Infrastructure Concepts for Tag-Based Ubiquitous Computing Applications

Kay Römer
Department of Computer Science
ETH Zurich
8092 Zurich, Switzerland
roemer@inf.ethz.ch

Thomas Schoch
Department of Computer Science
ETH Zurich
8092 Zurich, Switzerland
schoch@inf.ethz.ch

ABSTRACT

Object tagging systems such as Radio Frequency Identification (RFID) enable the implementation of a wide variety of ubiquitous computing applications. Up to date, most of these prototypical applications have been implemented from scratch. Our goal is to develop concepts and software frameworks to ease the development of such applications. In this paper we present our experience with with a collection of prototypical ubiquitous computing applications based on tagged physical objects. We point out a number of basic tasks common to this kind of application, before presenting design concepts that we have found useful for structuring and implementing such applications. Building upon these concepts, we are creating software infrastructures to support the development of tag-based ubiquitous computing applications.

1. INTRODUCTION

Object tagging systems such as *Radio Frequency Identification* (RFID) are an enabling technology for many interesting ubiquitous computing applications [5]. By attaching small *tags* to physical objects, these objects can be identified when brought into the vicinity of an antenna connected to a device known as a *tag reader*. State-of-the-art RFID systems such as Icode [6] allow the simultaneous detection of multiple tags within a space of up to approximately one cubic meter. Tags not only hold a unique ID, but also provide a limited amount (Icode tags currently provide about 60 bytes) of non-volatile read/write memory.

Such tagging systems enable the implementation of a wide range of novel ubiquitous computing applications by bridging the gap between the physical world (i.e., real-world objects) and the virtual world (i.e., application software). During the last two years we have developed a number of such RFID-based applications in areas like smart games, home and office automation. Our first prototype systems were implemented from scratch, the only piece of software they had in common being the driver software for the RFID system. Based on our experience with these applications, we identified a number of tasks common to this type of application, which led to the design of concepts which we found useful for structuring and implementing applications using tagged physical objects. Based on those mechanisms we are designing and implementing software infrastructure to support the development of tag-based ubiquitous computing applications.

We begin this paper with a short overview of some of the applications we developed, go on by pointing out common tasks, and then present pertinent infrastructure design concepts.

2. SELECTED UBICOMP APPLICATIONS

In this section we outline the type of applications we intend to support with our infrastructure by sketching some of the RFID-based ubiquitous computing applications we have developed over the recent years. Note that all the applications are based on multiple interacting tagged physical objects. These applications will serve as a basis for identifying common tasks that should be supported by a generic Ubicomp infrastructure.

RFID Chef

Grocery items are equipped with RFID tags (instead of the barcodes that are commonly used today). When placed on the kitchen counter, a nearby display suggests dishes that can be prepared with the grocery items available, or shows missing ingredients. The suggested dishes not only depend on the available ingredients, but also on the preferences of the cook, who might for example prefer vegetarian or Asian dishes. To implement this functionality, the cook is identified by an RFID tag attached to his wristwatch, such that the tag enters the range of the kitchen counter RFID antenna when grocery items are placed on the counter. [1] contains a detailed description of the system.

Smart Playing Cards

Ordinary playing cards are equipped with RFID tags. An RFID antenna mounted beneath a table monitors the game moves of the players. A nearby display shows the score, the winner, and a cheat alarm if one of the players does not follow suit, and gives hints to beginners by assessing the players' moves. This is implemented by having each card remember the contexts in which it has been used and whether the trick in question was won or lost. [2] contains a detailed description of the system.

Smart Agenda

Agendas are equipped with RFID tags. If two or more people want to make an appointment, they place their agendas on the "appointment table", which is equipped with an RFID antenna. A nearby display shows possible dates that are compatible with the schedules of all the participants.

Smart Tool Box

Tools are equipped with RFID tags, and the tool box contains a mobile RFID system. The tool box issues a warning if a worker attempts to leave the building site (or a sensitive maintenance area such as an airplane) while any tools are missing from his box. The box also monitors how often and for how long tools have been in use. Based on this information, tools can be replaced before they wear out. Additionally, the tool owner can charge for tool rental based on actual tool usage.

Smart Medicine

This application helps to avoid trouble with medication by monitoring medicine from production to use. For this, medicine bottles are equipped with RFID tags. The environmental temperature of the medicine is constantly checked in order to avoid it going bad. Within the medicine cabinet, the bottle checks for other pharmaceuticals which are not compliant if taken together. A warning is issued if such dangerous situations are detected.

3. COMMON TASKS

Based on the applications sketched in Section 2 we can highlight a number of tasks common to these applications.

Events

In order to enable an application to react to actions in the physical world, a link has to be established between tagged physical objects in the real world and the application. Since RFID systems detect presence and absence of tags in a certain physical space, this link can be established by notifying the applications about tags entering and leaving this space. A natural way to implement these notifications is by means of an event notification system. The system has to support two basic events `enter(X)` and `leave(X)`, which are sent to the application when a tag with ID X enters and leaves the detection range of the RFID system, respectively. Additionally, applications need a way of expressing their interest in a subset of all possible tags, since a single antenna might be used by multiple applications at the same time.

Note that the RFID system and the application may run on different systems and platforms, as for example in the Smart Tool Box application, which consists of a mobile RFID system in the tool box cooperating with a stationary system at the workshop.

Event Generation

Although from an abstract point of view the RFID system detects entering and leaving tags, matters are complicated by the actual low-level interface provided by the RFID system and certain application requirements. The Icode system [6], for example, periodically scans (typically at sub-second intervals) for present tags by sending a short RF pulse and waiting for answers from the tags. When receiving the pulse, a tag waits a random number of discrete time slots before answering, in order to avoid time-consuming collisions with other tags sending concurrently. The maximum number of time-slots N a tag may wait before answering influences both the time needed for a single scan and the expected number of collisions. A small N value results in fast scans (down to 60ms according to [4]) but many collisions, whereas a large N value results in slow scans (more than one second) but few collisions.

This kind of low-level interface has several implications. First, applications are typically only interested in *changes* of the detected set of tags, i.e., they want to receive enter and leave event notifications. So an appropriate software component has to convert scan results to event notifications. However, the task of this component is non-trivial, since the scan results are typically imperfect due to tag collisions, i.e., not all tags are detected in every scan. The latter can result in event flickering: the fast generation of alternating leave and enter events for a tag that is in fact present all the time. Filters which cancel out spurious leave/enter events are required in case of such imperfect tag detection.

Secondly, many applications require that objects be detected as fast as possible. This is necessary if tags stay in the detection range only for a rather short time. Even if the tags stay long enough, long delays in tag detection can cause problems with human computer interaction. The Smart Playing Cards application exemplifies the latter, because the user expects an immediate reaction from the system on placing a card on the table. The optimum detection performance can be achieved by selecting the number of time-slots N to be slightly greater than the actual number M of tags in the range of the antenna. However, M is typically unknown. Therefore, nontrivial algorithms are required for selecting N in order to read all present tags in a minimal amount of time [4].

Context

Typically the application's action when a tag enters or leaves the antenna's range not only depends on the identity of the tag, but also on the presence or absence of other tags during this event – which we call the context of the event. Consider for example the RFID Chef application: the dishes that have to be displayed when a new grocery item is placed on the kitchen counter not only depend on the grocery item itself, but also on the cook. In the Smart Playing Cards application, the action taken when a playing card enters or leaves the antenna's range depends on the other playing cards lying on the table. Note that this notion of context – the presence or absence of tagged objects – is a specific instance of a more general concept [3].

Often applications are only interested in a certain subset of events or events with a certain context. Consider the Smart Playing Cards for example, where the application only wants to be informed when the last of four players has played his card in the trick. Such a selection of events can be performed at several levels, for example in the application. However, scalability and performance of a system can be increased by performing this selection as close as possible to the source of events. This, however, requires a way of expressing the event contexts applications are interested in.

Location

An RFID system provides only a very simple notion of location – the reading range of an RFID antenna. A tagged object is at this location if its tag can be read by the antenna. Though very simple, this notion of location is useful for many applications, because it serves as a means of grouping tagged objects by their location. In the Smart Tool Box application, for example, all the tools in the range of the tool box antenna belong to the same tool box. As in this example, "cooperating" physical objects are often collocated. In order to establish cooperation among collocated objects, the concept of neighborhood has to be supported in an adequate way. Note that this may be a nontrivial task, since physically collocated objects might be logically separated (e.g., by a wall), thus making them non-neighbors.

In general, there is a need for an adequate representation of primitive locations and for concepts to build higher-levels of location information with explicit support for neighborhood relations.

Composition

Often physical objects are an aggregation of other physical objects. In the Smart Medicine application, for example, a medicine cabinet contains many medicine bottles. Many applications are only interested in manipulating composite objects, for example in order to perform a certain manipulation on all the objects contained in a composite object. In order to support such applications, it is necessary to explicitly model "part of" relationships among objects. Note that composition is different from the neighborhood concept, since neighboring objects do not necessarily belong to the same composite object.

Time

Some of the applications require a notion of time. The Smart Tool Box, for example, has to determine the amount of real time that has elapsed between removing a tool from the box and replacing it. The Smart Playing Cards application knows which player played which card by means of the temporal order of the card enter events. In general, there is a need to time-stamp enter and leave events. In the case of multiple tag readers, the time stamps of events originating from different readers should be comparable, even if some of the readers have been offline during event generation.

State and Behavior

Applications typically assign state and behavior to physical objects. In the Smart Tool Box application the state of a physical object (i.e. a tool) consists of its usage pattern. In the Smart Agenda application, the state of an agenda consists of a schedule. However, there are also stateless applications such as RFID Chef, where the reaction or "output" of the application depends only on the tags currently present.

The applications also differ in the way they assign behavior to physical objects. In the RFID Chef application, for example, all the grocery items and the cook have a "common" behavior – the display of a list of dishes. In the Smart Tool Box application, physical objects have a more "individual" behavior – raising an alarm if they are missing, and calculating tool usage. Moreover, a single physical object can contribute to the behavior of more than one other physical object. In the Smart Playing Cards application, for example, a single card contributes to the "usage context" of all the other playing cards on the table.

A flexible mechanism is therefore needed for assigning state and behavior to physical objects.

History

Some applications not only react immediately to entering and leaving tagged objects, but are also queried about their history later on. Consider the Smart Tool Box example, where tools can be queried regarding how long they were used in which tool box on which building site. Therefore, a generic mechanism for logging and querying the history of physical objects seems appropriate.

Communication Infrastructure

All the applications we have developed require a TCP/IP-based, Internet-like communication infrastructure. However, there may not always be global connectivity, as in the case of the Smart Tool Box application. The tool box contains a mobile RFID system and an associated computing system, which are able to operate offline. The toolbox is only connected to the background communication infrastructure when it is returned to the workshop. Such disconnected operations must also be supported.

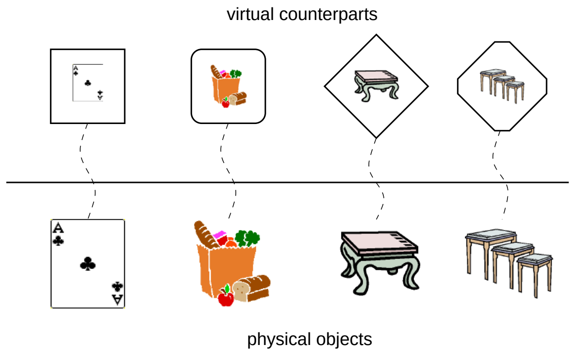

Figure 1: Virtual Counterparts

4. INFRASTRUCTURE CONCEPTS

Having identified a set of common tasks in Section 3, we now elaborate on some of the infrastructure concepts that are intended to support these tasks.

Virtual Counterparts

The central concept of our infrastructure is the *virtual counterpart* (VC). A virtual counterpart encapsulates the state and behavior of a tagged physical object. Figure 1 shows four different types of virtual counterparts. The simplest form appears in the playing card VC, which is denoted as a rectangle in the figure. There is a one-to-one mapping between tagged physical objects and their VC. A virtual meta-counterpart, on the other hand, implements "collective" state and behavior for a group of tagged physical objects; that is, it implements an n-to-one mapping of physical objects to state and behavior. Virtual meta-counterparts are depicted as a rounded rectangle, such as the grocery meta-counterpart in Figure 1.

There is a special kind of VC representing a location such as the range of an RFID antenna. These VCs are called virtual locations (VLs) and are depicted as diamonds. VLs offer location-dependent functionality, such as measuring local temperature as in the Smart Medicine application. Moreover, they play a key role in supporting location management as described below. Analogous to virtual meta-counterparts, there are also virtual meta-locations, which represent a whole set of locations. Virtual meta-counterparts are denoted by rounded diamonds. We will use the term "counterpart" as a genus for virtual (meta-) counterparts and locations.

Counterpart Events

As noted in Section 3, information flowing from the physical world to virtual counterparts consists of enter and leave events caused by tags entering and leaving the range of an RFID antenna. This link between tagged physical objects and their virtual counterparts is indicated by the dashed lines in Figure 1. A natural approach for informing virtual counterparts of tag events is the use of a Virtual Counterpart Event Service (VCES).

In general, an event notification system consists of producers generating events, consumers receiving events, and a component that forwards events from producers to consumers. Consumers subscribe to the types of events they are interested in. Based on these subscriptions, the VCES delivers events only to interested consumers.

In our context, producers correspond to software components that generate enter and leave events from the RFID scans, which we call event driver (ED). Consumers correspond to the various types of virtual counterparts. Since multiple counterparts can subscribe to events originating from the same physical object, we can implement an n-to-m mapping between physical objects and virtual counterparts by using the VCES.

In order to support context as described in Section 3, the event service can be programmed by the counterparts. By means of a rule-based program, the application can tell the system the context in which the application is interested in events. If an event and its context match a rule, then the event service generates a context event, which contains context information as well as the information from the triggering event.

Consider for example the Smart Playing Cards application. It only needs to be notified of completed tricks, i.e. when all four players have put a playing card on the table. Therefore the application programs the event service with a rule that says "notify me of an enter event only if there are already three cards on the table". If this rule is matched, the event service will generate a context event which contains the identities of all four cards.

Counterpart Management

Counterpart management consists of three subtasks: life-cycle management, location support, and composition.

Life-cycle management deals with the instatiation, migration, and destruction of virtual counterparts based on the tag enter and leave events received from the counterpart event service. If a tagged object is detected for the first time, the counterpart associated with that object has to be instantiated. Based on the unique tag ID contained in the enter event, the code executable for the associated counterpart is retrieved from a code repository and executed. The counterpart state should be saved and the counterpart destroyed if the tagged object leaves the vicinity of an RFID antenna for a long period of time. If the object shows up later at a different location, its counterpart must be re-instantiated using the saved state.

The life cycle of virtual locations is somewhat simpler. Each virtual (meta-) location is associated with one or more RFID systems. Upon seeing the first enter event from one of these systems, the counterpart associated with these "locations" is instantiated and never destroyed. Based on the unique location ID of the RFID system contained in the enter event, the code executable for the associated counterpart is retrieved from a code repository and executed.

Location support is based on virtual locations by assigning each virtual counterpart its respective virtual location. Virtual meta counterparts allow for grouping several physical locations into a logical location. By querying their associated virtual location, virtual counterparts can find out their potential neighbors.

Composition of virtual counterparts is supported by arranging them into a hierarchical structure. There, the counterpart of a composite object is the parent of the counterparts of its contained objects. Applications may use this hierarchy to perform a certain action on all objects contained in composite object, or to automatically update composite objects upon certain actions in contained objects.

Artifact Memory

In Section 3 we pointed out the need for persistently storing the state information and event histories of virtual counterparts. The Artifact Memory (AM) fulfills this task. A very limited amount of state can be stored in tag memory[1], this method has the advantage that the saved state is available wherever a tag is. Larger amounts of state and event history are stored in a database-like system.

Based on the stored event histories, the AM provides an interface for queries regarding tags and their location at certain points in time:

- find(TAG, TIME): returns the location of TAG at TIME
- with(TAG, TIME): returns the set of tags at the same location as TAG at TIME
- look(LOC, TIME): returns the set tags at location LOC at TIME
- history(TAG): returns a list of recent locations visited by TAG

Using these queries, counterparts can look for other counterparts they want to cooperate with. More generally, the AM can be used for data mining regarding certain behavioral patterns of tagged objects.

5. CONCLUSION AND OUTLOOK

In this paper we have given a brief overview of some tag-based ubiquitous computing applications we have developed, derived tasks common to these applications, and presented concepts to support these tasks. Key concepts are virtual counterparts, counterpart events, counterpart management, and artifact memory. Based on these concepts, we are developing software infrastructure to support the development of tag-based ubiquitous computing applications.

The general goal here is to learn about appropriate concepts for a general infrastructure for smart environments and to gain experience by implementing and applying those concepts in typical application scenarios.

6. REFERENCES

[1] M. Langheinrich, F. Mattern, K. Römer, and H. Vogt. First Steps Towards an Event–Based Infrastructure for Smart Things. In *Ubiquitous Computing Workshop, PACT 2000*, Philadelphia, PA, October 2000. www.inf.ethz.ch/vs/publ/papers/firststeps.pdf.

[2] K. Römer. Smart Playing Cards - A Ubiquitous Computing Game. In *Workshop on Designing Ubiquitous Computing Games, Ubicomp 2001*, Atlanta, USA, September 2001. www.inf.ethz.ch/vs/publ/papers/ubicomp01-smart-playing-cards.pdf.

[3] D. Salber, A. K. Dey, and G. D. Abowd. The Context Toolkit: Aiding the Development of Context-Enabled Applications. In *CHI 99*, Pittsburgh, USA, May 1999.

[4] H. Vogt. Efficient Object Identification with Passive RFID Tags. In *Pervasive 2002*, Zurich, Switzerland, August 2002. www.inf.ethz.ch/vs/publ/papers/rfid_obj.pdf.

[5] R. Want, K. Fishkin, A. Gujar, and B. Harrison. Bridging Physical and Virtual Worlds with Electronic Tags. In *ACM Conference on Human Factors in Computing Systems (CHI 99)*, Pittsburgh, PA, May 1999.

[6] The Philips I–Code System. www-us2.semiconductors.philips.com/identification/products/icode.

[1]We expect the amount of memory available in tags to increase over the next few years, however.