Deploy Spontaneously: Supporting End-Users in Building and Enhancing a Smart Home

Fahim Kawsar, Tatsuo Nakajima
Department of Computer Science
Waseda University, Tokyo, Japan
{fahim,tatsuo}@dcl.info.waseda.ac.jp

Kaori Fujinami
Department of Computer, Information and Communication Sciences
Tokyo University of Agriculture and Technology, Tokyo, Japan
fujinami@cc.tuat.ac.jp

ABSTRACT

This paper explores system issues for involving end users in constructing and enhancing a smart home. In support of this involvement we present an infrastructure and a tangible deployment tool. Active participation of users is essential in a domestic environment as it offers simplicity, greater user-centric control, lower deployment costs and better support for personalization. Our proposed infrastructure provides the foundation for end user deployment utilizing a loosely coupled framework to represent an artefact and its augmented functionalities. Pervasive applications are built independently and are expressed as a collection of functional tasks. A runtime component, FedNet maps these tasks to corresponding service provider artefacts. The tangible deployment tool uses FedNet and allows end users to deploy and control artefacts and applications only by manipulating RFID cards. Primary advantages of our approach are two-fold. Firstly, it allows end users to deploy ubicomp systems easily in a *Do-it-Yourself* fashion. Secondly, it allows developers to write applications and to build augmented artefacts in a generic way regardless of the constraints of the target environment. We describe an implemented prototype and illustrate its feasibility in a real life deployment session by the end users. Our study shows that the end users might be involved in deploying future ubicomp systems if appropriate tools and supporting infrastructure are provided.

Categories and Subject Descriptors: D.2.11 [Software Engineering]: Software Architectures

Keywords: Augmented Artefact, Pervasive Application, System Infrastructure, End User Deployment

INTRODUCTION

One of the consequences of the convergence of ubicomp technologies is the integration of processors and sensors into everyday objects resulting in the emergence of innovative pervasive applications. We envision that this trend will eventually bring these applications into our home. A pervasive application usually involves sensors, instrumented everyday objects (augmented artefacts), mobile devices, displays, etc. Ideally, these applications should be similar to the home appliances e.g., a table lamp, a dish washer, a TV etc. and should be easy to setup, adaptive to users' needs, and interchangeable with new models. A user may buy one or multiple physical artefacts and applications for them and should be able to install them just like other home appliances. In addition, one can incrementally enhance the artefact functionalities by upgrading its features or installing new applications e.g., consider a hypothetical table lamp application that proactively turns the lamp on and adjusts its brightness adapting ambient room lighting. A user can initially buy a regular lamp, and few weeks later he/she can buy a light sensor, attach it to the lamp and download this application into the lamp to make it proactive. There are several advantages in involving end users to form a smart home in this fashion as observed by Beckmann and his colleagues including less cost, greater user-centric control, more acceptance, better personalization and frequent upgrade support [3].

Involving end users in the deployment process requires the development of plug and play artefacts that are deployable in a *Do-It-Yourself (DIY)* fashion. Similarly pervasive applications should be built in such a way that they can use any compatible instrumented artefact. In addition, the installation process of these applications has to be seamless without complex configuration and administration. These issues raise two requirements: i) a general infrastructure for building plug and play augmented artefacts and generic pervasive applications and ii) a simple, easy to use tool that allows end users to deploy the artefacts and the applications. Although these two requirements (infrastructure and user interface) are contrasting, it is hard to draw a clear distinction because both are tightly coupled and complement each other for supporting end user deployment. That is, without having a proper infrastructure the user interface will be inadequate to support the deployment process and vice versa.

To address these requirements, we present an infrastructure that provides the foundation for end user deployment using an artefact framework. This framework represents an instrumented artefact by encapsulating its augmented functionalities (e.g., proactivity of the table lamp) in one or multiple service profiles atop a core and allows additions of profiles incrementally. Applications in our approach are represented as a collection of implementation independent functional tasks. These tasks are atomic actions that represent the artefacts' services, e.g., *"sense current light sensitivity"*, *"turn on the lamp"*, etc. An infrastructure component Fed-

Net, manages these applications and artefacts and maps the task specifications of the applications to the underlying artefacts' services by matching respective documents (that express the applications and the artefacts). A tangible deployment tool uses FedNet and supports the deployment activity. Each application and artefact comes with an RFID card that end users manipulate to install and control them. The combination of the infrastructure and the deployment tool enables end users to build and enhance a smart home. Consequently, the contributions of this paper are two-fold: i) an architecture that provides the foundation for involving end users in the deployment process, and ii) a tangible deployment tool that supports the end user deployment activity.

In the next section we justify end users involvement in the deployment process. Then, we present the design aspects, followed by the technical detail of our approach. Next, we proceed to the feasibility of our solution by using real life deployment examples. Although, the end user experiment positively evaluated our system's support for the deployment activity, it revealed some usability issues. We report these findings along with the experiment descriptions. Finally, we position our research with respect to the related work and conclude the paper.

INVOLVEMENT OF END USERS

The rapid proliferation of ubicomp technologies makes it essential to understand how to place and manage ubicomp systems into the environment. This is particularly important for the home where the dwellers have a greater control. One essential property of our home is its evolutionary nature and receptibility to continual change [23]. We incrementally organize our homes with furniture and appliances according to our preferences and styles. Previous studies have shown how end users continuously reconfigure their homes and technologies within it to meet their demands [21, 23]. Edwards et al. observed that the networked home of the future will emerge in a piecemeal fashion [10]. To support the evolutionary nature of our homes it is essential that ubicomp systems support the incremental deployment. Ideally, deployment should be carried out by the end users. The end users have in-depth knowledge of the structure of their home and their activities, resulting in a better understanding of where and which physical artefact and application to deploy. Furthermore, involving end users in the process leads to higher acceptability and a greater feeling of having control due to their active participations. It also reduces deployment cost as professional assistance is not needed. Considering these factors, we reckon that the end users will be deploying ubicomp systems at their home by themselves in the future. The work presented in this paper supports this notion.

DESIGN ISSUES

A central issue to support end user deployment is a suitable infrastructure that enables isolated development of plug an play artefacts and applications. To date several infrastructures are proposed in the ubicomp literature. However, these systems' supports are not compelling to enable end users in deploying smart artefacts and applications. Consider, Figure 1 where four different use cases for a ubicomp environment are shown. In case 1, artefacts are stand-alone providing

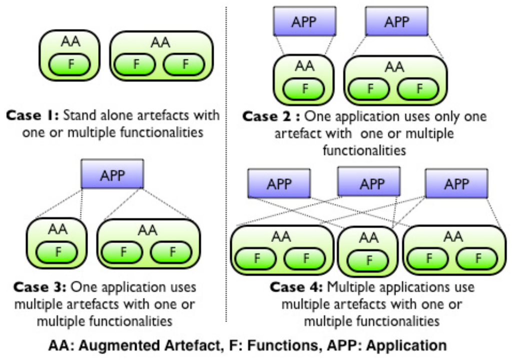

AA
F
APP
Case 1: Stand alone artefacts with one or multiple functionalities
Case 2 : One application uses only one artefact with one or multiple functionalities
Case 3: One application uses multiple artefacts with one or multiple functionalities
Case 4: Multiple applications use multiple artefacts with one or multiple functionalities
AA: Augmented Artefact, F: Functions, APP: Application

Figure 1. Different use cases for ubicomp applications

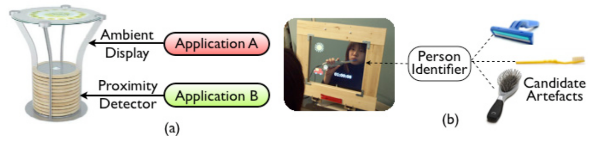


Figure 2. A single artefact with multiple roles and multiple artefacts with similar roles

a single or multiple built-in functions without any applications. Whereas in cases 2-4 three different modalities of application association are shown. Although the latter cases are supported by existing infrastructure [6,14,24,25] by providing a wrapper that is tightly glued with the rest of the infrastructure, but they have no clean support in case 1. Artefacts are inherently dependent on the infrastructure and cannot run in a stand alone mode. Also, to use augmented artefacts, applications are bound to follow the infrastructure semantics. These constraints limit the development of artefact and applications independent to infrastructure which makes it difficult for casual users to deploy ubicomp systems.

Another interesting aspect is the augmentation modalities of the artefacts. Augmentations depend on the designer's intuition and it is hard to confine the augmentation scope. Consider Figure 2, depicting two ideal situations, a) a single everyday artefact capable of playing multiple functional roles and b) multiple artefacts sharing an identical functional role. In Figure 2(a) we have a smart table providing two supplementary functions: an ambient display and a proximity detector. In Figure 2(b) we have a mirror whose display functionality can be triggered by any of the three augmented artefacts, e.g., a toothbrush, a comb or a razor. The suitable augmentation of these artefacts depends on the underpinned scenario, regardless of the multiple functionalities that can be afforded. Existing infrastructures provide a widget notion to encapsulate the artefacts [6]. However, such widgets are not capable of hosting augmented features in a plug and play manner. Adding a new feature to an existing artefact requires regeneration of the widget. This limits an end user to augment features of an existing artefact thus hindering the DIY support.

The final aspect related to the end user deployment is the tool to install artefacts, applications and enhance their functionalities gradually. Several researchers looked at the simple rule based authoring tool and *programming by examples* for the end users to configure an applications' proactive behaviors like [7,8,22]. However, these tools are not suitable for casual users with no technical background. As previous studies have shown, the deployment process has to be very simple with minimal configuration complexities [1,3]. Also, the process should resemble the current practices as closely as possible with which the end users are familiar, e.g., installing a home appliance like a washing machine, a microwave etc. These observations leads us to the following design decisions:

1. **DIY Instrumented Artefacts:** Artefacts should be reusable and augmented features should not be tightly coupled with the artefact. We need to package the artefact in a generic executable so that the end users can deploy them. We have designed a loosely coupled framework to represent an artefact and its augmented functionalities. Each augmented functionality is termed as a *profile* in our framework and profiles can be added to an artefact incrementally. The separation of artefacts and profiles enables DIY support, e.g., an artefact can be instrumented with any suitable profile.

2. **Infrastructure Independent Application:** Applications should be developed considering the functionalities only. To make an application independent of the infrastructure, it is imperative to know an application's runtime requirement ahead of the execution. Furthermore, the application needs a generic access mechanism to interact with the environment. We have addressed these challenges by representing an application as a collection of functional tasks written in a task description file and allowing an application to access the artefact services using popular web techniques (SOAP for push and RSS Feed for pull). We have opted to use SOAP/RSS over other event driven models since it makes an application less dependent on the infrastructure as the developers do not need to use the infrastructure specific APIs.

3. **Spontaneous Federation:** For creating a spontaneous federation among the applications and the artefacts, FedNet provides the runtime intermediation as shown in Figure 3. It is essential to understand the semantics of the movable data pattern ahead of the execution for such spontaneous federation. FedNet analyzes the task description file to extract the service requirements and then maps these tasks to underlying service provider artefacts by matching artefact description files. FedNet then assigns a generic intermediation component to the application that allows the application to access the services of the artefacts.

4. **Tangible End User Deployment Tool:** The DIY deployment process of a ubicomp system requires two tasks: i) installing a ready-to-run instrumented artefact or attaching sensors or actuators to an existing artefact, and ii) installing applications and associating the application with an artefact when necessary. We have adopted a tangible interface for supporting these tasks. Each artefact, profile or application is disseminated in a generic binary and comes with a corresponding RFID card. The end users can install/uninstall, run/stop artefacts and applications by using these cards and the deployment tool. Furthermore, they can associate a profile or an application to a specific artefact.

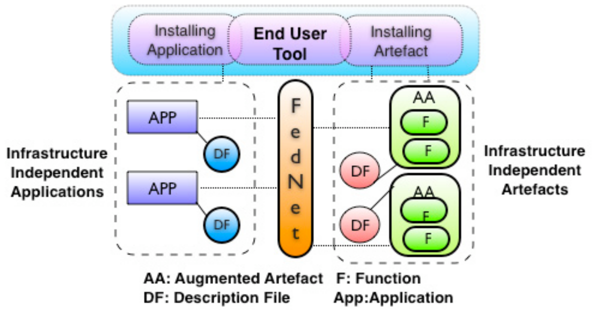


Figure 3. Basic workflow of our approach

SYSTEM DETAIL

In this section, we present the artefact framework followed by the task centric application framework. Then, we show how FedNet utilizes these frameworks to create a spontaneous association between the artefacts and the applications. The frameworks and FedNet are implemented in Java.

Artefact Framework

Artefact framework provides a layered architecture where basic artefact functionalities are combined in a core component. Additional augmented features can be added as plugins into the core. Each augmented feature is called a profile in our approach. These profiles are artefact independent and represent a generic service, For example: sensing room temperature could be one profile, and multiple artefacts (e.g., a window, an air-conditioner, etc.) can be augmented with a thermometer for supporting this profile.

Internal Architecture of Artefact Framework

The internal architecture of the artefact framework is shown in Figure 4 and consists of the following:

1. **Core Component:** Typically instrumented artefacts have some common characteristics e.g., capable of communication [4,26], provides perceptual feedback, possesses memory etc. The core component of the artefact framework encapsulates all these functionalities. The communication module facilitates communication support and encapsulates the transport layer where as the discovery module allows service advertisement. The notification module enables the rest of the modules to indicate their status. The artefact memory contains property data, profile descriptions, and other temporal data. The client handler is the request broker for services and delegates the external requests to specific profiles. Finally, the profile repository hosts the array of profiles. The profile repository has dynamic class loaders to load the profiles dynamically when requested. The entire core is packaged in an executable binary and runs independently.
2. **Profile:** Each profile represents a specific functionality and implements the underlying logic of the functions, e.g., providing context by analyzing the attached sensors' data (e.g., room temperature) or actuating an action by changing the artefacts' state (e.g., increasing the lamp brightness etc.). Each profile is of type sensor or actuator and has a profile handler, a template to plug-in device code and context calculation or service actuation logic. The profile handler has an abstraction layer that hides the heterogeneity of the underlying devices.

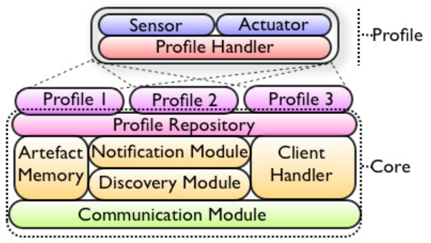
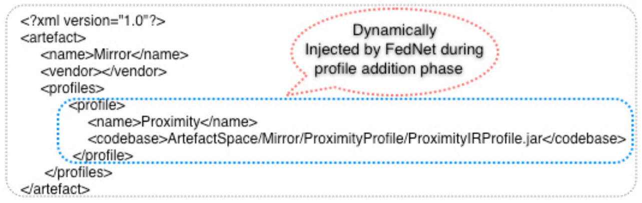


Figure 4. Architecture of Artefact Framework


Dynamically Injected by FedNet during profile addition phase

```
<?xml version="1.0"?>
<artefact>
    <name>Mirror</name>
    <vendor></vendor>
    <profiles>
      <profile>
         <name>Proximity</name>
         <codebase>ArtefactSpace/Mirror/ProximityProfile/ProximityIRProfile.jar</codebase>
      </profile>
    </profiles>
</artefact>
```

Figure 5. Artefact Description File for a Mirror with *Proximity Profile*

Documents to represent Artefacts

The artefact framework's core is packaged as a ready-to-run binary with a description document called Artefact Description File (ADF) as shown in Figure 5 for a mirror artefact with a *Proximity Profile*[1]. Profiles are packaged as plug-ins with a Profile Description File (PDF) (Figure 6) that run atop the core. The PDF specifies the data semantics of the corresponding profile and contains a *detector* or an *actuator* node based on the profile type. The sensor profile's description follows the specification of the Sensor Modeling Language (SensorML) [20] (Figure 6(a)). The primary strengths of SensorML are its soft typed attribute, reference frame and parameters, with which the semantics of different sensor data platforms can easily be understood and interchanged. For an actuator profile[2], our custom designed XML based *Artefact Control Language (ACL)* is used (Figure 6(b)) where the *state* attribute is used to abstract the operational states of the artefacts. It contains the input parameters to change the states along with their data type. PDF also contains a quality of service(QoS) block which specifies profile's quality. Furthermore, these files contain an *installation-instruction* block that provides hardware installation guidelines.

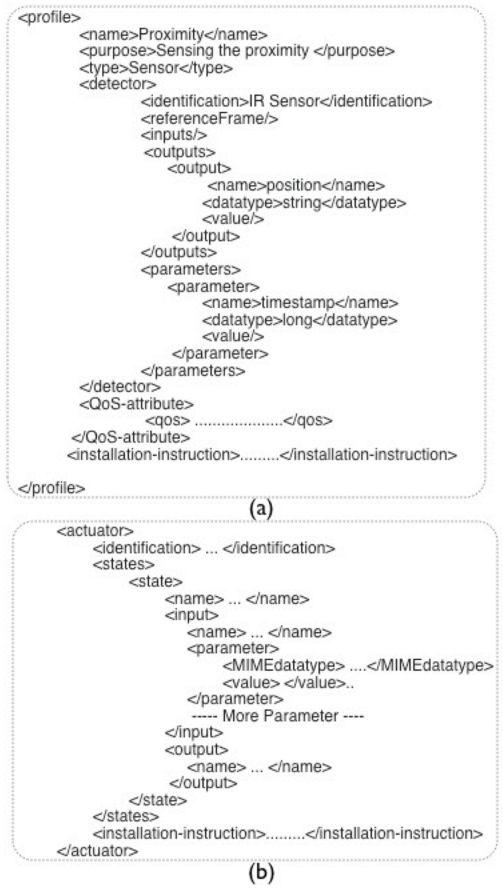
Task-Centric Application Framework

An application is expressed as a collection of functional tasks independent of the implementation. This specification allows FedNet to map these tasks to respective service provider artefacts. An application developer can follow any library and implementation language to code the execution logic. The two things necessary to work in a FedNet environment are the task specification, and the generic access mechanism.

Any application is composed of several functional tasks, i.e., atomic actions. In ubicomp applications, these atomic actions may be: *"turn the air-conditioner on"*, *"sense the proximity of an object"* etc. An application is expressed as a collection of such functional tasks in a Task Description File

[1]Proximity Profile's sole purpose is to recognize the presence of an object in front of the artefact.

[2]Please note that the protocol to handle the underlying device is implemented in the profile implementation.



```
<profile>
        <name>Proximity</name>
        <purpose>Sensing the proximity </purpose>
        <type>Sensor</type>
        <detector>
                <identification>IR Sensor</identification>
                <referenceFrame/>
                <inputs/>
                <outputs>
                    <output>
                         <name>position</name>
                         <datatype>string</datatype>
                         <value/>
                    </output>
                </outputs>
                <parameters>
                    <parameter>
                         <name>timestamp</name>
                         <datatype>long</datatype>
                         <value/>
                    </parameter>
                </parameters>
        </detector>
        <QoS-attribute>
                <qos> ...................</qos>
        </QoS-attribute>
      <installation-instruction>........</installation-instruction>

</profile>
```

(a)

```
<actuator>
     <identification> ... </identification>
     <states>
          <state>
               <name> ... </name>
               <input>
                    <name> ... </name>
                    <parameter>
                         <MIMEdatatype> ....</MIMEdatatype>
                         <value> </value>..
                    </parameter>
                    ----- More Parameter ----
               </input>
               <output>
                    <name> ... </name>
               </output>
          </state>
     </states>
     <installation-instruction>........</installation-instruction>
</actuator>
```

(b)

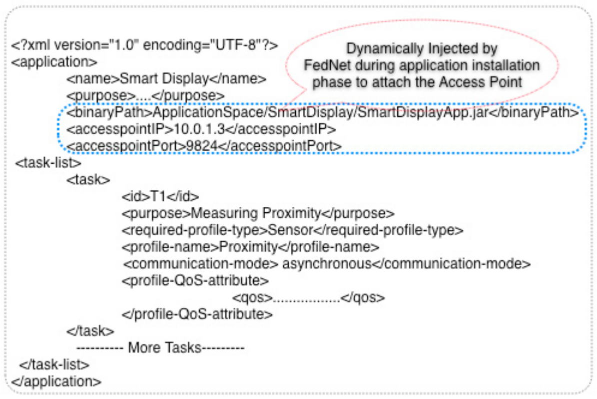
Figure 6. (a)Profile Description File for *Proximity Profile*, SensorML is used in the *detector node*. (b) Artefact Control Language is used for actuator profile

(TDF). Each task specifies the respective profiles it needs to accomplish its goal. Figure 7 shows part of the task description file for a smart display application. Each task contains Quality of Service (QoS) requirements for the target profiles.

The second requirement for an application is to use generic web protocols to access artefact services. During application installation in FedNet, an access point is assigned to the application. An application needs to access this point to send requests and receive responses from the underlying artefacts. In our current implementation the application uses a SOAP request for polling or sending an actuation request to the artefacts. For continuous polling (i.e., subscription), auto discoverable RSS feeds are used. During the application's instantiation time, the required physical artefacts data semantics (*detector* and *actuator* nodes of the Profile Description File) are send to the application by FedNet, to let the application prepare for the moveable data accordingly. In the current implementation, we have provided a simple library in Java comprised of a SOAP Client and Auto Discoverable RSS Parser for the application developer.

FedNet System

FedNet provides the runtime association between the applications and the artefacts by utilizing respective documents. FedNet is composed of four components (Figure 8).



```
<?xml version="1.0" encoding="UTF-8"?>
<application>
        <name>Smart Display</name>
        <purpose>....</purpose>
        <binaryPath>ApplicationSpace/SmartDisplay/SmartDisplayApp.jar</binaryPath>
        <accesspointIP>10.0.1.3</accesspointIP>
        <accesspointPort>9824</accesspointPort>
 <task-list>
        <task>
                <id>T1</id>
                <purpose>Measuring Proximity</purpose>
                <required-profile-type>Sensor</required-profile-type>
                <profile-name>Proximity</profile-name>
                <communication-mode> asynchronous</communication-mode>
                <profile-QoS-attribute>
                                <qos>................</qos>
                </profile-QoS-attribute>
        </task>
        --------- More Tasks---------
  </task-list>
</application>
```

Dynamically Injected by
FedNet during application installation
phase to attach the Access Point

Figure 7. Task Description File (partly) for a smart display application

1. **Application Repository** hosts all the applications that run on FedNet. During an application's deployment, the binary executable and the Task Description File (TDF) are submitted to this repository. FedNet Core generates the an access point for the application and updates the respective TDF by dynamically injecting the identity of the corresponding access point as shown in Figure 7.
2. **Artefact Repository** manages all the artefacts running in FedNet environment. During artefacts' deployment, the executable binary implementing the artefact framework and the Artefact Description File (ADF) are submitted to this repository. When a profile is added to an artefact, the profile information is dynamically injected into ADF as shown in Figure 5 and the respective profile is attached to the artefact.
3. **FedNet Core** provides the foundation for the runtime federation. When an application is deployed the task specification is extracted from the application repository by the FedNet Core. It analyzes the task list by querying the artefact repository and generates an appropriate template of the federation and attaches it into a generic access point component for that application. When an application is launched, the access point is instantiated and the respective template is filled by the actual artefact available in the environment right at that moment thus forming a spontaneous federation.
4. **Access Point** represents the physical environment needed by an application. Since each application's artefact requirement is different and each application might not be running all the time, FedNet assigns a unique access point for each application; meaning multiple federations of artefacts can co-exist in the environment. Simultaneously, each artefact can participate in multiple federations. When an application is launched, the access point sends the federated artefacts data semantics, i.e., SensorML and ACL to the application. This allows an application to know the semantics of movable data in advance. From then on, the application delegates all its requests to the access point which in turn forwards them to the specific artefact. The artefacts' responses to these requests by providing their profile outputs either by pushing the environment state (actuation) or pulling the environment states (sensing) back to the access point that are fed to the application.

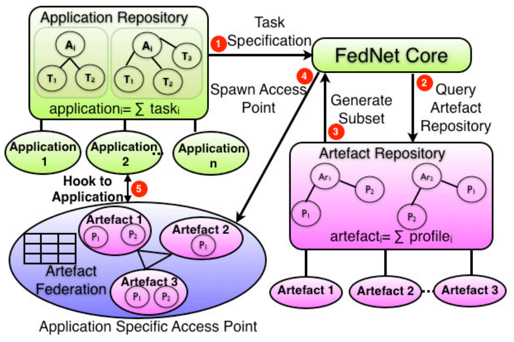


Figure 8. Architecture of FedNet

DEPLOYMENT TOOL FOR END USERS

To support end users in the deployment process, a tool is needed to install the artefacts and applications into the corresponding repositories, and to add profile plug-ins into the artefacts. Furthermore, this tool should enable the end users to control these artefacts and applications. We provide a tangible interaction tool for supporting these tasks. There are several user-centric design rationales behind our decision.

1. **Tangible Interface:** In the earlier stage of this work we provided a web based GUI tool for the deployment process. However, our pilot user study revealed several usability problems of such GUI as it contradicts end users' conceptual model of installing home appliances. Although, the deployments tasks were identical to regular desktop computing, e.g., software installation, the end users found it difficult to comprehend and suggested that the the process has to be more mechanical and tangible. Also, the GUI suffered from fragmentation of attention as users had to switch back and forth from GUI to physical artefact. Considering their suggestions we have designed a tangible interface for the deployment purpose. RFID (representing artefacts, application and profiles) and RFID Reader with touch buttons are provided for end users' interaction.
2. **Contextual Feedback and Guidelines:** It is necessary to provide feedback of users' actions and to guide the users with proactive suggestions [1]. We have provided visual (blinking LEDs) and sound feedback along with speech guidelines. These are contextual, i.e. the possible next steps are provided based on the user's latest activity and the state of the system.
3. **Semantic Mapping of User Actions:** To support distributed deployment with a centralized tool we need to identify which action is for which entity (artefact, application and profiles) i.e., we need a selection phase followed by the action phase. We have adopted RFID cards for the selection phase thus resolving the identity and provided touch buttons to resolve the actions.

Hardware

The deployment tool is built with an RFID reader, 3 touch buttons and 3 LEDs (Figure 9(d)). The touch buttons and LEDs are connected to an interface kit. The whole unit has a USB interface and can be connected to a regular PC/laptop that provides the power, audio output and controls the unit.

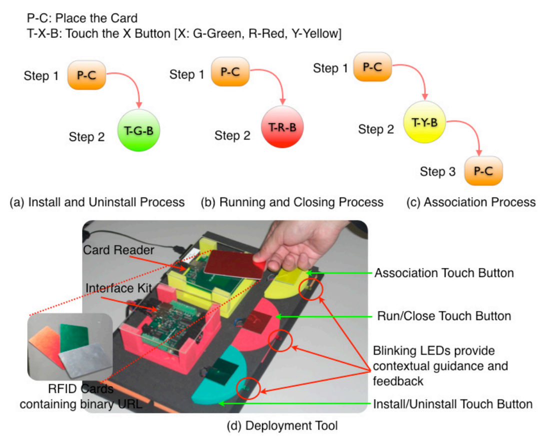

P-C: Place the Card
T-X-B: Touch the X Button [X: G-Green, R-Red, Y-Yellow]
Step 1
P-C
Step 2
T-G-B
Step 1
P-C
Step 2
T-R-B
Step 1
P-C
Step 2
T-Y-B
Step 3
P-C
(a) Install and Uninstall Process
(b) Running and Closing Process
(c) Association Process
Card Reader
Interface Kit
RFID Cards containing binary URL
Association Touch Button
Run/Close Touch Button
Blinking LEDs provide contextual guidance and feedback
Install/Uninstall Touch Button
(d) Deployment Tool

Figure 9. (a-c) Deployment Steps (d)End User Deployment Tool

Interaction Mechanism

We have mentioned that each application, artefact and profile is disseminated as an executable binary. We assume that each of these is packaged with an RFID card that embeds a remote URL from where the binaries can be collected. There are six functions (organized in 3 groups) that the end users can perform using these cards and the deployment tool.

- *Install/Uninstall an Artefact/Application:* These functions can be performed by placing the RFID card of the artefact/application and then pressing the green button (Figure 9(a)). During installation, the binary is downloaded from the remote URL embedded in the RFID[3].

- *Runing/Closing an Artefact/Application:* These are performed by placing the RFID card of the artefact/application and then pressing the red button (Figure 9(b)). Although most of the pervasive applications are assumed to run continuously, we have provided these explicit controlling functionalities to the end users since there is no secondary user interface in our system. Thus if a user needs to stop an application or artefact's functionality for some reason (e.g., going for a vacation), they can do so using these functions.

- *Associating/Removing a Profile/Application:* These functions are needed to associate a profile or an application to a specific artefact. Note that the applications that are not associated with any specific artefact can be installed following Figure 9(a). However, for the applications and profiles that are specific to one artefact an association phase is needed. This is done by placing the target artefact card first into the reader, touching the yellow button and then placing the newly installable profile's or application's card into the reader (Figure 9(c)). If a profile description file contains hardware installation instructions (Figure 6), the tool provides audio guidelines to support the installation.

Since each of the RFID cards represents the corresponding artefact or applications, these cards are used for controlling (running/closing) the artefacts and applications after the deployment process is finished.

[3]In the current implementation, RFID contains a unique number which is resolved by consulting a secondary file to extract the binary URL.

EVALUATION

There are two aspects of evaluation of our approach: one is from the system's perspective and the other is from the end users' perspective to evaluate the usability of the deployment tool. We have approached the first aspect by following the guidelines of Edwards et al. [9]. A couple of *proof-of-concept* ubicomp systems that include multiple artefacts, profiles and application are re-developed following FedNet's approach and are provided to end users for real time deployment. This deployment task is supported by the tangible deployment tool. So the end users could evaluate its usability and thus complementing the second aspect of our evaluation. In this section, first we present the two *proof-of-concept* systems and then present the user trial.

Two Sample Ubicomp Systems

The first system is composed of the following:

- *Mirror Artefact:* A regular display is augmented with an acrylic mirror panel (Figure 10(a,f)). The acrylic panel is attached in front of the display, and only bright colors from the display can penetrate the panel. The mirror display has an extension board for attaching sensors. An artefact framework instance (executable binary in an RFID Card) represents the mirror.
- *AwareMirror Application:* This application runs in a mirror and displays some up-to-date information [12]. The application's default functionality can be enhanced if the mirror is augmented with *Proximity* and *Bi-state Interaction* profile. The former enables the application to show information only when someone is in front of it and the latter enables the users to interact with it, e.g., to know detail information. This application adheres FedNet semantics, e.g., expresses tasks in a description file and access the artefacts using web techniques. The application comes in an RFID card (Figure 10(g)).

The second system is composed of the following:

- *Mirror Artefact:* Same as the above system.
- *Toothbrush Artefact:* A toothbrush (Figure 10(b,f)) is augmented with a wireless 3D accelerometer sensor. It can provide its state of use information and is represented by an instance (in an RFID) of the artefact framework with a *state-of-use*[4] profile plugged into it.
- *Virtual Aquarium Application:* This application has the objective of improving users dental hygiene by promoting correct toothbrushing practices [19]. The application turns the mirror artefact into a simulated aquarium. Fish living in the aquarium are affected by the users toothbrushing activity if a toothbrush is available. The application's default functionality can be enhanced if the mirror is augmented with *Proximity* profile that enables the application to show the aquarium only when some one is in front of it. This application adheres FedNet semantics and comes in an RFID card (Figure 10(g)).

Profiles

Both systems' functionality can be enhanced by adding one or multiple profiles into the mirror artefact. These are:

[4]The sole purpose of the *state-of-use* profile is to provide the usage state, e.g., toothbrush is in use, etc.

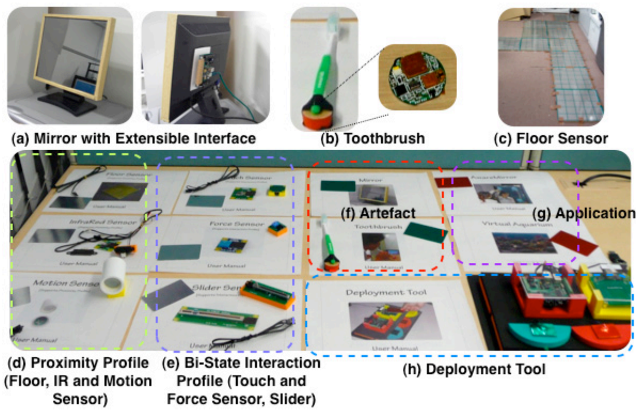

(a) Mirror with Extensible Interface
(b) Toothbrush
(c) Floor Sensor
(f) Artefact
(g) Application
(d) Proximity Profile (Floor, IR and Motion Sensor)
(e) Bi-State Interaction Profile (Touch and Force Sensor, Slider)
(h) Deployment Tool

Figure 10. Artefact, Profiles and Applications with their corresponding RFID cards, manuals and hardware

- Proximity Profile: This profile's purpose is to recognize the presence of an entity in front of its host artefact. This functionality can be achieved in multiple ways, i.e., using an infra-red sensor, a motion sensor, a camera, etc. We have provided three implementations for this profile (Figure 10(d)) using infra-red sensor, floor sensor (Figure 10(c)) and motion sensor respectively.
- Bi-State Interaction Profile: This profile enables a user to interact with its host artefact. It provides a simple two-state input facility suitable for applications that needs binary input. There are multiple instrument choices for the profile implementation and we have provided three implementations (Figure 10(e)): one with a touch sensor, one with a force sensor and the last one with a slider.

Each profile binary comes in an RFID card with associated hardware.

FedNet Infrastructure and the deployment Tool

The FedNet infrastructure runs in a laptop computer and the deployment tool is connected to it (Figure 10(h)).

Evaluation Methodology

Each of these applications, artefacts and profiles are developed following our architecture. Also, the same profile is built with multiple sensors. The successful deployment and incremental integration of these components by the end users will highlight the core features of our system and will reveal the usability of overall process. Accordingly, we have conducted a user study to understand the feasibility of our approach and to explore avenues for further research.

Participants

We invited 25 individuals (14 Male, 11 Female, age range 22-39) with moderate computing skills (familiar with web, email, and basic office applications) through an open invitation in a social networking site. 23 of them did not have any engineering background. Users were screened such that their professions were fairly disperse (e.g., law students, house wives, office workers, etc.) to balance the skill level.

Study Sessions

Each study session was held for 90 minutes and included four phases. In phase one we introduced the concept and presented a tutorial on the deployment tool. In phase two, they were given 10 minutes to get familiar with the tools. Next, in phase three, they were given the following four tasks:

- **Task 1:** Deploying and and running the mirror.
- **Task 2:** Installing and running either AwareMirror or Virtual Aquarium application.
- **Task 3:** Adding the Proximity Profile[5] into the mirror by selecting one of the three implementations if the user selected AwareMirror application or adding the toothbrush artefact if the user selected Virtual Aquarium application. This task ends with running the artefact and the application again.
- **Task 4:** Adding the bi-state interaction profile into the mirror by selecting one of the three implementations if the user selected AwareMirror application or Adding the Proximity Profile into the mirror by selecting one of the three implementations if the user selected Virtual Aquarium application. This task also ends with running the artefacts and the application again.

Finally in phase four, we had a questionnaire and in-depth follow-up interview session. The questionnaire contained 14 statements structured with a 5-item Likert scale to indicate their level of agreement or disagreement. Question 1-10 were designed following the System Usability Scale (SUS) [5] and the remaining four questions regarding the complexities of each tasks [6]. Following the questionnaire, we interviewed users to gain further insight into their assessments. Each session was video taped for later analysis.

Figure 11. Participants consulting manuals, deploying artefacts, installing applications, adding profiles, etc.

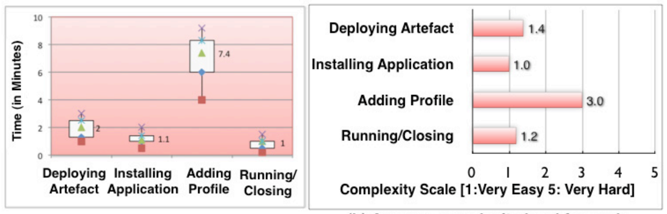
Evaluation Results

Figure 11 shows some snapshots of the experiment sessions. **System Performance:** FedNet and the target systems provided a stable performance in all the sessions and the end users' activities were properly converted into system events accordingly. We consider, the flawless deployment and the successful utilization of the two ubicomp systems evaluate the system aspects of our approach qualitatively.

1. *Plug and Play DIY Artefact:* The mirror artefact was deployed by the participants and its functionality was extended by attaching a couple of profiles (with multiple sensor choices) to enhance applications features. Regardless of the sensor type, profiles were seamlessly added into the artefact framework. Furthermore, the order of profile addition had no effect on the deployment process. So participants picked whatever profile they felt like adding. This highlights the capacity of our artefact framework for

[5]The profile hardware installation requires attaching the sensor to the mirror using double sided adhesive tape and connecting the sensor cable to the interface board located in the backside of the mirror. For the floor sensor, hardware installation was not needed except for placing the floor mat.

[6]Likert scale was normalized to complexity levels.



(a) Required time for each task

(b) Average complexity level for each task as perceived by the end users

Figure 12. Average time taken and average complexities for completing experiment tasks

hosting multiple profiles implementing different device interfaces. The combination of these are important for enabling the DIY support for the end users.

2. *Infrastructure Independency and Spontaneous Federation:* Both the artefacts and applications were expressed in high level descriptive documents and disseminated as executable binaries independent of the FedNet infrastructure. This allowed the end users to install them easily. FedNet provided the runtime association enabling applications to use the artefacts and to switch to respective advanced modes when new profiles were added. For the end users, these mechanisms were completely transparent as they could only see the effect of their actions. This highlights the simplicity and power of our approach to involve end users in the deployment process.

End Users' Performance: There were 100 tasks in total, four for each participant. All participants successfully finished the assigned tasks, though 6 participants needed active support in the early stages, primarily because of the misconception of the deployment process (explained later). On an average 16 minutes were required for the third phase (accomplishing four tasks cumulatively including the intervals between the tasks). Figure 12 shows the time (Figure 12(a)) required for each task type and the corresponding complexity (Figure 12(b)). Task type 1, 2, and 4 required fairly little time (1-4 minutes) since they consist of 2 step interaction (Figure 9(a,b)). However, artefact addition time was slightly higher because of the task order. Since, every participants' first task was to add an artefact, it required a slightly more time. The end users also found these tasks easy with an average complexity of 1.2 out of 5.0 for the task type of 1, 2 and 4. The profile addition task took maximum time (4-9 Minutes) where a large portion was spent for the hardware installation. Moreover, 8 participants made mistake in the association step (Figure 9(c)) e.g., placing the artefact card later than the profile card onto the card reader or not placing the artefact card at all. However, the deployment tool rejected these interactions and suggested the correct steps. This allowed the end users to accomplish the task without secondary assistance. These factors also impacted the profile addition task's complexity (average: 3) as shown in Figure 12(b). All participants have shown progress in repeating tasks and on an average they required 28.3% less time in redundant activities, e.g., when adding the second profile plugin, attaching hardware, etc. This indicates the fast learnability of the deployment process.

Subjective Results of End Users Feedback: The composite SUS score was 79.8 out of 100 (Standard Deviation: 11.7, Max: 91.2, Min: 62.7) regarding the overall usability of the deployment tool and the process. We consider these values are quite promising. Moreover, the individual frequency of the acceptance statement in SUS: *"I would like to have this system if it were available"* (Strongly Agree: N=17, Somewhat Agree: N=5) suggests users positive response regarding the acceptance of the overall approach.

Implications: Later interviews with the participants revealed several interesting aspects regarding their understanding and qualitative assessments of the entire process.

1. **Concept is difficult to comprehend:** The notion of artefact profile and application were difficult for the end users to comprehend and differentiate. For them the artefact profile and the application were the same. One participant commented *"I did not get this profile thingy, is not it an application for the artefact? It's a bit confusing, what is the difference between profile and application?."* Another participant remarked *"I understand that profiles are artefact features, but since the installation process is same, it is hard to differentiate the role of the artefacts, profiles, and applications. May be the profile addition button should be at the other end so that we know it has a different purpose."* They also had difficulties in understanding what a profile is, as they associated the term *profile* with someone's background or record. So, they could not correlate how a physical object could have multiple profiles, which also affected the performance of adding profiles as shown in Figure 12 (a,b). These facts imply that, our current notions are not self explanatory to end users and we need to provide a more comprehensive way of expressing these concepts. Also, as one of the above quotes pointed out, the orientation and placement of the interaction button for extending artefact feature should be different than installation buttons. We are in the process of addressing these concerns.

2. **Different packaging is preferred over DIY:** Several participants mentioned that they can buy a product with different functional granularity according to their preferences. one articulated participant pointed out *"Why don't you make different versions of this mirror with installed sensors and applications, then I can just go and buy whatever I need rather than trying to figure out where to put which sensor."* She goes on saying *"I would like to have this kind of cool stuff in my home but perhaps I would ask my boyfriend to install it for me."* 7 participants had similar view points. They concurred that the augmented artefact should be pre packaged, for example: one mirror could be packaged with a proximity profile and another with both profiles, etc. In this case they have the flexibility to buy different packages. Although, they agreed that the DIY approach is fun, interesting and inexpensive, but it limits the acceptability of the product to a mass population. One housewife remarked *"My husband would love to play with your mirror, but I don't know how often I will do this."* These views suggest that, to make augmented artefacts available to a larger user base, packaging with variant options are needed. The incremental DIY approach can further extrapolate the packaging scheme.

3. **Familiarity with sensors is crucial:** We have provided multiple sensor implementations for the same profile. We have found that different participants have picked different sensors. For proximity profile infra-red sensor was picked 17 times, the motion sensor 7 times and the floor sensor 1 time. For the bi-state interaction profile, the touch sensor was picked 21 times, and both slider and force sensor were picked 2 times each. The end users pointed out that their familiarities with the infra-red, motion and touch sensors in everyday life (e.g., in washroom faucet, magnetic door, garage, touch screen etc.) influenced their selection, since it was easy for them to understand how the sensors work and how to interact with them. Also domestic concerns were highlighted by a few participants. Most of them rejected the floor sensor since they found it big, and problematic while cleaning the floor. Similarly, they mentioned that the slider's sharp edge might harm their kids (3 housewives mentioned it). Moreover, since they knew what the sensor does, they revealed that it would be very simple for them to deactivate it. The latter findings actually confirms what Beckmann et. al. concurred about domestic concern and greater feelings of control [3]. These factors imply that the artefacts and the profiles should use the sensors that are common in our everyday life to involve end users in the deployment process.

4. **Tangible interface has potential:** We mentioned that in our pilot study we used GUI for the deployment process, which failed to attract users. They could not relate such GUI based deployment process with installing other home appliances. However, most of the users found the tangible tool familiar and user friendly. One user who participated in both the studies commented *"I like this tool because it gives me the feeling of installing a physical thing, this touching, pressing are something I am familiar with, for example using my TV or washing machine, its simple and more user friendly."* Similar comments were received from other participants too where they emphasize that tangible interaction for household appliances is more familiar and suitable to them. Considering their projections, we envisage that tangible interface might be a potential candidate for deploying ubicomp technologies.

RELATED WORK

Involving end users in the deployment process relies on augmented artefacts, device integration technologies and end user tools. We look at the related work in these areas.

Augmented Artefacts

One of the very first prototypes of smart object was Mediacup [4] where a regular coffee cup was instrumented to provide its contextual state infromation. Although the Mediacup project and its succeeding SmartIts [14] provide solid insight into the augmentation of physical artefacts with sensing and processing, they did not provide any generic representation model that can make them usable with any general purpose applications. Kohtake and his group introduced Smart Furniture and u-Textures to build custom furniture [17], however their approaches are also closed and tightly coupled with their underlying scenarios. The artefact framework presented in this paper takes a generic approach and uses a service profile based framework to represent the instrumented artefacts in a scenario independent way thus allowing any applications to utilize them.

Device Integration Technologies

To date several methods have been proposed to address device integration mechanism. One approach is interface and protocol standardization as attempted by Jini[7] and UPnP[8] respectively. Jini describes devices using interface description and language APIs allowing applications to utilize those interfaces where as UPnP attempts to standardize protocols to allow devices to intercommunicate seamlessly. These infrastructures focus primarily on the developers rather than the eventual users, consequently their support to enable incremental deployment by the end users is limited. For example, it is hard to add features in an existing artefact and using that feature immediately in the application with these infrastructures. Patch Panel [2] is a programming tool that provides a generic set of mechanisms for translating incoming events to outgoing events using EventHeap [16] communication platform. It allows new applications to leverage the services of existing components. Our overall approach is close to Patch Panel as we seek to support incremental integration. However, we exploit a distributed state model with an artefact framework that enable incremental addition of features to both artefacts and applications by the end users. In SpeakEasy [11] mobile codes (typed data streams and services) are exchanged among heterogeneous devices to create an interoperable environment. SpeakEasy does not consider the incremental extension of artefact services or end user deployment as its primary focus is on service composition. InterPlay [18] is a home A/V device composition middleware and uses pseudo sentences to capture user intent, which is converted into a higher level description of user tasks. These tasks are mapped to underlying devices that are expressed using device description. Although our approach is very close to InterPlay as we employ similar mapping of tasks to device services, our challenge is to support incremental extension and deployment of both artefacts and applications by the end users. Our artefact framework is a major leap from InterPlay which signifies our contribution. A range of middlewares for pervasive systems [6,24,25] specify their application development processes strictly. These middlewares usually provide end-to-end support for the application developer, i.e., instrumented artefacts are encapsulated into wrappers and an array of APIs is provided to the applications to manipulate them. The problem of this approach is that the applications and the instrumented artefacts become virtually incompatible in other environments. We have adopted a document centric approach allowing development of infrastructure independent applications and artefacts and the runtime association between them is provided by FedNet.

End User Tools

Most of the works that approached end user support in pervasive literature have taken either rule based or recognition based perspective to customize the pro-activity of the application. Rule based tools like iCAP [8], Stick-e-notes [22], Alfred [13] provide visual tool, or sound macros to the end

[7]Jini - http://www.sun.com/software/jini

[8]Universal Plug and Play - http://www.upnp.org

users to define conditional rules based on the context to connect input and output events. Similarly recognition tools, or more formally Programming by Demonstration systems like CAPpella [7] uses machine learning techniques to allow end users to associate personalized rules with real world events.These approaches are valid for rapid prototyping and also to personalize the pro-active behavior of the applications. However, they do not provide any general guideline regarding application or instrumented artefact deployment by the end users. Moreover, their support are primarily for developers to assist rapid prototyping and are not suitable for casual users with no or minimal technical background. One notable example is the Jigsaw Editor [15] that uses puzzle metaphor and allows non expert users to configure services intuitively by assembling available components (e.g., connecting a doorbell to a camera for taking a photo shot when someone rings the bell). Their study shows that end users understand the semantic association of devices and can manipulate them in order to meet their changing house hold demands. We are highly influenced by their findings and our overall approach of providing simple, easy-to-use deployment tool with FedNet is aligned with their projection. We extrapolated our system by allowing end users to introduce new applications or to extend existing artefacts' services by manipulating RFID and touch buttons regardless of the application or artefact types which simplifies end users involvement considerably.

DISCUSSION AND CONCLUSION

We reckon that in the near future, end users will be involved in building smart homes and this involvement must support the evolving nature of the home, i.e., incremental deployment. To provide this support, we presented a system infrastructure utilizing an artefact framework. Also, a tangible deployment tool is provided that simplifies the end users involvement significantly. We have reported a real life deployment experiment to evaluate our approach which revealed several interesting usability issues.

The profile notion has the potentially serious implication that standard common vocabularies or ontologies will be needed to support general interoperability of profiles and applications. However, by profile abstraction, we are not trying to define the ontology for profiles. Instead we are providing a structure that designers can use to disseminate their implemented ontology and glue it with rest of the infrastructure.

One immediate avenue of our future work is providing a mobile deployment tool using an RFID reader augmented mobile device (phone or PDA) to support spatially distributed deployment. We are currently in the process of placing our system in a number of users' domestic environments for prolonged evaluation. Furthermore, responding to end users' concerns we will replace the term *"profile"* with *"feature"*.

The contributions of this paper are two-fold. First, the system architecture that allows the development of plug and play artefact and generic applications regardless of the constraints of the target environment. Second, a simple and carefully designed tangible deployment tool that allows end users to deploy and incrementally enhance a smart space without going through complex authoring or configuration steps. Importantly, our findings put forth the fact that the end users might be involved in deploying future ubicomp systems if appropriate tool and supporting infrastructure are provided. We consider the findings from our experiment are very useful for further research exploration in the ubiquitous computing domain, particularly one that involves augmented artefacts.

ACKNOWLEDGEMENT

We like to thank Simo Hosio for his contribution to this work, all the participants for their voluntary contributions and the anonymous reviewers and our shepherd for their valuable feedback.

REFERENCES

[1] S. Antifakos, F. Michahelles, and B. Schiele. Proactive instructions for furniture assembly. In *4th International Conferenc on Ubiquitous Computing*, 2002.

[2] R. Ballagas, A. Szybalski, and A. Fox. Patch panel: Enabling control-flow interoperability in ubicomp environments. In *PerCom 2004*, 2004.

[3] C. Beckmann, S. Consolvo, and A. LaMarca. Some assembly required: Supporting end-user sensor installation in domestic ubiquitous computing environments. In *Ubicomp 2004*.

[4] M. Beigl, H. W. Gellersen, and A. Schmidt. Media cups: Experience with design and use of computer augmented everyday objects. *Computer Networks, special Issue on Pervasive Computing*, 35-4, 2001.

[5] J. Brooke. *SUS: A quick and dirty usability scale*, pages 189–194. Usability Evaluation in Industry. Taylor and Francis, London, 1996.

[6] A. K. Dey, G. Abowd, and D. Salber. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human Computer Interaction*, 16(2-4):97–166, 2001.

[7] A. K. Dey, R. Hamid, C. Beckmann, I. Li, and D. Hsu. a cappella: Programming by demonstration of context-aware applications. In *ACM CHI 2004*, 2004.

[8] A. K. Dey, T. Shon, S. Streng, and J. Kodama. Supporting end user programming of context-aware applications. In *Pervasive 2006*.

[9] W. K. Edwards, V. Bellotti, A. K. Dey, and M. W. Newman. Stuck in the middle: The challenges of user-centered design and evaluation of infrastructure. In *CHI 2003*.

[10] W. K. Edwards and R. Grinter. At home with ubiquitous computing: Seven challenges. In *Ubicomp 2001*, 2001.

[11] W. K. Edwards, M. Newman, J. Sedivy, T. Smith, and S. Izadi. Challenge: recombinant computing and the speakeasy approach. In *th ACM MobiCom*, 2002.

[12] K. Fujinami, F. Kawsar, and T. Nakajima. Awaremirror: A personalized display using a mirror. In *Pervasive 2005*, 2005.

[13] K. Gajos, H. Fox, and H. Shrobe. End user empowerment in human centered pervasive computing. In *International Conference on Pervasive Computing*, 2002.

[14] H. Gellersen, G. Kortuem, A. Schmidt, and M. Beigl. Physical prototyping with smart-its. *IEEE Pervasive Computing*, 03(3):74–82, 2004.

[15] J. Humble, A. Crabtree, T. Hemmings, B. K. Karl-Petter Åkesson, T. Rodden, and P. Hansson. Playing with your bits': user composition of ubiquitous domestic environments. In *Ubicomp 2003*.

[16] B. Johanson, A. Fox, and T. Winograd. The interactive workspaces project: experiences with ubiquitous computing rooms. *IEEE Pervasive Computing*, 1-2, 2002.

[17] N. Kohtake, R. Ohsawa, M. Iwai, K. Takashio, and H. Tokuda. u-texture: Self-organizable universal panels for creating smart surroundings. In *Ubicomp 2005*.

[18] A. Messer, A. Kunjithapatham, M. Sheshagiri, H. Song, P. Kumar, P. Nguyen, and K. H. Yi. Interplay: A middleware for seamless device integration and task orchestration in a networked home. In *IEEE PerCom 2006*.

[19] T. Nakajima, V. Lehdonvirta, E. Tokunaga, and H. Kimura. Reflecting human behavior to motivate desirable lifestyle. In *The Conference on Designing Interactive Systems (DIS 2008)*, 2008.

[20] O. G. C. Inc. *Sensor Model Language (SensorML) implementation specification*.

[21] J. O'Brien, T. Rodden, and M. R. J. Hughes. At home with the technology: an ethnographic study of a set-top-box trial. *ACM Transactions on Computer-Human Interaction*, 6, 1999.

[22] J. Pascoe. The stick-e note architecture: Extending the interface beyond the user. In *Intelligent User Interfaces*, 1997.

[23] T. Rodden and S. Benford. The evolution of buildings and implications for the design of ubiquitous domestic environments. In *ACM CHI 2003*, 2003.

[24] M. Roman, C. K. Hess, R. Cerqueira, A. Ranganathan, R. H. Campbell, and K. Nahrstedt. Gaia: A middleware infrastructure to enable active spaces. *IEEE Pervasive Computing*, pages 74–83, 2002.

[25] J. P. Sousa and D. Garlan. Aura: an architectural framework for user mobility in ubiquitous computing environments. In *3rd Working IEEE/IFIP Conference on Software Architecture*, 2002.

[26] M. Strohbach, H.-W. Gellersen, G. Kortuem, and C. Kray. Cooperative artefacts: Assessing real world situations with embedded technology. In *UbiComp 2004*.