Prototyping Smart Objects for the Mass

Geert Vanderhulst, Fahim Kawsar, Johan Criel and Lieven Trappeniers
Alcatel-Lucent Bell Labs
Copernicuslaan 50, 2800 Antwerpen, Belgium
{*geert.vanderhulst, fahim.kawsar, johan.criel, lieven.trappeniers*}*@alcatel-lucent.com*

***Abstract*—Several do-it-yourself projects and toolkits aim to empower people in bringing intelligence into their personal environments. This is particularly challenging because there is still a large gap between people who can imagine how technology could simplify their life and engineers who can actually make things behave smart. To unite both camps, we present a platform that enables ordinary people to become the inventor of their own smart objects, assisted by a community of field experts. Our software, called D-TALE, fosters the creativity of people by giving them a forum where they can express and prototype their innovative ideas. Moreover, D-TALE gives rise to a crowd-sourcing platform where non-technical users can request developers to provide an implementation for their project. Our approach is outlined in a preliminary case study.**

***Keywords*-Smart objects, DiY prototyping, Crowd-sourcing**

I. Introduction

Whereas out-of-the-box products compete each other in terms of connected features, we witness their smart building blocks – sensors actuators – hitting the market at affordable prices (e.g. Phidgets, Particles, ...). With these components getting smaller, cheaper and more power-efficient, it becomes feasible to integrate them in the home environment and attach them to everyday objects. Not only this supports Weiser's vision [12] of technology that assists people in their daily life, but it also feeds the opportunity to let users invent their own smart objects supporting their activities. For example, a health conscious person could install a sensor on a cookie jar that can measure the number of cookies taken out of the jar and warn if a threshold has been reached. Since we cannot envision all possible cases of how technology can enrich a person's life, we must rely on the creativity of people to make their own vicinity (re)act smart. However, most end-users are not capable of developing the software needed to make their objects behave smart.

Although initiatives have been taken to empower users in creating ubiquitous applications for quite a while [4], [3], [8], [6], we are still confronted with a barrier for mass deployment and usage of end-user development tools which might be due to the fact that they merely target technically skilled people. In addition, we witness a lack of distribution channels enabling visionary users to communicate their ideas with developers who can help realizing them. Initiatives such as Quora[1] illustrate the success of crowd-sourcing, yet up till now we are not aware of such platform targeted at building smart environments. In this paper we focus on *supporting the diffusion of smart objects by the people for the people*. To this end, we allow ordinary users to prototype their own DiY projects and delegate the development task to expert users, hoping that one day prototyping and development tasks will seamlessly merge together. We bring ordinary users in touch with a forum of DiY experts who can provide assistance to realize their case and e.g. give advise on the type of sensors and actuators that could be installed.

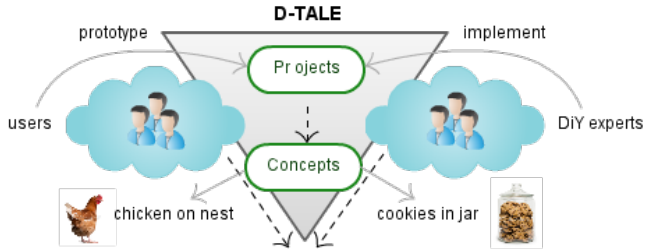

Figure 1. D-TALE fosters user-driven innovation by bringing people who have ideas about making their environment a smarter place in touch with experts in the field. By introducing high-level concepts whose implementation(s) can be reused, D-TALE aims to make the gap between ordinary individuals and technical experts smaller.

The main contributions of this work are illustrated in figure 1 and can be summarized as follows. First, we present D-TALE from an end-user's point of view and explain how users can prototype several aspects of a DiY project (section IV), using a motivating scenario (section II) as running example. Second, we highlight D-TALE from a developer's point of view and discuss our efforts towards a crowd-sourcing community where experts – called DiY'ers – assist end-users with the development of their projects (section V). We introduce high-level concepts such as 'the number of cookies in a jar' that abstract away underlying technologies such as the sensors and application logic beneath. By contributing such concepts to the community, we aim to shorten the distance between ordinary users and DiY'ers. We also report on the difficulties we encountered when transforming user created prototypes into actual applications (section VI)

[1]http://www.quora.com/

and finally, we performed a preliminary evaluation of our approach by means of a case study (section VII).

II. The chicken or the egg?

In this section we present a motivating scenario that illustrates how smart objects help to enhance a specific task for which it is unlikely that an out-of-the-box solution exists. The scenario describes the situation of grandmother who takes care of three chickens. Behind a fence in her garden, the chickens live their life and produce eggs in an old henhouse. On average, these hens produce one egg in two days, but it is hard to predict the number of eggs in the henhouse at a given day. Hence, every few days grandmother checks the henhouse for eggs and collects them. As this takes quite some effort considering her age, she would like to know in advance how many eggs are awaiting in the henhouse so she can decide whether it is time to collect them. Besides, she would like to get informed if one of her chickens tends to bread on eggs as she knows that this kind of behaviour prevents other chickens from entering the occupied nest. To help grandmother with her 'chicken or the egg problem', we need to keep track of the current number of eggs in the henhouse and warn about possible breeders on a display inside the house. To measure this information, we can think of several DiY solutions that involve one of the following sensor devices: a camera, a weight sensor, a temperature sensor, etc. Consider for example the case where a temperature sensor is installed in each nest in the henhouse. When a hen enters the nest, the sensor will measure the chicken's presence as a sudden increase in temperature. Similar, when the hen leaves the nest, the temperature will decrease again. By measuring the time a hen is on the nest (i.e. the time between a sudden increase and decrease of the nest's temperature), we can assume an egg has been produced (e.g. time less than 10 minutes) and detect when a hen is eager to bread on eggs (e.g. time more than 10 minutes). This sensor data is then sent to an application running on the display in the house where it is processed and visualized. Note that using just a temperature sensor, the application is still unable to detect when grandmother has collected eggs. Hence grandmother should inform the application to reset the egg counter (e.g. by pressing a button on the display) or we need to extend this project with extra sensors to detect when eggs are collected.

III. Related work

Empowering end-users to create personalized smart objects requires a sound ecosystem with a coherent interplay among hardware, software, toolkit and interfaces. End-user programming in desktop computing has a rich history of research, but in the domain of pervasive computing it is still very prospective. Early seminal work includes the Jigsaw Editor [8] which uses a puzzle metaphor to allow non-expert users to configure pervasive services intuitively by assembling available components, and the SpeakEasy [5] platform where end-users can compose pervasive services spontaneously. Other work looked at end-user deployment of sensors in a physical environment in order to meet changing household demands [2], [1], [9]. It shows that users understand the semantic association of physical devices and software components when assisted by clear installation guides. Some work also addressed support for personalizing behaviour through rule-based or recognition-based tools. Rule-based tools like iCAP [4] and Alfred [6] provide a visual tool or sound macros which allow end-users to define conditional rules to connect input and output events. Similarly, recognition-based tools or more formally 'Programming by Demonstration' systems like CAPpella [3] use machine learning techniques to allow end-users to associate rules with real-world events. Exemplar [7] also shows that events can be derived from patterns recognized in continuous sensor data after teaching the system about these patterns, i.e. by providing examples. While these approaches are valid for rapid prototyping, they primarily focus on facilitating initial configuration tasks and are less suited for casual users with no or a limited technical background.

Taking lessons from the state-of-the art, we adopt a community-driven approach where an individual can participate in the prototyping process either by doing it herself or by leveraging the skills of expert communities. Our approach enables an individual to specify her needs in a structured fashion using highly abstracted tools, henceforth performing the very basic stage of programming. The output from these tools provides the foundation for the community developers to perform the advanced programming using our proposed system and thus responding to the naive users prototyping needs. This two-sided participatory development approach atop of our platform empowers ordinary individuals to realize their personalized smart objects.

IV. Prototyping DiY projects

D-TALE provides a platform and integrated tools that support end-users in prototyping smart objects. We treat users as *engineers of their own smart environments* and help them express their ideas in a structured way. By sharing prototypes with DiY'ers, users can engage in a dialog with a technical expert. Experts can then give feedback, iterate over a prototype and provide assistance to come up with a working application that can be deployed in the user's environment. Prototyping a DiY project is achieved in three iterative mock-up phases which are illustrated in figure 2 and discussed in the next sections.

A. Situation mock-ups

Situation mock-ups let users virtually augment their environment with sensors and actuators. They help people to provide insight in the environment wherein their DiY project will be deployed. To this end, the D-TALE editor

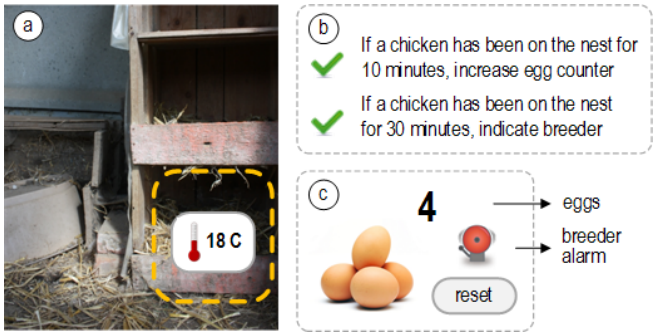

Figure 2. Situation mock-up (a), behaviour mock-up (b) and interaction mock-up (c) of a smart henhouse.

integrates photos (or drawings) of the user's environment as a convenient means to quickly represent a situation and share it with others. This set of integrated images corresponding to the user's virtual environment can then be augmented with DiY components. These components are selected from a toolbox and are for instance dragged on an image to indicate which sensors and actuators should be installed where. As such, we allow users to model their DiY project in a virtual environment before starting to implement it in the real world. Figure 2(a) shows a situation mock-up for a henhouse augmented with a temperature sensor.

- **Tag tool:** some parts of an image illustrating a situation deserve extra attention because it reveals objects that play an important role in a DiY scenario. These can be marked using the tag tool and augmented with descriptive labels (i.e. tags), similar to tagging people on a social networking site such as Facebook.
- **Sensor tool:** the sensor tool maintains a link to a repository of hardware components that can be used for measuring particular concepts. A sensor selected from the list of available components can be dragged on an image to indicate where it would be installed in the physical environment.
- **Actuator tool:** similar to the sensor tool, the actuator tool is aware of a set of components that can be used for actuating purposes. Examples include an actuator for a television that supports commands to change its video source or an actuator for an interaction device enabling the deployment of a user interface component.
- **Search tool:** the search tool is used to look up concepts as illustrated in figure 3. A user looking for a way to measure the presence of a chicken in a nest can use the search tool to find out that others measured the 'presence' concept before and shared their implementation. For instance, it can be seen that a weight sensor embedded in a chair was used in an office environment to detect whether colleagues are sitting behind their desk. If a concept implementation is dragged on an image, the sensors and actuators required by the implementation are added to the mock-up and installation tips are displayed.

By restricting the set of available DiY components to sensors and actuators that are at hand on the market, users are forced to think in terms of available technology such that unrealistic ideas are filtered from the beginning. Nevertheless, it can be hard for a user to find a suitable sensor for measuring a particular property. By looking at what others have built using a given sensor, users can get familiar with the sensor and become inspired by its possibilities. Hence we put effort in a semantic search feature that assists users in selecting appropriate sensors given a concept they would like to measure such as whether a chicken is on the nest or not. Depending on the situation at hand, a concept can have several implementations associated, new ones can be provided anytime and existing ones can be reused. A dedicated installation guide linked with a concept's implementation helps the user to figure out how to install sensors and actuators. Alternatively, situation mock-ups from projects that make use of the concept can serve as examples.

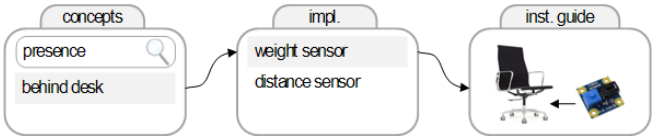

Figure 3. Searching for concepts.

Moreover, physical sensors can be connected to situation mock-ups for simulation and testing purposes. Sensor data will appear in overlay on the image(s) with the corresponding sensor component and actuator components are presented with a small interface for sending commands to their counterpart in the real world. Apart from sharing mock-ups with DiY experts, sharing real-time sensor data and actuator controls is equally important as it provides DiY developers with a realistic testbed to configure application logic for a specific environment.

B. Behaviour mock-ups

Behaviour mock-ups allow people to express how they want their augmented environment to react on events. To relieve users from programming the environment's behaviour, we ask users to describe an application's behaviour on a high level using natural language such that expert users with programming skills can understand the needs of a particular DiY project. Figure 2(b) shows a behaviour mock-up describing a set of rules for the scenario discussed in section II. For each rule, it can be specified whether this is a minimal requirement for the project to succeed or an optional rule that can be switched on or off. As these rules are described on a high level – e.g. 'the time a chicken is on the nest' is used instead of 'if a weight sensor measures an increase during a period of time' – they are independent of the hard- and software that is used to realize the project.

C. Interaction mock-ups

Interaction mock-ups enable users to specify how they want to interact with their environment. To grasp this information, we let users design their own user interface prototypes and share them using the D-TALE platform. For example, the interaction mock-up shown in figure 2(c) is merely a visualization of the current number of eggs in the henhouse accompanied by a potential breeder notification and a reset button to set the counter to 0 after having collected the eggs. Note that interaction mock-ups are closely connected to behaviour mock-ups since a control interface could be used to enable or disable a particular behaviour.

V. Towards a DiY community

By sharing projects and the concepts they introduce, D-TALE allows users to find out what others have created and reuse concepts that were implemented before. If additional help is needed, users can rent a DiY'er to support their case.

A. What did other users build?

Previously accomplished DiY projects could serve as an important source of inspiration to develop an individuals' ideas. We support different ways to search for projects:

- starting **from a sensor** or a set of sensors that were used in a project. This is similar to asking the system 'What did others do with this sensor?'.
- starting **from an actuator** or a set of actuators that were used in a project. This provides an answer to the question 'How did others incorporate this actuator?'.
- starting **from tags** that categorize a DiY project and the concepts that were measured. Using search engine-style keyword phrases such as 'chicken eggs henhouse' we search for high-level concepts (e.g. number of eggs in a nest) and project descriptions (e.g. grandmother's henhouse project) by matching tags that were attached to projects (i.e. situation mock-ups) and concepts. Since we make use of linguistic relations in our search algorithm (see section VI-A), search results are not limited to exact matches of given keywords but will also include semantically equivalent results.
- starting **from a combination** of the above. This search strategy is particularly useful if a user wants to build a case from a specific set of sensors and actuators and/or if she has a clear idea of the scope of her project.

Existing projects can be reused as templates from which users can instantiate a personalized version. This is achieved by adjusting mock-ups to reflect their own needs and substituting the hardware components that were used in the original project with their own sensors and actuators. In this case, high-level concepts that were used in the original project are reused in the new project along with their implementation that transforms raw sensor data into something meaningful.

B. Rent a Do it Yourselver

In section VI-B we have discussed some of the problems users are likely to stumble upon when converting a prototype into a working application. These issues merely deal with the (in)capability of end-users to program their own applications. Even simple DiY projects might be too complex and domain-specific to fit within the scope of end-user programming tools, in which case assistance from experts will be indispensable. Hence we believe the option to 'rent a Do it Yourselver' is a major requirement to effectively realize a successful DiY community. Even if ordinary users get acquainted with tools for composing smart application logic, it might just cost them too much time and effort. Users might rather want to pay a fair rate to an expert user who can deliver a qualitative solution in fewer time.

A DiY'er can contribute to the community in several ways. First, he can iterate over mock-ups and collaborate with users to come up with a realistic prototype. For instance, a first version of the situation mock-up shown in figure 2(a) might have listed a random temperature sensor that was picked by a user to express her idea. In a second iteration, the sensor might be replaced by a particular type of temperature sensor that is considered best suited for the given situation by a DiY'ers. Second, a DiY'er can deliver new concepts to the community which are extracted from projects and generalized for (re)use by a broader audience. This includes sharing implementations for a concept and annotating them with a step-by-step installation guide. To motivate DiY'ers to engage in projects and to ensure the quality of their work, an incentive program should be incorporated in D-TALE [10]. Incentives not only have proven to stimulate contributers to acquire a high rating, but they might also trigger ordinary users to participate in the community.

VI. Implementation and technical challenges

In this section we discuss the architecture of D-TALE and outline the major difficulties we faced when creating a working application from a prototype.

A. Architecture and models

Figure 4 shows an overview of D-TALE's architecture and the models it employs. The two major models, i.e. the *project and concept model*, define a semantic structure upon concept and project instances and their respective relations. We make use of light-weight ontologies for mapping equivalent concepts, building up a catalog of hierarchically organized concepts that evolves over time and maintaining links between projects and concepts. Moreover, by adding tags linked with the WordNet[2] lexicon to newly created content, we can avoid ambiguity (i.e. differentiate between words with different meanings) and improve the accuracy of search results by employing linguistic relations. Adequate

[2]http://wordnet.princeton.edu/

search results are important to stimulate reuse of existing concepts as much as possible. Additional models are used to describe sensors and actuators. For instance, SensorML profiles specify a.o. the datatype and unit of outputted data.

Furthermore, the architecture is composed of a set of repositories. The *project repository* stores information about situation mock-ups, behaviour mock-ups and interaction mock-ups per project. The *concept repository* stores high-level concepts, contributed by end-users. Each concept can have several implementations, each depending on one or more sensors and actuators. Concept implementations are further annotated with an installation guide indicating how e.g. sensors should be installed for optimal results. The projects in which a particular concept was used act as examples that enrich a concept's installation guide. The *data repository* aggregates sensor data originating from a user's personal environment for i) collaboration with expert users and for ii) sharing information with others. Similar to Pachube[3], applications can subscribe to sensors and get notified of data updates. Since each repository has an open REST interface, tools can connect with them to retrieve and store information. For instance, a tool can push new concepts into the concept repository or subscribe to sensor data via the data repository's REST interface.

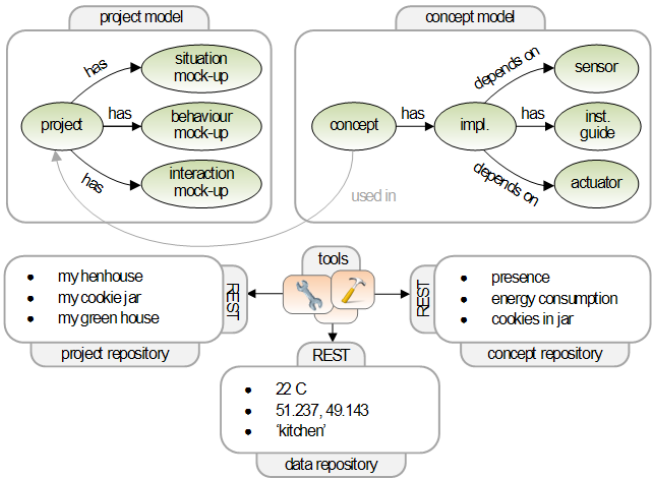

Figure 4. D-TALE architecture and models. Design and development tools interact with project, concept and data repositories through REST interfaces.

B. Transforming prototypes into working applications

Even though D-TALE enables users to express the design and functional requirements of the hard- and software needed to realize a project, there is still a difference between a prototype of an application and its actual implementation. From the experience we gathered when building working applications from prototypes, we have compiled a list of three main reasons that might still prevent ordinary users from creating their own DiY applications:

- **Meaning derivations:** many sensors produce raw data which needs to be transformed into something meaningful such that applications can start using it. Transformations include simple conversions from a discrete arbitrary number that to be converted into an official measurement unit (e.g. degrees Celcius), but also more advanced sensor data processing might be required to derive facts from continuous sensor data such as when and for how long a chicken has been on the nest.
- **Component deployment:** to run transformation blocks, logic that steers smart object behaviour and user interface components, an execution environment is needed. However, the multitude of devices typically available in a ubiquitous environment makes it a non-trivial task to decide upon which components should be installed where [11]. For example, a component that collects sensor data should run in the home network whereas a transformation block could basically run anywhere. Besides, the deployment of user interface components is challenging due to the variety of typical interaction devices (e.g. various screen sizes and modalities).
- **Programming behaviour:** in section III some end-user development tools for programming behaviour in a ubiquitous environment were discussed. The applicability of these tools is typically limited to specific target domains in order to keep them usable. However, since the domain of a DiY project is not known in advance, it can be cumbersome to program behaviour within the boundaries of a generic end-user development tool.

D-TALE mainly contributes to the first issue. Since high-level concepts share implementations that include transformation blocks, they can be reused. Hence, the more populated the concept repository becomes, the higher chance an ordinary user can create realistic DiY prototypes that can be translated into working applications with minimal effort.

VII. Case Study

In section II, we introduced the scenario of a DiY project dealing with a smart henhouse. Throughout the paper, we have explained D-TALE's prototyping process using this project as running example. In this section we describe how we built a working application from the mock-ups depicted in figure 2 and explain how high-level concepts are extracted from it and contributed to the community. We used a Phidget SBC board with embedded WiFi chip as execution environment and developed a small SDK for the SBC that provides methods to interact with D-TALE's open REST interface. On the SBC we installed an application that transforms input from an attached temperature sensor into three outputs: i) an output that indicates whether there is a chicken on the nest based on sudden temperature increases/decreases, ii) an output that counts the number of eggs in the nest and iii) an

[3]http://www.pachube.com/

output that notifies about breeders. When sharing is enabled, this information is forwarded to the data repository depicted in figure 4 and made available on the D-TALE platform. This allows compatible tools like the D-TALE editor to access information related to a project and e.g. visualize live sensor data on top of situation and interaction mock-ups for testing purposes. To support switching sharing on/off and resetting the egg counter, we programmed the SBC board to listen on an input port for incoming commands. Furthermore, we developed an iPad application with a graphical user interface whose design adheres to the interaction mock-up shown in figure 2(c). This application fetches information about eggs and breeders directly from the Phidget SBC board. Pressing the reset button sends a reset command to the SBC.

From this project, we can extract the following concepts and share them with the community: 'chicken on nest' (e.g. categorized under a 'presence' concept), 'chicken is breeding' and 'chicken laid egg'. These concepts share the same implementation, namely the application that was developed for the SBC. By adding a description of the implementation's hardware dependencies (i.e. a Phidget temperature sensor) and an installation guide that explains how to install them, these concepts can be reused in other projects. For example, the SBC implementation for the 'chicken is breeding' concept can be found using the search tool illustrated in figure 3 and reused in a DiY project that aims to discourage breeders to stay on the nest.

VIII. Conclusions and outlook

On the one hand, D-TALE provides end-users with a forum to express the requirements of a DiY project they want to accomplish using mock-ups. The result of three mock-up phases is a prototype that can be iterated upon by DiY'ers and converted into a working application. On the other hand, high-level concepts can be contributed to a community established by the D-TALE platform. By stimulating the reuse of concept implementations (e.g. by means of step-by-step installation guides), we expect amateur and expert users to close in on each other. Still, an incentive strategy and privacy mechanism should be further explored to motivate people to participate in the community and gain confidence.

We conclude with an outlook to two future extensions for D-TALE. First, we aim to incorporate additional constraints in our prototyping tools. For example, if power constraints (electricity available?), network constraints (wired or wireless network within reach?) and budget constraints (what is the maximal price the user is willing to spend on this project?) are taken into account, prototypes become more personal and implementations generated from the prototype are more likely to fit in the target environment. In the case study outlined in section VII, we could make use of powered Phidget components because there is a power plug available in the henhouse. Yet, for a user on a smaller budget, a Phidget SBC controller might be considered too expensive, etc. Second, we see an opportunity in building a social network of personal smart objects that share information with others using D-TALE. Looking at the popularity of social networks, we can imagine social communication channels go beyond people and shift to smart objects as well. For instance, users might want to share facts about their pets such as when they sleep or walk around or compete with each other in how 'green' (energy conscious) they are. The notion of concepts and the ability to reuse their implementation puts D-TALE on the front line to provide such social features.

References

[1] S. Antifakos, F. Michahelles, and B. Schiele. Proactive Instructions for Furniture Assembly. In *UbiComp'02*, pages 351–360, 2002.

[2] Chris Beckmann, Sunny Consolvo, and Anthony LaMarca. Some Assembly Required: Supporting End-User Sensor Installation in Domestic Ubiquitous Computing Environments. In *UbiComp'04*, pages 107–124, 2004.

[3] Anind K. Dey, Raffay Hamid, Chris Beckmann, Ian Li, and Daniel Hsu. a CAPpella: Programming by Demonstration of Context-Aware Applications. In *CHI'04*, 2004.

[4] Anind K. Dey, Timoty Shon, Sara Streng, and Justin Kodama. iCAP: Interactive Prototyping of Context-Aware Applications. In *Pervasive'06*, pages 254–271, 2006.

[5] W. K. Edwards, M. Newman, J. Sedivy, T. Smith, and S. Izadi. Challenge: Recombinant Computing and the Speakeasy Approach. In *MobiCom'02*, pages 279–286, 2002.

[6] Krzysztof Gajos, Harold Fox, and Howard Shrobe. End User Empowerment in Human Centered Pervasive Computing. In *Pervasive'02*, pages 1–7, 2002.

[7] Björn Hartmann, Leith Abdulla, Manas Mittal, and Scott R. Klemmer. Authoring Sensor-based Interactions by Demonstration with Direct Manipulation and Pattern Recognition. In *CHI'07*, pages 145–154, 2007.

[8] Jan Humble, Andy Crabtree, Terry Hemmings, Boriana Koleva Karl-Petter Åkesson, Tom Rodden, and Pär Hansson. Playing with the Bits: User-Configuration of Ubiquitous Domestic Environments. In *UbiComp'03*, pages 256–263, 2003.

[9] Fahim Kawsar, Tatsuo Nakajima, and Kaori Fujinami. Deploy Spontaneously: Supporting End-users in Building and Enhancing a Smart Home. In *UbiComp'08*, pages 282–291, 2008.

[10] Uuong-Sik Lee and Baik Hoh. Sell Your Experiences: a Market Mechanism Based Incentive for Participatory Sensing. In *PerCom'10*, pages 60–68, 2010.

[11] Geert Vanderhulst, Kris Luyten, and Karin Coninx. PerCraft: Towards Live Deployment of Pervasive Applications. In *IE'10*, pages 191–196, 2010.

[12] Mark Weiser. The computer for the 21st century. *Scientific American*, 265(3):66–75, 1991.