MultiKit: A User Interface Toolkit for Multi-tag Applications

Robert Hardy
School of Computing and Communications, Lancaster University, UK
hardyr@comp.lancs.ac.uk

Enrico Rukzio
paluno, University of Duisburg-Essen, Germany & Lancaster University, UK
enrico.rukzio@uni-due.de

Matthias Wagner, Paul Holleis
DOCOMO Euro-Labs, Germany
{holleis, wagner}@docomolab-euro.com

ABSTRACT
This paper documents MultiKit: a toolkit that supports the development of sophisticated user-interfaces for applications using Near Field Communication (NFC). The usage of the toolkit is exemplified using three implemented prototypes, each with significant differences. These differences highlight the complexities involved in producing such NFC applications, and the benefits of a toolkit that can adapt to varying interface types. As the toolkit is tailored to NFC technology, NFC-specific features are explored. These include switching between dynamic (e.g. projection) and static interfaces (e.g. posters), and interface generation from NFC tag descriptions.

Categories and Subject Descriptors
H5.2 [Information interfaces and presentation]: User Interfaces - Graphical user interfaces.

General Terms
Algorithms, Design, Human Factors.

Keywords
Mobile, NFC, Multi-tag, toolkit.

1. INTRODUCTION
Public, situated displays are prolific in human-computer interaction research and a ubiquitous part of our environment. We see increasing numbers of digital displays in public places due to technology advances driving down costs and their ability to show up-to-date information. Yet, non-digital displays (e.g. posters and flyers) still remain prevalent due to their simple deployment and low cost. Public, situated displays play a vital role in communicating information about our environment, supporting activities in our environment, and providing resources for which activities and communication can be centered. In recent years, we have seen these displays begin to support user interaction, such as interactive kiosks, and ultimately provide multi-user interaction for shared, collaborative interaction – inherently apt to large screens in social settings.

Ballagas et al. [3] survey various interactions that allow smart phones to play the role of input devices applicable to such displays. The survey covers various smart phone features (e.g. cameras, buttons, touch displays, accelerometers, and microphones). However, these features only typically support indirect interaction with the displays. The focus in this paper is on direct touch-based interaction with a variety of display types (digital displays, posters, flyers, etc.). We will now describe a tourist scenario where a user can interact with a range of travel services by touching their phone on desired options on the various displays around the city. Say you have flown to an unfamiliar city; you may wish to find the next train from the airport to your hotel. You see various digital, situated displays around the airport providing the latest transport and navigation information and ticket services. By touching the options on one of these displays with your phone, you are able to find the platform for the next train and purchase a ticket, which is saved to the phone. Also saved to the phone are walking directions from the train station to the hotel. Once settled at the hotel, you may wish to visit a restaurant recommended by a friend. You find an interactive map outside the hotel with options that can also be touched with your phone for querying the next bus travelling to the restaurant. You fold the map up and carry it with you for future reference. When you arrive at the restaurant, you see interactive menus located on every table. The menu provides food and drink items that can be touched with the phone to show additional information on you phone's display about the items and suggestions of other items that compliment your selection. In this scenario, we see a high level of ubiquity pertaining to such touch-based displays. We also see that they can be represented many different physical forms.

The underlying technology we use to support touch-based mobile interaction with both dynamic (digital) and static (non-digital) displays is Near Field Communication (NFC). NFC is a short-range radio frequency communication technology suitable for touch-based interaction between these devices due to limited range. We have begun to see its integration into mobile phones allowing them to read/write a few kilobytes of data from/to powerless NFC tags (with a size of 42mm in diameter).

In recent years, we have seen the trend of augmenting physical objects, such as maps, posters, information boards, menus or interactive displays, with one or more NFC tags. *Tagging* these displays enables the user interface on the phone to be extended to the physical object; thus, making mobile services more visible. The physical interface can fall into two categories: single-tag interfaces and multi-tag interfaces. Single tag interfaces have only one tag attached and multi-tag interfaces have multiple tags attached. Therefore, such a multi-tag interface may support sequences of tag reads in order to achieve a common goal. Multi-tag interfaces will be the focus of this paper.

The development of multi-tag applications is currently a costly and time-consuming process as they are usually developed from scratch. The most problematic aspect is that the user interface is distributed over the mobile phone and the tagged physical interface. Additional issues are the fact that the geometry of the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
MUM'11, Dec 7–9, 2011, Beijing, China.
Copyright © 2011 ACM 978-1-4503-1096-3/11/12…$10.00.

physical interface must match the tags used and support different display types. Currently available user interface widget libraries typically support a single particular device (e.g. Java/SWING for Laptops or Java/LCUID for mobile phones), but none are able to span two different devices (e.g. mobile phone screen and LCD). In addition, there is a granularity mismatch between the desktop GUI elements and the physical dimensions of the tags.

The MultiKit was developed to address this problem by accelerating and simplifying the development of user-friendly multi-tag applications. Firstly, and most importantly, a multi-tag widget library supports the development of multi-tag user interfaces that span the mobile device and the multi-tag physical interface. Secondly, MultiKit provides an overall infrastructure for the communication between physical interface, mobile phone, and server components. Three sophisticated prototypes exemplify the generality of this user interface toolkit.

2. RELATED WORK

In recent years, we have seen an increasing use of mobile phones for interactions with people, places and things in the real world [10]. One of the most popular interaction techniques in this field is *touching*, which allows a mobile phone to interact with other objects by touching them. The technology behind this is NFC (Near Field Communication): a short-range RF communication technology with a read range of up to 10 cm. The potential of this technology is suggested by ABI Research who forecast that by 2012, more than 400 million NFC chipsets will be shipped that will be used not only for payments at points of sale, but also to access information from smart objects [1]. More and more mobile phone makers are integrating NFC in their handsets (e.g. Samsung Nexus S / Galaxy S II, Blackberry Curve, HTC Amaze 4g) and and several such as Nokia are making strong commitments [2].

The applications, which are based on NFC or passive RFID tags read by mobile phones, are usually single-tag as just one tag is attached to an object. Want et al. were among the first who augmented objects such as books, documents or business cards with a single RFID tag [24]. An RFID reader was connected to a mobile device and used to read the links to corresponding services, such as ordering the book, getting the electronic version of the document or picking up the email address from a business card. Such single-tag interactions are now widely used in Japan via i-mode FeliCa service offered by NTT DOCOMO [14]. Here, i-mode Felica enabled handsets can be used for electronic payments, access control, and as a commuter pass simply by touching corresponding readers.

Currently, we see the trend towards multi-tag interfaces in which physical objects are augmented with many NFC tags that represent different actions or parameters related to the object. Reilly et al. augmented a paper map with a number of RFID tags beneath points of interest [16]. A mobile device was connected to a RFID reader in order to read these tags and provide information about the touched sights. Further examples are posters augmented with RFID/NFC tags that provide several services, such as getting additional information, downloading media content, and ordering movie tickets [2, 4]. Augmenting menus with RFID technology for ordering food has been trialed by McDonalds in Korea [12], by Häikiö et al. for a touch-based user interface for elderly people [6], and by VTT for a restaurant in Oulu, Finland [17]. In those examples, a menu was used in which each item (e.g. burger, salad or drink) was touchable through a corresponding NFC tag on the back. Further examples are "infotags" which provide different hyperlinks of which 1500 were deployed also in Oulu [22] and local traffic news poster trialed by Vodafone [14].

The multi-tag interfaces discussed so far focus on static physical interfaces such as maps, posters or menus. The Touch & Interact [7] and the Touch & Select [21] systems show dynamic physical interfaces through the usage of projections or LCD displays augmented with NFC tags. Here, the dynamic physical interface turns into an interactive screen that reacts to the interactions of the user. The Touch & Interact system supports the interaction with an interactive display in a tourist office supporting interactions with maps, points of interests and routes [7]. The Touch & Select system supports interactions with the NFC-equipped display of a laptop by touching it with the mobile device in order to upload and download pictures between the two devices [21].

Many of the previously mentioned examples for multi-tag applications have been evaluated in user studies or have been trialed. Although this previous research shows the usefulness and the advantages of multi-tag interfaces, all have been developed from scratch leading to time consuming and expensive prototypes, or systems often with serious user interface problems.

An early framework that supported different kinds of mobile interactions with smart objects is the Physical Mobile Interaction (PMIF) Framework [18]. This framework also addressed RFID/NFC-based interactions and is focused primarily on technical aspects needed to establish a connection between mobile phone and augmented object. It did not focus on multi-tag applications at all. This framework was extended to the Pervasive Service Interaction (Perci) framework, which focused primarily on the automatic generation of mobile user interfaces for physical mobile interactions based on semantically enriched service descriptions [4]. Again, no support for multi-tag user interfaces was provided. Further frameworks that focus rather on the technical aspects of single-tag NFC applications than on user interface aspects, were developed by Koskela et al. [11], Guinard et al. [5] and Sanchez et al. [20].

Almost all graphical user interfaces are built using existing user interface widgets such a buttons, sliders or menus (provided by user interface toolkits). Those are usually focused on device classes, such as desktop systems (Java Swing, Macintosh Cocoa and Windows MFC) or mobile devices (Symbian UIQ, Android View System und iPhone UIKit). Developing a system with a distributed user interfaces – as it is the case for multi-tag applications – requires currently the development of two different user interfaces and of a corresponding application logic which combines them.

Systems in which the user interface spans several devices are, for example, the Pebbles system focusing on indirect interactions of mobile devices with remote screens [13], the Stitching interaction technique that supports pen gestures that span multiple displays [8] or Pick & Drop supporting data transfer between different devices [17]. None focus on touch-based mobile interactions with physical objects, and most were focused rather on application and interaction rather than on the provision of a user interface toolkit.

The related work shows, on the one hand, the trend towards multi-tag applications, and on the other hand, the lack of a corresponding toolkit which addresses the special requirements regarding the support of user interfaces that span mobile device and physical interface.

3. MULTI-TAG APPLICATION EXAMPLES

Three prototypes that exemplify particular usages of the toolkit are described in this section. The fundamental configuration remains the same for all three prototypes (Figure 1). A set of NFC

tags is attached to the rear of a display layer. This layer hides the tags from the user and may come in the form of a paper sheet (e.g. for posters, labels or a sheet used to project onto) or a direct-view screen (e.g. an LCD or TFT screen). With this configuration, the phone reads through the display layer, which, in turn, provides a representation of the role of the tag. In Figure 1, we see only a single tag, whereas a multi-tag interface will have multiple tags attached to the display layer. When contrasting single-tag and multi-tag interaction, the need for a multi-tag toolkit does not arise from the fact that there is a different development process for either. Essentially, multi-tag interaction comprises of multiple, single tag interactions and the toolkit could be used to cater for both types of interaction. However, with multi-tag interaction, there is capacity for higher complexity with respect to the NFC interface. It is this potentially greater complexity that drives the need for a toolkit that can accelerate the development process.

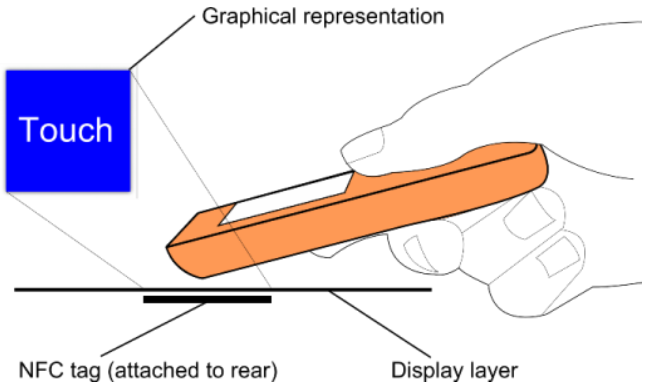

Figure 1. The basic configuration shared between prototypes

3.1 The Tags-to-context Prototype

The tags-to-context prototype provides a physical interface that allows users to make their latest experiences viewable for their friends via a web site or application on supported phones [8]. The experiences are categorized into groups such as activity, status, and mood. The prototype has three components: a user interface for configuration, an NFC phone, and a set of tags.

The configuration user interface allows the user to specify a number of user experiences (e.g. drinking coffee). These experiences are specified by touching the experience options with the phone (Figure 2-1). Each option has a corresponding NFC tag behind it that stores the meta-data for the option (Figure 1). Once one or more experiences have been specified (read to the phone), they can be written to a context tag. As soon as the experiences have been written, the context tags are used to update the user's experience with subsequent reads (Figure 2-2).

Figure 2. The tags-to-context interaction sequence

The tags can then be labeled using a pen or by attaching a sticker and the tag is ready to be attached to a meaningful physical object. Once deployed, whenever the user drinks coffee, they can touch the context tag in order to make their friends aware of their action. On notification, their friends may then join the user for a coffee. While the options are read from the configuration user interface, the phone screen is used to display currently read options.

3.2 The Static Tourist Guide Poster

The second example is a static (non-digital display) tourist guide poster (Figure 3). The printed poster is augmented with NFC tags that are used to provide extra functionality, yet take nothing away from the traditional use of the map. One deployment scenario is that many posters can be created and installed at various bus shelters around the city. The poster can then be used to provide users with information regarding nearby points of interest (POI) (e.g. hotels and restaurants). This is achieved by using a matrix of tags that provides complete coverage of the map.

Figure 3. The tourist poster static physical interface

The map grid is used to indicate the position of the tags behind the map. In order to read a POI, the user simply touches the corresponding grid area with the phone. When there are multiple POIs in a grid area, the POI identifier can be selected using the phone keypad. POIs are identified locally within the bounds of the tag (e.g. if there are three POIs in one map grid, their identifiers will be 1, 2, and 3). Global identification is achieved by combining the local identifier with the grid reference represented by the tag (e.g. A7-1).

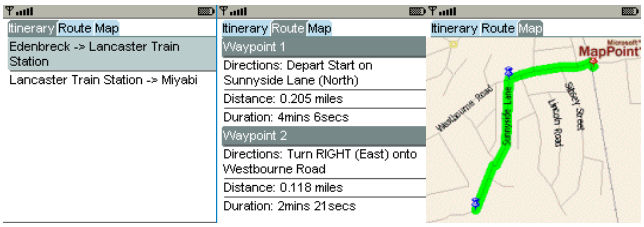

Figure 4. (*left*) an itinerary on POIs (*middle*) directions between POIs (*right*) a map for rendering the directions

The map also allows routes to be constructed by selecting multiple POIs. Dependence on the phone display is used in order to support route planning. Figure 4 illustrates this use. The itinerary (left) shows the POIs that have been selected from the map. Directions (middle) between two POIs can then be requested from the itinerary and can be visualized graphically (right).

3.3 The Dynamic Tourist Prototype

The third example is a dynamic (digital display) prototype, which looks similar to the previous, static example. With the dynamic prototype, a projector displays the feedback from the poster. The goal of the dynamic prototype is to extend the features of the static poster prototype by taking advantage of a large dynamic multi-tag interface. Further utilizations of the dynamic display are: map panning, zooming functionalities, navigational route overlays, information overlays, a new dynamic rollout menu widget (Figure 5, left), POI filtering, and enlarged areas for POI selection (see Figure 5, right).

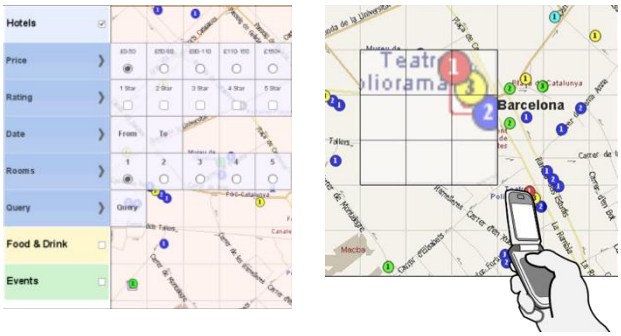

Figure 5. (*left*) The rollout menu (*right*) tag enlargement for the dynamic map-based prototype

When considering both the static and dynamic tourist guides, one can imagine utilizing the advantages of both. The dynamic guide has the advantage that it can abstract data (e.g. in the rollout menu), and thus, provide a great deal of functionality to its users.

The static poster is cheap to produce and replace, in addition to it being potentially portable. It therefore could be produced as a snapshot of the current state of the dynamic guide. For example, tailored to hotel POIs for a user, or tailored to relevant POI search results from the dynamic version. The posters could be placed outside the POIs themselves with a snapshot of their information options made visible.

3.4 Summary of the Prototypes

A theme connecting all three prototypes is the sharing of user interfaces between the phone and a physical display. With the (static display) prototypes 1 and 2, the phone interface is dependent on the physical interface, whereas with (dynamic display) prototype 3, the phone user interface takes more of a complementary role.

However, there is also much dissimilarity between the prototypes. There are two main causes for this; firstly, the display types are different, and secondly, the networking support is different. Pertaining to the display type, prototypes 1 and 2 use static paper displays. As it is not necessary to use a server to drive the interface we cannot assume that a short-range wireless technology, such as Bluetooth, is required. It is more likely that the phone will make requests over the Internet (incurring significant round-trip times and monetary cost to the user). As an alternative implementation for prototype 3, a Bluetooth connection can be used if the server is directly connected to the display.

When considering a toolkit that is capable of supporting these three types of multi-tag applications (plus further varieties), we first need to be certain of its requirements. From our experiences with the development of the three aforementioned prototypes, we conclude that the list shown in Table 1 provides a minimum set of requirements for the toolkit. These target the ability to develop a broad-range of prototypes and also deal with the idiosyncrasies of multi-tag interfaces, such as differing NFC display types.

Table 1. Toolkit Requirements

1	Support user interface sharing between the phone and physical displays.
2	Provide networking support via Bluetooth and Internet connectivity.
3	Provide support for physical interfaces of varying display feedback types (both static and dynamic)
4	Provide a multi-tag NFC record type.
5	Provide GUI creation that suits the tag geometry rather than an adaptation of existing desktop GUI toolkit.
6	Provide a mobile client that can work with each type of prototype.
7	Support the prototyping of static interfaces using dynamic interfaces.

4. OVERVIEW OF THE CORE TOOLKIT

The toolkit provides support for the developments of NFC solutions with particular focus on multi-tag interaction. At the hardware level, there are three main components: the NFC tags, the phone, and the server (Figure 6). The NFC tags are attached to a display in order to convert this into a physical user interface. The phone is the "smart stylus" used to interact with the physical interface by touching them together. The server takes the role of the back-end for the physical interface (providing services and application logic) and is also used in order to generate the physical interface.

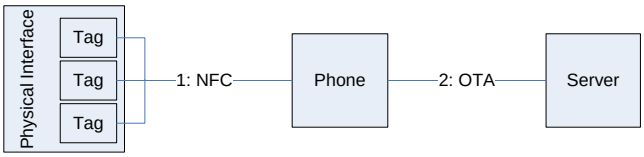

Figure 6. The hardware components of multi-tag applications

4.1 Data Flow Between the Hardware Components

Typically, when a physical interface is read (Figure 7), the communication flow travels from the tags to the phone and onwards to the server (1). Subsequent flow may take place between the phone and server using the various input modalities of the phone (2). This process is true for every dynamic physical interface (Figure 8a) tag as there is no strict mapping between the tags and the corresponding data. But when a static physical interface is used, in many cases, the data will be static on the representative tags. In these situations, the application needs not communicate with the server (3) as the tags can store the absolute data rather than a link to the data on a server (Figure 8c). Storing the actual data on the tags offers various advantages over sending OTA (Over-The-Air) requests from every tag read and the option on downloading the entire application on the first tag read. Firstly, OTA requests over the Internet can incur significant round-trip times that are often impractical considering user interaction responsiveness. Secondly, downloading the whole application forces a greater barrier to entry regarding first-time use, not to mention forcing the user to have many applications on their phone rather than one generic one.

When absolute data on the tag is not possible due to its rigidity, an ID-based widget link can be established to the server widget representation (Figure 8b).

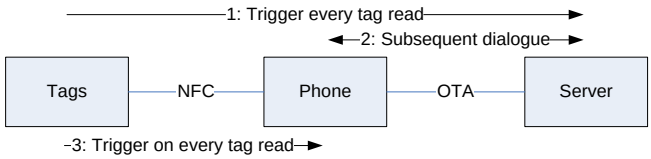

Figure 7. Reading from the tags

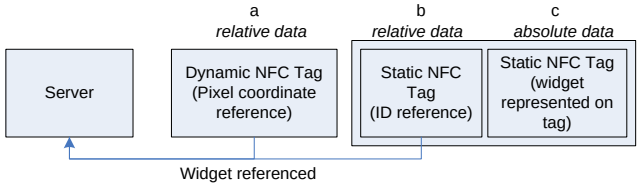

Figure 8. Various tag combinations

To summarize, the toolkit reliance on each hardware component depends on the type of interface and presence of networking infrastructure. For example, with a static poster, it is likely that the poster must be supported by Internet connectivity (rather than Bluetooth, as there is no nearby server driving a display, cf. dynamic displays). By reducing server connectivity as much as possible, one can alleviate round-trip-time delays and, consequently, the undesirable effect on interaction feedback timeliness.

4.2 Multi-tag Widgets

The core software components are focused around a reusable collection of multi-tag user interface widgets. The majority of multi-tag widgets are synonymous with those used in desktop widget libraries (e.g. radio buttons, drop-down boxes, etc.). However, the distinctions with multi-tag widgets are that:

- They transform to suit both dynamic and static display types (in accordance with req. 3).
- The widgets are represented on the *tags* via XML descriptions, the *phone* interface via Lightweight User Interface Toolkit (LWUIT) widgets [22], and the *server* via a multi-tag scalable vector graphics (SVG) widget set.

To support the transformation of the widgets to a different display type, the model-view-controller pattern is applied within each widget. By separating the model from the views and controllers, multiple views and controllers can look onto a single model; thus, the phone and physical interfaces could run in parallel. Moreover, the views and controllers can be replaced with without affecting each other or the model. The dynamic views and controllers act very much like any of those for traditional desktop graphical user interfaces. Controllers receive events (from the phone), update the model, and instruct the view/s to render the changes (Figure 9, top). However, the static views and controllers typically operate by controlling the phone through XML-based messages. The same messaging format can also be used by the local tags to control the phone, e.g. to display a radio button of the phone UI (Figure 9, bottom).

The server user interface is supported by a powerful implementation that marries Piccolo (for efficient interface management), SVG Salamander (for per widget graphics), and bespoke look-and-feel, animation, and event-handling mechanisms. By using SVG for the server widgets, the interface supports lossless rescaling capabilities to suit different NFC tag sizes (in accordance with req. 5). In addition, the developer has a great deal of freedom when creating graphics using a visual graphics editor. The phone client uses LWUIT widgets as a mobile version of the multi-tag widgets. This toolkit provides much greater flexibility than the standard Java ME user interface toolkit, whilst additionally retaining a small memory footprint.

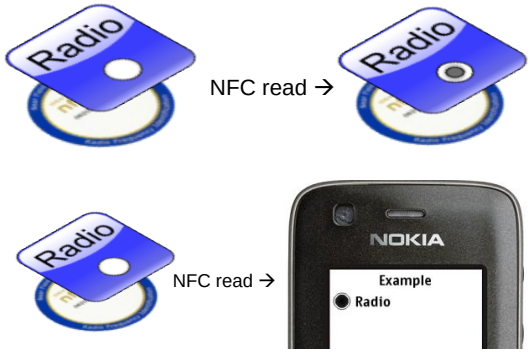

Figure 9. (*top*) initial tag graphic on static display (*bottom*) the phone screen used to show the new state

A look-and-feel is used to dictate the mappings between the models, views and controllers. The look-and-feel is used in the same way as in many traditional desktop look-and-feels (e.g. Microsoft Vista). However, with traditional examples, there is a greater weighting towards the look (cf. *skinning*), whereas with the multi-tag look-and-feel, the feel is radically changed.

5. COMPONENT-LEVEL DETAIL

The various software modules that contribute to the toolkit can be allocated into four groups. These groups are: (a) the NFC tag, (b) the phone client, (c) the common communication library, and (d) the server.

5.1 The NFC Tag

There are currently two types of NDEF records for the toolkit: a multi-tag record (MIME format) and a multi-tag command record (Text RTD). The multi-tag record is mandatory for all tags as it maps each tag to its virtual counterpart. The optional multi-tag command record contains a WAP-based XML file. This can be used to control the phone client through the various generic methods provided.

5.2 The Phone Client

The phone client's responsibilities include the detection of NFC tags and parsing of the corresponding records (in accordance to the NFC multi-tag record type). The phone also contains a collection of generic methods that can be used to: display, store, and retrieve command descriptions, navigate to the previous interface, and store undo states.

The phone client uses LWUIT widgets as a mobile version of the multi-tag widgets. This toolkit provides much greater flexibility than the standard LCDUI API, whilst additionally retaining a small memory footprint. Persistent storage can be used for storing command descriptions and corresponding results for archival. The phone's storage is primarily targeted at passing data between services and offline retrieval by the user through the phone interface.

5.3 The Common Communication Library

This common library is shared between the server and phone client. It contains an RPC client and server implementation and allows communication between them. The RPC calls contain a *Multi-tag Event Header* that enables the phone client to provide the server with a range of event details. These details specify whether the event is from a tag read or not, whether the tag is configured for a static or dynamic display, tag-to-widget mappings, tag read options, gesture types, and specific phone keys that have been pressed (normally in conjunction with a tag read).

5.4 The Server

The server's main responsibilities are to orchestrate the deployment of the multi-tag applications and connect the user interface to the application logic. A widget module manages the multi-tag widgets. Figure 10 shows the multi-tag widget hierarchy (the patterned widgets are those that can be directly mapped to LWUIT widgets).

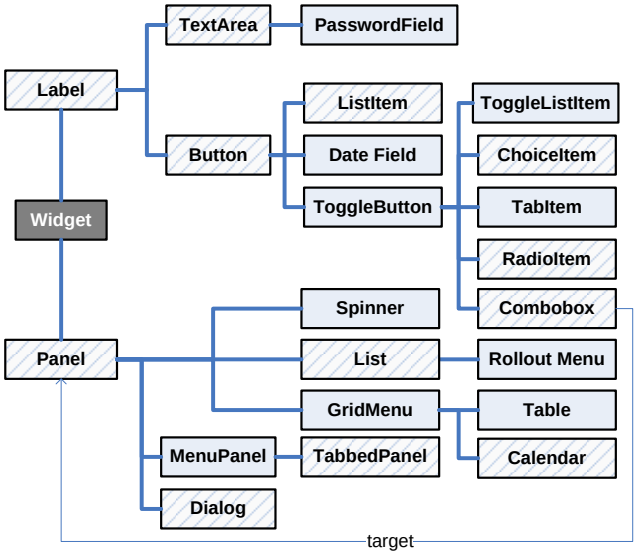

Figure 10. The widget hierarchy (patterned widgets are directly mapped to LWUIT)

Alternate widgets can be substituted for widgets that are not directly mapped (e.g. a ChoiceItem for a ToggleButton). The hierarchy is designed for a high level of reuse as this core set of widgets is designed to be extended for widgets that are greater tailored to the needs of a specific application.

Regarding the widgets, the server's role involves widget creation (using the assigned look-and-feel), managing the state of the widgets, and pushing events to the widgets. It centralizes the widgets into a single storage container and is, therefore, the ideal place to perform user interface transformations (via look-and-feel switches).

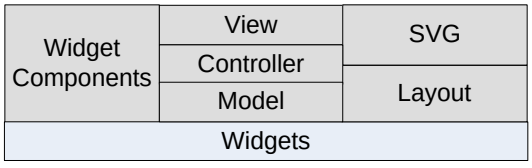

Figure 11. The major components of the widgets

The widgets can also be broken down into *Widget Components*. Examples of these components include icons and text that can be reused and painted separately on rendering updates. The layout component is responsible for the layout of widgets themselves and any widget components associated to it. It can also be seen that each widget contains the MVC structure; the widget component views and models can also be decoupled for greater reuse.

6. USER INTERFACE TRANSFORMATIONS

A look-and-feel switch allows a new collection of widgets to replace the existing widgets. For example, these widgets maybe tailored towards static posters rather than dynamic displays. This enables the whole interface type to automatically switch at runtime. The components that are affected by a switch in the look-and-feel are the views and controllers; the models are unaffected. Consequently, to facilitate a switch at runtime, the controllers and views must have no state that would otherwise be shared with alternate views and controllers. This shared state must be centralized in the models.

During an interface switch from a dynamic display to a static display, we find that we cannot select options that are not currently visible (e.g. options hidden in a combo box) as we have no dynamic feedback. Therefore each widget *that is currently visible to the user* is iterated by the toolkit to check whether there are other widgets hidden behind the current, parent widget. For example, in Figure 12, left, the options behind the rollout menu are not visible if a poster is used. Therefore, a static look-and-feel will use widgets that command the phone to show these hidden options using its display. However, if the rollout menu is expanded (Figure 12, right) to make all the options visible, the phone UI does not have to be used at the cost of greater display space usage on the NFC display.

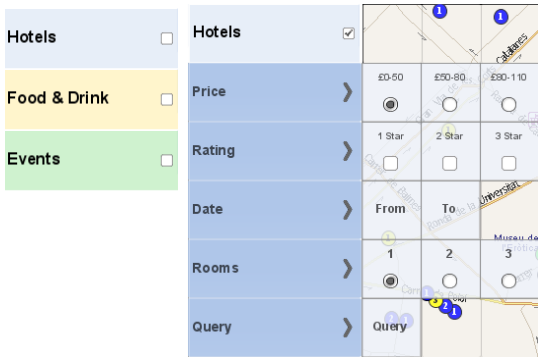

Figure 12. The rollout menu from the dynamic tourist guide application (left) the menu tabs (right) a selected tab

The interface switch feature allows a dynamic version of a user interface to prototype a static version. With the dynamic version, the designer has the power to configure the interface in a particular way (by opening certain screens, menus, etc.) before it is switched to a static version.

Exemplifying the possible options for a designer, the *Hotels* tab (Figure 12, left) could remain deselected, which would force the phone interface to display the combo boxes (e.g. price) as tabs (each displaying their corresponding options). Alternately, the user could select the *Hotels* tab, but not expand the combo boxes. This would allow a particular tab (e.g. price) to be directly accessed on the phone. The final approach is to occlude the map and allow all the options to be directly accessed via the tags.

In order to push the user interface to the phone, hybrid widgets can be used. Hybrid widgets are widgets on the NFC interface that link to widgets on the phone interface. The theory behind hybrid widgets is twofold: they save space (and tags) and they enable static interfaces to provide dynamic feedback. An example use is the replacement of a static, physical calendar widget for a calendar hybrid widget. The physical calendar widget can occupy up to 31 tags. By trading off interaction consistency with interface space, a hybrid widget could be used that occupies a single tag that prompts a phone-only calendar widget on the phone interface. Moreover, this widget may also have dynamic feedback such as an indication of the current date. In the case of the design seen in Figure 12, left, that is applied to a poster, three hybrid widgets would push the content behind the roll-out menu tabs to the phone.

7. PROTOTYPE ADAPTATIONS

The placement of the MVC components changes in accordance to *providing support for physical interfaces of varying feedback types* (Table 1, req. 3). This is also true for the deployment procedure for varying types of multi-tag application.

7.1 Creating a Dynamic Multi-tag Interface (Deployment & MVC Migration)

When developing the application (Figure 13-1), the look-and-feel is set to a dynamic version. The GUI can then be created and is purposefully similar to many traditional desktop GUIs (e.g. SWING). The physical dimensions of the NFC tags can be input to set default sizes for the graphics. The business logic is then connected to the GUI widgets through the use of event listeners.

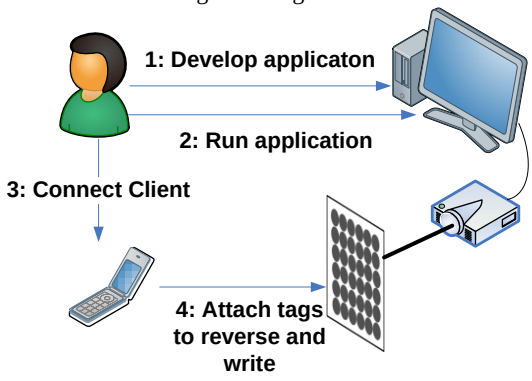

Figure 13. Deployment scenario (Dynamic)

The developer can also configure the toolkit to begin in deployment mode (Figure 13-2). In this mode, once the client connects (Figure 13-3), the server orchestrates tag writing in order the write the corresponding x/y coordinates to every tag (Figure 13-4). The tags are arranged in a matrix prior to writing.

When focusing on the migration of the MVC components, the models, views, and controllers for each widget are all located on the server during the development of the given application (Figure 14-1).

This is necessary as the server orchestrates the migration process. Once the client is connected (Figure 14-2), a view role is pushed to the phone V1/2 (where V1 represents a mirroring of the server view and V2 represents a complementary view (via a hybrid widget)). Once the phone begins writing relative data to the tags (also part of Figure 14-2), a new model on the server is placed on each tag (M2). M2 provides a link to the corresponding controller (C1) on the server. The server retains its representation of the widget; therefore, the model is updated on the server and rendered on a dynamic display.

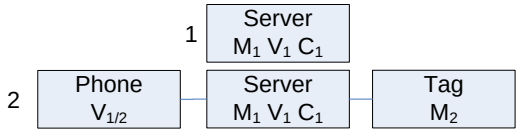

Figure 14. Dynamic MVC migration

7.2 Creating a Static Multi-tag Interface

The development process (Figure 15-1) is the same as that of the dynamic display (Figure 13-1). However the look-and-feel must be set to a static version. The appropriate static widget set can then be used.

Once the GUI has been created, the application can be run so the GUI can be rendered for printing (Figure 15-2/3). The tags can then be attached to the back of the printout (Figure 15-4).

Deployment mode is then entered once the client connects and the data is written to each tag (Figure 15-5/6). The level of data depends on whether the widgets are represented completely by the tags or simply a link to a server representation. In the former case, a XML file is built that represents a description of the widget to which it is associated. Once deployment is complete, the system continues running and is ready to be used by users.

There are two MVC migration perspectives, which are based on whether the server is required or not. An example when a server is not required would be when a tag represents, for example, a radio button. Here, the tag stores an XML file with details about the radio button. The phone is able to parse this option and display its status on its display. Only when a query needs to be sent does the server become involved. Therefore, various tags may represent local parameters for a query that can be selected without requests from a server. Once the desired parameters have been selected, final tag can be used to push the parameters to a remote server.

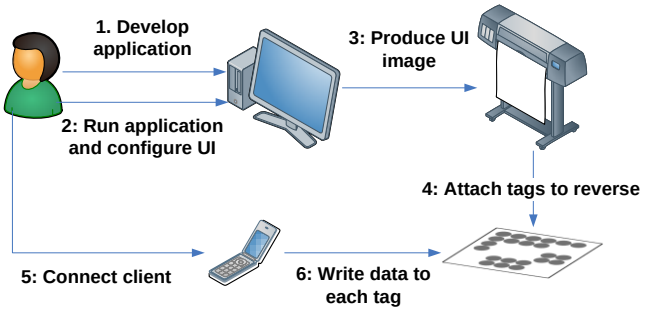

Figure 15. Deployment scenario (Static)

We will begin with the example where a server is not required, for example, reading experiences from the configuration interface to the phone in Prototype 1. The process can be seen in Figure 16. Here, the components are centralized on the server (Figure 16-1) – again to orchestrate the migration process. When the application is run, we move to stage (Figure 16-2). The display is printed and a view on the tag (V_1) signifies that the printout represents the model, though from a processing perspective, this is produced by the server. When the tags are written to, a description of the widget fully represents the model of the widget (M_1).

The controller (C_1) is moved to the phone as this is now responsible for updating the model. Although there is initially no representation of the model on the phone, once the tag is read, the model (M_2) is transferred from the tag to the phone (Figure 16-3). The controller then updates the model in accordance to the tag

read (e.g. a selection) and takes the role of the view (V_1). V_1 is required on the phone because the V_1 on the tag may not represent the model now as it is static (e.g. show that a radio button is selected).

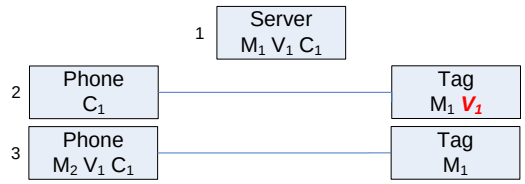

Figure 16. MVC Migration (absolute data on tag)

M_1 on the tag can be assumed to be out-of-date (as the new state is not re-written to the tags) when M_2 is created on the phone. However, M_1 is still retained as it links to M_2. Therefore, for future reads to a tag that has already been read (i.e. has a corresponding M_2 on the phone), M_1 is used to access (and trigger) an update on M_2.

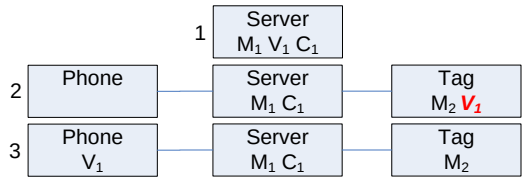

Figure 17. MVC Migration (relative link on tag)

If a server is required, as in Figure 17 (i.e. the widget must link a counterpart on the application), the migration of the MVC components differs. In Figure 17-2 (on application initialization,), the server must retain the model (M_1) and controller (C_1) as they are used to link to the application. However, as the display type is static, the dynamic view is no longer required on the server. As with the previous case, the view is initially represented by the printout. However, only a link represents the M_2 on the tags so the full representation on the server (M_1) can be accessed by the phone. After a tag read (Figure 17-3), the representation between the V_1 on the printout and the M_1 on the server may be erroneous; therefore, the phone takes the role of the view (V_1).

7.3 Switching from a Dynamic Interface to Static Interface

When switching from a dynamic to a static interface, the switch takes place at runtime. In order to achieve this, the views and controllers must have no dependencies. In addition, any partial dialog between the server and phone will be lost.

1. The GUI is developed and initialized using a dynamic display look-and-feel.
2. The GUI is then set in a particular state (e.g. opening a tab to make various options visible).
3. The switch is then made to a static look-and-feel. This triggers the widget controllers and views to be changed accordingly. The current GUI is then exported for printing. Finally, the deployment stage is entered once more to write the new tag data that provides the tags to directly specify the widget they represent.
4. The MVC migration involves replacing the views and controllers for each widget in correlation to the new look-and-feel. They can then be migrated to the phone and tags in accordance to Section 7.2.

8. PHONE AND PHYSICAL INTERFACE MIRRORING

Mirroring is a technique in which the phone interface mirrors the focused section the physical interface. The partitioning of the physical interface into sections allows effective mirroring on the phone. Each section represents a logical grouping of widgets by the developer. Each group (e.g. a radio button group) is represented by a container widget or subclass thereof. A mandatory requirement for the groups is a label, as each group is mapped to a tab on the phone interfaces (Figure 18). Nested groups can be accommodated on the phone through the use of nested tabs.

A set of rules dictates the mapping process, which can be seen below.

1. A logical grouping of interface options (e.g. a radio button group) maps to a tab on the phone screen.
2. Developers must be conscious of the physical interface option grouping and its effect of the phone interface (with regards to the number of options to display via the phone).
3. Ungrouped physical interface options are automatically grouped into the first tab. Subsequent tabs are mapped to the top groups if present. Groups within groups are displayed on the phone as nested tabs (Figure 18, right).
4. Grid layouts are retained (e.g. calendars, timetables, grid-based games, maps) by default. There are various zoom levels for the phone that are useful for large grids such as a map. Non-grid layouts are represented as single column lists by default.

Figure 18. Mapping from groups (from the static tourist guide poster) to phone interface tabs

If mirroring is not required, rules must govern when tags require relative data or absolute data. The choice can be made when considering what dependencies the widget has to: (1) the developer's business logic (e.g. retrieve a precipitation forecast), and (2) other widgets that are observing the widget (e.g. a textbox may change its value when the widget is selected). From an implementation level, these dependencies are easy to infer as the listener and observer lists can be searched.

The phone also supports partial-mirroring in order to support greater consistency between the phone interface and the physical interface. With this approach, the phone displays all the options in a particular group, the difference with full mirroring being that the user cannot access other groups through the tabs. Partial mirroring can be used without full server dependency, providing there is not a plethora of options that cannot all be stored on tags, and is used primarily in the *tags-to-context* prototype.

9. SUMMARY OF TOOLKIT FEATURES

One of the very core features of the toolkit is the support of multi-tag interfaces that can be represented on a variety of display mediums (posters, dynamic displays, etc.). In addition to providing support, the same interface can be transformed to one that is tailored for another display medium. For example, a dynamic display interface could switch to a poster interface. A scenario to demonstrate this feature may be a tourist information

office that provides multi-tag tourist services using the dynamic display prototype (see Section 3.3. As the display is dynamic, it may show a dynamic map that can be panned and zoomed in order to view POIs in greater detail. It may also show information options for a particular POI. To complement, augmented posters could be installed at each POI and provide the same information options as the dynamic display for the POI the poster represents. These can be automatically generated as a *snapshot* of the fully functional dynamic multi-tag interface, simply requiring printing and NFC-tag augmentation, see for example, the prototype in Section 3.2.

A by-product of supporting the concept of interface type switching is that dynamic multi-tag interfaces can be used to prototype static designs. Once the developer is happy with a particular design, the switch to a poster interface can be easily made by printing the poster and writing to the tags. This process is much more forgiving regarding mistakes as they are easy to rectify during the dynamic prototyping stage. What's more, it is not necessary to print potentially many posters when evaluating poster design usability.

A key advantage of using a mobile phone as an interaction device is that a mobile phone contains all the required input/output components to continue interaction away from the physical interface. Phone-only interaction is by no means considered an alternative method of interaction when a physical interface is present; rather it is considered a complement. However, it can be used as a suitable alternative when no physical interface is present. For example, a user may be browsing POIs (nearby their chosen destination) whilst waiting at a train station. When the train arrives, they are forced to leave the physical display and can continue browsing POIs on the train using only the phone interface. In order to achieve this offline continuation, the phone interface and physical interface must be mirrored.

A generic client serves the purpose of lowering the barriers to entry for accessing the new multi-tag applications. This is owing to a number of reasons; the most obvious is the fact that the client does not have to be downloaded and installed more than once. This encourages new users to try a particular application. Another advantage of a generic client is that the user does not build a large collection of separate applications in the mobile device and consequently saves on memory usage and searching time for the application. A further advantage is that the multi-tag services can use the single application to pass information between one another. For example, a user may begin to build a user profile that can be used globally across multiple services, rather that demanding re-entry of details for each service.

10. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a toolkit that is targeted at providing developers with the resources to support the development of multi-tag NFC applications. As well as providing developer support, the toolkit can support a diverse range of NFC applications whilst retaining interaction consistency. The development of the toolkit is driven by the complexities of sharing the user interface between the phone and physical interface, and the difficulty of adapting a GUI that is intended for desktop use to an interface that suits the geometry of the tags. The toolkit provides a tailored model for the widgets that allows a very high level of reuse for extending the widget hierarchy. Moreover, the developers have complete freedom over the appearance of the widgets with the advantage that they have lossless scaling properties.

Three prototypes exemplify the versatility of the toolkit; these vary in sophistication, display of feedback, NFC demands, and networking demands. One of the main features of the toolkit is not only to support dynamic and static interface types, but also to switch between them, as both have their own advantages and disadvantages depending on the context they are used in.

Future work will be focused around preventing malicious rewrites to the tags through a security mechanism on the tags. Also, eliciting further usability aspects with multi-tag interfaces for integration into the toolkit. The Keystroke-Level Model (KLM) could be adapted for multi-tag interaction and used to gain an insight into the efficiency of different interface designs.

11. ACKNOWLEDGEMENT

The presented research was conducted in the Multitag project which is funded by DOCOMO Euro-Labs and supported by the Emmy Noether research group "Mobile Interaction with Pervasive User Interfaces" funded by the DFG.

12. REFERENCES

[1] ABI Research: NFC Chipsets to Grow Steadily into 2012. http://parts.ihs.com/news/2008/abi-nfc-chipsets.htm

[2] All new Nokia smartphones to come with NFC from 2011, http://www.nearfieldcommunicationsworld.com/2010/06/17/33966/all-new-nokia-smartphones-to-come-with-nfc-from-2011/

[3] Ballagas, R., Rohs, M., Sheridan, J., and Borchers, J. The Design Space of Mobile Phone Input Techniques for Ubiquitous Computing. *In Handbook of Research on User Interface Design and Evaluation for Mobile Technologies.* IGI Global, Hershey, PA, USA, 2008.

[4] Broll, G., Siorpaes, S., Rukzio, E., Paolucci, M., Hamard, J., Wagner, M., and Schmidt, A. Supporting Mobile Service Usage through Physical Mobile Interaction. *In Proceedings of the Fifth Annual IEEE International Conference on Pervasive Computing and Communications (Percom 2007)*. IEEE, pp. 262-271. 2007.

[5] Guinard, D., von Reischach, F., Michahelles, F., and Fleisch, E.: MobileIoT Toolkit: Connecting the EPC Network to MobilePhones. *In Proceedings of Mobile Interaction with the Real World (MIRW 2008)*. 2008.

[6] Häikiö, J., Wallin, A., Isomursu, M., Ailisto, H., Matinmikko, T., and Huomo, T. 2007. Touch-based User Interface for Elderly Users. *In Proceedings of the 9th international Conference on Human Computer interaction with Mobile Devices and Services (Mobile HCI 2007)*. ACM, pp. 289-296. 2007

[7] Hardy, R. and Rukzio, E. Touch & Interact: Touch-based Interaction of Mobile Phones with Displays. *In Proceedings 10th International Conference on Human-Computer Interaction with Mobile Devices and Services (Mobile HCI 2008).* ACM. 2008.

[8] Hardy, R., Rukzio, E., Holleis,P. and Wagner, M. MyState: Sharing Social and Contextual Information through Touch Interactions with Tagged Objects. *In Proceedings 13th International Conference on Human-Computer Interaction with Mobile Devices and Services (Mobile HCI 2011).* ACM, pp. 475-484. 2011.

[9] Hinckley, K., Ramos, G., Guimbretiere, F., Baudisch, P., and Smith, M. Stitching: Pen Gestures that Span Multiple Displays. *In Proceedings of the Working Conference on Advanced Visual interfaces (AVI 2004).* ACM, pp. 23-31. 2004

[10] Kindberg, T., Barton, J., Morgan, J., Becker, G,. Caswell, D., Debaty, P., Gopal, G., Frid, M., Krishnan, V., Morris, H., Schettino, J,. Serra, B., Spasojevic, M.: People, Places, Things: Web Presence for the Real World. *In: Mobile Networks and Applications*, 2002.

[11] Koskela, M., Ylinen, J., and Loula, P. A Framework for Integration of Radio Frequency Identification and Rich Internet Applications. ITI 2007.

[12] McDonald's Pilots RFID Self-Ordering System in Korea, http://www.rfidupdate.com/articles/index.php?id=1444

[13] Myers, B.A. Using Hand-Held Devices and PCs Together. *Communications of the ACM 44 (11)*. pp. 34 - 41. 2001

[14] NTT DoCoMo i-mode Felica, http://www.nttdocomo.co.jp/english/service/imode/make/content/felica/index.html

[15] O'Neill, E., Thompson, P., Garzonis, S. and Warr, A. Reach Out and Touch: using NFC and 2D Barcodes for Service Discovery and Interaction with Mobile Devices, *In Proceedings of Fifth International Conference on Pervasive Computing (Pervasive 2007)*. Spring, pp. 19-36. 2007

[16] Reilly, D., Rodgers, M., Argue, R., Nunes, M., Inkpen, K.: Marked-up Maps: Combining Paper Maps and Electronic Information Resources. *In: Personal Ubiquitous Comput.*, 10 (4), pp 215-226.

[17] Rekimoto, J. Pick-and-drop: a Direct Manipulation Technique for Multiple Computer Environments. *In Proceedings of the 10th Annual ACM Symposium on User interface Software and Technology (UIST 1997)*. ACM, 31-39. 1997

[18] Rouru-Kuivala, O., Project Manager, Interaction, as simple as touch. http://www.nfc-forum.org/resources/multimedia/gsm08_press_luncheon_presentations/City_of_Oulu_-_Barcelona.pdf

[19] Rukzio, E. Physical Mobile Interactions: Mobile Devices as Pervasive Mediators for Interactions with the Real World. PhD Dissertation. Faculty for Mathematics, Computer Science and Statistics. University of Munich. 2007.

[20] Sánchez, I., Riekki, J., and Pyykkönen M. Touch & Compose: Physical User Interface for Application Composition in Smart Environments. *Proc. 1st International Workshop on Near Field Communication (NFC'09)*. 2009

[21] Seewoonauth, K., Rukzio, E., Hardy, R., and Holleis, P. Touch & Connect and Touch & Select: Interacting with a Computer by Touching it with a Mobile Phone. *In Proceedings 11th International Conference on Human-Computer Interaction with Mobile Devices and Services (Mobile HCI 2009)*. ACM, 2009.

[22] The Lightweight User Interface Toolkit (LWUIT): An Introduction, http://java.sun.com/developer/technicalArticles/javame/lwuit_intro/

[23] VVT, SMARTTOUCH newsletter, Dec. 2007. www.vtt.fi/liitetiedostot/muut/2007Newsletter_WEB.pdf.

[24] Want, R., Fishkin, K.P., Gujar, A., and Harrison, B.L. Bridging Physical and Virtual Worlds with Electronic Tags. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI 1999)*. ACM. 1999.