BIT – A Browser for the Internet of Things

Christof Roduner

Institute for Pervasive Computing, ETH Zurich, 8092 Zurich, Switzerland
roduner@inf.ethz.ch

Abstract. Mobile phones are increasingly able to read auto-id labels, such as barcodes or RFID tags. As virtually all consumer products sold today are equipped with such a label, this opens the possibility for a wide range of novel digital services building on physical products. In this paper, we discuss the problems that arise when such novel applications are deployed and present a *Browser for the Internet of Things* (BIT), which facilitates the development of such consumer services on the mobile phone platform.

1 Introduction

Personal mobile devices, such as mobile phones and PDAs, represent an important building block in many of the systems discussed in the ubiquitous computing community. Researchers have used them for prototypical implementations of a vast array of applications ranging from attaching digital annotations to physical objects [1, 2] and controlling large public displays [3, 4] to interacting with appliances of all sorts [5, 6].

Many of these applications leverage auto-id tags, such as visual markers or labels based on radio frequency identification (RFID) technology, for the identification of physical objects. With the advent of technologies that allow mobile phones to read such auto-id tags, the two fields of personal mobile devices and auto-id technology have moved closer together more recently. Most mobile phones already feature an integrated camera that has been shown to be capable of decoding the omnipresent EAN/UPC barcode symbols [7] that can be found on virtually all consumer goods. On top of this, some mobile phones already ship with a built-in Near Field Communication (NFC) module that is able to read passive RFID tags.

From an economic perspective, the convergence of auto-id technology and mobile phones opens many very attractive opportunities for businesses. While the benefits of auto-id tags were earlier limited to internal business processes (e.g., enhanced efficiency in supply chain management), it is now possible to leverage this technology throughout a product's life cycle. One of the most promising prospects is for businesses to be able to establish a direct link to consumers that can be followed by simply interacting with the physical product. More specifically, these developments have a number of implications for product manufacturers, third-party businesses, and customers alike:

For a *manufacturer*, it is now possible to deliver added value to customers by enriching its physical products with digital services that can be accessed in a straightforward and intuitive way. Unlike the physical product itself, such digital services are not static and can be evolved over time, thereby ensuring an ongoing appeal and repeated interaction with the product. If these services are tailored to a consumer's needs, this also allows for the personalization of an item, which is not feasible with a physical product alone. They further allow customers to easily get in touch with a product's manufacturer, which can be helpful in situations where support information or personal assistance is needed. Overall, a manufacturer can leverage the combination of tagged products and mobile devices to differentiate its offering and improve customer loyalty.

Opportunities for services based on tagged products are not limited to manufacturers, but arise for *third-party businesses* as well. For a consumer advocacy organization, for example, it is very attractive to "link" a review directly to a physical product, thus making it easily accessible when a buying decision is made in a brick and mortar store. Similar benefits can be reaped by many other businesses that currently offer product-related information on the internet. A price comparison service, for example, can greatly benefit from the fact that users can access its data directly at the point of sale and without manually typing the product name into a search form.

For *consumers*, finally, the possibility of retrieving information and services directly from a physical product is compelling due to the ease of interaction. Information is accessible immediately and intuitively as it is delivered by the physical object without the intermediate step of a manual search on the web. To further enhance ease of interaction and offer the services that are most relevant to a given situation, data that allows the user's context to be inferred (e.g., GPS coordinates) can be taken into account as well.

Typical usage scenarios that we have in mind include the following:

– Users with allergies can use their mobile phone to easily check whether a product contains ingredients that should be avoided.
– A price comparison service can inform shoppers that a product at hand costs less at a nearby store.
– In a store, consumers can look up product reviews before making their final buying decision. Similarly, they can rate tagged products directly on their phones.
– Consumer interest organizations can make users aware if a product that was just scanned conflicts with their views or preferences (e.g., its production possibly entailed child labor).
– Consumables (such as printer cartridges or coffee capsules) can be re-ordered directly on the mobile phone by simply reading the auto-id tag of a product (such as a printer or coffee maker).
– A movie trailer can be viewed on the phone by scanning the barcode printed on a movie poster.
– Users can look up troubleshooting information for a faulty appliance, such as a printer or a coffee maker, which provides a diagnostic code over an

integrated NFC interface. The phone picks up this diagnostic information and receives instructions on how to resolve the problem. Alternatively, a list of nearby repair centers can be displayed.
– For appliances (e.g., a coffee maker) equipped with an NFC interface, users can change settings (e.g., the time the device switches on automatically). The new settings are entered on the phone and then transmitted to the appliance over NFC.[1]

2 Technical Limitations Today

While the scenarios outlined above are likely to be appealing to most end users, developers of such services face a number of challenges due to the technology available today.

Application development for mobile phones is still a cumbersome process. Toolchain and framework support is generally poor. At the same time, the applications that implement the services described above are often very small and do not offer a lot of functionality. Acquiring deep skills in mobile application development and going through the process of setting up a full-fledged development environment is hardly justified. Moreover, many applications are provided by product manufacturers with no expertise in software development. While application development can certainly be outsourced, this is not the preferred way as it makes solutions static and hard to adapt as the associated physical product evolves. Manufacturers need a way to quickly deploy new services as they, e.g., launch new marketing campaigns. The same applies, of course, to traditional websites accompanying a physical product. Unlike mobile application development, however, the programming of an interactive website is a considerably simpler task that requires skills that are much easier to acquire.

This is further complicated by the fact that today's mobile phone market is still fragmented into a number of different, incompatible platforms, such as Java ME, Symbian, iPhone, Android, and Windows Mobile. Every mobile application thus needs to be ported to all major platforms. As an obvious answer to this dilemma, manufacturers may opt for web-based solutions. The problem with this approach is that a mix of HTML, JavaScript, and server-side applications does not enable developers to access phone-specific hardware, such as barcode or RFID readers as well as GPS receivers. Another problem arises from the relatively high latency in today's mobile networks. Applications based on web technology require frequent request-response cycles that are triggered by simple user input. Given the latency commonly observed, this has a negative impact on user experience. Finally, there are still some places without network coverage, such as subway stations. Even without network coverage, a physical object may offer information or services directly to the mobile phone through, e.g., NFC technology or Bluetooth.

[1] The value of mobile phones for carrying out such tasks was investigated in more detail in [6].

From a usability perspective, another challenge stems from the necessity to integrate the various services into a single interaction framework. While services can easily be created and deployed independently, this results in the user's phone being littered with a large number of small applications. Worse yet, the execution of these applications is not automatically coordinated. In order to use a service, the user first needs to manually start the corresponding application before a product can be scanned. Since many objects will not be associated with a given service, scanning will not render a result in many instances, which will discourage users from further interaction. A user's need to simply see "everything my favorite services offer for this product" cannot be addressed in current architectures. A user would have to manually launch one application after another to check whether it offers a service for a given product – an approach that is simply not feasible. Finally, interaction with physical products often happens spontaneously as users are on the go. Users will often discover that a product offers a new service that was previously not known to them. In such a situation, going through the process of installing an application that supports the new service is not desirable. If new software is needed on the mobile phone, it should be deployed on the fly and without disturbing the interaction flow between the user and the physical object.

We believe that the majority of these challenges can be best addressed by a *Browser for the Internet of Things* (BIT). Such a browser can offer a single runtime environment that is home to all sorts of services as described above. From a user's point of view, this browser represents a one-stop shop for interaction with auto-id tagged physical products.

Several other projects have explored frameworks to facilitate the creation of services that allow users to interact with physical objects through mobile phones. The Physical Mobile Interaction Framework (PMIF) [8] provides developers with a generic framework to write applications that support different interaction techniques, such as touching, pointing, and scanning. It frees developers from the need of dealing with the specific technologies used to implement these techniques. The PERCI (PERvasive ServiCe Interaction) project [9] also aims at facilitating physical mobile interaction, however, the focus lies on the automatic generation of user interfaces from service descriptions based on Semantic Web Services. The REACHeS (Remotely Enabling and Controlling Heterogeneous Services) project [10], finally, proposes a system that implements the universal remote control paradigm. NFC tags are used to enable users to physically interact with their environment.

While all of these projects are related to our work, our focus is more on single, low-value physical products than on smart environments. We aim at providing a runtime environment in which services offered by a large number of interested parties can be deployed and executed. In contrast to many other scenarios discussed in the community, the physical objects tend to be more inexpensive in our examples, and both tagged products as well as interaction devices are highly mobile. The emphasis of our research is thus on a lightweight approach that allows for the fast and easy creation of new services in this specific domain.

3 Requirements

Based on the above considerations, we can derive a number of requirements that BIT needs to fulfill in order to support the wide array of services that can be implemented for tagged products:

Reader management. BIT must provide a single environment that manages tag readers (i.e., barcode and RFID readers).

Discovery and presentation. When a tag is read, BIT must discover which services are available and present this information to the user.

Lifecycle management. As soon as users indicate they want to use one of the discovered services, BIT needs to obtain and execute the code needed to offer the service on the mobile phone. Since many services will be invoked several times, their code needs to be cached on the phone, and BIT must regularly check if updates are available.

Unified user experience. In order to ease interaction with the plethora of services that may be provided, BIT must enforce some restrictions with regard to user interface design. For example, the steps needed to stop to use a service should always be the same no matter which manufacturer provides it.

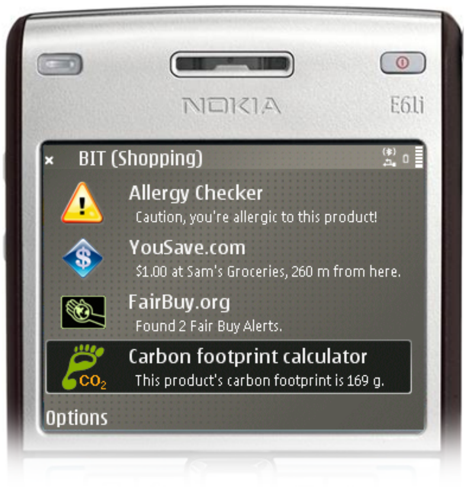
Platform and device independence. As services should run on any platform for which a BIT implementation is provided, device and platform peculiarites must be abstracted. For example, device-specific APIs for the use of RFID or barcode readers as well as location information should be wrapped. Also, independence from available screen estate must be ensured.

Programming abstractions. BIT should offer programming abstractions that free developers from dealing with, e.g., a particular tagging technology (such as RFID or barcodes) or communication technology (such as Bluetooth or GPRS). Instead, it should be possible for developers to simply process reads of physical objects, irrespective of the tagging technology used. Similarly, developers must be able to, e.g., retrieve a physical appliance's error status without needing to worry whether this information is available through NFC, Bluetooth, or IP.

Integration with backend infrastructure. In most cases, product-related information and services that are presented by BIT are stored in a backend infrastructure. In order to discover and retrieve this data, BIT must seamlessly integrate with such a standardized infrastructure.

Storage. Certain services, such as an allergy checker, need to permanently store data (e.g., the user's food allergens). BIT must provide persistent storage for such services.

Security and privacy. BIT must ensure that a service can only access those resources for which it is authorized. For example, a service must be constrained to access its own storage area only. As we envision BIT to be a single tool that mediates all interaction between a user and every physical product's digital offerings, it would be easy to track nearly every move of a user. Privacy is thus a major concern, and BIT must support users in revealing as little about themselves as possible if they wish to do so.



Allergy Checker
Caution, you're allergic to this product!
YouSave.com
$1.00 at Sam's Groceries, 260 m from here.
FairBuy.org
Found 2 Fair Buy Alerts.
Carbon footprint calculator
This product's carbon footprint is 169 g.

Fig. 1. Results list.

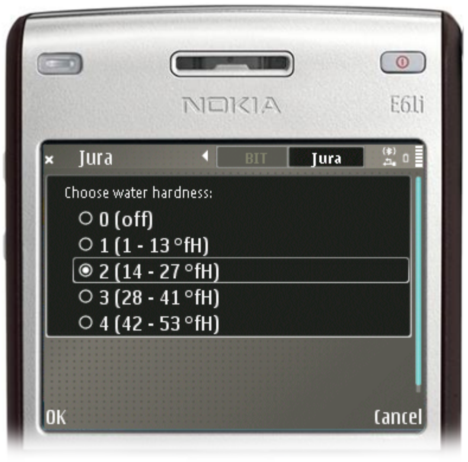
4 Prototype Implementation

Based on the requirements outlined in the previous section, we have implemented a first prototype of BIT to explore what is needed in such a system. We hope to bundle these concepts in a more comprehensive and detailed architecture soon.

Our prototype of BIT is implemented in Python for S60[2] for a Nokia E61i phone. It is capable of reading both EAN/UPC barcodes as they can be found on virtually all consumer goods as well as UHF RFID tags based on EPCglobal's Class 1 Gen 2 standard[3], which is today's most commonly used RFID standard in logistics. We used a Symbian C++ version of the BaToo toolkit[4], which offers robust barcode recognition from within Python. The E61i we use has been modified by Nokia Research and features an integrated RFID reader that we access via a proprietary UDP-based protocol.

In our prototype implementation of BIT, services are provided by *applets*. An applet is a small application that is implemented in BITML, our custom markup language for BIT. BITML allows for the definition of simple user interfaces. Similar to PHP or JSP, developers can embed scripts within BITML code. As a scripting language, we used Lua [11], which we ported to Nokia's Symbian S60 platform. For demonstration purposes, we built a number of applets that provide the following services: (1) a configurable allergy checking service, (2) a controller for a coffee maker (simulated on a Bluetooth-enabled laptop), (3) a product review service, (4) a price comparison service, and (5) a carbon footprint tracking service.

[2] `http://opensource.nokia.com/projects/pythonfors60/`
[3] `www.epcglobalinc.org/standards/uhfc1g2`
[4] `http://people.inf.ethz.ch/adelmanr/batoo/`



Choose water hardness:

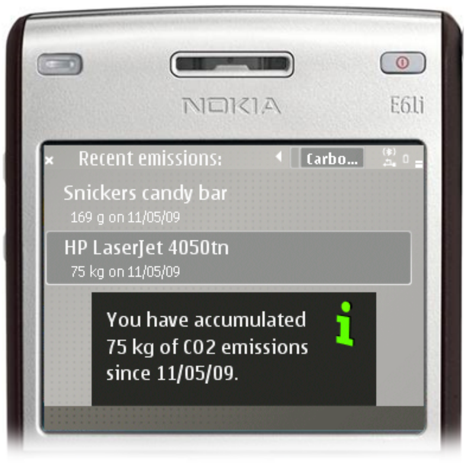
0 (off)
1 (1 - 13 °fH)
[✓ 2 (14 - 27 °fH)]
3 (28 - 41 °fH)
4 (42 - 53 °fH)

Fig. 2. Detailed view of an applet that lets users control their coffee maker. The screenshot shows how the coffee maker's water hardness settings can be adjusted.

When a user scans a physical product's EAN/UPC barcode or RFID tag, BIT decodes the product's identifier, i.e., the EAN number or, in case of RFID, the Electronic Product Code (EPC) as it is used in logistics today. BIT then runs all applets that are installed and passes the detected identifier to all of them. Every applet generates a terse output (e.g., "Caution: contains peanuts") that can be empty optionally. Based on all collected non-empty output returned by the applets, BIT generates a list of results as shown in Figure 1. A user can now browse this list and open applets in a more detailed view. Figures 2 and 3 show the detailed views of two example applets. While Figure 2 shows an applet that lets the user change a coffee maker's settings, Figure 3 illustrates an applet that lets users keep track of the carbon emissions produced by the products they buy.

BIT comes with a number of pre-installed applets. Users are, however, free to install new applets and remove existing ones. In many instances, users will not be aware of all services that a physical product offers. In order to give them a means to discover new services, BIT integrates seamlessly with our open lookup infrastructure [12]. Among other features, the open lookup infrastructure allows both manufacturers as well as third parties to offer applets for download. It also provides a context-aware lookup mechanism, which helps users to find those services that are most relevant to their current situation. BIT leverages this mechanism to suggest new applets, which are installed and executed on-the-fly if the user selects a new service.

In order to facilitate applet development, developers can use the BIT API. The BIT API is a collection of methods that allow applets to access lower-level functions, such as sensors, networking, and storage in a platform-independent way. All applets listed above could thus be implemented using a mix of BITML,



Snickers candy bar
169 g on 11/05/09
HP LaserJet 4050tn
75 kg on 11/05/09
You have accumulated 75 kg of CO2 emissions since 11/05/09.

Fig. 3. Detailed view of an applet that lets consumers keep track of the carbon emissions produced by the products they buy.

the BIT API, and the Lua language only and did not need to rely on system-level features, such as sockets.

5 Summary and Outlook

We presented an overview of possible services that build upon tagged consumer products and future mobile phones that are able to read auto-id labels. We then discussed a number of obstacles that arise when such services are implemented with tools available today. Since we believe that the idea of enriching physical products with digital information and services will become increasingly popular, we see the need for simpler ways to build such services. To address this need, we presented the idea and a prototype implementation of a Browser for the Internet of Things (BIT) that accommodates such services in the form of portable applets. In future work, we will further detail the architecture of BIT, improve the current implementation, and thoroughly test these concepts with more complex services.

6 Acknowledgements

We would like to thank Nokia Research for providing us with the modified E61i phone and its integrated Gen2 RFID reader.

References

1. Carter, S., Churchill, E., Denoue, L., Helfman, J., Nelson, L.: Digital Graffiti: Public Annotation of Multimedia Content. In: CHI '04: CHI '04 Extended Abstracts on Human Factors in Computing Systems, Vienna, Austria (2004) 1207–1210

2. Rohs, M., Roduner, C.: Camera Phones with Pen Input as Annotation Devices. In: Pervasive 2005 Workshop on Pervasive Mobile Interaction Devices (PERMID), Munich, Germany (2005) 23–26
3. Myers, B.A., Stiel, H., Gargiulo, R.: Collaboration using multiple PDAs connected to a PC. In: CSCW '98: Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work, Seattle, WA, USA (1998) 285–294
4. Ballagas, R., Rohs, M., Sheridan, J.G., Borchers, J.: BYOD: Bring Your Own Device. In: UbiComp 2004 Workshop on Ubiquitous Display Environments, Nottingham, UK (2004)
5. Myers, B.A., Nichols, J., Wobbrock, J.O., Miller, R.C.: Taking Handheld Devices to the Next Level. IEEE Computer **37**(12) (2004) 36–43
6. Roduner, C., Langheinrich, M., Floerkemeier, C., Schwarzentrub, B.: Operating Appliances with Mobile Phones – Strengths and Limits of a Universal Interaction Device. In: Proceedings of the 5th International Conference on Pervasive Computing (Pervasive 2007). Volume 4480 of Lecture Notes in Computer Science, Toronto, Canada, Springer (2007) 198–215
7. Adelmann, R., Langheinrich, M., Floerkemeier, C.: Toolkit for Bar Code Recognition and Resolving on Camera Phones – Jump Starting the Internet of Things. In: Informatik 2006 Workshop on Mobile and Embedded Interactive Systems (MEIS'06), Dresden, Germany (2006)
8. Rukzio, E., Broll, G., Wetzstein, S.: The Physical Mobile Interaction Framework (PMIF). Technical Report LMU-MI-2008-2, Lugwig-Maximilians-Universität München, Munich (2008)
9. Broll, G., Siorpaes, S., Rukzio, E., Paolucci, M., Hamard, J., Wagner, M., Schmidt, A.: Supporting Mobile Service Usage through Physical Mobile Interaction. In: Proceedings of the 5th Annual IEEE International Conference on Pervasive Computing and Communications (Percom 2007), White Plains, NY, USA, IEEE Computer Society (2007) 262–271
10. Riekki, J., Sanchez, I., Pyykkönen, M.: Universal Remote Control for the Smart World. In: Proceedings of the 5th International Conference on Ubiquitous Intelligence and Computing (UIC 2008). Volume 5061 of Lecture Notes in Computer Science, Oslo, Norway, Springer (2008) 563–577
11. Ierusalimschy, R., de Figueiredo, L.H., Celes, W.: The Evolution of Lua. In: Proceedings of the Third ACM SIGPLAN Conference on History of Programming Languages (HOPL III), San Diego, CA, USA (2007) 2.1–2.26
12. Roduner, C., Langheinrich, M.: Publishing and Discovering Information and Services for Tagged Products. In: Proceedings of the 19th International Conference on Advanced Information Systems Engineering (CAiSE 2007). Volume 4495 of Lecture Notes in Computer Science, Trondheim, Norway, Springer (2007) 501–515