provide information about the location of the glass:

```
location_and_weight(table, 25, 25, glass, 400, added, 2105)
```

This is a result from applying rule R2 which entails a broadcast query. This query is replied by the glass with the corresponding observation that was made at nearly the

same time. The identity of the glass can now be used to query the glass about its filling state:

```
filling_state(glass, FILLING_STATE, T2)
```

In order to answer this query the glass must cooperate with the table: rules R2 and R3 rely on the table's observations. Again, the table issues a broadcast query asking for the observation made by the table. The table replies with his observations and the glass replies with the conclusion of rule R3:

```
filling_state(glass, empty, 2103)
```

Again the changes in the environment have been detected and the display can be updated:

Fig. 5. Screenshot of the available knowledge after action 2. In the demonstrator question marks are used to represents unknown artefacts, their size is relative to their weight. Here the question mark represents the bottle.

After the third action the knowledge bases of both artefacts are updated. In this state queries to the table always return the location of the unknown artefact (the bottle) as the observation stored in the glass relates to an earlier observation (cf. Table 6).

Table 6. Observations after action 3

Glass	Table
`location_and_weight(_, _, _, me, _, removed, 2876)`	`location_and_weight(me, 25, 25, _, 400, added, 1325)`

As we fill the glass with water no new observations are added to either of the knowledge bases. After action 4, cooperation between artefacts is similar as after action 2 and we obtain the observations in Table 7:

Table 7. Observations after action 4

Glass	Table
	`location and weight(me, 25, 25, _, 400, added, 1325)`
`location_and_weight(_, _, _, me, _, added, 3469)`	`location_and_weight(me, 120, 60, _, 400, added, 3472)`

5 Related Work

Our work is generally related to other ubiquitous computing research concerned with instrumentation of the world and with systems that adapt and react to their dynamically changing environment [10, 22, 1]. In contrast to our work, most of these previously reported systems and infrastructures are based on instrumentation of locations (e.g. office [1, 7, 20], home [6, 15, 29], or of users and their mobile devices [19, 23, 27]).

Wireless sensor networks are also generally related to our work as they use similar technology, i.e. wireless nodes with generic sensors [2]. However, they differ mainly in one point: functionalities implemented on wireless sensor nodes only include data acquisition and routing, i.e. taking measurements from sensors and sending the data to a backend infrastructure where it is processed and potential models of individual nodes are maintained.

Previous research has also considered the role of artefacts in addition to locations and users, e.g. the Cooltown project provides a digital presence for people, places and things [16], and SPECs is a proximity sensing hardware platform for activity recognition [17]. Closer to our work are systems directly concerned with artefacts and their situations, e.g. for tracking of moveable assets [18, 26]. Particulary close in spirit is the eSeal system in which artefacts are instrumented with embedded sensing and perception autonomously monitor their physical integrity [9].

Several levels of integration of artefacts in ubiquitous computing systems have been explored, e.g. visual tags [21] and RFID tags [18, 26] to support unique identification. SPECs have been attached to artefacts to capture movement information of users [17]. Collective behaviour of augmented artefacts has been explored in the context of the Smart-its project, e.g. by integrating different kind of sensors in user's belongings [12], furniture [3], and cups [5]. A more generic framework, based on event-condition-action rules (ECA rules), has been provided by the Ubiquitous Chip platform [30].

6 Conclusion

In this paper we demonstrated the potential of the CA concept for activity recognition applications. We have described the design and implementation on an embedded systems platform in the context of a case study in which an augmented table and household goods can be used to track surface-based user activities. There are three innovative aspects to be noted:

- It is a novel approach to acquire and maintain knowledge on activity and changes in the world, distinct in being entirely embedded in moveable artefacts.
- Embedding of generic reasoning capabilities constitutes a new quality of embedded intelligence not previously demonstrated for otherwise non-computational artefacts.
- This approach has the potential to leverage activity recognition applications by providing rich knowledge about situations in the world that can be especially

useful for deployment in users homes where installing external infrastructure might be critical.

Currently we are working on a software framework for embedded devices called *arteFACT* that fully implements the CA architecture. Among power efficiency and responsive, our current prototype has also revealed the following two issues that we seek to address in our future work:

- Activity recognition applications rely on timestamped data and history information. While we are currently extending our devices with FRAM and Flash memory to store history information, it will be crucial to include time as a fundamental concept. This will be especially important for querying history information. We are planning to look into possibilities of using temporal logic in our current implementation of the embedded inference engine.
- In order to improve the scalability of our architecture, we plan to include subscriptions to changes in the artefact knowledge bases. This will entail to include forward reasoning as a more effective inference algorithm

References

1. Addlesee, M., Curwen, R., Hodges, S., Newman, J., Steggles, P., Ward, A., Hopper, A. Implementing a Sentient Computing System. IEEE Computer 34(5), Aug. 2001, pp. 50-56.
2. Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless Sensor Networks: A Survey. In Computer Networks, 38(4), March 2002, pp. 393–422.
3. Antifakos, S., Michahelles F., Schiele, B,: Proactive Instructions for Furniture Assembly. Proc. Ubicomp 2002, Gothenburg, Sweden, Sept. 2002.
4. Basten, T., Geilen, M., de Groot, H., (eds.): Ambient Intelligence: Impact on Embedded System Design. Kluwer Academic Publishers, Boston, 2003.
5. Beigl, M., Gellersen H., Schmidt, A. Mediacups: Experience with Design and Use of Computer-Augmented Everyday Artefacts. Computer Networks 35(4), March 2001.
6. Brumitt, B., Meyers, B., Krumm, J., Kern, A. and Shafer, S. EasyLiving: Technologies for Intelligent Environments. Proc. of HUC 2000, Bristol, UK, Sept. 2000.
7. Cooperstock, J.R. Fels, S. S., Buxton, W. and Smith, K.C. Reactive Environments: Throwing Away Your Keyboard and Mouse. Comm of the ACM 40(9), Sept. 1997.
8. Crabtree, A., Rodden, T., Hemmings, T., Benford, S.: Finding a Place for Ubicomp in the Home. In Proceedings of the 5th International Conference on Ubiquitous Computing Ubicomp 2003, Seattle, WA ,USA, October 2003.
9. Decker, C., Beigl, M., Krohn, A., Robinson, P. and Kubach, U.: eSeal - A System for Enhanced Electronic Assertion of Authenticity and Integrity. In Proc. Of Pervasive 2004, Vienna, Austria, April 2004.
10. Dey, A.K., Salber, D. Abowd, G.D.: A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications In Human-Computer Interaction (HCI) Journal, Vol. 16 (2-4), 2001, pp. 97-166.
11. DIY Smart-its Homepage, http://ubicomp.lancs.ac.uk/smart-its
12. Holmquist, L.E., Mattern, F., Schiele, B., Alahuhta, P., Beigl, M., Gellersen, H-W.: Smart-Its Friends: A Technique for Users to Easily Establish Connections between Smart Artefacts. In Proc. Ubicomp 2001, Atlanta, USA, Sept. 2001.

13. Holmquist, L. E., Antifakos, S., Schiele, B., Michahelles, F., Beigl, M., Gaye, L., Gellersen, H.-W., Schmidt, A., Strohbach, M.: Building Intelligent Environments with Smart-Its. SIGGRAPH 2003, Emerging Technologies exhibition, San Diego USA.
14. Horn, A.: On sentences which are true of direct unions of algebras. Journal of Symbolic Logic, 16, 14-21, 1951.
15. Kidd, C., Orr, R., Abowd, G., Atkeson, C., Essa, I., MacIntyre, B. Mynatt, E., Starner, T and Newstetter, W.: The Aware Home: A Living Laboratory for Ubiquitous Computing Research. In Proc. Cooperative Buildings, CoBuild'99, Pittsburgh, Oct 1999.
16. Kindberg, T., et al.: People, Places, Things: Web Presence for the Real World. In MONET Vol. 7, No. 5, Oct. 2002, Kluwer Publ.
17. Lamming, M., Bohm, D.: SPECs: Another Approach to Human Context and Activity Sensing Research. In Proceedings of Ubicomp 2003. Seattle, WA, USA, October 2003.
18. Lampe M. and Strassner M.: The Potential of RFID for Movable Asset Management. Workshop on Ubiquitous Commerce at Ubicomp 2003, Seattle, October 2003
19. Lukowicz, P. et al.: Recognizing Workshop Activity Using Body Worn Microphones and Accelerometers. Proc. Pervasive 2004, Vienna, Austria 2004.
20. Pentland, A.: Smart rooms, *Scientific American*, vol. 274, pp. 54-62, 1996.
21. Rekimoto J. and Ayatsuka, Y.: CyberCode: Designing Augmented Reality Environments with Visual Tags. Proc. Designing Augmented Reality Environments (DARE 2000), 2000.
22. Schilit, B. Adams, N. and Want, R.: Context-aware computing applications. Proc. WMCSA'94.
23. Schmidt, A., Aidoo, K.A., Takaluoma, A., Tuomela, U., Van Laerhoven, K., Van de Velde, W.: Advanced Interaction in Context. In Proc. of HUC99, Karlsruhe, Germany, 1999.
24. Schmidt, A., Strohbach, M., Van Laerhoven, K., Friday, A., Gellersen, H.-W.: Context Acquisition based on Load Sensing. In Proc. of Ubicomp 2002, Gothenburg, Sweden.
25. Schmidt, A., Strohbach, M., Van Laerhoven, K., Gellersen, H.-W.: Ubiquitous interaction - Using surfaces in everyday environments as pointing devices. In Lecture Notes in Computer Science (LNCS), Volume 2615, N. Carbonell & C. Stephanidis (Eds.). Springer Verlag, 2002, pp. 263-279.
26. Siegemund, F. and Flörkemeier, C.: Interaction in Pervasive Computing Settings using Bluetooth-enabled Active Tags and Passive RFID Technology together with Mobile Phones. Proc. IEEE PerCom 2003, March 2003, Fort Worth, USA.
27. Starner, T., Schiele, B. and Pentland, A.: Visual Context awareness in Wearable Computing. Proc. Intl. Symp. on Wearable Computing (ISWC'98), Pittsburgh, Oct. 1998, pp. 50-57.
28. Strohbach, M., Gellersen, H.-W., Kortuem, G., Kray, C.: Cooperative Artefacts: Assessing Real World Situations with Embedded Technology. Accepted for Publication in Proc. of Ubicomp 2004, Nottingham, UK 2004.
29. Tapia, E. M., Intille, S. and Larson, K.: Activity Recognition in the Home using Simple and Ubiquitous Sensors. Proc. Pervasive 2004, Vienna, April 2004.
30. Terada, T., Tsukamoto, M., Hayakawa, K., Yoshihisa, T., Kishino, Y., Kashitani, A. and Nishio, S.: Ubiquitous Chip: a Rule-based I/O Control Device for Ubiquitous Computing. In Proc. of Pervasive 2004, Vienna, April 2004.
31. Weiser, M. and Brown, J. S.: Designing calm technology. PowerGrid Journal 1.01, July 1996.