The Value of Handhelds in Smart Environments[⋆]

Frank Siegemund, Christian Floerkemeier, and Harald Vogt

Institute for Pervasive Computing
Department of Computer Science
ETH Zurich, Switzerland
{siegemund|floerkem|vogt}@inf.ethz.ch

Abstract. The severe resource restrictions of computer-augmented everyday artifacts imply substantial problems for the design of applications in smart environments. Some of these problems can be overcome by exploiting the resources, I/O interfaces, and computing capabilities of nearby mobile devices in an ad hoc fashion. We identify the means by which smart objects can make use of handheld devices such as PDAs and mobile phones, and derive the following major roles of handhelds in smart environments: (1) mobile infrastructure access point, (2) user interface, (3) remote sensor, (4) mobile storage medium, (5) remote resource provider, and (6) weak user identifier. We present concrete applications that illustrate these roles, and describe how handhelds can serve as mobile mediators between computer-augmented everyday artifacts, their users, and background infrastructure services. The presented applications include a remote interaction scenario, a smart medicine cabinet, and an inventory monitoring application.

1 Introduction

As pointed out by Weiser and Brown [12], "Ubiquitous Computing is fundamentally characterized by the connection of everyday things in the real world with computation". Computer-augmented everyday artifacts – also called *smart everyday objects* – epitomize this vision of Ubiquitous Computing in that they are everyday objects augmented with small sensor-based computing platforms (cf. Fig. 1). Smart objects are aware of their environment, can perceive their surroundings through sensors, collaborate with peers using short-range wireless communication technologies, and provide context-aware services to users in smart environments.

But the computational capabilities of smart objects are very limited because their computing platform needs to be small and unobtrusive. Furthermore, they do not possess conventional I/O interfaces such as keyboards or displays, which restricts the interaction with users. And finally, because of their limited energy resources, smart objects support only short-range communication technologies,

⋆ Parts of this work were supported by the Smart-Its project, which is funded by the European Commission (contract No. IST-2000-25428) and the Swiss Federal Office for Education and Science (BBW No. 00.0281).

which makes it difficult to access background infrastructure services when no access point is near by. Combined, all these limitations cause severe problems for the design of applications in environments of smart objects.

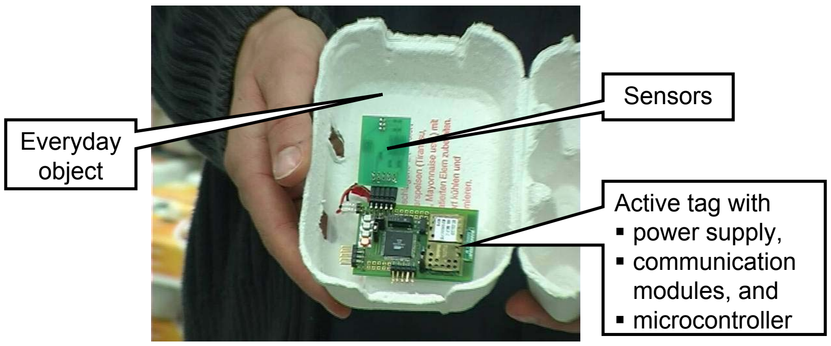


Fig. 1. A smart everyday object: an egg carton augmented with a sensor-based computing platform

We argue that most of these problems can be overcome when smart objects can spontaneously access the capabilities of nearby handheld devices. In smart environments, people move around who carry their personal devices with them. By exploiting the features of nearby handhelds in an ad hoc fashion, new possibilities for the design of applications on smart objects evolve. We identify and illustrate six different means by which computer-augmented everyday artifacts can make use of handhelds: (1) as mobile infrastructure access point, (2) as user interface, (3) as remote sensor, (4) as mobile storage medium, (5) as remote resource provider, and (6) as weak user identifier.

Given these roles, handhelds can enrich the interactions among smart objects, users, and background infrastructure services (cf. Fig. 2). As *mobile access points*, handhelds facilitate the ad hoc interaction between smart objects and a background infrastructure. A handheld's input and display capabilities enable new forms of user interactions with smart objects. And finally, the cooperation among smart objects themselves can be improved by utilizing handheld devices as *remote resource providers*.

As mediators between smart objects, users, and background infrastructure services, handhelds enable new forms of applications in smart environments. To support this thesis, we present three applications that make extensive use of handheld devices. We start with a remote interaction application. Here, interaction patterns that people associate with a specific type of handheld device (e.g., making phone calls) are translated to smart environments. In particular, we assign phone numbers to smart objects and describe how users can "call" and interact with objects from remote locations. The remote interaction application uses handhelds as *mobile storage medium*, *user interface*, and *weak user identifier*. We then present the Smart Medicine Cabinet. It improves medical compliance

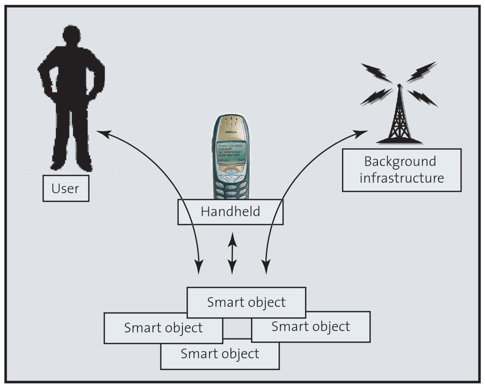


Fig. 2. Handhelds as mediators in smart environments: handheld devices enrich the interaction between different smart objects, between smart objects and their users, and between smart objects and background infrastructure services

and facilitates a more effective treatment of mobile patients by using handheld devices together with "smart medicine". In this application, handhelds serve as *mobile infrastructure access points*, and again as *user interface* and *mobile storage medium*. Finally, we present an inventory monitoring application. Its goal is to illustrate how smart objects can spontaneously outsource computations to nearby handheld devices. It illustrates a handheld's ability to serve as *mobile resource provider* and also as *user interface* for smart objects.

In the following section we review related work. In section 3 we identify the roles of handhelds in smart environments. Section 4 discusses how these roles can actually be implemented. Sections 5 through 7 present three applications that illustrate these roles. Section 8 concludes the paper by summarizing the lessons learned from our applications.

2 Related Work

Gellersen et al. [6] proposes to integrate low-cost sensors into everyday objects and mobile user devices to facilitate the development of context-aware applications. In their MediaCup project [2], active sensor tags are embedded into everyday objects to derive and provide services based on the situational context of users. For example, the context "meeting" can be inferred from the presence of many hot cups in a meeting room. However, their work assumes stationary access points that facilitate the cooperation between active artifacts and existing infrastructure services, while we explicitly focus on utilizing nearby handheld devices in an ad-hoc fashion.

Hartwig [7] integrates Web servers into active Bluetooth-enabled tags, attaches them to physical objects, and controls augmented items with nearby Bluetooth-enabled mobile phones by means of a WAP (Wireless Application Protocol) interface. WAP is used for local interactions with augmented items, whereas we use WAP (among other technologies) for remote interactions with smart objects.

Want et al. [11] augments physical objects with passive RFID tags to associate objects with a representation in the virtual world. In the Cooltown project [8], everyday items are also equipped with tags in order to link them with a representation in the World Wide Web. In both approaches, the functionality of a tag consists primarily in providing a link to information in the virtual world. As tags are usually read out by a mobile user device, the actual application is implemented on the handheld or the backend infrastructure, but not on the tag. In our approach, the active tags and sensors attached to physical objects process data autonomously, derive context-information in collaboration with other smart objects, and coordinate the actual applications. Nearby handheld devices do not implement applications for smart objects, but are only means by which computer-augmented artifacts can dynamically extend their own capabilities.

3 Characteristics and Roles of Handhelds in Smart Environments

The roles of handheld devices in environments of computer augmented everyday artifacts are manifold. Handhelds can serve as primary user interface, they can be a mobile infrastructure access point, provide mobile data storage, act as a user identifier, supply energy and computational resources, or offer sensing capabilities. In this section we identify the main reasons for this versatility. Thereby, we name important characteristics of handheld devices and from that derive the major roles of handhelds in smart environments.

Habitual presence. As mobile phones, PDAs, and other handheld devices are habitually carried around by their owners, they are always in range of a smart object when a physical interaction with it is about to take place. This is especially important because the smart objects themselves generally do not have access to resources beyond their peers, and handheld devices are the only local devices able to provide powerful resources and sophisticated services. The habitual presence of handheld devices during physical interactions with smart objects is the most important characteristic of handhelds in smart environments. It entails their general function as mediator between smart objects, users, and background infrastructure services, and is therefore a precondition for the roles of handhelds presented in this paper.

Wireless network diversity. Mobile phones and PDAs usually support both short-range as well as long-range wireless communication technologies, such as Bluetooth, IrDA, WLAN, GSM, or UMTS. This enables handhelds to not only interact with smart objects directly via short range communication standards but also to relay data from augmented items to powerful computers in an infrastructure far away. The characteristic of wireless network diversity makes it possible for handhelds to serve as *mobile infrastructure access points*.

User interface and input capabilities. Tags attached to everyday objects have to be small, unobtrusive and are ideally invisible to human users.

Consequently, they do not possess conventional buttons, keyboards, or screens. Interaction with augmented objects therefore has to take place either implicitly by considering sensory data of smart objects, or explicitly by using the input and display capabilities of other devices [10]. As people are usually familiar with the features provided by their handhelds, interactions with smart objects that are based on these well-known interfaces should imply a more comfortable and easy usage of smart objects. As a result, handhelds often serve as the primary *user interface* for smart objects.

Perception. Handheld devices can serve as remote sensors for a smart object, which are accessed wirelessly using a communication technology supported by all participating devices. The way handheld devices perceive their environment strongly depends on their functionality. Cellular phones, for example, know to what cell they currently belong and can serve as *remote location sensors* for augmented items.

Mobility. Active tags can transfer data such as how to reach a smart device from remote locations to a handheld device, where it is permanently stored and accessible for users independent from their current location. Here, handhelds serve as *mobile storage medium* for smart objects.

Table 1. The roles of handhelds in smart environments and the underlying characteristics that entail these roles

Handheld's role	**Underlying characteristic**
Mobile infrastructure access point	Wireless network diversity
User interface	Input and display capabilities
Remote sensor	Perception
Mobile storage medium	Mobility
Remote resource provider	Computational resources Regularly refilled mobile energy reservoirs
Weak user identifier	Personalization

Computational resources and regularly refilled mobile energy reservoirs. Although the energy consumption of a handheld device such as a cellular phone should be as small as possible, people are used to recharge its batteries at regular intervals. PDAs are often shipped with a cradle that offers both host access to the device and automatic recharging. A similar procedure, however, is not feasible for smart objects because there are just too many of them. As a result, smart objects may exploit handheld devices in range as remote energy reservoir, for example for carrying out complex and energy consuming computations. Because of regularly renewed energy resources, handhelds can also offer more powerful resources regarding memory and bandwidth, which allow smart objects to use them as *remote resource providers*.

Personalization. PDAs and mobile phones are most often personalized, i.e. they belong to a certain person who uses the device exclusively. Smart devices can therefore adapt their behavior according to the current handheld devices in range and thereby offer functions tailored towards certain persons. In this context, handhelds can serve as *weak user identifiers*.

The relation between the described characteristics of handheld devices and the roles we derived from these characteristics are summarized in Tab. 1.

4 Interfacing Handhelds from Smart Objects

After having identified the major roles of handhelds in smart environments on a more conceptual level (cf. Tab. 1), we now describe how these roles can actually be implemented.

Thereby, we focus on smart objects that are equipped with BTnodes [3]. BTnodes are small computing platforms, consisting of a microcontroller, Bluetooth communication modules, an autonomous power supply, and externally attached sensor boards (cf. Fig. 3). As Bluetooth is integrated into an increasing number of consumer devices, BTnodes are suitable to illustrate the roles of handhelds in smart environments. These roles, however, do not depend on Bluetooth or any other specific communication standard (cf. [9] for a discussion about communication issues in smart environments).

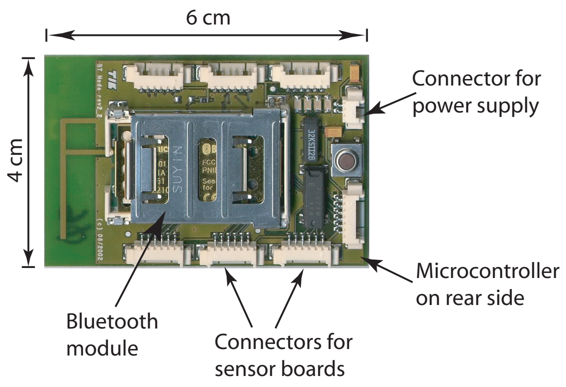

6 cm
4 cm
Connector for power supply
Microcontroller on rear side
Bluetooth module
Connectors for sensor boards

Fig. 3. BTnodes are used as a device platform to make everyday objects "smart"

Mobile infrastructure access point. Because of their wireless network diversity – i.e., their support of short-range as well as wide-range communication technologies – handheld devices can serve as mobile gateways to background infrastructure services. Technically, this is achieved by establishing a local short-range connection from a smart object to a handheld device, and a long range communication link from the handheld to a background infrastructure server (cf. Fig. 4). The wireless technology used to build up the long-range connection to the backend infrastructure depends on the capabilities of the handheld device. In case of PDAs this might be an IEEE 802.11 link to a base station, and a GSM (Global System for Mobile Communication) or GPRS (General Packet Radio Service) connection in case of mobile phones.

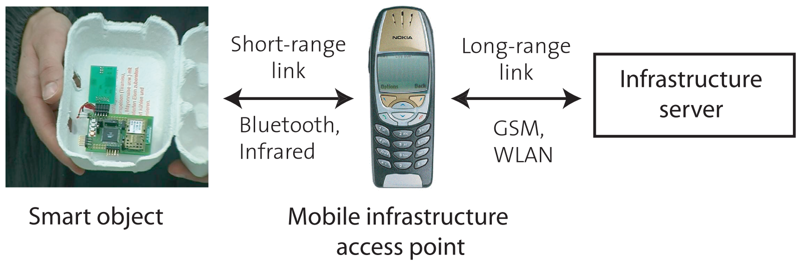


Fig. 4. Mobile access points: smart objects use nearby handheld devices to communicate with background infrastructure services in an ad hoc fashion

In our applications, we have realized a handheld's role as mobile infrastructure access point as follows. When a handheld device (e.g., a mobile phone) comes within range of a smart object that needs to access the background infrastructure, the object establishes a local Bluetooth connection to the handheld. The smart object then sends AT commands over this local Bluetooth link in order to establish a long-range GSM data connection from the mobile phone to a background infrastructure server. Given this connection, arbitrary data can be exchanged between the smart object and the background server. There is a standardized set of AT commands supported by all GSM-enabled mobile phones. Besides using explicitly established GSM data connections, it is also possible for smart objects to embed data into an SMS (Short Message Service) message. As the latter approach does not require the overhead of GSM data connection establishment, it is the preferred way to exchange data with a background infrastructure server in our applications.

User interface. The user interface and input capabilities of handheld devices can be exploited by smart objects to notify users acoustically or to allow interactions with smart items based on a graphical user interface. Mobile phones and PDAs offer several popular features by means of which such an interaction can be realized. They range from (1) custom alarms, (2) SMS messages, (3) WAP pages, (4) calendar entries and business cards to (5) whole Java user interfaces that can be downloaded over a local connection from a smart object to a handheld device. We have prototypically implemented all these different means to facilitate the user interaction with smart objects. Thereby, BTnodes are used as prototyping platform to augment everyday objects, and mobile phones or PDAs as handheld devices.

(1) Alarms are written to cellular phones by transmitting standardized AT commands from smart objects over a local Bluetooth connection to a mobile phone.

(2) Similarly, smart objects initiate the exchange of SMS messages with remote users by sending AT commands to a local GSM gateway. This GSM gateway transmits the SMS messages to remote users and relays incoming messages to the corresponding smart objects (cf. Sect. 5).

(3) User interaction with smart objects can also take place via WAP interfaces. In our implementation, user interaction with WAP takes place by means of a background infrastructure service, the *background infrastructure representative (BIRT)* of a smart object. Smart objects synchronize their state with the BIRT whenever an access point is in range. Based on this information, the BIRT provides WAP pages that reflect the current state of an object. By means of their handheld devices users can then access these WAP pages and exchange information with the BIRT of a smart object. User input is relayed to the actual smart object during the next synchronization phase (cf. Sect. 6).

(4) Calendar entries and business cards can be exchanged via Bluetooth OBEX (Bluetooth object exchange protocol) with Bluetooth-enabled mobile phones and PDAs in range of a smart object. Calendar entries can be used as an alternative to custom alarms in mobile phones. Their advantage is that they not only trigger an acoustic alarm at the time specified but also display information on the handheld's screen.

(5) Finally, we have also implemented means for more sophisticated interactions with smart objects. Thereby, a Java user interface is stored on the smart object. People can select smart objects in their environment by means of a small program on their handheld device and download the user interface from the selected object (cf. Sect. 7).

Remote sensor. The percepts of handheld devices are of potential interest for smart objects, which often simply do not have sufficient resources to deploy sophisticated sensors. Some sensor data – as, for example, the information about the current cell id of a mobile phone – can be easily retrieved from nearby handheld devices. In our implementation, mobile phones can serve as remote location sensors for smart objects. Thereby, a short-range connection is established from a smart object to a mobile phone, whose location information is queried by exchanging AT commands on top of the local communication link.

Mobile storage medium. Data transmitted from a smart object to a mobile device is available to users independent from their current location and their overall situational context. In our applications, smart objects transmit contact information in form of telephone book entries and templates that specify the commands supported by a specific smart object to mobile phones (cf. Sect. 5). These information enable users to start an interaction with smart objects from anywhere. As in our software package, data are transmitted to mobile phones by sending standardized AT commands over a local Bluetooth communication link when a user is in range of a smart object.

Remote resource provider. The previously described roles show how handhelds mediate between smart objects and their users, or between smart objects and background infrastructure services. As *remote resource provider*, however, a handheld primarily enriches the interaction among smart objects themselves in that it provides a platform for outsourcing complex computations and offers sophisticated data storage capabilities. Our goal was to spontaneously integrate handheld devices into already existing groups of collaborating smart objects.

This goal is achieved by introducing an infrastructure layer facilitating the collaboration among computational entities. This layer is a distributed tuple space [4] for smart objects and handheld devices, which is part of our implementation. Smart objects that want to collaborate form a tuple space and write their sensory data into the space. When a handheld device comes into the range of collaborating objects it also joins this distributed tuple space. Thereby, smart objects can instantaneously make use of the memory resources of handheld devices on the basis of resource-aware tuple space operations. Our resource-aware tuple space operations try to identify the most suitable place to store sensor tuples. As the actual location of a tuple becomes transparent through the tuple space, they are stored on the device with the most spare resources – which is often the handheld device.

The most important reason for introducing the tuple space abstraction, however, is that the location where code is executed becomes transparent. This is because all devices operate on the same data basis of the distributed tuple space. Smart objects can therefore simply transfer code to a nearby handheld device participating in their tuple space and thereby exploit its computational resources. In our implementation of this concept, Java classes are stored on a smart object that are spontaneously transmitted to nearby handheld devices when the handheld joins the distributed tuple space (cf. Sect. 7).

Weak user identification. In many interaction scenarios, the identity of an involved user is important for authorizing certain actions, or adapting services. Handheld devices offer user identification capabilities in varying flavors. They range from a PIN that is necessary to operate a mobile phone, to fingerprint sensors. Knowledge of the PIN gives a hint on the user's identity if one can assume that a device is personalized to a single user, while a biometric sensor provides much stronger confidence in the user's identity. From the point of view of the smart object, the downside of using an external identification mechanism is the necessity of putting trust in the handheld's correct operation (and in the user, e.g. to keep the PIN secret). Since the assurance level on the identity of the current user is usually rather low, we call this feature weak identification. The software package we developed to demonstrate the roles of handhelds in smart environments supports authentication using PIN codes as part of the implemented Bluetooth protocols.

5 Smart Object-Human Interaction: as Easy as Making Phone Calls

In this and the following two sections we present applications that illustrate the previously identified roles of handhelds in smart environments. The application in this section demonstrates how mobile devices serve as as *mobile storage* *medium*, *weak user identifier*, and as *user interface* for remote interactions with smart objects.

People usually associate a specific type of handheld device with a specific way to interact with communication partners: teenagers write SMS messages to arrange a fun meeting using their mobile phones, and business people organize their appointments with PDAs. Adopting device specific behavior to smart environments while maintaining interaction patterns people expect from their handheld devices is a key approach for successfully integrating handhelds into smart environments.

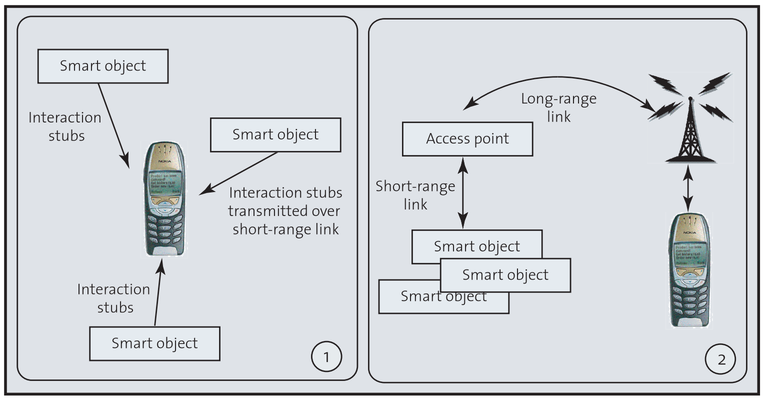
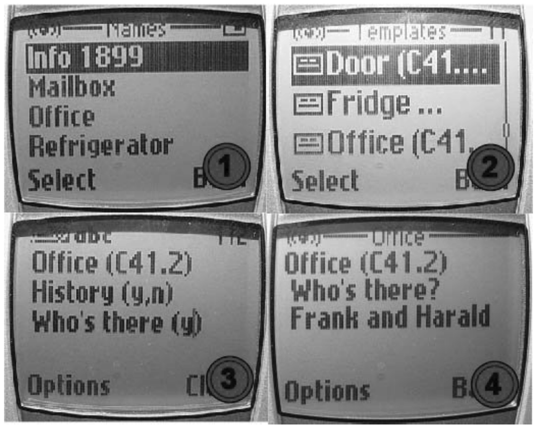


Fig. 5. Remote interaction with smart objects using handheld devices: when people are in range of smart objects, handhelds serve as *mobile storage medium* for interaction stubs (1); when far away, interaction stubs are executed to trigger interactions with remote objects using the handheld as *user interface* (2)

In the following, we illustrate how device specific interaction patterns like making phone calls can be used as a metaphor for implementing remote interactions with smart objects. Enabling remote interactions is a two step process: (1) when a user is in proximity of a smart object, it stores interaction stubs in the user's handheld device; (2) later, when not in vicinity of the object, a user selects a suitable interaction stub stored on the handheld to trigger a remote interaction with an augmented item (cf. Fig. 5).

The interaction stubs are the key mechanism to establish the remote communication link. They consist of a human readable name for a smart object, a set of commands that can be executed by it, and its address. In our current implementation, mobile phones serve as handhelds and the actual communication with a smart object takes place by exchanging SMS messages. Here, an interaction stub is composed of a phone book entry for the smart object and an SMS template. The phone book entry indicates the human readable name and the object's address, which is a telephone number in this case. The SMS template contains a range of predefined commands that can be activated and sent to the smart object (cf. Fig. 6).

We illustrate this approach with an office room as an example of a (rather large) smart object. The "smart office" knows who is currently working in it and what noise level is inside.



Office (C41.2)
History (y,n)
Who's there (y)

Office (C41.2)
Who's there?
Frank and Harald

Fig. 6. Interaction stubs transmitted from smart objects to a mobile phone (phone book entries (1), SMS templates (2)), an edited SMS template with activated command (3), and the corresponding reply from a remote smart object (4).

A BTnode equipped with several sensors, such as a microphone, is placed in the office and provides information about the noise level inside (cf. [1] for a description of the sensor boards used). Furthermore, according to the concept of weak identification we can infer from a handheld's presence who is currently in the room, and utilize the handheld as *weak user identifier*.

Given these capabilities, the smart office can keep track of entering and leaving persons, maintain a short history of events, and derive its current situational context (e.g., an ongoing meeting). Based on this information, interaction stubs (phone book entries and SMS templates) are transmitted to a user's handheld device. For example, the person most frequently in the office is given a special stub that allows her to remotely interact with the office after hours.

Interaction stubs (phone book entries and SMS templates) are transferred over a short-range wireless link between smart object and handheld (cf. Sect. 4). These transmissions are completely transparent to the user and are initiated by a smart object based on its current context and history information. Later, when people want to remotely interact with the smart office, they select the corresponding phone book entry from their phone book and compose a new SMS message using the appropriate SMS template. The SMS message is received by a stationary access point with a GSM gateway and relayed to the corresponding smart object in range of the access point. The smart object then executes the embedded commands and returns a message to the user's mobile phone (cf. Fig. 6). Consecutive messages can be exchanged between user and smart object. Besides the described SMS-based approach we have also implemented a similar solution based on WAP.

Direct remote interaction with a smart object requires a nearby stationary access point. The next section shows how we can get rid of this stationary gateway by using nearby handheld devices as *mobile infrastructure access points*.

6 The Smart Medicine Cabinet

The smart medicine cabinet illustrates a handheld's role as *mobile infrastructure access point*, *mobile storage medium*, and *user interface*. It was designed to improve the drug compliance of patients with chronic diseases by reminding them to take their medicine. It also knows about its contents and can be queried remotely with a WAP-enabled mobile phone. Interaction with the information technology inside the cabinet is implicit – i.e., transparent for the patients – who might not even know that the cabinet is "smart". By using small RFID tags attached to the folding boxes and an off-the-shelf medicine cabinet equipped with a BTnode connected to an RFID reader, the information technology becomes completely invisible to the users (cf. Fig. 7). When a patient removes a certain kind of medicine she needs to take during the day, the active tag in the cabinet establishes a connection through the user's Bluetooth-enabled mobile phone to a background infrastructure service querying it about prescription information concerning this medicine. The active tag then utilizes the user's mobile phone as *mobile storage medium* and stores a corresponding alarm and a calendar entry in it that are activated when the patient has to take the medicine.

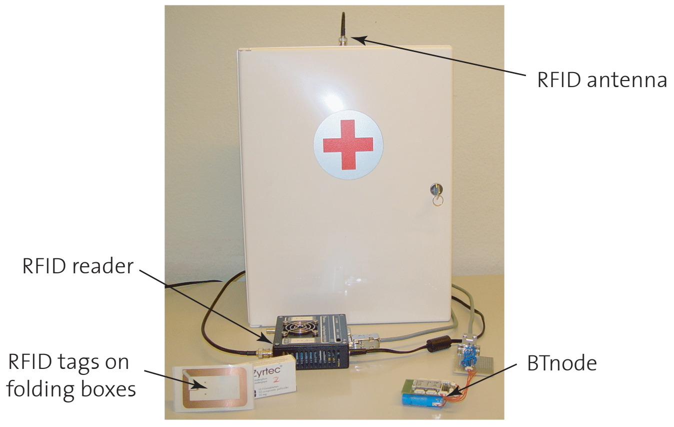


Fig. 7. The smart medicine cabinet

A WAP-based interface allows patients to remotely interact with the smart medicine cabinet – or its representation in the background infrastructure, the background infrastructure representative (BIRT) of the cabinet. Since the patient's mobile phone serves as *mobile infrastructure access point* for the cabinet, it operates in a disconnected mode whenever there is no mobile phone present. It hence requires a BIRT in the background infrastructure that represents the medicine cabinet continuously. Whenever a mobile phone is in the vicinity of the medicine cabinet and provides connectivity, the cabinet synchronizes its state with the BIRT. Afterwards, remote WAP-based queries need to address the BIRT of the cabinet. Information displayed on WAP pages regarding the contents of the cabinet and recently taken medicine therefore reflect the status of the cabinet during the last synchronization.

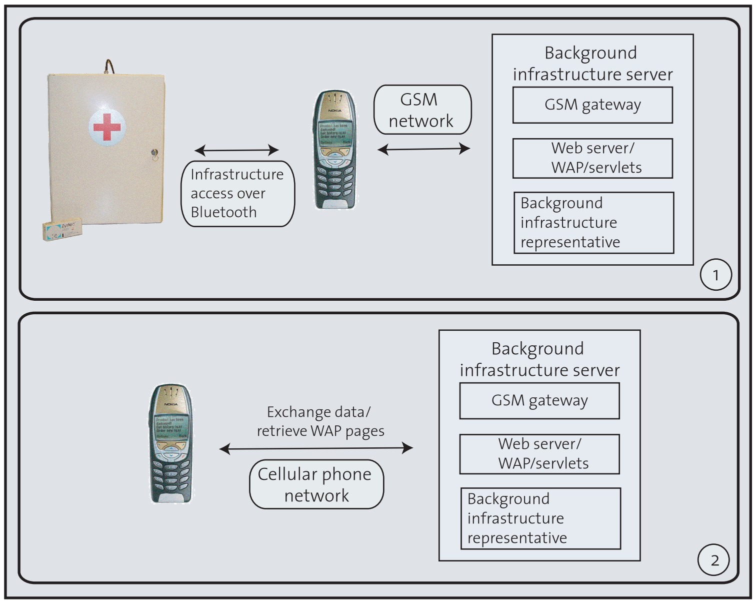

Infrastructure access over Bluetooth
GSM network
Background infrastructure server
GSM gateway
Web server/ WAP/servlets
Background infrastructure representative
1
Exchange data/ retrieve WAP pages
Cellular phone network
Background infrastructure server
GSM gateway
Web server/ WAP/servlets
Background infrastructure representative
2

Fig. 8. When the patient is in range of the cabinet her mobile phone serves as *mobile access point* to the background infrastructure server (1), when not in range of the cabinet she interacts with the background infrastructure representative of the cabinet using the handheld as *user interface* (2)

The following technologies have been incorporated into an ordinary medicine cabinet: (1) passive RFID tags on the folding boxes, (2) an RFID reader attached to the medicine cabinet, and (3) a BTnode that processes the information from the RFID reader and communicates via Bluetooth with a (4) mobile phone (cf. Fig. 7).

An actual use case would be the following: A patient approaches the smart medicine cabinet and takes a package out of the cabinet. The active tag in the cabinet notices that a box of medicine disappeared and connects to a background service (the BIRT of the cabinet). Thereby, the patient's mobile phone acts as a *mobile infrastructure access point* to the cellular phone network for the smart object, and no stationary access point is required near the cabinet. The Bluetooth-enabled active tag in the cabinet queries the BIRT about when the patient has to take the medicine. It then stores a corresponding calendar entry into her mobile phone that reminds the patient to take the medicine during the day. While there is a connection to the background infrastructure through the patient's mobile phone, the cabinet also synchronizes its state with that of the BIRT, which provides WAP pages based on this information. When the patient visits a pharmacist, the WAP interface can be used to query the contents of the cabinet (cf. Fig. 8). This information is a good basis for the pharmacist to decide whether another kind of medicine is compliant to that in the smart cabinet.

7 Spontaneous Integration of Handhelds into Smart Environments

The inventory monitoring application presented in this section illustrates a handheld's role as *remote resource provider* and *user interface*. As *remote resource provider*, a handheld provides data storage capabilities and serves as a platform for executing computations on behalf of smart objects. The possibility to outsource computations to nearby handheld devices also allows us to transmit sophisticated *user interfaces*, which facilitate the local user interaction with smart objects.

As already described in Sect. 4, a handheld's role as *remote resource provider* is based on a distributed tuple space implementation. In a typical pervasive computing environment, smart objects need to collaborate in order to implement an application. Such collaborating entities form a distributed tuple space as a shared data structure distributed over participating entities (and without the use of a background infrastructure). Smart objects write data (e.g., perceived sensory data) into the tuple space, read data from it, process these data, and write the corresponding result again into the space. Thereby, the origin of data becomes transparent for participating objects when it is not explicitly coded into tuples. Consequently – and this is the major point – the location where code is executed becomes irrelevant because it operates on the same data basis regardless of the node in the distributed tuple space chosen for code execution. In order for a handheld to serve as resource provider for smart objects, it joins their distributed tuple space and receives code from these objects. As likewise previously mentioned (cf. Sect. 4) resource-aware tuple space operations automatically use the memory capacity of a handheld after it has joined the tuple space.

We have implemented the idea of using handhelds as *remote resource provider* and *user interface* for local interactions with smart objects in a software framework called *Smoblets*. The main goals of Smoblets are to enable interactions with smart objects without the need of a supporting backend infrastructure, to outsource computations to handheld devices in order to save energy, and to foster the collaboration among smart objects and handheld computers.

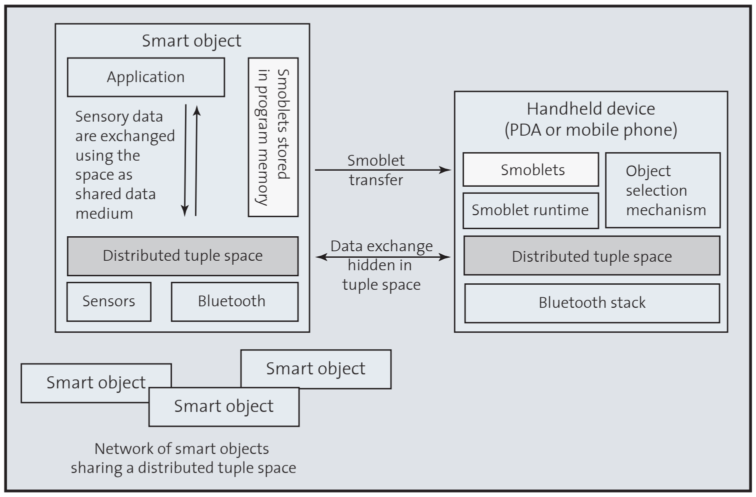

Smart object
Application
Sensory data are exchanged using the space as shared data medium
Smoblets stored in program memory
Distributed tuple space
Sensors
Bluetooth
Smoblet transfer
Data exchange hidden in tuple space
Handheld device (PDA or mobile phone)
Smoblets
Smoblet runtime
Object selection mechanism
Distributed tuple space
Bluetooth stack
Smart object
Smart object
Smart object
Network of smart objects sharing a distributed tuple space

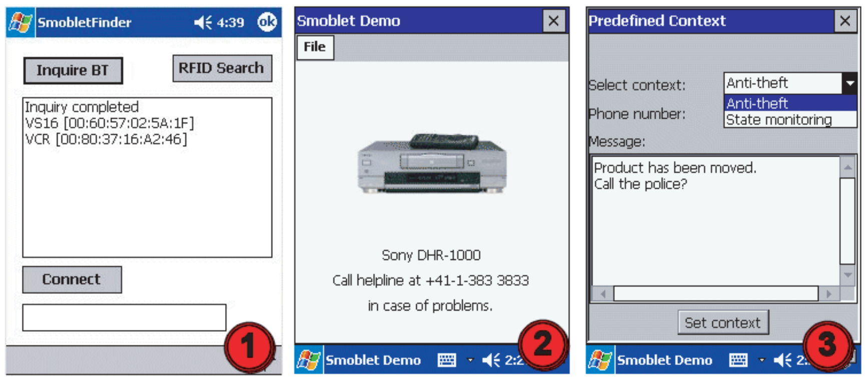
Fig. 9. The Smoblet concept: ad hoc interaction with nearby handheld devices

The main components of a Smoblet system are (1) a set of Java classes – the actual Smoblets – stored in the program memory of an active tag (a BTnode in our case), (2) a runtime environment for the execution of Smoblets on a handheld device, (3) a mechanism to select smart objects and to transfer Smoblets from smart objects to handheld devices, and (4) a distributed tuple space [5] implementation that serves as a shared content-based data structure for participating entities (cf. Fig. 9). It is important to note that smart objects themselves cannot execute Java code; their program memory merely serves as storage medium for the code. Java classes can only be executed on nearby handheld devices, which communicate with the smart objects using a distributed tuple space abstraction.

A Smoblet transmission can either be initiated automatically by smart objects or manually triggered by human users. To manually initiate the transmission of Smoblets from a smart object to a handheld device, the user selects a smart object for interaction by means of a small program on her handheld. The device address of the object is then used to establish a connection and to retrieve the Java classes from the smart object.

Together with the Smoblets, their functionality is moved and therefore outsourced from a smart object to a handheld device. Because of the distributed tuple space the actual source of data becomes transparent for the Smoblets. Although they are executed on the handheld they can operate on the data provided by the object they are originating from and perform computations on its behalf that were not feasible because of the object's limited resources. Outsourcing computations from smart objects and using handhelds as remote resource provider is the core motivation for the Smoblet idea.

We have created a Java framework that simplifies the development of Smoblets. It provides methods to access data stored in a distributed tuple space, which can be used to transparently access remote sensor data of smart objects via Bluetooth. The distributed tuple space implementation provides a convenient high-level communication abstraction to the application programmer, who does not have to care about low-level communication issues. The tuple space also detaches Smoblets from any particular communication technology that is being used by the smart objects.



```
Inquiry completed
VS16 [00:60:57:02:5A:1F]
VCR [00:80:37:16:A2:46]
```

Sony DHR-1000
Call helpline at +41-1-383 3833
in case of problems.

Select context: Anti-theft (Anti-theft / State monitoring)

Message: Product has been moved. Call the police?

Fig. 10. A typical Smoblet interaction – after a user has searched for smart objects (1), Smoblets are transferred to a handheld device and offer a small user interface (2), which allows users to graphically interact with active tags and to access data shared through the distributed tuple space (3)

To illustrate the idea of Smoblets, we have implemented an inventory monitoring application (cf. Fig. 10). Here, BTnodes together with sensor boards are attached to expensive products – in our case a video cassette recorder (VCR), but it could also be a bottle of expensive wine, a book, etc. – in order to notify its owner or another person when it is being stolen or damaged. Smoblets stored on the smart video cassette recorder allow a user to customize the behaviour of the product.

The scenario is as follows: When nearby, a person selects a smart object by using the SmobletFinder application, which triggers the transmission of the corresponding Java code from the selected object to her handheld device (cf. Fig. 10). The code contains a small user interface for adapting the behaviour of the smart object. Here, a user can specify a telephone number and associate messages with certain situations the product can be in. For example, in case of damage a notification message must be sent to a nearby repair service. The user input (e.g., the telephone number and the message text entered into the user interface) is embedded into a tuple and written into the underlying distributed tuple space. As the smart object that provided the user interface is also a member of the tuple space, it can access the information input by the user. In our implementation, the smart object registers a callback on tuples that represent relevant user input. Is a corresponding tuple is written into the space the smart object is automatically notified by the tuple space implementation. Therefore, the smart object can immediately react on the user input and adapt its behaviour accordingly. When the user closes the application the handheld leaves the distributed tuple space.

8 Conclusion

This paper formulated and supported the hypothesis that smart objects can provide increasingly sophisticated services to users in smart environments when they are able to exploit the capabilities of nearby handheld devices in an ad hoc fashion. We identified six roles that describe how smart objects can spontaneously make use of nearby handhelds: (1) as mobile infrastructure access point, (2) as user interface, (3) as remote sensor, (4) as mobile storage medium, (5) as remote resource provider, and as (6) weak user identifier. We then presented an implementation of these roles tailored towards everyday objects which have been augmented with Bluetooth-enabled sensor nodes. Finally, three applications – a remote interaction scenario, a smart medicine cabinet, and an inventory monitoring application – illustrated some of the usage scenarios that become feasible when smart objects are able to spontaneously collaborate with nearby handheld computers. Table 2 summarizes the roles of handhelds in the presented applications.

Table 2. The roles of handhelds in the applications presented in this paper

Application	**Handheld's role**
Remote interaction	Mobile storage medium, user interface, weak user identifier
Smart Medicine Cabinet	Mobile infrastructure access point, mobile storage medium, user interface
Inventory monitoring	Remote resource provider, user interface

Considering the presented applications, the basic function of handheld devices is to mediate between smart objects and infrastructure services, between smart objects and their users, and among smart objects themselves. For example, in the Smart Medicine Cabinet application, a patient's mobile phone serves both as *mobile infrastructure access point* (i.e., it enables smart objects to communicate with background infrastructure services) and as *primary user interface* (i.e., it facilitates the user interaction with smart objects). As a *remote resource provider* a handheld provides data storage capabilities and serves as platform for outsourcing computations from smart objects. By providing resources for smart objects, handhelds enrich smart object's computational abilities and their interaction among each other.

References

1. M. Beigl and H. W. Gellersen. Smart-Its: An Embedded Platform for Smart Objects. In *Smart Objects Conference (SOC) 2003*, Grenoble, France, May 2003.
2. M. Beigl, H.W. Gellersen, and A. Schmidt. MediaCups: Experience with Design and Use of Computer-Augmented Everyday Objects. *Computer Networks, Special Issue on Pervasive Computing*, 25(4):401–409, March 2001.
3. J. Beutel, O. Kasten, F. Mattern, K. Roemer, F. Siegemund, and L. Thiele. Prototyping Sensor Network Applications with BTnodes. January 2004. Accepted for publication, IEEE European Workshop on Wireless Sensor Networks (EWSN 04).
4. N. Davies, S. Wade, A. Friday, and G. Blair. Limbo: A Tuple Space Based Platform for Adaptive Mobile Applications. In *Proceedings of the International Conference on Open Distributed Processing/Distributed Platforms (ICODP/ICDP '97)*, pages 291–302, Toronto, Canada, May 1997.
5. D. Gelernter. Generative Communication in Linda. *ACM Transactions on Programming Languages and Systems*, 7(1):80–112, January 1985.
6. H. W. Gellersen, A. Schmidt, and M. Beigl. Multi-Sensor Context-Awareness in Mobile Devices and Smart Artifacts. *Mobile Networks and Applications (MONET)*, 7(5):341–351, October 2002.
7. Stephan Hartwig, Jan-Peter Strömann, and Peter Resch. Wireless Microservers. *Pervasive Computing*, 1(2):58–66, 2002.
8. T. Kindberg et al. People, Places, Things: Web Presence for the Real World. In *WMCSA 2000*, Monterey, USA, December 2000.
9. A. Schmidt, F. Siegemund, M. Beigl, S. Antifakos, F. Michahelles, and H. W. Gellersen. Mobile Ad-hoc Communication Issues in Ubiquitous Computing. In *Personal Wireless Communication (PWC 03)*, September 2003.
10. F. Siegemund and C. Floerkemeier. Interaction in Pervasive Computing Settings using Bluetooth-enabled Active Tags and Passive RFID Technology together with Mobile Phones. In *IEEE Intl. Conference on Pervasive Computing and Communications (PerCom 2003)*, pages 378–387, March 2003.
11. 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, USA, May 1999.
12. M. Weiser and J. S. Brown. The Coming Age of Calm Technology, October 1996.