CEPIS UPGRADE is the European Journal for the Informatics Professional, published bimonthly at <http://cepis.org/upgrade>.

Publisher

CEPIS UPGRADE monographs are published jointly with Novática, that published them in Spanish (full version printed; summary, abstracts and some articles online first published by Novática and INFORMATIK/INFORMATIQUE, bimonthly journal of SVIFSI (Swiss Federation of Professional Informatics Societies, <http://www.svifsi.ch/>).

CEPIS UPGRADE is the anchor point for UOPENET (UPGRADE European NETwork), the network of CEPIS member societies’ publications, that currently includes the following ones:

- infoTumal, the magazine from the Serbian CEPIS society JISA
- Informatica, journal from the Slovenian CEPIS society SDI
- Informatik-Spektrum, journal published by Springer Verlag on behalf of the CEPIS societies Gi, Germany, and SI, Switzerland
- ITNOW, magazine published by Oxford University Press on behalf of the British CEPIS society BCS
- Mondo Digitale, digital journal from the Italian CEPIS society AICA
- Novática, journal from the Spanish CEPIS society ATI
- OCG Journal, journal from the Austrian CEPIS society OCG
- Pliroforiki, journal from the Greek CEPIS society CCS
- Tölvumal, journal from the Icelandic CEPIS society ISIP

Editorial Team
Chief Editor: Llorenç Pagés-Casas
Deputy Chief Editor: Rafael Fernández Calvo
Associate Editor: Pierna Fanning

Editorial Board
Prof. Vaclav Baletka, CEPIS President
Prof. Wolfthord Stucky, CEPIS Former President
Hans A. Frech, CEPIS Vice President
Prof. Nello Scarabottolo, CEPIS Honorary Treasurer
Fernando Perea Gómez and Llorenç Pagés-Casas, ATI (Spain)
Francois Louis Nicotet, SI, (Switzerland)
Roberto Carniel, ALSI – Tecnoteca (Italy)

UPOPENET Advisory Board
Dobrava Dolic (infoTumal, Serbia)
Matjaž Gams (Informatica, Slovenia)
Hermann Engesser (Informatik-Spektum, Germany and Switzerland)
Brian Runsum (ITNOW, United Kingdom)
Franco Hippluzzi (Mondo Digitale, Italy)
Llorenç Pagés-Casas (Novática, Spain)
Vedh Risak (OCG Journal, Austria)
Panos Mavroudis (Pliroforiki, Cyprus)
Thordurur Kari Odlason (Tölvumal, Iceland)
Rafael Fernández Calvo (Coordination)

English Language Editors: Mike Andersson, David Cash, Arthur Cook, Tracey Darch, Laura Davies, Nick Durr, Rodney Fennemore, Hilary Green, Roger Harris, Jim Holder, Pat Moody.

Cover page designed by Concha Arias-Pérez “Devourer of Fantasy” © ATI 2011.

Layout Design: Francois Louis Nicotet
Composition: Jorge Llàcer-Gil de Ramales

Editorial correspondence: Llorenç Pagés-Casas <pages@ati.es>
Advertising correspondence: <info@cepis.org>

Subscriptions
If you wish to subscribe to CEPIS UPGRADE please send an email to info@cepis.org with “Subscribe to UPGRADE” as the subject of the email or follow the link “Subscribe to UPGRADE” at <http://www.cepis.org/upgrade>.

Copyright
© Novática 2011 (for the monograph)
© CEPIS 2011 (for the sections Editorial, UPOPENET and CEPIS News)
All rights reserved under otherwise stated. Abstracting is permitted with credit to the source. For copying, reprint, or republication permission, contact the Editorial Team.

The opinions expressed by the authors are their exclusive responsibility.

ISSN 1684-5285

Monograph of next issue (April 2011)
“Software Engineering for e-Learning Projects”
(The full schedule of CEPIS UPGRADE is available at our website)

Monograph
Internet of Things
(published jointly with Novática*)
Guest Editors: Germán Montoro-Manrique, Pablo Haya-Coll, and Dirk Schnelle-Walka

2 Presentation. Internet of Things: From RFID Systems to Smart Applications — Pablo A. Haya-Coll, Germán Montoro-Manrique, and Dirk Schnelle-Walka

6 A Semantic Resource-Oriented Middleware for Pervasive Environments — Aitor Gómez-Goiri, Mikel Emaldi-Manrique, and Diego López-de-Ipiña

17 “Creepy IOn i.e.”, System Support for Ambient Intelligence (AmI) — Francisco J. Ballesteros-Cámara, Gorka Guardiola-Múzquiz, and Enrique Soriano-Salvador


34 Model Driven Development for the Internet of Things — Vicente Pelechano-Ferragud, Joan-Josep Fons-Cors, and Pau Giner-Blasco

45 Digital Object Memories in the Internet of Things — Michael Schneider, Alexander Kröner, Patrick Gebhard, and Boris Brandherr

52 Ubiquitous Explanations: Anytime, Anywhere End User Support — Fernando Lyaret and Dirk Schnelle-Walka

59 The Internet of Things: The Potential to Facilitate Health and Wellness — Paul J McCullagh and Juan Carlos Augusto

UPOPENET (UPGRADE European NETwork)

69 From Informatica (SDI, Slovenia)
Online Learning
A Reflection on Some Critical Aspects of Online Reading Comprehension — Antonella Chifari, Giuseppe Chiazzese, Luciano Seta, Gianluca Merlo, Simona Ottaviano, and Mario Allegra

75 From infoTumal (JISA, Serbia)
eGovernment
Successful Centralisation in Two Steps. Interview with Sasa Dulic and Predrag Stojanovic — Milenko Vasic

CEPIS NEWS

78 Selected CEPIS News — Fiona Fanning

* This monograph will be also published in Spanish (full version printed; summary, abstracts, and some articles online) by Novática, journal of the Spanish CEPIS society ATI (Asociación de Técnicos de Informática) at <http://www.ati.es/novatica/>.
Internet of Things: From RFID Systems to Smart Applications

Pablo A. Haya-Coll, Germán Montoro-Manrique, and Dirk Schnelle-Walka

1 Foreword

"The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it" [1].

This sentence, written by Mark Weiser in 1991, can describe the current idea of the Internet of Things (IoT). The technology has to disappear but, as if it was a contradiction, it has to do so by becoming an essential and indistinguishable part of the everyday objects that surround us. Thus, the Internet jumps from traditional devices to real objects, thanks to the use of technologies such as wireless sensing or radiofrequency identification (RFID). The number of users surfing the web will be overtaken by the number of objects that will communicate with each other. And they will do it with one main objective: to offer users advanced computational services in a subtle and efficient way. People will not need to be aware that they are assisted by technology and, as far as possible, will interact with these objects in the most natural way. That is how the Internet of things will make technology disappear, by disappearing from the eyes and the "feeling" of the people.

As we can see from the vision of Mark Weiser, although the idea of the Internet of Things is relatively new, it is supported by concepts and technologies that have been around for decades. The ability to integrate these technologies, the advances of the miniaturization required to incorporate them into everyday objects, and the capacity of communication of these objects with the network and with each other make it possible to talk now about the realization of those concepts. This vision has already become a reality. For instance, DHL offers the DHL Paketbox where customers can frank parcels on the Internet and take them to the Parcelbox [2]. The customers open a door, place the franked parcel inside and close the door. Here, the franking code is automatically read and used for further processing steps. Right after the customer closed the door, he or she will be able to monitor the delivery status on a special web site.

The term Internet of Things, IoT, was originally coined within the RFID development community around 2000. They referred to the possibility of querying information about a tagged object by browsing an Internet address or a database entry. Today, this approach has merged with the concepts of Ubiquitous Computing (UC) or Ambient Intelligence (AmI), centred on shifting the information and interaction from screens and traditional devices to the physical environment. In this new approach, everyday objects can be discovered, located and controlled via the Internet. This does not only refer to the smart devices that we carry or that surround us in a specific moment, but also to the things that we discover while we are on the move. It is not restricted to things that have any intelligence integrated, but can be extended to ordinary things of our daily life, such as food or clothing. Hence, the basic idea is an Internet of connected everyday objects, creating the ground for two major research fields:

- Machine-to-machine communication.

The Guest Editors

Pablo A. Haya-Coll is a full-time Lecturer at the School of Engineering of the Universidad Autónoma de Madrid (UAM), Spain, and has been a researcher and member founder of AmILab (Ambient Intelligence Laboratory at UAM) since 2000. He received his doctorate in Computer Science Engineering and Telecommunications from the same university in 2006. His research interests currently focus on Human Computer Interaction applied to Ubiquitous Computing. More details and publications can be found at <http://amilab.ii.uam.es> or <http://www.eps.uam.es/~phaya>. <pablo.haya@uam.es>

Germán Montoro-Manrique is an Associate Professor in the Dept. of Computer Engineering of the Universidad Autónoma de Madrid (UAM), Spain. He received his PhD in 2005 in the area of spoken dialogue systems for smart environments. Since then he has researched in the field of advanced user interfaces related to ambient intelligence environments. Further information and publications can be found at <http://amilab.ii.uam.es> and <http://www.eps.uam.es/~montoro>. <german.montoro@uam.es>

Dirk Schnelle-Walka received his PhD in 2007 in the area of voice user interface design at the Technical University of Darmstadt, Germany. He transferred the idea of design patterns to the design of voice user interfaces. His research focus is on voice-based interaction in pervasive environments. He currently leads the Talk&Touch group at the Telecooperation Lab at the Technical University of Darmstadt. He is also the head of several open source projects on speech technology, for example the open source voice browser JVoiceXML. More details and publications can be found at <http://www.ik.informatik.tu-darmstadt.de/de/people/dr-dirk-schnelle-walka/>. <dirk@ik.informatik.tu-darmstadt.de>
Machine-to-human and human-to-machine communication.

2 Characteristics of the IoT Systems: Overview

A good overview of the technologies needed to make the IoT concept a reality can be found in [3]. It specifies the kind of applications that can be obtained and the open issues that are still to be addressed. Below we will summarize some of their ideas as a review of these research fields.

2.1 Technologies and Interaction

These systems can be built thanks to the integration of identification, sensing and communication technologies, as well as the presence of a middleware layer. Interaction is also an important issue, and many systems have to deal with the development of new ways of interaction.

RFID systems are one of the key parts for the development of the IoT. In these environments objects can be identified by a unique ID corresponding to a RFID tag. Readers monitor these tagged objects to map the real world into a virtual world. In [4] authors show that building applications with RFID data in the IoT is challenging because tag-read events provide only low-level information and the metadata associated with tags, antennas, and events must be personalized and carefully controlled to create a safe, meaningful user experience. This implies some problems that must be overcome, like achieving a sufficient density of tags and users or finding techniques that improve or compensate for low tag-read rates. As for privacy, a major concern in the IoT, they conclude that context-aware access control seems to be a useful, easily understood abstraction for managing location privacy. Nevertheless in this case more evaluation is needed to determine whether it meets users’ needs when privacy concerns are magnified.

On the other hand, wireless sensor networks can help RFID technology to better track the status of objects by augmenting the awareness of the environment. They consist of distributed sensor nodes that monitor the environment and pass low level information (temperature, luminosity, pressure, etc.) to an interpreter which provides high level information about the state of the environment. For the successful development of the system it is necessary to choose the appropriate kind of sensors for the tasks that are to be measured. It is also important to employ some kind of technique for processing sequential sensor data, so that it can reduce noise and infer context beyond what the sensors actually measure. John Krumm proposes in [5] some well-accepted approaches:

- Mean and median filters: a simple technique that averages together multiple samples.
- The Kalman filter: a more sophisticated method that explicitly accounts for sensor noise and employs a dynamic model of the system to keep up with changes over time.
- The particle filter: a less restrictive method that does not require a liner model for the process in question.
- The hidden Markov model (HMM): which works only for discrete state variables but which can be useful for reducing the frequency of transitions between states.

The next necessary element is a middleware. By middleware we mean a software layer that connects the software and physical components hiding the unnecessary details for their communication. The middleware simplifies the development of new services and their integration with existing ones and helps to network sensors and augmented objects to talk to each other. The presence of a middleware is necessary since devices employ very diverse standards and ways of interaction. [6] describes some basic requirements for these specific middlewares:

- Communication: Synchronous and asynchronous (event-based) communication.
- Resilience: Replication, isolation and graceful handling of failures.
- Security: Secure communication among distributed services.
- Ease of use: Natural and intuitive usage for each target language.

One of the most common approaches is the use of a service-oriented middleware. For instance, the Hydra project is a major European effort to develop a middleware based on a Service-oriented Architecture for networked embedded systems. Its objective is to allow users to create AmI applications utilizing device and sensor networks [7].

As mentioned above, the IoT requires new ways of interaction, with concepts like tangible user interfaces (TUI) [8], where users interact with digital information through the physical environment or embedded interfaces [9], where the interfaces are integrated or co-located with the devices. These interfaces follow a new paradigm of design, far from the ideas of the classical user interfaces. In [5] Aaron Quigley defines ten key rules that should be followed by designers and developers of these new systems:

- Bliss: Learning to interact with a new kind of UI should not require people to learn another skill or complex command language.
- Distraction: Do not demand constant attention in the interface.
- Cognitive Flow: Systems that are everywhere must allow the user to retain total focus on the task at hand.
- Manuals: Do not require a user to read a manual to learn how to operate the interface.
- Transparency: Do not rely on users to hold application state in the mind to operate the interface.
- Modelessness: Avoid "modes" where the system responds differently from the same input stimulus dependent on some hidden state information.
- Fear of Interaction: Provide easy ways to undo actions.
- Notifications: Feedback to the user can be piggybacked and layered into interactions with their physical environment.
- Calming: Interfaces will rely on a wide array of human inputs and human senses.
- Defaults: Good interfaces judiciously exploit what the system knows or can deduce, reusing, for instance, user inputs.
2.2 Application Domains

Therefore, taking these considerations as a starting point we can develop a huge number of applications. One possible division would classify them into the following domains [3]: transportation and logistics, healthcare, smart environments, and personal and social.

The transportation and logistics domain includes real-time processing technology based on RFID and NFC to monitor the supply chain in a logistic environment; vehicles provided with sensors and actuators that can offer information for assisted driving; monitoring of the environmental parameters for the distribution and consumption process; or augmenting maps to provide richer contextual information.

The healthcare domain groups technologies that track and monitor objects, staff and patients in health environments; patient and staff identification; automatic data collection; or sensing to provide real-time information on patient health.

The smart environments domain includes homes and offices that become more efficient, comfortable and easy to interact by their inhabitants; industrial plants monitored by numerous RFID tags and sensors; or smart leisure environments such as museums that provide a personalized experience to the users.

The personal and social domain embraces applications that allow users to interact with other people. These can be applications to study and recognize trends in their activities; automatically update their social information; help users to locate lost objects; or warn them if they are stolen.

3 Open Issues

Finally, although many of these technologies, projects and systems are already developed, some open issues need to be addressed in the field of the IoT. They include:

- Standards for the integration of RFID or for the wireless and communication networks.
- Addressing and naming to provide mobility support and mapping between objects and references.
- Transport protocols and traffic characterization that can deal with the new traffic patterns derived from the IoT.
- Authentication and data integrity architectures adapted to the new ways of communication of the elements in the IoT.
- Privacy to ensure that only the appropriate information is shared between the different nodes of the system.
- Data integrity and privacy are two of the major concerns when we talk about the IoT. [10] presents an analysis of security and privacy challenges in the IoT. It divides the IoT into eight topics and describes the state of the art in the different aspects involved for each of them:
  - Communication: Mechanisms for securing communication are well-established but unfortunately seldom applied, especially in small devices with weak processing power.
  - Sensors: their major problem is privacy, users should be aware that they are being sensed, be able to choose whether they are being sensed and be able to opt-out and remain anonymous.
  - Storage: Mechanisms for integrity and confidentiality in storage are well-established, but unfortunately they are often complex to be employed, besides anonymization and pseudonymization mechanisms must be used to ensure that data does not contain information sensitive to privacy.
  - Devices: The integrity of devices is an unresolved issue since fully Trusted Platform Computing capable operating systems are still lacking.
  - Processing: Mechanisms for preserving privacy exist in data mining but are applied seldom.
  - Localization, Tracking and Identification: In the three cases the user should be able to opt-out and be notified of these processes.

4 IoT in this Special Issue

Following these ideas we present in this special issue seven papers that cover some of the most relevant areas presented in the IoT. We have divided them into three categories that try to illustrate the different aspects involved in the creation of an IoT environment: system support, creating applications for IoT, and IoT applications.

4.1 System Support for IoT

In this category we present two papers that describe different approaches to create the necessary base that will allow building applications and services on top of it.

A Semantic Resource Oriented Middleware for Pervasive Environments the authors describe a Triple Space middleware that allows communication between heterogeneous devices through different communication links. To do this, they bring the idea of tuplespace based distributed computing to ubiquitous systems. They describe the characteristics of the implemented middleware and show their preliminary test results. As a conclusion they prove that this approach is appropriate for the integration of devices and services in the IoT paradigm.

Creepy II-on, i.e., System Support for AmI also tries to provide a solution to the integration of different applications and technologies in the field of ambient intelligence environments. Their proposal consists of a new kernel for machines providing IoT services, a distributed file system protocol to allow communication between the different components and a new user interface management system to let users interact with the environment. They are responsible of moving the data fast between the different components, providing a common language to exchange information and access devices and bringing distribution, replication, persistence, and interaction heterogeneity by default.

4.2 Creating Applications for IoT

Here we include two new papers that describe methods and mechanisms to deploy IoT systems.

The Mundo Method - An Enhanced Bottom-Up Ap-
systems in a bottom-up way, in a different way to other bottom-up methods employed so far. To do this, the authors analyse the main problems and limitations of existing approaches and propose alternatives to facilitate the design and implementation of ubiquitous computing systems.

In "Model Driven Development for the Internet of Things", the authors show how the Model Driven Development can be employed in the IoT to specify physical mobile workflows of business process-supporting systems. They explain how to capture the requirements to link the virtual and physical information by means of modelling techniques that can be employed by the designers. They present a development strategy and describe a framework that defines an abstract architecture and code generators that produce, from a model based in that architecture, the implementation assets required for the technological infrastructure of a specific platform.

4.3 IoT Applications

The IoT applications category is formed by the papers "Digital Object Memories in the Internet of Things", "Ubiquitous Explanations" and "The Internet of Things: The Potential to Facilitate Health and Wellness".

In "Digital Object Memories in the Internet of Things" the smart objects feature a memory that can be used to store information about the object’s properties, state and usage history. Hence, objects become self-representative and enable new kinds of applications in the IoT. The digital memory is realized by a layered architecture that is described in this article.

"Ubiquitous Explanations: Anytime, Anywhere End User Support" introduces a middleware to increase the ability of users to interact with devices that they encounter while they are on the move. Therefore, a middleware has been developed to facilitate the exchange of knowledge between centralized knowledge bases and the devices.

"The Internet of Things: The Potential to Facilitate Health and Wellness" investigates how home telehealth monitoring and mobile classification systems for movement activities can benefit from the IoT. The goal is to help improve health and wellness with the help of autonomous 'things' (sensors, processing and communication devices, and displays).

The papers of this issue can cover only a part of the many faces of the IoT which are reflected by the different views on the selected categories. The creation of IoT applications combine several fields. Each of them is needed to make the vision of the disappearing computer come true.

Finally, we would like to express our gratitude to the authors for their valuable papers and to the Editorial Teams of GRADE and Novática for the opportunity of presenting a review of IoT concepts, applications and technologies. We hope readers enjoy them and consider them as interesting and instructive as it has been for us to help bring out this special issue.
A Semantic Resource-Oriented Middleware for Pervasive Environments

Aitor Gómez-Goiri, Mikel Emaldi-Manrique, and Diego López-de-Ipiña

Pervasive environments are highly dynamic with lots of heterogeneous devices which share information through increasingly interconnected networks. In this context semantic models can be used to describe the context that surrounds them in a very expressive manner, usually stored in centralized knowledge bases. The applications built on top of these knowledge bases use heterogeneous protocols to transmit their data, but do not capture the dynamism of the network. The presented middleware facilitates the exchange of knowledge between different sensors and actuators in a highly distributed, decoupled and resource-oriented manner following the Triple Space paradigm. This middleware has been tested on a stereotypical scenario, which illustrates how different peers can exchange data whilst keeping them autonomous and yet with a reasonable footprint for devices with reduced computational capabilities.

Keywords: Distributed, Embedded, Mobile, Semantics, Triple Space.

1 Introduction

In context-aware environments, a lot of devices communicate with each other and share changes of state in order to trigger actions within the environment. Different approaches to modelling and storing context data have been presented in several works [1], coming to the conclusion that ontology-based models are the most expressive models and fulfill most of the requirements of these environments. The blackboard model is one of the main context management models [2] which post messages into a shared media, usually centralized in a server.

Triple Space computing is a coordination paradigm based on tuplespace-based computing, which comes from the parallel computing language Linda [3]. In tuplespace computing the communication between processes is performed by reading and writing data structures in a shared space, instead of exchanging messages. The Semantic Web vision aims to offer machine-understandable persistent data forming a knowledge network for machines instead of the current World Wide Web which is more human-centered and require user intervention (web services offer remote functionality to machines, but they are not really Web-based since they are driven by message exchange). Triple Space (TS) computing performs a tuplespace based communication using RDF triples, in which the information unit has three dimensions: “subject predicate object”, to express this semantic data. TS offers reference autonomy (processes can communicate without knowing anything about each other), time autonomy (because of the asynchronous communication) and space autonomy (processes can be executed in very different computational environments), which cannot be achieved by message exchange-driven communication.

In this paper middleware for pervasive environments which uses the blackboard model through a Triple Space decentralized implementation is presented. This middleware is specially designed to run over devices with limited computational resources and embedded devices which may be part of the Internet of Things where common objects share their contextual information on the network.

The remainder of the paper is organized as follows. Section 2 discusses related work. Section 3 presents our middleware. Section 4 details an experimental environment. Section 5 examines the results of using the proposed solution in the experimental environment. Finally, Section 6 concludes and outlines the future work.

2 Related Work

Several approaches exist in the field of semantic tuplespace [4]. Conceptual Spaces, or cSpaces, were born to study the applicability of semantic tuplespaces to different scenarios including ubiquitous computing. Semantic Web Spaces propose some new primitives defining two different data coordination views: data view (with syntacti-
Pervasive environments are highly dynamic with lots of heterogeneous devices which share information through increasingly interconnected networks. Normally valid RDF and Linda primitives) and information view (with consistent and satisfiable data and new primitives). sTuples was conceived by Nokia Research Center as a pervasive computing work and provides description logics reasoning and a semantic extension of JavaSpace tuplespace middleware.

None of these projects was fully distributed and they were deployed over more or less a client-based architecture which did not implement the tuplespace paradigm itself in mobile peers and which restricted the reasoning process to few powerful devices.

In Triple Space Computing the tuples are expressed in the form of triples. Currently, two main pure Triple Space Computing middleware implementations exist: tsc++ and TripCom.

TripCom has different kernels hosted in servers which can distribute the semantic data through themselves, but once again, is too server centered. TripCom clients are not part of the space and they could hardly be, because of the complexity of this software which is oriented to run on powerful machines (it is designed to be able to run even different modules of the same kernel in different machines).

The first Triple Space project was called TSC. In TSC, triples can be interlinked to form graphs and semantic algorithms are implemented for template matching. It also offers a transactional context and a simple form to publish and subscribe to certain patterns. tsc++ [5] is a new version of the former TSC project [6] which basically offers the same API in a distributed way. To do that, tsc++ uses Jxta Peer To Peer framework to perform the coordination and Sesame [7] and Owlim [8] to store triples of each peer.

The nodes in tsc++ not only can query the space, but they can also store their own information, enabling the distribution of the space over all the peers by means of the strategy known as negative broadcast. In negative broadcasting the nodes of a group write the information locally and read querying to the rest of the nodes. This seems to adapt to ubiquitous system, where different devices share heterogeneous data entering and leaving the system, compromising data consistency and availability. In this aspect a sensor can provide information, but when it leaves the space, its information is automatically removed from there since it is no longer available to the rest of the nodes. However, as will be explained below in Section 3.4, negative broadcasting needs to be adapted to those cases in which a device wants to remotely change the state of actuators managed by another device.

Nevertheless, tsc++ lacks some of the advantages of other alternatives: it does not make inferences, it does not allow expressive querying and last but not least, it has not been designed for devices with reduced computing capabilities, because tsc++ middleware is focused on architecture and implementation in large scale and we focus on small scale aspects (local area networks with an intelligent environment).

Our work also aims to build an Internet of Things where everyday objects have connectivity and share their data through it. Thus, some of our work aspects have been presented in other solutions which belong to this area such as [9] or the Web of Things (WoT) paradigm [10]. The first one describes how the Jxta framework can facilitate the communication between objects with connectivity capabilities, and even if our solution uses Jxta as communication protocol, it is more centered on sharing knowledge rather than on the underlying layers. The second solution advocates the convenience of making objects part of the web to use existing web techniques to create applications and therefore they focus on embedding web servers on them. Despite the simplicity of this approach, there are certain aspects which are not considered as the discovery method of new devices (other works try to correct this limitation of the web approach as [11]), the instability of the nodes which serve webs in a network or the use of semantics.

As it has been seen, even if some works have analysed the convenience of the semantic tuplespace approach in Ubiquitous computing, to the best of our knowledge TS has never been specifically designed and implemented to use mobile and embedded devices as another peer of these spaces and not only as simple clients. This approach allows heterogeneous devices communicate with each other limiting the necessity of fixed infrastructure and previous configuration enabling more dynamic environments.

3 Infrastructure Description

Our main goal was making Triple Space middleware suitable for pervasive environments. To do this, it should enable the communication between devices of heterogeneous nature (mobiles, embedded devices, PDAs, Tablets, even PCs) through different communication links using standard

© Novática
Internet of Things

3.1 API

In order to interact with the Triple Space, an API is offered to the developer. Although it was originally inspired in the tsc++ API, it has been adapted taking into account the nature and restrictions of pervasive environments. It is composed by several primitives which are presented in Table 1.

The space management primitives allow the nodes to share knowledge with different groups of nodes. Query retrieves all the triples which match an specific template in a given space, while queryMultiple divides a SPARQL query into templates which are sent to all the nodes, the responses obtained from those nodes are merged and the query is made again over them potentially obtaining new results. Read retrieves just one complete graph which contains at least a triple which matches the template and take does the same but it removes that graph from the semantic repository of

<table>
<thead>
<tr>
<th>Space management</th>
<th>createSpace(space)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>joinSpace(space)</td>
</tr>
<tr>
<td></td>
<td>leaveSpace(space)</td>
</tr>
</tbody>
</table>

| Querying          | query(space,template): triples |
|-------------------| read(space,graph): triples    |
|                   | take(space,graph): triples    |
|                   | queryMultiple(space,spatql): triples |

| Writing           | write(space,triples): URI |
|-------------------| demand(space,template,timeout) |

| Subscriptions     | subscribe(space,template,listener): URI |
|-------------------| unsubscribe(space,URI)                  |
|                   | advertise(space,template): URI           |
|                   | unadvertise(space,URI)                   |

| Services          | register(space,service) |
|-------------------| unregister(space,service) |
|                   | invoke(space,invocation,listener): URI |

Table 1: Primitives of the designed Triple Space Implementation grouped by their Nature. space is the URI which identifies the knowledge of a group of nodes, graph is an URI which identifies a set of triples written into the space, triples is a set of triples and template expresses a sequence of adjacent triple patterns which specify WHERE-clauses of SPARQL queries. Service and invocation are interfaces which express the data needed to define a service and its invocation.

So, as one of the main concerns was allowing mobile and embedded devices to run that Triple Space middleware, effort was put into developing a Java-embedded distributed triple space implementation, namely tscME, which was still compatible with tsc++ [5]. It has also been developed a module to allow embedded devices such as SunSPOTs or XBee sensors which do not support IP stack be part of that space through a gateway. Finally, the former Triple Space API adopted from [5] was also enhanced to offer more expressive queries which are spread through all devices within the space, service management primitives and a new writing approach.

In the following sections the API and the different stages of the implementation of the middleware are described.

![Subscribe and Advertise Use Example](image-url)
the node which had it. Both read and take are overloaded to retrieve an specific graph identified by its URI.

Although write primitive was used to write triples in a local repository returning the URI which identified the new graph in which they had been stored, its implementation has been modified not to always follow the negative broadcasting strategy. Demand lets the developer announce to other nodes that it will be responsible for the graphs which match the given template for a period defined by the timeout parameter. The solution will be explained further in Section 3.6, below.

Subscribe primitive is used to made a node aware of the notifications made over the given template or any particularization of it. Advertise lets a node propagate a notification to all the nodes in an space to warn them about something. Those primitives can be used to build a notification system upon the space (see Figure 1).

Finally, a service API was conceived in top of other basic primitives as a first approach to solve the problems described in the Section 3.4 [12]. With register and unregister the notation of a service (inputs and outputs) is stored in a node and advertised to the rest of nodes. With invocation some inputs are written into the space and advertised to the service provider. The service provider can perform that invocation and, optionally, retrieve the output of the service. Most of the time, the output may imply changes in the knowledge base of the provider as a result of change in the context. This process will be explained in detail in Section 3.4, below.

### 3.2 1st Stage: tscME

The first step towards implementing our middleware was creating a library for a MIDlet which could be deployed on Java ME CLDC compliant JVM called tscME. This version provides space management, querying (except for queryMultiple), writing and subscription primitives in a native way. The communication has done using Jxme (Java ME’s Jxta framework) and the semantic data was managed using Microjena [13].

Due to the fact that Jxme does not use multicast, a special Jxta element called Rendezvous must be used to propagate mobiles’ messages. This element centralizes a little bit the solution and makes the mobile nodes dependent on them, but whenever multicasting is implemented on Jxme, this problem should be automatically avoided.

Since Jxta communication method was slightly different from the one used in tsc++ peers, a new communication layer was attached to those peers. To made subscriptions and advertising compliant with the method used by tsc++, a tsc++ node must act as a gateway for tscME nodes breaking the distribution principle in mobile nodes. Subscription

![Figure 2](image)

**Figure 2:** Services over tsc++: (a) registration, (b) invocation from the consumer point of view, and (c) invocation from the service provider point of view.
primitives are, however, not part of the Triple Space paradigms themselves.

Finally, the basic template has recently been improved, thus creating a new comparison template both in tscME and in the expanded tsc++ versions. This new template allows one to check whether a triple matches it by comparing its literal value.

Comparison Template Examples

The first example is equivalent to ?s ?p ?o . The second one checks whether a triple has myont:leength predicate and a numeric literal object not equal to 3. The last one checks if a triple has :aitor as the subject, :has_age as the predicate and a numeric literal object less than or equal to 30.

1. [{s ?p ?o . , ?o <= ?lit .}
2. [{s myont:leength ?o . , ?o != "3"^^xsd:int .}
3. [{:aitor :has_age ?o . , ?o <= "30"^^xsd:int .}

3.3 2nd Stage: tsc++ extension

As described above, to make tsc++ compatible with tscME a new communication layer was attached. Apart from that, a service API and queryMultiple was provided in these nodes. Service API is described in the following section.

QueryMultiple uses an SPARQL Construct query as input and splits it up into basic templates which are spread to the rest of nodes of the space. Then it filters all the received responses locally with the original query. Different strategies to propagate the queries have been proposed in [14] and [15], but due to the nature of negative broadcasting the nodes do not know anything about each other and therefore they cannot redirect those basic templates to an specific node.

Since the SPARQL query decomposition cannot be done in mobile peers, this primitive is not mandatory.

3.4 3rd Stage: Service API

The necessity of providing this service infrastructure in Triple Space or not could be argued since the knowledge can be directly obtained from the space or written into it, working with resources in a very RESTful way. Although in Section 3.6 a more resource-oriented approach is described, in [12] some primitives to use services inside Triple Space were proposed.

In pervasive environments, the sensed data can be obtained querying the space, but some limitations when modify the actuator were discovered:

- Security. Since tsc++ does not implement any kind of access control list, somebody might modify more knowledge than he or she wanted by mistake.
- Concurrency. If two different peers modify the same information at the same time, what information should be taken into account?
- Location of the information. Due to the nature of tsc++, when any information which initially belongs to peer a is modified by peer b, it is stored in peer b instead of being stored in peer a. If peer b leaves the space, some crucial information about the actuator will disappear. It seems logical that the information about a sensor or an actuator should be stored in the device which manages it (in the example, peer a).

Our service solution aims to provide this control to the device which controls the actuators being respectful of the asynchronous nature of TS. For that purpose, a really simple A Service invocation approach, which is very independent of the way the semantic services are defined (we use our own service definition language for the scenario, but other standard languages can be used), was designed. First, the service provider should register its service in the space (see Figure 2a). The consumer would discover it querying the space, and then it would create an invocation using the master-worker pattern and advertise it (see Figure 2b). An invocation is basically composed by an URI identifier and the input data the service may need.

The service provider, which is subscribed to its services invocation templates, will notice the event (see Figure 2c) and will retrieve the input data and perform the service (typically, performing a change in the environment using an actuator). When the invocation has been completed, the provider may write some output triples into the space and advertise the consumer in a similar fashion to the invocation.

3.5 4th Stage: Embedded Devices’ Gateway

In this stage, we adapted a gateway which enables the integration of SunSPOTs devices into TS. We interact with the SunSPOTs through a gateway designed to follow the WoT paradigm [10]. It provides access to them and theirs sensors and actuators through RESTful services (see Figure 3).

Besides the default representations provided by the server (XML, JSON and HTML), we have added another one which returns a set of triples with the semantic data corresponding to the resource identified by the user’s given
Internet of Things

Figure 3: Message exchanging between the Nodes beyond the Space, the WOT Gateway and a SunSPOT.

URL. This representation allows the seamless integration of SunSPOT devices in the TS by simply replacing the layer which interacts with the semantic repository of any node with the one which translates TS primitives into HTTP calls addressed to the gateway. The translated primitives are read, write and query.

Read has two different implementations. The first one returns a graph identified by an URL requesting it to the gateway. For example, to get the graph which describes the temperature of a SunSPOT, the read primitive would be mapped to an HTTP GET to the following URL: http://{host}:{port}/sunspots/{SpotName}/sensors/temperature/

The second implementation, returns the first graph found in the gateway which has a triple that matches a given template. To do that, it requests the following URL passing the template and the space as parameter: http://{host}:{port}/read?spaceURI={spaceURI}&template="{s ?p ?o}"

Internally, the gateway collects all currently available knowledge for SunSPOTs in a temporary repository and runs the template obtaining the desired result. The take primitive has been mapped to a read primitive since it does not make sense to remove a graph which is only generated on demand and therefore is not stored anywhere.

The write parses the contents of the triples passed to it extracting the values needed to make an HTTP POST request as follows: http://{host}:{port}/sunspots/{SpotName}/leds/led{0-6}?switch={true/false}&redColor={0-255}&blueColor={0-255}&greenColor={0-255}

The query gets all the triples that match a specific template in a given space. To do that, a GET request is addressed to a URL like http://{host}:{port}/query?spaceURI={spaceURI}&template="{s ?p ?o}" and the server executes the template over all available graphs.

3.6 5th Stage: Remote Writing

Although negative broadcasting perfectly suited to scenarios where nodes frequently enter and leave the local net-

Figure 4: A Node which joins a Space asks to the Rest of the Members about the Demands they have received.
work sharing their own data (i.e. data from its sensors or the profile of the owner of the mobile phone), this approach presents a problem when a node changes the context of an actuator managed by another node (quite common on the Internet of Things). Those problems have already been discussed in Section 3.4.

The first approach to solve this problem was the use of services over Triple Space. The inherent problem with it is the same that exists between WS*-like web services and RESTful web services [16]: excessive complexity. As it was described in [17], Triple Space paradigm is basically a resource-oriented paradigm, and since all the primitives use semantic resources (triples) as basis, it seems logical to find a solution consistent with the paradigm triples to be used as base. To keep the API simple for the developer, we have modified the implementation of the write primitive in order for him not having to care about the write operation to be used.

In essence, a demand primitive has been created to allow each node to reflect its responsibility for a piece of knowledge by using a template. Write primitive has been overwritten to send such knowledge to other nodes of the space when the responsibility for a set of triples is known to belong to another node. See Figure 4.

The demand primitive has a maximum lifetime after which the nodes will remove such claim from their registers to avoid the indefinite accumulation of them. The nodes which join a space will interrogate others to get an idea of the responsibilities they have claimed.

Finally, the write will have the following behavior.
If a node a tries to write a set of triples and another node is known to be responsible for the knowledge which is node a trying to write (somebody has performed a demand with a template that at least one triple matches), it will be suggested to the other nodes that the node a wants to modify that knowledge with that set of triples. This is done using an operation transparent to the users of the middleware called suggest (see Figure 5).

Each node which has performed a demand that matches the given triples, will call a callback method passing the original set of triples to it. In this method, ideally the environment should be changed using an actuator and,
4 Experimental Environment

There are many home automation or urban instrumentation scenarios where the proposed middleware could be used. One of these stereotypical cases could be the control of the room temperature. In this scenario, which uses all the primitives described previously, there are at least five peers, which are shown in Figures 7 and 8.

There are two different context providers: SunSPOTs and weather provider. The first types are physical devices which share their sensed data through the node described in Section 3.5. The weather provider, on the other hand, is a virtual context provider which gets information from the Internet and writes it as semantic information into the space. To do that, the peer polls for new environments in the space, and checks if Yahoo! has a place id for this environment name. If so, it queries the Yahoo! Weather service and it gets the current temperature in this location. The space has also an actuator device: an air conditioning system which will decrease the temperature. It can be turned on or off to try to regulate indoor temperature. Current temperature can be also monitored with a mobile phone.

The regulator node uses all the sensed information whenever someone is in a place (inferred when there is a device who belongs to somebody in the space) to decide when to cool it. To regulate the temperature, this peer checks whether there is an air conditioning system in the location it wants to regulate and turns it on.

### Basic Indoor Temperature Control Algorithm

```plaintext
while( indooort>26 ) // unbearable temperature
if( outdoort<26 ) // user should open windows
else // turn on the air conditioning
```

In order to implement this scenario two different approaches can be used depending on what method they use to perform actions over the environment. The first approach (Figure 7) uses the described service primitives to create a service which turns on or off the connected air conditioning. Then the regulator node invokes it whenever it is necessary.

The second approach (Figure 8), the recommended one, relies in the demand primitive explained above. The SunSPOT node demands a template related with the LEDs of the controlled sunspots and the air conditioning node does the same with a template related with itself. When the regulator node decides that the indoor temperature should be decreased, it tries to write that knowledge on the space but since both SunSPOT and air conditioning nodes have claimed to be responsible of that knowledge the primitive is transformed on a `suggest` primitive. Then, SunSPOT and air conditioning nodes decide what to do with these information. In this example, SunSPOT node turns on or off all the LEDs of all the SunSPOTS connected to them using the gateway, so their status is automatically updated and the air conditioning node turns it on or off and updates its status.

This scenario is only a proof of concept. Technically, a SWRL (Semantic Web Rule Language) could be introduced to define the temperature control rules in a more expressive and decoupled way.

5 Experimentation

In this section, the new `demand` primitive in conjunction with the `write` primitive and the SunSPOT gateway will be evaluated. The rest of the primitives have already been assessed in [12] for tscME and in [5] for tsc++.

<table>
<thead>
<tr>
<th>Number of registered demands</th>
<th>10</th>
<th>20</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>Former version</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>write</td>
<td>0.004</td>
<td>0.004</td>
<td>0.004</td>
</tr>
<tr>
<td>Current version</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>demand</td>
<td>0.096</td>
<td>0.099</td>
<td>0.098</td>
</tr>
<tr>
<td>write</td>
<td>0.006</td>
<td>0.015</td>
<td>0.027</td>
</tr>
<tr>
<td>suggest</td>
<td>0.133</td>
<td>0.159</td>
<td>0.186</td>
</tr>
</tbody>
</table>

**Table 2:** Demand Primitive Performance in Mobile Node (in seconds).

<table>
<thead>
<tr>
<th>Number of graphs</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Spots</td>
<td></td>
</tr>
<tr>
<td>SunSPOTs</td>
<td></td>
</tr>
<tr>
<td>write</td>
<td>0.037</td>
</tr>
<tr>
<td>read(URL)</td>
<td>0.020</td>
</tr>
<tr>
<td>read(template)</td>
<td>0.198</td>
</tr>
<tr>
<td>take</td>
<td>0.164</td>
</tr>
<tr>
<td>query</td>
<td>0.170</td>
</tr>
</tbody>
</table>

**Table 3:** WOT Gateway Performance (in seconds).
Finally, the stereotypical scenario presented in the previous section has been implemented using the demand primitive and will be compared with the service-based solution assessed in [12].

5.1 Demand Primitive
In Table 2 we can appreciate how the demand primitive behaves as a networking primitive and therefore it takes the time which is needed to propagate over the network until the other node is reached, and it may vary depending on the network status. The write is slightly penalized by the demands checking process, when the node decides whether it should write on the repository or propagate a suggest primitive, but anyway it is nearly insignificant (even for 30 demands, it just takes 20ms more comparing with the measures obtained in the previous version of this primitive). When the write is propagated using suggest, the time needed is also similar to the propagation time.

According to those measures, we can assume that the new implementation of the write is fast enough to be run instead of the former implementation, with “always local writing” behavior.

5.2 SunSPOT Gateway
As can be seen in Table 3, the number of connected SunSPOTs does not affect write and read(URL) primitives performance, because the gateway accesses directly to the request URL provided by them. With the read(template), the take and the query, however, the delay increases whenever more SunSPOTs are available since the gateway must collect the graphs from all the resources of each SunSPOT to filter the results using the given template.

5.3 Scenario
In order to deploy the scenario in the two ways described in the previous section, all the nodes in this test have been configured to wait for up to one second for responses (due to its asynchronous nature, we cannot predict when all the responses will arrive). In any case, the performance of this scenario could be improved by decreasing the waiting time.

Table 4 shows the time required for activating the air conditioning in each implementation. With demand based implementation, the invocation is done faster. Moreover, this difference will become more noticeable if more air conditioning machines are activated. While the service-based invocation will take as much time as needed to perform each invoke primitive, write will still need a single call taking the same time.

Finally, from a qualitative point of view, it has been experienced that the demand-based approach is a much easier way to develop the scenario than the service-based approach.

6 Conclusions and Future Work
This paper explores the possibility of bringing semantic tuplespace-based distributed computing to ubiquitous systems, where many heterogeneous devices share knowledge asynchronously and in a resource-oriented manner, which fits perfectly with the idea of the Internet of Things.

The results obtained in our stereotypical scenario have proved that the middleware has an acceptable performance. However a more exhaustive evaluation of the middleware should be done to check scalability issues and to explore how the intensive use of it would affect mobile and embedded nodes (battery consumption, usefulness of the middleware...).

In addition, some implementation problems found on mobile and embedded nodes can be corrected by applying effort in improving the P2P framework (using multicast) and creating a "reasoning" feature for mobile and embedded devices. Otherwise, the usefulness of the proposed middleware and resulting applications will be limited.

For our future work, we are planning to create new gateways for other embedded devices using the same approach as for SunSPOTs, developing a mobile "reasoning" feature, considering new alternatives for the network layer and exploring security issues in the middleware. Finally, a performance analysis both in a simulator and in a heavily instrumented deployment scenario should be taken into account.

Acknowledgments
This project has been financed under grant PC2008-28 A by the Department of Education, Universities and Research of the Basque Government, Spain, for the period 2008-10.

References
the Department of Education, Universities and Research of the Basque Government, Spain, for the period 2008-10.

References


"Creepy Πon i.e.", System Support for Ambient Intelligence (AmI)

Francisco J. Ballesteros-Cámara, Gorka Guardiola-Múzquiz, and Enrique Soriano-Salvador

AmI, Ambient Intelligence, environments are a challenge for building applications. These environments are data-rich, highly heterogeneous environments requiring fast systems on small devices and conventional machines, working together to build complex applications. Moreover, these applications need to be accessible from a wide range of I/O devices present in the environment, from high latency mobile networks. To build systems for AmI that work well in practice we need support them on three fronts: (i) a kernel which leverages modern hardware capabilities to move data fast, otherwise AmI environments will be slow and, therefore, dumb; (ii) a common language to exchange data and access devices, tolerating high latency links, available everywhere, otherwise intelligence will be confined to isolated components; and (iii) UIMS, User Interface Management System, capable of providing distribution, replication, persistence, and interaction heterogeneity by default, otherwise the user will need to make use of a PC to interact with the ambient intelligence. We are currently writing a new kernel, Πon ("PI-on"), leveraging the features of modern machines to provide fast services fast. Πon is designed to simplify building synthetic file systems as interfaces for AmI services, perhaps running on tiny devices. The common language of Πon is a new file system protocol, "Creepy", designed to work well despite high latency links, for both data and device access. User Interfaces, UI, for final users are to be provided by the UIMS we call "i.e.", which builds on the infrastructure provided by Πon and uses the Creepy protocol to distribute UI elements among highly heterogeneous I/O devices. This paper describes the requirements and design for these systems and, briefly, the status of their implementations.

Keywords: AmI, System Support, Creepy Πon i.e., File System Protocol, Kernel, Shared Address Space, UIMS, Zero-copy.

1 Introduction

An AmI, Ambient Intelligence, environment needs to sense the user needs and respond to them in real time. In

Authors

Francisco J. Ballesteros-Cámara received his MS in Computer Science on 1993 and his PhD in the same matter on 1998, at the Universidad Politécnica de Madrid, Spain. While an undergraduate, he received several grants from European research projects where he developed run-time software for programming languages. Later, he worked for a number of years in telecommunications companies producing systems software (he is the co-author of LiS, a STREAMS framework for Linux.) Since 1995 he has been a Lecturer at several Spanish universities where he has been teaching and developing Operating Systems. He developed the Off++ kernel, for the 2K Operating System jointly with the SRG at University of Illinois at Urbana Champaign. 2K evolved later into the Gaia OS. Since then, he has been developing Plan 9 from Bell Labs software. Recently, he developed Plan B and Octopus, two different ubiquitous systems derived from Plan 9 from Bell Labs that he uses to perform daily computing tasks. He is the head of the Systems Lab, <http://lsub.org/>, where Pon is being developed. Additional info can be found at <http://lsub.org/who/nemo/>. <nemo@lsub.org>

Gorka Guardiola-Múzquiz obtained his MSc in Telecommunication Engineering from the Universidad Carlos III de Madrid, Spain, in 2003, and his PhD from the Universidad Rey Juan Carlos, Spain, in 2007. He is currently working as a full time lecturer at the Universidad Rey Juan Carlos de Madrid. He has been an intern at Bell Labs NJ, Bell Labs Antwerp and Austin Research Lab (IBM), where he has worked on Plan 9 related projects, including security for the femto base station, an implementation of IKE and layered block devices. His research areas are operating systems and ubiquitous systems. He was part of the team developing Plan B and Octopus operating systems. He has been working in R&D on both the Plan 9 from Bell Labs, Inferno and the Octopus Operating Systems. He is a member of the Operating Systems Lab, <http://lsub.org/>, where Pon is currently being developed. <paurea@lsub.org>

Enrique Soriano-Salvador obtained his MSc in Computer Science from the Universidad Rey Juan Carlos de Madrid, Spain, in 2002. He had been awarded a four-year F.P.I. full time grant from the Spanish Ministry of Science and Technology. and his PhD from the Universidad Rey Juan Carlos in 2006. He is currently working as a full time lecturer at the same university. As a postgraduate, he worked as software developer for the said university and as a UNIX trainer before becoming a member of the Systems Lab. His research interests include security, operating systems and pervasive computing. He was part of the team developing Plan B and Octopus operating systems. He has been working in R&D on both the Plan 9 from Bell Labs, Inferno and the Octopus Operating Systems. He is a member of the Operating Systems Lab, <http://lsub.org/>, where Pon is currently being developed. <esoriano@lsub.org>
order to do so, it needs to process data from sensors, some of which are bandwidth hungry, such as video input. Then, it needs to integrate them with models and heuristics for the user behaviour. Finally, it needs to respond by controlling actuators and user interfaces. All this must be done within a real time feedback loop while interacting with the user. Latency is a critical parameter for the whole system to work.

As a result we need servers, protocols, and user interfaces customized to work well in this data-rich environment. Most work done in the past to this effect has been fragmentary, with special customized hardware, applications, and services, but not thinking of the system as whole. More than just focusing on some special services or applications, we need to focus on how to integrate everything into a whole system that the user can perceive as the "Environment".

In the past, we have built different systems and protocols for pervasive computing, including Plan B [1][2][3], Octopus [4], Omero [5], etc. We are now using what we learned from building them to provide appropriate systems services to bring intelligence to the environment.

In particular, we are focusing on three specific components that we consider critical in this respect:

1. A new system kernel for machines providing AmI services.
2. A new file system protocol to let them speak to each other.
3. A new UIMS technology to let users interact with the environment.

Current general purpose kernels are not optimized for serving data fast, especially since recent advances in hardware have rendered many of their assumptions obsolete. Pon, or "PI-on", our new kernel, considers that the common case is either a tiny low-end machine (e.g. a small ARM machine for embedded sensors) or a powerful multi-core computer with copious resources.

The abstractions it provides take both cases into account, aiming at being able to pipe data fast, with low latency, from/to devices embedded in the environment.

Pon uses synthetic file systems as its main abstraction, as our previous systems did [3][4]. Synthetic file systems have proved to be a very general abstraction on which to build distributed systems [6][7]. They address many of the problems of distribution, including protection, naming, access transparency, heterogeneity, and concurrency control.

However, existing protocols for distributing file systems can be slow, especially in the presence of high latency networks like mobile networks (where latency is due to retransmission and sophisticated channel coding). Moreover, most of them are not suitable for accessing devices represented by synthetic files (e.g. devices under /dev in Linux).

Creepy is a new file system protocol specially designed to export synthetic files through the network on high latency environments.

Previously, we developed several file-based UIMS (Omero [5] and O/Live [8]) that transparently permit the distribution and replication of UI on heterogeneous I/O devices. We can take advantage of the new kernel and protocol to improve upon them, and the result is a new UIMS called i.e., based on synthetic files and persistent event channels.

i.e. is well adapted to AmI, because it decouples the application from its user interface, both from the bandwidth point of view and from the adaptation point of view. Interactions are confined to the user interface whenever possible, and only very high abstraction messages cross the network.

The resulting environment is depicted in Figure 1. In what follows, we describe the requirements, design and, briefly, the implementation status of each of these components.

2 Pon

Most of the assumptions behind current general purpose OS architecture have changed:

- The virtual space now is much larger than it used to be. Even though pages or more specifically TLB entries may be a scarce resource, the address space itself is not.
- Concurrent programming is a must, in and out of the kernel. Cores are not getting faster, in fact, sometimes the opposite; they are just growing in number per machine. Concurrency requires automatic garbage collection, otherwise making correct programs is difficult.
- Because of the way the multicore architecture is built,
it is very easy for one process to poison the cache of another. Particularly the L1/L2 which are usually local to the processor. Spin locks and, in general, memory locking operations where processors alternate locking the access to a memory area, are a bad idea in this environment without special provisions. Playing ping-pong with the L1/L2 cache is specially bad for performance. Processors tend to grind to a halt on memory coherence protocols and the software runs slow as a consequence. Which cache limits us most, the data or the text, is still something we need to measure carefully.

- In most machines, RAM memory is also huge, spanning tens of GB, but caches have not grown as fast as RAM and are now scarce resources.
- On the other hand, there are small machines (e.g. ARM machines) designed for embedded systems. They are not different from a single core in desktop computers.

Considering these points, it is desirable for a kernel to include the features described next. We are building a kernel, Pon, which does so.

### 2.1 Pon Design

Considering that most I/O devices are capable of performing gather/scatter, a zero-copy framework can move data by copying pointers and not data itself.

In order to exploit concurrency on multicore machines, applications need garbage collection. A buffer framework supporting zero-copy is an opportunity to provide it, relieving the user from having to manually manage memory or having runtimes to do so. Pon is designed to include an immutable buffering scheme similar to the one in the io-lite [9], but intended for fast communication in modern multicore machines and providing automatic garbage collection. Following [10] we plan to make our protection coarser than it currently is, sharing a big part of the address space between processes. This will enable us to use this channels to pass references around.

Because machines are ccNUMA these days (each core has its own cache attached and communicates with different cores by using a hypertransport bus), it is important to rely on message passing as the primary communication and synchronization mechanism. Pon includes message passing and channels to communicate data between processes and also between processes and the kernel. ** These channels are typed, and protection for the referenced data changes when a message is sent through them. **

Considering the previous point, scheduling should be closely tied to communication. For example, the processor might be handed directly from sender to receiver without going through the kernel. The aim is to keep the data cache warm. At the same time, the concurrency model can build on this.

For latency, channels can be buffered in order to have the receiver (kernel or process) asynchronously running in another core, batching operations to reduce the number of context switches in a similar manner to FlexSC [11].

When scheduling, there is always a compromise between CPU usage and cache usage. If a process is tied to a processor, it is likely that the entries in the caches can be reused by using some form of tagging with process identifiers. However, if all the processes are tied to a subset of available processors, we may be wasting resources. The trade-off becomes harder if we consider other effects not described here.

Pon uses a Warp Scheduling Drive inspired by Apertos computational field [12]. A curved space of costs can be set for scheduling, where highly coupled processes find themselves inside a potential well, and are unable to move to another processor; whereas non interactive processes are free to move, although they may find a potential barrier to using processors depending on the cache state. The idea is that the system automatically computes the forces in the field by self measuring, and moves processes accordingly. Not only caches but other shared resources in the processor can be taken in account as well. For example, hyperthread (HT) cores share more resources with each other than real cores. Processes running in the same HT core can repel each other so that one of them migrates to a proper core.

In modern computing, shared libraries [13] have a large cost in execution time because of the overload added by dynamic linking, and the size of the linker and symbols themselves. More importantly, they can have a great cost in manageability [14][15]. However, if many instances of the same library are running on different processes (with different binaries), RAM and, more importantly, cache misses, can be saved. Pon gets the same benefits by using a new approach: shared, but statically linked, libraries (SSLL).

In SSLL, we have static linking, but the library is shared between the different binaries. Each binary is statically linked against an immutable version of the libraries required. Each library is assigned a global, unique portion of the virtual address space. Therefore, different libraries occupy different portions of the global virtual space, no matter the

"Creepy is a new file system protocol specially designed to export synthetic files through the network on high latency environments"
In modern computing, shared libraries have a large cost in execution time because of the overload added by dynamic linking. Binaries and libraries are to be paged on demand. The abundance of virtual addresses is used to distinguish between libraries.

3 "Creepy", a New Distributed File System Protocol

The new distributed file system protocol called Creepy is intended to become the ION file protocol. Creepy supports a distributed file system for the Internet, capable of working well to access devices both on the local and wide area; therefore with caching in mind. This is desirable for bringing together devices and services as required for building AmI environments.

We need a file system protocol supporting access to distributed files and devices. For this, 9P [16] or Styx [17] suffice. However, we have extra requirements that make 9P not so well suited for our purposes. This is the full list of requirements:

- The protocol must support access to synthesized files and devices, as we want it to reach sensors, actuators, and software devices in general. 9P and Styx meet this requirement, but not the following ones.
- The protocol must work well across high-latency and wide-area, network links. Otherwise, we would not be able to use it on mobile networks, from overseas or from our (poorly connected) homes.
- The protocol must permit client disconnections, without discarding the client state. Otherwise, suspending clients would be impractical or require additional software.

3.1 "Creepy" Design

To work well with devices, the protocol must support file descriptors as found on clients. Just having request and responses send names or resolving them on demand is not enough. There has to be a state in the server per open file to be able to keep the state of the client, e.g. the offset. This state is indexed by a file descriptor. The equivalent concept in the protocol is the fid, as used in 9P [16]. Like in 9P, a fid is a small number used as a file identifier. Unlike in 9P, the server assigns fid numbers on behalf of the client. Clients must maintain their own data structures to keep track of fids, and they do not usually look them up by fid number, which means that it is better to let the server assign them; the server can use the fid number as an index and save a hash table.

Support for disconnected operation requires the server to be able to keep track of fids in use by a client that is disconnected. This is achieved by using "sessions", a container for the fids of the client. A session is established prior to the Creepy dialogue for file access. In some cases creating a new session; in other cases associating to a session from a previous connection. Sessions may be collected by the server (and their state released) after a disconnection, after some time past the disconnection, or after a server reboot. When to do it is suggested by the client. The server decides.

To reduce latency effects, and utilize effectively each round trip, protocol requests exchanged between clients and servers are batched. Each batch behaves as an RPC but, instead of using a single request and a single reply, a batch consists on a series of individual operations, each one with its request (and response).

To clarify, a batch, or RPC, is a series of operations, but does not need to be a single message (i.e., a single "write" on the channel). Each operation is a separate request but it is part of a logical RPC, or batch.

A client sends a series of requests (one per operation) for a single RPC, and then waits for a series of response messages from the server. For each operation there is a request message (or "transaction" message, or T-message) and a corresponding reply message (or R-message). This is similar to 9P.

Requests to agree on a protocol version, to agree on a session to use, and to flush outstanding requests, are RPCs on their own. But all other requests must be made within a batch of requests.

A batch (or RPC) has a begin request, followed by one or more requests, followed by a final end request. This has important implications. A server is free to read from the network all requests in a batch, without starting any of them before reading the end of the batch. Because all such requests are considered a single RPC, with several operations in it, an error in one operation means that the rest of the batch is ignored by the server; the final error response effectively terminates the RPC when seen by the client.

Batching is needed to achieve a single round trip time in those cases where the client, or an intermediate cache, wants to perform a series of operations and does not need the result of any of them before issuing the next one.

A batch is not packaged as a single message, in order to permit multiplexing of the communications channel. Should a batch be a single message, no requests from other clients

For enabling AmI environments, it is paramount to provide applications with multimodal, replicated, distributed user interfaces (UI).
may be interleaved with it, and a long batch would effectively block other clients sharing the channel. Packaging each different operation as a different message is also beneficial for the implementation, which can pack, unpack, and handle all messages in the same way. To help intermediate caches to decide what to batch when they need to ask the server for data, each batch speaks about a single fid. An intermediate cache may read a series of requests (from a single batch) and, when all of them are known, decide if it makes sense to forward all of them at once to the server. It might decide instead to serve a prefix of the batch from local data, forward the rest to the server, and then reply to the client. Other approaches, mixing requests in a more general way, make it not clear for the cache how long to wait before asking the server: asking too soon leads to multiple roundtrips, asking too late leads to extra delays in the service.

A client (or an intermediate cache) must know something about the semantics of each file, to know how to cache it, if at all. There are several types of file to consider:

- Conventional files. No extra requirements on them.
- Device files. They should not be cached at all. All requests on them are to be forwarded to the final server, or the semantics will change and the device interface will break.
- Append only files. These are only appended, which may be exploited by clients and caches regarding what to cache.
- Timely files. These correspond to audio and video media, where it is usually more important to deliver data timely than it is to retransmit every piece of data.

- Read-only files. Immutable files have nice properties for caches and clients. Those files that are meant to be read only for their whole life, belong to this category. For example, system binaries. Previous versions of mutable files may be considered also immutable.
- Synchronous files. For some files, it is important for clients to see all writes made before (i.e., UNIX semantics). In all other cases, it may suffice to see all writes made prior to the last close of the file (i.e., session semantics). The later is better for caching and works fine in most cases; thus, it is a desirable default. However, the former is sometimes necessary and, therefore, some files may have to be flagged as "synchronous".

Being able to recognize different types of file, as enumerated, is essential for AmI environments, where some files may correspond to device interfaces, others might represent logs of events (perhaps append-only files), yet others might be conventional files.

4 "i.e., a New User Interface Management System"

Having a fast kernel designed for providing AmI services, and a Creepy protocol to bring components together, does not suffice.

For enabling AmI environments, it is paramount to provide applications with multimodal, replicated, distributed user interfaces (UI). Regarding the UIMS and the window system, this implies the following:

- Applications must be able to create interfaces independent of the technology used for I/O. In many cases, graphics and voice may be equivalent (e.g., a menu can be shown using graphics or may be presented by reading options). The application should remain unaware of these details.
Applications should not be concerned with how many copies of their UIs are deployed on the environment. It should be feasible to make a duplicate of (part of) a UI to access it from more than one place at once.

- Interfaces should not be blocks. They are made of parts (widgets, or UI elements). There is no reason why UI elements cannot be handled independently. They might be rearranged by the user, or perhaps replicated without replicating the rest of the interface.

- How interfaces are rendered, or presented to the user, should be independent of the structure of the UI as created by the application. Depending on the device, a set of interfaces might be presented one-at-a-time, or perhaps using tabs, or windows, or by a 3D representation, or by any other means a viewer (or presenter) for the interface deems reasonable.

I/O interactions should be confined to the viewer or presenter at hand. Otherwise, editing or interacting with the interface would require a permanent connection to the application or to the UIMS, which is not always reasonable.

Operations on an interface should not require much regarding throughput and latency of the communication link used to reach the UIMS and the application. Inserting a line on a text could require sending a few bytes, with the new text, but it is not reasonable to require all parties involved to reside in a well-connected network to be able to achieve this.

UI elements should be able to move around, as requested by the user, perhaps crossing machine boundaries. Otherwise we wouldn’t be able to move UI elements to other displays or I/O devices.

The state for the session, as seen by the user from one location (and perhaps after moving to another and recalling it), should persist. There is no need for users to recreate their preferred session once and again each time they "connect" to the system.

Concurrent interaction with multiple copies of an UI element must be feasible, perhaps overseas. Otherwise, replication of UI elements would not be actually perceived as a replication. If a button or text is shown in three different places, it must be feasible to edit it from all three places, at the same time. It is likely that some of them would be co-located, and that the user chooses one of them at one time, and a different one next, at will.

These requirements are feasible today, given the technology available. Current window systems, including research ones, fail in one or more points. The two previous UI services we implemented for pervasive computing, Omero [5] and O/Live [8], achieve some of them, but not all. A descendant of these systems, called i.e., is being built in the hope to meet all of them.
The work described here builds on the previous operating systems we have built for ubiquitous computing and AmI: Plan B and Octopus.

4.1 "i.e." Design

i.e. is both a UIMS and a window system. It is designed by dividing the tasks in two different parts:
- **Id:** This part is the UIMS. It is responsible for keeping the state of the UIs created. At one time, there will be many copies of **Id**. Perhaps running at different machines.
- **Est:** This part is the UI viewer. It is responsible for taking different UIs (or parts of them) from one or more **Id** instances, and presenting them. All user interaction is confined to **Est**.

It should be understood that at one point in time, multiple copies of **Id** and multiples copies of **Est** would be running in different machines.

In essence, the task performed by **Id** is to receive an abstract stream of updates to UI Elements, and to multicast it to **Est** instances interested on the elements (see Figure 2). At any point in time, **Est** may have performed several updates to UI Elements shown. Some of them have been received by **Id** and echoed back to **Est**, some of them (following the formers) not. The latter are still speculative changes, and, due to concurrent updates performed by several **Est** or by the application, might be rejected if they are not echoed back in the order expected (for example, if an external insertion on text is seen by **Est** when it was expecting the echo for one of its own edit operations).

In Figure 3, both **Est** components are presenting a particular element. The one above has been updating it (due to user interaction) and has performed updates from **a** to **z**. Only updates from **a** to **k** have been echoed back from **Id** (from the one supporting the element considered). Those are seen by everyone, including the application. Updates from **I** to **z** are still speculative and presented only from the **Id** above. As shown in the figure, the application is not different from **Est** regarding delivering of updates.

4.2 Interfaces

**Id** is a synthetic file server. Its main interface to other programs, including **Est**, applications, and external programs, is a synthesized file tree. UI elements are represented by synthetic files. This is the same approach followed by the predecessors Omero, and O/live [5][8].

Programs operate on interface elements kept by **Id** by operating on the files it implements. Figure 4 shows an example file tree served by **Id**. It is important to notice that these are not actual files. Instead, they are just the interface for the abstractions implemented by **Id**.

If Interface Elements (IE, in what follows) files are accessed by using a network file system protocol designed to avoid latency problems, like Creepy, i.e. can perform well even through high latency WANs, as its predecessor O/live [8] does.

4.3 Input Devices and Clipboards

Input devices and clipboards are independent of **Id** and **Est**, but cannot be left apart.

As shown in Figure 5, there may be different input devices involved. The same happens to output devices, but deploying one **Est** instance per output device suffices to coordinate them through the **Id** instances involved.

Input devices are connected to **Est**, because they are usually close to the terminal used for I/O. Unlike on other systems, we consider I/O devices as stand-alone devices. Each device is able to send its output as input to one **Est** or to another. At any point in time, the device may change its connection and address a different **Est**.

For the sake of convenience, I/O devices may be grouped so that redirecting one to one **Est** also redirects the others in the group, in the same way. Groups are named so that different **Est** implementation might choose, for example, to show different pointers for different pointing devices attached, but only one per I/O group.

At a given point in time, there may be multiple input devices attached to a given **Est**. Suppose that in the scenario depicted in figure 5, the machine running the middle **Est** seems to be suffering some activity. The pointer device from the top-most machine is directed towards the middle **Est**. However, the keyboard device close to it is left alone for use on the top-most **Est**. Devices from the bottom-most machine are all directed towards the middle **Est**.

Clipboards are handled in the same way, like any other input device. They differ just in that they are bi-directional. They can send input to **Est** as well as they can accept output from it. For example, cutting some text with a mouse or multitouch associated to a given editing device would copy the text into all clipboards associated to such edit device.

5 Related Work

The work described here builds on the previous operating systems we have built for ubiquitous computing and AmI: Plan B [3] and Octopus [4].

Plan B [3] is a peer to peer operating system which can adapt to changes on the environment by mounting dynami-
ally resources as they appear on the network. Plan B is a descendant of Plan 9 [6] and, in the same manner, uses synthetic files to model resources. The Plan B UI, Omero [5], is an ancestor of i.e., and therefore related. However, Plan B used a kernel built using abstractions that were designed for hardware as it was in the 80s (as most other popular OSes do). Pon uses ideas from different existing kernels, like immutable buffers [9], system call batching and the Apertos computation field [12] but adapting them to present hardware.

Octopus [4] is a descendant of [7] and can run hosted on different operating systems. The Octopus protocol, Op [18], an evolution of 9P [16] and Styx [17], designed to work on high latency connections. The problem is that Op loses the connection whenever the network has a transient failure. Creepy is a descendant of Op, designed to fix this and other problems. O/Live [8], the UIMS for Octopus is also an ancestor of i.e. and O/Live inherits from Octopus the problem with transient network failures, and therefore we had to design i.e. to address this and other problems.

6 Conclusions

In this paper we have identified three different critical components for building AmI environments (specialized kernel support, file system protocols, and UI support). We have enumerated different requirements that, after our experience in the field, we have found to be relevant and, therefore, should be considered for building them. We have also described a set of design guidelines for each one, which we are following to implement prototypes for them.

As of today, we have a prototype for i.e. almost finished, a implementation for the Creepy protocol finished, a prototype for a Creepy file system (speaking the protocol) near to be functional, and have just started the work on Pon. In the near future we will bring these three components into operation to port our smart space (i.e., AmI) into a more powerful environment; to evaluate the result and measure the benefits or drawbacks of the ideas presented here when used in practice.

References

The Mundo Method — An Enhanced Bottom-Up Approach for Engineering Ubiquitous Computing Systems

Daniel Schreiber, Erwin Aitenbichler, Marcus Ständer, Melanie Hartman, Syed Zahid Ali, and Max Mühlhäuser

Deploying ubiquitous computing systems into real world scenarios can realistically only be done in a bottom-up way, using smart building blocks. Our everyday environments are just too chaotic to allow top-down design of ubiquitous computing systems. Creating ubiquitous computing systems in a bottom-up manner has some inherent problems, which have not been successfully addressed in any existing approach, hence the scarcity of real-world ubiquitous computing systems. This article describes the Mundo Method for designing and implementing ubiquitous computing systems, which addresses two of these problems: structuring, i.e., separating the spontaneously emerging system into meaningful ensembles and orchestration, i.e., providing meaningful behaviour for an ensemble. The Mundo Method heavily relies on the MundoCore communication middleware, which is especially suited for ubiquitous computing applications.

Keywords: Bottom-Up Development, Methodology, Middleware, Smart Products, Ubiquitous Computing.

1 Introduction

There is an abundance of ubiquitous computing system infrastructures described in literature (see [1] for a survey). With them comes an almost equally large number of approaches for developing ubiquitous computing systems. Within these, we distinguish two main paradigms, which we call top-down and bottom-up.

- The top-down paradigm is characterized by the assumption that the complete system is under control of the engineer and that all components are known when a system implementation is designed.
- The bottom-up paradigm assumes that instead of being designed by an engineer as a whole, ubiquitous computing systems result from spontaneous integration of independent components.

To illustrate the difference between these two paradigms, we consider the example scenario of a kitchen that should be equipped with a ubiquitous computing system, i.e., the kitchen should become smart, similar to the ones described in [2][3][4].

Following the top-down paradigm, the system design would start by careful observation of user needs, thereby eliciting requirements for the system behaviour. The implementation is then designed assuming complete knowledge over all aspects of the room, and assuming control...
over all details of the system. Of course, the engineer would try to avoid re-inventing the wheel and leverage existing components (e.g., a smart blender), established architectures (e.g., goal based [5]) and patterns (e.g., tuplespace for communication [6]) for the implementation where possible.

If the kitchen system is built following the bottom-up paradigm, there would be no designer of the overall system in the first place. Instead, different self-contained items are deployed into the kitchen. They are expected to start interacting with each other in a spontaneous way, the behaviour of the ubiquitous computing system emerges from the single parts.

The advantage of the top-down paradigm is that it can result in well designed, useful systems. However, the pure top-down approach can hardly be applied in practice outside the laboratory, as realistic environments are inevitably constructed in a bottom-up way [7]. Therefore, the bottom-up approach is more applicable in practice but, there is no guarantee that the high-level requirements of the users are met by a system created in this way.

In this paper we present a method for engineering ubiquitous computing system, called the Mundo Method. It adheres to the bottom-up paradigm and provides solutions to its two key problems, which have been unsolved so far: (i) how to structure bottom-up created systems into meaningful ensembles and (ii) how to orchestrate these ensembles so that the system behaviour reflects the requirements of the user.

In the scenario of the smart kitchen structuring means logically separating the components in the kitchen from those in the living room. Orchestrating the kitchen ensemble could mean integrating the oven with a scale so that the former automatically selects the appropriate settings based on the weight of a roast.

We present a more detailed overview of the advantages and problems of the bottom-up paradigm in Section 2. This section provides an overview of the proposed four-step method. The following four sections correspond to the four steps of the Mundo Method. Section 3 explains our approach towards building the individual components from which ubiquitous computing systems are composed. We provide a versatile hardware and software setup for turning devices without ubiquitous computing features into components for ubiquitous computing systems. Section 4 describes how these building blocks are deployed and automatically set up communication with each other. The following two sections present our solutions for the two key problems found in bottom-up engineered systems. Section 5 explains how the spontaneously connected devices can be structured according to different organizational principles. In a classical top-down solution this would have been done by the designer up-front. Section 6 deals with the orchestration of these ensembles, i.e., how to define and deploy behaviour in ensembles.

2 The Mundo Method for Engineering Ubiquitous Computing Systems

We propose a four-step method for engineering ubiquitous computing systems, shown in Figure 1. The four steps are:

**build**: Implement the physical and software components for ubiquitous computing systems.

**deploy**: Deploy these components into an environment, where they establish communication links.

**structure**: Define which components should interact as ensembles.

**orchestrate**: Define higher-level behaviour and functionality of the system.

This is the opposite of a top-down paradigm, which would start with a definition of the overall functionality and structure of the system before building and deploying the components.

As stated above, the bottom-up paradigm better reflects everyday practices. This is its most obvious advantage. We shape, construct and engineer our non-ubiquitous computing systems mostly in a bottom-up way, e.g., by buying new utensils and appliances for the kitchen. The utensils and appliances are relatively autonomous blocks. Instead of introducing functional components, e.g., for "logging", found in other approaches for building for ubiquitous computing systems, we propose to utilize these existing components as much as possible.

Moving the focus in the system decomposition away from the software engineer to the end-user resembles the change from components to services in the service oriented architecture, which provide functionality meaningful to the business user. Because the components provide meaningful functionality to the end-user, we use the term *product* to describe them. A product is a pre-confectionated package of functionality, which can be reused easily. Thus, another advantage of the bottom-up paradigm is that it results in a high degree of reuse.

Products that can be used as building blocks of a ubiquitous computing system we characterize as smart products. Smart is a rather vague term and needs to be defined. Smart is often used to describe reasoning or artificial intelligence features [8]. In contrast, in this article we define as follows: A smart product is a self-sustaining component of a ubiqui-
A ubiquitous computing system should be more than the sum of its components. The Mundo Method addresses this problem on three different levels. At the networking level, all smart products constituting a system must be enabled to communicate with each other. This problem can be addressed by suitable communication middleware. We provide the MundoCore middleware, which is specifically targeted towards the needs of ubiquitous computing systems. However, besides low-level communication a system is characterized by:

- (higher-Level) structure, and
- (higher-Level) behaviour.

Thereby, higher-level refers to the structure and behaviour of multiple smart products. Experience has shown that...
Internet of Things

neither structure nor behaviour can be made to emerge automatically, e.g., by resorting to a common ontology for structuring the world-knowledge or planning. We propose therefore a more practical solution to these problems as part of the Mundo Method.

However, as we provide well defined interfaces to handle these problems in our approach, it can be easily adapted to use other methods. In the remainder of the paper, we will explain how the smart kitchen system shown in Figure 2 has been built using the Mundo Method. The goal was to build a system which is able to support the user in cooking a recipe with step by step instructions.

3 Building Smart Products

In order to integrate everyday electronic appliances into ubiquitous computing systems, these appliances must have built-in data interfaces. Unfortunately, e.g., kitchen appliances with integrated network interfaces are barely available on the market today.

The main reason for this is that a network software stack still requires considerable CPU and memory resources. However, appliances often only have very small 8-bit microcontrollers (e.g., the coffee machine described below) or no microcontroller at all (e.g., the blender described below). Using 8-bit microcontrollers is cheap (approx. $1), because they contain a full system-on-chip (SOC), while more powerful CPUs currently require external RAM and flash memory. This easily increases the overall system price by a factor of 10 or more. Fortunately, SOCs become more powerful from year to year, and as soon as complex networking software can be deployed on a single SOC, all kinds of appliances will come with built-in networking functionalities.

To emulate the capabilities of such future appliances with built-in network interfaces, we have designed the Smart Products Hardware Platform (SPHP).

3.1 Smart Products Hardware Platform

To turn an ordinary product into a smart product, we tap into the operating elements of the device, i.e., buttons, switches, dials, sensors, lights, displays, etc. Based on our earlier hardware modifications of off-the-shelf appliances [9], we have recognized that these modifications often have very similar requirements. Hence, we have designed the SPHP as a common basis for such modifications. The SPHP consists of:

- A computer-on-module, based on the Gumstix Verdex platform and the Linux operating system. It runs the device service, which encapsulates the functionality of the appliance as a software service. This allows monitoring and control of the device over the network (Figure 3a).
- A custom general purpose I/O (GPIO) board with an Atmel 8-bit microcontroller to run low-level, real-time control tasks. This board provides digital and analog inputs and outputs. Optionally, the input and output lines can be isolated by optocouplers or isolation amplifiers. In addition, an OEM RFID reader module can be plugged onto this board (Figure 3a).
- A power line modem, based on the Devolo dLAN technology. It enables data communication over the power line with up to 14 Mbit/s (Figure 3b).

Using power line communication for smart products seems to be an elegant solution, for several reasons. Wired Ethernet cannot be used, because kitchens and similar environments do not have a network infrastructure. USB or RS232 would require a nearby PC and special attention needs to be paid to galvanic isolation. Wireless networks are difficult to set up by end-users, while power line communication offers zero-configuration.

3.2 Smart Kitchen Appliances

In the following, we describe how components for a smart kitchen scenario as mentioned in the introduction are created using this platform.

Smart Coffee Machine: This appliance is based on an ordinary off-the-shelf espresso machine. When used out of the box, the user can choose between espresso, small coffee, large coffee, and hot water. There is a coffee bean container and a water tank attached to the machine. If either is empty, the display prompts to refill them.

To attach the SPHP, we modified the front panel circuit board (Figure 3c). This allows us to detect key presses, simulate key presses, check if the water tank is empty and read the pump control signal. The latter indicates that the machine is actually dispensing fluid and gives a very accurate measure of how much fluid has run through the pump; one pulse on this signal corresponds to 0.4 milliliters. An RFID reader is used to identify cups placed under the coffee dispenser. With all this information, we can detect the machine state and user actions to automatically start processes or proceed in a workflow, e.g., for descaling the machine.
To emulate the capabilities of future appliances with built-in network interfaces, we have designed the Smart Products Hardware Platform (SPHP).

**Smart Blender:** The modifications to the blender are similar to those described above (Figure 3d). However, the original blender does not use a microcontroller and does not even have a galvanically isolated power supply. Hence, all digital signal lines have to go through optocouplers and the analog signal lines have to go through isolation amplifiers.

Our Smart Kitchen System (Figure 2) also uses the following two devices, which are not based on the SPHP and are simply connected to a PC:

**Smart Scale:** This scale from Kern has a built-in RS-232 data interface. It is connected to the PC through an USB adapter cable.

**Transponder Reader:** This is a custom-built reader for SimonVoss electronic door keys. Such door keys are used by several departments of our university. Building on this large deployment base (approx. 4,000 users), we use this technology for identifying users in our applications.

The SPHP turns ordinary appliances into network-capable smart products. All further aspects of system integration are now a concern of software.

### 4 Deployment and Networking

The individual components as described in the previous section are deployed in a kitchen environment. To become a ubiquitous computing system, the individual components need to communicate with each other. However, in a bottom-up approach, we cannot rely on hard-wired communication links. The components must establish a communication infrastructure autonomously. Just like they use the power supply network provided in the kitchen they may connect to the LAN. We call a group of communicating smart products a *smart products ensemble*.

We use the MundoCore [10] middleware to realize communication between smart products. It is an open-source communication middleware for developing, deploying and managing highly dynamic distributed systems composed of services. In the following, we first describe the basic concepts and elements of MundoCore-based systems. The nodes and services involved in the smart kitchen scenario are shown in Figure 2.

#### 4.1 Node

A MundoCore node is an execution container that contains the MundoCore runtime environment and an arbitrary number of services. Each node corresponds to an operating system process. It is possible to run any number of nodes on a single computer. Nodes are autonomous and use peer-to-peer communication.

MundoCore versions are available for Java and C++. A Java-based node hosts Java services, while a C++ based node hosts C++ services. Because both versions use the same communication protocol, services can arbitrarily talk to each other, no matter in which language they are programmed. Hence, the developer has the freedom of choice as to whether an application service is implemented in Java or C++. In general, higher-level services, such as workflows and user interfaces are easier to develop in Java, while C++ is more efficient and provides much better access to the hardware.

In the kitchen scenario (Figure 2), every smart product contains one C++ node on the embedded Gumstix computer. This allows good access to the hardware. Java is practically not usable on this resource-constrained platform. The interaction PC hosts two nodes: one based on C++ and one based on Java. Here, C++ services are used to access the USB devices, while the main application services are implemented in Java.

MundoCore is well-suited for the bottom-up construction of ubiquitous computing systems as it supports the automatic discovery of neighbour nodes in the same subnet. This is realized by the following concepts:

- **Broadcast and Multicast discovery:** By using IP broadcast or multicast packets, nodes advertise their presence in the network. This discovery mechanism works inside the local subnet or the multicast scope.
- **Join via the primary port:** MundoCore guarantees that if at least a single node is running on a computer, then a node is reachable on the primary port. If the node holding the primary port terminates, then another node will try to allocate the primary port. This concept allows nodes in external networks to join a MundoCore overlay network when a single computer running a node is known.
- Each MundoCore node runs a publish/subscribe broker. All brokers in the overlay network are also automatically interconnected. Most of the communications mechanisms that are used by services, such as eventing and remote method calls are based on this publish/subscribe system.

#### 4.2 Service

The network discovery happens at the layer of nodes that discover each other as described above. Insides a node, smart product functionality is made available as services.

Appliances must have built-in data interfaces in order to be part of ubiquitous computing systems.
For example the smart coffee maker functionality is implemented as a MundoCore service, called SaecoService.

A service has defined input and output ports which are specified in the service interface. Ports are interconnected using channels provided by the publish/subscribe system. They can provide unicast or multicast connections.

### 4.3 Communication

MundoCore provides support for the three major traffic classes: events, request/reply, and media streaming.

The publish/subscribe abstraction is well-suited for distributing events. For example, when the SaecoService detects an RFID tag on the cup holder surface of the coffee machine, it publishes an RFID Event to the channel saeco.rfid. All services interested in this event can subscribe to the channel saeco.rfid and will receive all such events.

Request/reply interactions are supported via remote method calls, which are also performed over publish/subscribe in most cases. This enables a good decoupling of the services and provides location transparency as well as execution transparency. Hence, service consumers as well as service providers can be moved in the network, without requiring any adaptations on the side of the respective communication partner.

Consequently, when developing a smart product, one does not have to hard-wire connections to other products it depends upon, but these can be dynamically satisfied at runtime.

### 4.4 Service Discovery

Besides the implicit location of objects via channel names, MundoCore also provides service discovery functionality. A service can explicitly search the MundoCore environment for services fulfilling certain criteria, e.g., implementing an interface. Once discovered, a one-to-one connection for remote method calls is established.

### 5 Structuring Smart Products Ensembles

In a top-down designed system, all possible communication paths between products are modelled statically in the design phase. This will inevitably lead to a closed system. In contrast to that, we want to facilitate ad-hoc interactions between products that were not designed in beforehand with our bottom-up approach. Communication links are automatically formed using node discovery and products can search for required functionalities in the environment using service discovery.

However, we cannot permit complete freedom in these automatic discovery processes. For example, if products in the kitchen need to cooperate to realize some functionality, the products in the living room should not be considered for cooperation, as they are too far away. Hence, setting up smart product ensembles consists of two steps:

1. Structure the available smart products using scopes.
2. Within the boundaries defined by scopes, the system can automatically discover and compose services.

Communication scopes have been proposed as an adequate means for organizing and structuring distributed systems [11][12]. They emerge naturally from technical implementations, e.g., subnet or multicast groups introduce communication scopes into a system. We generalize the idea of communication scopes and propose a model for communication scopes in a middleware, called zones, which can be used to implement different real-world structures, even in parallel. Thereby, the zones model provides a common abstraction for different kinds of structures, also those that

![Figure 4: World View – Location-Based Zones.](image)
do not directly emerge from the network level.

The technical communication scope is often a manifestation of an underlying structure in the real world. For example, in a ubiquitous computing system scopes can be based on location in the real world [13], organizational hierarchy, or trust relationships [14]. By taking the structure management out of applications and moving it into the middleware, applications can dynamically adapt to the changing underlying structure in the real world.

5.1 Publish/Subscribe Communication with Zones

We extend the classic publish/subscribe system model containing consumers, producers and brokers with zones and zone arbiters. A zone is a logical grouping of consumers and producers. For smart product ensembles, a zone is an organizational unit of nodes in the publish/subscribe system, in which events do not propagate beyond the zone in which they were generated. For example, smart products in the kitchen zone can only observe the events that are created by other nodes in the kitchen.

To implement zones, the channel-based publish/subscribe model of MundoCore remains unchanged. Smart products can still subscribe and publish to channels, as described above. However, the message broker has been augmented to take the zone of the smart product into account. It will only dispatch messages to a product if it is a member of the required zone.

Every zone is identified by a URI, which may have a hierarchical structure, depending on the aspect the zone models. Currently, we model zones for the following aspects:

- **Location**: Smart products are joined into zones based on their location, e.g., smart products are joined into the kitchen zone if they are physically located in the kitchen.
- **Organizational Structure**: The zone structure can base on organizational hierarchy, e.g., the Telecooperation Lab is part of the Computer Science Department, which is in turn part of TU Darmstadt.
- **Task**: Ubiquitous computing systems can only work effectively when they have a model of the task or even the process the user is currently performing. Task-based zones are automatically created and contain the products involved in the current task of the user.

5.2 Zone Arbiter

Zones are managed by a *zone arbiter* service, which is implemented in a distributed way, running on every MundoCore node. In a zone-enabled publish/subscribe system, the zone arbiter service provides the following operations to smart products:

- `create(X, Z)` smart product X creates a new zone Z
- `join(X, Z)` smart product X joins an existing zone Z
- `leave(X, Z)` smart product X leaves zone Z
- `change(X, Zl, Zj)` smart product X leaves zone Zl and joins Zj
- `destroy(X, Z)` smart product X destroys zone Z

The zone arbiter may restrict smart products from creating and joining certain zones, based on the structure one wants to model with the zones. Currently, we have implemented basic support for zones based on location models, as described in the next section.

5.3 Location-Based Zones

As location is an obvious basis for structuring smart products ensembles, we explain the implementation of this case in more detail. Figure 4 shows the floor plan of the home containing the kitchen that should become smart. As one can see, several smart products have been deployed throughout the home. The goal of the structuring in this case is to separate the smart products in the kitchen from those in the living room.

We assume the home is equipped with a location tracking system which provides geometric coordinates to smart products. In addition to the geometric model, a symbolic location model is required that describes the structure of the home at the room level, i.e., living room, kitchen, dining room, etc.

The MundoCore middleware comprises a service for hosting such symbolic location called WorldModelStore. The model used by this service consists of named rectangular regions. The model can be edited using the WorldView tool (shown in Figure 4). This tool does not only allow offline editing of the symbolic location model, but also live inspection of the current setup.

Another service, the Context Server, also coming with MundoCore, performs the transformation between the geometric coordinates from the tracking service to the symbolic locations. The zone arbiter now continuously checks the location reported for each smart product, joining them into zones reflecting their current symbolic location. This takes into account changes in the location of the smart product, i.e., if a TV is carried from the kitchen into the living room its zone changes, as well as structural modifications of the symbolic location model. The latter allows the imposition and maintenance of a structure after smart products have been deployed in a bottom-up manner.

6 Orchestrating Smart Products Ensembles

To provide a substantial benefit to the end-user, the ubiquitous computing system must support higher-level behaviour or task knowledge which goes beyond invoking the functionality of a single product via a network interface. If we have to deal with more complex functionality, the required task knowledge can usually be specified by a set of steps that need to be performed. For example, the system should be able to guide the user through complex proce-
dures, like the descaling of a coffee machine in the kitchen. In this case, the task knowledge can be built-in to the coffee machine.

However, in many cases, higher-level behaviour spans multiple products. A classical example being a printer and a scanner, which can together offer copy-machine functionality. In a top-down designed system, the copy-machine functionality would be modelled up-front. If the system is constructed in a bottom-up way, the question arises how this higher-level behaviour is introduced to the system.

It could possibly come with a product, which provides task knowledge beyond its own functionality, e.g., a printer that has knowledge about copying. It is hard to generate such complex task knowledge in a completely automated way, e.g., through reasoning. For that reason, the end-user or developer should be able to specify and deploy task knowledge at runtime in an interactive way.

6.1 Context Aware Workflows

In order to execute tasks in a ubiquitous computing system, the products at first require a representation of the usually procedural knowledge necessary to carry out the task. In non-ubiquitous computing systems, workflows are used for a similar purpose [15][16][17]. Workflows are a formalized means to model higher-level procedural knowledge in a business organization. Based on the definition of the Workflow Management Coalition [18], we define a workflow as follows: "A workflow represents a process and organizes it as a finite set of activities, containing information about their interrelations (called transitions). Activities contain information about involved organizations and users (called groups and participants), required input and output data, and tools which need to be applied".

The activities and participants in ubiquitous computing systems differ from those found in traditional workflow systems. For example, they do not take the user interaction into account which is essential in ubiquitous computing systems. Further, ubiquitous computing settings pose special requirements on workflows. As they are used to model higher-level behaviour, spanning multiple smart products, they must also support activities that cannot be executed by the smart product running the workflow itself. In this case, the activity can be carried out by a product nearby, or as a last resort, the user might be asked to perform the activity.

We proposed a novel workflow language for ubiquitous computing systems called XPDL4USE [9]), which is based on the XPDL [20] workflow language. Using this language the activities in a workflow are described in a way that can be mapped to the service discovery of the underlying middleware. This way, an activity in one workflow may trigger a remote workflow on another smart product. For example, we consider a workflow for making a cocktail in which one activity requires crushed ice for preparing a cocktail. Further, we assume only ice cubes are available. The ice cubes can be processed into crushed ice by a blender. Therefore, this workflow activity can be performed with the help of the blender, and the activity should trigger a sub-workflow with the blender.

Figure 5 shows an example workflow for preparing a cocktail [21]. The currently active activity ‘putGlass’ is highlighted (Figure 5 a), which needs to be performed by the user and is thus rendered as a user interface (‘Take a glass and add ice’, Figure 5 b).

6.2 User Interaction and Context in Workflows

The user needs to be approached, whenever an activity cannot be executed by any available smart product. In this case, the user must find a way to carry out the task.

Smart products often provide very limited means to interact with the user directly, e.g., a coffee machine usually only has a few buttons and a one or two line display. For that reason, the required interaction with a user should be reduced to a minimum.

This can be done by automating actions and support of implicit interaction instead of requiring explicit user input. For that purpose, the component executing the workflows...
must be aware of the current state of the ubiquitous computing system and the user’s activities, i.e., they must be context aware.

This enables a smart product for example to detect when the user finished a step and thus to automatically proceed to the next step. For example, a water sensor can automatically detect whether the user has filled water in the water tank instead requiring the user’s explicit acknowledge via a user interface. In this case, context triggers the next activity in the workflow.

7 Summary

In this paper, we proposed a novel method called the Mundo Method to build ubiquitous computing systems in a bottom-up way as we think this approach is more applicable in practice than the top-down approach. However, the current approaches for building ubiquitous computing systems in a bottom-up way have several problems, i.e., structuring the different components in a ubiquitous computing system and specifying higher-level behaviour which involves several components. We showed how these problems can be tackled with the Mundo Method and how its different steps can be realized in practice.

Acknowledgements

Part of this research was conducted within the SmartProducts project funded as part of the Seventh Framework Programme of the EU (grant number 231204).

References


Model Driven Development for the Internet of Things

Vicente Pelechano-Ferragud, Joan-Josep Fons-Cors, and Pau Giner-Blasco

The Internet of Things vision is about reducing the gap between the physical and the digital world to make daily activities more fluent. By providing a digital identity to real-world objects, Information Systems can handle them in an automatic way. This enables physical objects to participate actively in business processes. In such systems, the technological heterogeneity in Auto-ID and the fast-changing requirements of business processes hinders their construction, maintenance and evolution. This paper shows how the Model Driven Development can help to systematize the development of business process-supporting systems that integrate physical elements.

Keywords: Auto Identification (Auto-ID), Business Processes, Implicit Interaction, Internet of Things, Mobile Workflows, Model Driven Development, Models.

1 Introduction

The introduction of Information Technologies creates a digital world where information can be automatically processed, improving the Information System efficiency. However, computers have a limited vision of the real world they are managing. Thus, there is still a challenge in automating the linkage between digital and physical worlds.

Nowadays, Information Systems that deal with real-world objects (such as baggage pieces in an airport or products in a supermarket) are normally informed by humans. We consider this use of humans as information carriers to be inefficient and error-prone. The gap between the physical and the digital world commonly results in mishandled luggage or long queues at the supermarket.

Internet of Things vision [1] is about reducing this gap to make daily activities more fluent. By providing a digital identity to real-world objects, Information Systems can handle them in an automatic way. This enables physical objects to participate actively in business processes by reducing the gap between physical and virtual worlds [2]. In addition, the wide availability of mobile devices with advanced capabilities allows users to access the information and services where they need them.

Although developing such systems is feasible, the technological heterogeneity in Auto Identification (Auto-ID) and the fast-changing requirements of business processes hinders their construction, maintenance and evolution. Therefore, there is a need to move from ad-hoc solutions to sound development methods in order to assure the quality of the final product. Model Driven Engineering techniques [3] can help the developers to provide development principles for constructing such systems.

The main goal of this paper is to show how Model Driven Development can help to systematize the development of business process-supporting systems that integrate physical elements. The development process defined is focused on the particular requirements of the Internet of Things domain. Then, a software solution is obtained from this specification by following a set of systematic steps. This solution is supported by an architecture specifically designed to cope with the Internet of Things requirements and to survive technological evolution.

Specifically, our target applications are physical mobile workflows, which are business processes that take advantage of the capabilities of mobile devices for the identification of physical elements. The high heterogeneity in identification technologies, the fragmentation in mobile platforms and the fast-changing nature of business processes, make it hard to develop such systems in a sound manner. In this paper, modelling techniques are applied in order to develop such systems from a higher abstraction level.

The remainder of the paper is structured as follows. Sec-
tion 2 presents the main ideas for integrating business processes with the Internet of Things principles. Section 3 discusses taking into account a modelling strategy to help defining systems by avoiding problems regarding technical details. Section 4 presents the main ideas for developing Internet of Things systems by taking into account a model driven strategy. A tool support for this method is described in Section 5. Finally, Section 6 concludes the paper.

2 Business Processes and the Internet of Things

Integrating real-world objects in business processes has been successfully demonstrated to reduce media breaks, human errors and delayed information problems [2]. Many benefits are obtained in economic [4] and process improvement terms [5][6]. A better integration of real and virtual worlds not only improves business processes, but also enables new business models [7][8].

However, it is not an easy task to develop systems of this kind. Business processes are constantly changing, which in turn requires the corresponding evolution in the supporting Information System. In addition, systems in the Internet of Things context, involve a great diversity of technologies to bridge physical and digital worlds. This heterogeneity forces the developer to know the details of each technology involved in the system, making these systems difficult to develop and to maintain. From the methodological perspective, there is a need for a systematic development method that can free developers from technological details and that also allows a fast propagation of requirement changes to technological solutions.

This work presents a method that provides a mechanism for defining the desired degree of automation for the physical-virtual linkage of a given business process. In order to systematize the development of such systems, the method is based on the Business Process Management (BPM) initiative principles. BPM is an initiative that promotes the continuous re-engineering of business processes. Since current solutions for BPM are mainly focused on the digital world (i.e. orchestration of digital services), support is lacking for coping with the particularities of the physical-virtual linkage in the different stages of the BPM cycle. This work builds onto existing BPM techniques and extend them to integrate business processes with the physical world at different levels. Existing BPM techniques are complemented with support for capturing the identification requirements, evaluating the user participation in a real environment, and executing the workflow in a software platform.

3 Why a Modelling Approach?

Traditionally, the application of Auto-ID to business processes has mainly been approached from a technological perspective (by developing integration middleware and architectural designs). However, deploying an Auto-ID-enabled system involves a lot more than purchasing the right tags and installing the right readers [9].

The way in which a business goal is achieved depends on the properties of the physical-virtual integration. Certain business models are only feasible with an adequate level of automation in the physical-virtual linkage [7]. For example, using RFID for identifying products in a supermarket allows the checkout process to be automated [10], and it does not require the participation of a cashier in the process. Thus, when modelling a business process it is not possible to determine which tasks are required for handling physical elements (e.g. requiring a cashier to handle them or not) if there is no notion of how they participate in the process. Models are key in our proposal to provide this notion by linking identification requirements to technological requirements in a gradual manner.

Abstraction is one of the fundamental principles of software engineering in order to master complexity [11]. Our approach makes use of modelling techniques in order to promote abstraction in the development of physical mobile workflows. By abstracting technical details, we can describe the physical-virtual connection of a workflow regardless of the particular technology used for the implementation. In the case of physical mobile workflows, modelling techniques are applied to obtain the following benefits:

- **Focus on the process.** Separation of concerns is promoted by our approach in order to allow designers to focus on a specific aspect of the workflow at a time. Business analysts can define the way in which physical elements participate in a business process without considering technology limitations. They can think on the way they want the process tasks to flow, and later, the appropriate identification mechanisms can be chosen to cope with their requirements.

- **Explore the solution space.** The use of models allows the capture of not only a specific solution but also the rationale behind it. In this way, alternative solutions can be re-considered and the design knowledge can be better reused for similar problems. In addition, support for traceability allows to easy identification of the model elements affected when different issues are detected during the system evaluation.
Support system evolution. The fast changing nature of business processes, and the technological heterogeneity of identification technologies, suggests that systems in this area must be designed to evolve. By analyzing the knowledge captured in models by our approach, it can be easily determined whether or not a new technology fits better with the requirements of the physical-virtual linkage for a given process.

Our approach involves the manipulation of models in different manners. An overview of the steps involved in our method is provided in the following section.

4 Method Overview

This section presents the development method introduced in this paper. The design stage is the initial stage in our method (see Figure 1). Since we follow a model-driven approach, the specification obtained at design drives the later stages in the development of the system. Thus, the design stage becomes central to the development method. The design method captures, by means of models, the concepts that are relevant in the development of physical mobile workflows.

Modelling techniques describe a system by handling abstractions of the problem space. This allows designs to be expressed in terms of concepts from the application domain instead of concepts from the technical space [12]. By raising the level of abstraction, systems can be developed without taking into account much of the implementation details of the underlying platform. Models are used to organize knowledge about the problem domain in order to...
guide the development. Furthermore, when models are machine-processable and precise-enough, they can be used to automate the production of a software system.

Thanks to Model Driven Engineering (MDE) techniques, it is possible to traverse the gap between the high-level concepts used in design and the technical details of the particular mobile platform that are used for the system implementation.

Figure 1 illustrates how our approach can connect the concepts used at design (e.g. task, obtrusiveness level, physical interaction, etc.) with a particular implementation platform. Our approach introduces two layers to cover the abstraction gap between the design concepts and the software platform:
**Parkour.** The design concepts defined are formalized into a modeling language named Parkour. Designers can define the requirements for a workflow by following our design method and use the system specification during the development process. By formalizing design concepts, system descriptions can be processed to validate their consistency and automate some steps in the development. It is possible to transform workflow specifications based on Parkour into a final software solution using MDE techniques.

**Presto.** Presto [13] is an architectural framework specifically defined to support applications in the physical mobile workflow domain. Presto is a sustainable software architecture, that is, an architecture that can evolve over time throughout several technological cycles. The architecture elements are defined in a technology-independent fashion and code generation techniques are used to translate the generic architecture into components for each particular mobile platform targeted for development.

We believe that it is important to describe the physical-virtual linkage for a workflow at design time since the way in which a business goal is achieved depends on the properties of the physical–virtual integration. Our approach is based on existing notations for business process modelling such as Business Process Modelling Notation (BPMN). Parkour complements BPMN with the modeling of three aspects that affect the physical–virtual linkage:

- **Obtrusiveness level for each task.** We identify the obtrusiveness level required for the different tasks of the workflow. Each task can be carried out at a different level of initiative and attention according to the conceptual framework introduced in [14].
- **Interaction techniques to use.** Users can interact with the objects in their surroundings in different manners. The interaction technique that allows the task to be performed at the adequate obtrusiveness level is selected for each task in the process.
- **Supporting technologies.** The different mobile device involved in a workflow must be equipped with technologies that support the interaction technique that was chosen in the previous stage.

More detail on these aspects are provided below. In order to illustrate these aspects we are using the Smart Library case study [13] were different workflows are supported in the context of a Library.

### 4.1 Obtrusiveness Level for Each Task

The first step in our method is to detect the tasks to support in a workflow and determine at which obtrusiveness level they should be supported. Each task in a workflow can be provided at a different obtrusiveness level. The design method introduced in this work allows designers to specify a workflow by indicating (1) to which extent the different tasks should intrude the users mind and (2) which technologies can be used to support such interactions. Since mobile devices provide rich interaction mechanisms, we are not considering interactions to be either implicit or explicit in a binary fashion but part of a continuum. In this work we use the concept of obtrusiveness level to indicate to which degree an interaction is implicit or not.

In order to specify the obtrusiveness level for a task, we make use of the conceptual framework presented in [14]. This framework defines two dimensions (see Figure 2) to characterize implicit interactions: *initiative* and *attention*. According to the initiative factor, interaction can be reactive (the user initiates the interaction) or proactive (the system takes the initiative). With regard to the attention factor, an interaction can take place at the foreground (the user is fully conscious of the interaction) or at the background of user attention (the user is unaware of the interaction with the system).

We found it very useful to consider initiative and attention as independent concepts. In the case of physical mobile workflows, automation and user awareness are factors that usually vary independently. For example, an automated task (i.e., proactive in terms of attention) can require the user to be aware of it (i.e., foreground in terms of the attention dimension) or not (i.e., at the background of user attention) depending on different context factors (such as the user workload).

Figure 3 shows some of the most representative tasks of the Smart Library case study and their obtrusiveness level. The obtrusiveness space in this case was defined by dividing each axis into different parts. The initiative axis in this case is divided into two parts: *reactive* and *proactive*. The attention axis is divided into three segments, which are associated with the following values: *invisible* (there is no way for the user to perceive the interaction), *slightly noticeable* (usually the user would not perceive it unless he/she makes some effort); and *completely aware* (the user becomes aware of the interaction even if he/she is performing other tasks). Designers can divide each axis into as many parts as they require for describing the obtrusiveness level for the process. This division is later considered when selecting the appropriate interaction mechanisms for each el-
Figure 5: An Example of Different Mediums and Specialization Relationships

Figure 6: Auto-ID Services involved in the Smart Library
This work is part of a comprehensive proposal that includes a framework called Presto

The ‘Medium’ concept is a technological-independent mechanism for describing identification requirements.

The medium concept is a technological-independent mechanism for describing identification requirements. This concept is useful for guiding the technology selection process. Designers can organize mediums into a hierarchy in order to better capture their commonalities and variability. In the example, the paper medium is specialized in two mediums which are considered as sub-types. These sub-types distinguish between an identifier that is expressed by means of numbers from one that is expressed by means of an image. In the example in Figure 5, image on paper and numbers on paper are paper-based mediums but the interaction mechanisms they support are different. The image on paper medium supports the pointing mechanism, whereas the numbers on paper medium requires users to perform the identification and type of the associated identifier (i.e. user-mediated interaction).

With this analysis we can determine the impact of the technology selection in the process. For example, if RFID (based on the radio medium) were considered not appropriate for its use by the library members due to the lack of RFID-capable mobiles widely available in the market, we can explore the consequences of using another kind of identification technology such as a paper-based one. In this case, using a visual marker (based on the image on paper medium) would require to use the pointing interaction technique instead of the touching one. The final decision depends on the designer but our approach allows the consequences of these decisions for the workflow to be tracked. Figure 6 shows the identification technology selected for different tasks of the Smart Library (at the bottom of each figure).

This study of the commonalities and variability of medium properties is also useful for supporting process evolution. When business processes are re-engineered new technologies can be considered for their support. By classifying the new technologies according to the existing medium hierarchy, it can be determined whether these new technologies are really adding value to the process or not.

4.3 Technology Selection

Many technologies have existed for long time to bridge the gap between the physical and the virtual worlds [21]. Thus, physical elements can be identified in a myriad of different ways (e.g. using a sequence of bars on paper, or a radio wave emitted by a RFID tag). The selection of the most appropriate one depends on many different factors. For example, RFID may not be appropriate for the users of the example library if we want them to autonomously borrow books with their device since RFID-reading capabilities are not present on the average mobile device whereas as far as we know cameras, to capture images, are more often present. Nevertheless, for taking such decisions it is necessary to understand the impact that each technology has in the process. For example, we can wonder if there is another technology that can be used to replace RFID and supports our workflow as it was described (and in the case that changes are required, easily identify them).

In order to organize the knowledge about the different identification technologies we define the medium concept. Mediums are defined in this work as physical supports for identifiers. Designers can define mediums to represent a genre of technologies in an abstract manner. For example, designers can define the paper medium (see Figure 5) in order to describe in an abstract manner a set of identification technologies that have some requirements in common: (1) they are cost-effective by using identifiers that are cheap to produce and that require simple devices for their capture; and (2) the technologies require direct line-of-sight since identifiers are recognized optically. At this point of the design we are not interested on the specific technology used (barcodes, visual markers, Quick Response (QR) Codes, etc.) but in a broad category of identification technologies with common properties. In particular, we consider which interaction techniques they support.

The medium concept is a technological-independent mechanism for describing identification requirements. This concept is useful for guiding the technology selection process. Designers can organize mediums into a hierarchy in order to better capture their commonalities and variability. In the example, the paper medium is specialized in two mediums which are considered as sub-types. These sub-types distinguish between an identifier that is expressed by means of numbers from one that is expressed by means of an image. In the example in Figure 5, image on paper and numbers on paper are paper-based mediums but the interaction mechanisms they support are different. The image on paper medium supports the pointing mechanism, whereas the numbers on paper medium requires users to perform the identification and type of the associated identifier (i.e. user-mediated interaction).

With this analysis we can determine the impact of the technology selection in the process. For example, if RFID (based on the radio medium) were considered not appropriate for its use by the library members due to the lack of RFID-capable mobiles widely available in the market, we can explore the consequences of using another kind of identification technology such as a paper-based one. In this case, using a visual marker (based on the image on paper medium) would require to use the pointing interaction technique instead of the touching one. The final decision depends on the designer but our approach allows the consequences of these decisions for the workflow to be tracked. Figure 6 shows the identification technology selected for different tasks of the Smart Library (at the bottom of each figure).

This study of the commonalities and variability of medium properties is also useful for supporting process evolution. When business processes are re-engineered new technologies can be considered for their support. By classifying the new technologies according to the existing medium hierarchy, it can be determined whether these new technologies are really adding value to the process or not.
4.4 Deployment Distribution

Functionality in support of a mobile business process is normally distributed across different computing resources. In order to support the integration of physical elements in the different tasks from a business process, the identification functionality should be organized. This involves defining the setting of the different resources for the system. We use the concept of deployment unit to encapsulate the functionality required for the support of different tasks by means of a set of technologies that is deployed in a particular device.

To provide an abstract view of the Smart Library setting, the different deployment units are specified in our approach. Figure 6 represents a diagram for the Auto-ID related deployment units for the Smart Library case study.

The following properties are defined for each deployment unit; the task that it supports, the physical elements that are involved and the technologies that are used. The Member Mobile deployment unit represents a set of software components that support the book loan and pick up reserved book tasks by making use of QR Code technology for their completion. The software solution for this deployment unit is accessed by the library members from their mobile devices. The Return Box deployment unit is in charge of automatically detecting the returned books by means of RFID. Thus, each return box requires one or several RFID antennas capable of detecting its content. The Librarian Mobile deployment unit is accessed by the librarians from their RFID-equipped mobile devices in order to transfer the returned books from the return box to their shelf. The Shelf Detector deployment unit is also supporting the place the book in the shelf task. In this case, it detects whether a book is placed in a wrong shelf.

When defining the different deployment units it is essential the selected technology to be consistent with the other aspects considered during design. That is, the technology selected must use a medium that allows the interac-
tion technique defined for the task, and this interaction technique supports the obtrusiveness level specified for this task. For example, the Member Mobile deployment unit makes use of QR Codes which is based on a paper medium that supports the pointing interaction technique since we considered this technique adequate to replace the touching interaction technique initially considered (as discussed in Section 4.3). This kind of validation can be difficult to perform as the workflow model grows. For this reason, tool support has been provided to automate this process.

5 Tool Support

Parkour introduces design concepts to specify different aspects of the physical-virtual linkage for a workflow. Tool support has been provided to model and validate workflows that are designed according to our method.

For the specification of business process models we took the BPMN metamodel defined in the SOA Tools Platform Project (STP) as a basis. The STP metamodel defines the modelling constructs for the BPMN modelling language. The STP metamodel has a very complete support for the BPMN specification covering almost all BPMN shapes, connections and markers except the layouts and appearance of the lanes inside a pool and the group-artifact. The STP project also provides a functional editor for BPMN diagrams (see Figure 7, top) which is integrated with other Eclipse-based modelling tools.

In addition to the editing support, we provided validation capabilities to verify that the description of the physical-virtual linkage in the workflow is consistent. Eclipse Modelling tools are used to formalize the concepts introduced in Parkour and to specify the different constraints on them. For example, the following expression checks whether a specific medium supports a particular interaction tech-
nique by taking into account the medium hierarchy:

```java
Boolean supportsTechnique (Medium this, InteractionTechnique inter):
  this.InteractionTechnique.contains (inter) ||
  (this.parent != null?
  this.parent. supportsTechnique (inter) : false);
```

By enforcing the check of different constraints, inconsistencies are automatically detected in workflow specifications. This allows designers to anticipate the detection of problems in the workflow before effort is put into the implementation and deployment. In this way, designers can foresee the impact of removing, adding or changing a specific identification technology by simply modifying the model. This allows the answering of questions such as ¿Can RFID be used by library members to borrow books by pointing at them?

Figure 8 shows a Parkour instance model where the editor has detected some inconsistencies. The editor verifies the constraints each time the model is saved (or by user demand). The errors and warnings detected are integrated into the standard error view provided by Eclipse. In this case, the model contains one error and one warning. The error is produced because the deployment unit is supporting the borrow book task by means of RFID while the book was only tagged with a paper-based identifier in the example. Thus, the task cannot be supported with the technology defined.

6 Conclusions

This paper has presented a design method to specify physical mobile workflows. It has illustrated how to capture the requirements for physical-virtual linkage in a gradual manner by means of modelling techniques. The use of models helped to centralize the knowledge about the workflow and organize it in a way that is easy to handle by designers (e.g. work with technology independent concepts, detect inconsistencies, etc.).

The design method provided relies on proven techniques and frameworks for business process modelling, implicit interaction design and physical interaction patterns. The tool support provided for Parkour enabled the anticipation of errors in the workflow described. However, the use of the tools required some advanced knowledge in modelling techniques.

This work is part of a comprehensive proposal that includes a framework called Presto and code generators to fully support the development method presented in Figure 1. Presto defines an architecture that fits with the specific needs of physical mobile workflows. This architecture has been defined in a technological independent manner. In this way, components in this architecture can be mapped to different technological platforms. Our approach provides support to automatically obtain an architecture design for a given workflow specification. Code generation is also provided to produce, from a model based in our abstract architecture, the implementation assets that are required for the technological infrastructure in a given technology platform. In particular we have defined code generation support for the Android platform that frees developers from dealing with Android-specific components and focus on implementing the required business logic in Java.

Acknowledgments

This work has been developed with the support of MICINN under the project EVERYWARE (TIN2010-18011).

References

[10] George Roussos, Juha Tuominen, Leda Koukara, Olli Seppala, Panos Kourouthanasis, George M. Giaglis,


Digital Object Memories in the Internet of Things

Michael Schneider, Alexander Kröner, Patrick Gebhard, and Boris Brandherm

Digital Object Memories, DOM, comprise concepts and technologies to physically and conceptually associate digital information with physical objects in an application-independent manner. By storing information about an object’s properties, state, and history of use in its digital memory, objects become self-representative, which allows for novel kinds of open-loop applications in the Internet of Things. In this paper we report on work performed on architectures for Digital Object Memories, concepts for interacting with Digital Object Memories, and the application of Digital Object Memories in the context of the Internet of Things.

Keywords: Digital Object Memories, Internet of Things, Semantic Product Memories, Semantic Web.

1 Introduction

Digital Object Memories, DOM, comprise hardware and software components that physically and/or conceptually associate digital information with real-world objects in an application-independent manner. Such information can take many different forms (structured data and documents, pictures, audio/video streams, etc.) and originate from a variety of sources (automated processes, sensors in the environment, users, etc.). If constantly updated, Digital Object Memories provide a meaningful record of an object’s history and use over time.

From a technical point of view, Digital Object Memories provide an open-loop infrastructure for the exchange of object-related information across application and environment boundaries. Besides fostering information re-use and reducing the risk of information inconsistencies, they allow for novel classes of applications in which rich object histories are created and exploited. From the user’s point of view, Digital Object Memories create a new design space for everyday interactions. Physical objects could become sites for their owner’s personal stories, but also afford people the opportunity to explore an object’s provenance and connections to other elements of physical and digital life. In this sense there is the potential for designers to augment or even transform our relationship with objects and the services that they mediate.

The concept of Digital Object Memories on the one hand and the idea of the Internet of Things on the other hand complement each other perfectly. As a collection of general object-related information, Digital Object Memories make physical items self-representative and hence provide a valuable knowledge source that can be exploited by applica-

Figure 1: Relation between Digital Object Memories and the Internet of Things.

Authors

Michael Schneider works as a researcher at the German Research Center for Artificial Intelligence (DFKI, <http://www.dfki.de>) in the SemProM project. His interests include proactive applications in the context of smart objects and the Internet of Things. <michael.schneider@dfki.de>

Alexander Kröner is coordinating research activities at the German Research Center for Artificial Intelligence (DFKI, <http://www.dfki.de>) in the SemProM project. His interests comprise adaptive user interfaces, user modelling, and smart environments. <kroener@dfki.de>

Patrick Gebhard works as a senior researcher at the German Research Center for Artificial Intelligence (DFKI, <http://www.dfki.de>) in the projects SemProM and INTAKT. His research interests comprise the employment of emotional virtual characters for computer games as well as in entertainment and educational applications. <patrick.gebhard@dfki.de>

Boris Brandherm works as researcher at the German Research Center for Artificial Intelligence (DFKI, <http://www.dfki.de>) in the SemProM project. His research interests are embedded dynamic Bayesian networks, Dual Reality and smart objects in the context of the Internet of Things. <boris.brandherm@dfki.de>
In this article we report on work that we performed over the last year on different aspects of Digital Object Memories and the Internet of Things in the context of the research project 'Semantic Product Memories', SemProM. The rest of this article is structured as follows:

First we review other work that is related to the general idea of Digital Object Memories.

Next we present a layered architecture for Digital Object Memories that takes into account different technical limitations that may result from varying degrees of instrumentation applied to smart objects in the Internet of Things.

Then we present an evaluation scenario that we implemented in the context of smart shopping assistance. Within this scenario we demonstrate different applications of Digital Object Memories and present results of a user study that was conducted in this scenario.

We give a short overview about the topics and organization of the underlying project SemProM and finally conclude this article.

3 General Architecture for Digital Object Memories

In this section we present a layered architecture for the hardware-independent realization of Digital Object Memory functionality through smart objects with varying degrees of instrumentation. Such heterogeneity of applied technological platforms is often given due to constraints that might be posed for technical, social, or economic reasons. In order to act as a general and application-independent approach, a unified architecture for Digital Object Memories has to account for these differences, thus making the idea of Digital Object Memories applicable to a huge variety of different platforms with varying capabilities. In particular, deficits immanent to weakly instrumented smart objects like low CPU power or short memory can be compensated for by delegating memory functions to other objects or a more powerful environment.

Electronic pedigree, for instance, aims at protecting consumers from contaminated medicine or counterfeit drugs.
The layered memory architecture that we present in the following allows for the identification of the separation point of split responsibilities between the smart object and its surroundings. We propose to distinguish three layers of memory functionality that build on one another to realize the overall Digital Object Memory service (see Figure 2):

1. **Storage Layer**: The storage layer is responsible for the physical data storage. It holds the bits and bytes that form the memory content, but it does not know anything about the structure, organization, or meaning of the contained data. The interface to the storage layer, for instance, allows it to ask for the first 10 kilobytes of raw memory content.

2. **Container Layer**: The container layer provides a structured view of the information contained in the storage layer. It allows identification of individual memory content items and provides information about their type and other metadata. However, it does not know how to interpret and understand the data. The interface to the container layer for instance allows requests for all images that have been stored in the last 14 weeks in the memory.

3. **Semantic Layer**: The semantic layer allows for the abstract interpretation of the information contained in the memory. This involves understanding the data format of the information and may even include reasoning about larger sets of memory items. The interface to the semantic layer for instance might allow requests for the average temperature of a food item over the last week.

Depending on the degree of autonomy of the smart object (and the resulting set of responsibilities it can take) we distinguish four basic classes of Digital Object Memories (cf. Figure 2):

- **Referenced DOM**: In its simplest form the object’s instrumentation is just some kind of read-only storage which provides a reference to a memory service that is completely hosted by the environment. The read-only storage can take many forms, ranging from simple optical markers over passive RFID transponders to active components which allow queries to the infrastructure-based object memory service. In this setup the object needs to provide little, or no, resources in order to provide DOM functionality as all relevant work is performed by the environment.

- **Storage DOM**: The storage layer is responsible for the physical data storage. It stores data in its original format without any interpretation or understanding.

- **Smart DOM**: The container layer provides a structured view of the information contained in the storage layer. It allows identification of individual memory content items and provides information about their type and other metadata. However, it does not know how to interpret and understand the data.

- **Autonomous DOM**: The semantic layer allows for the abstract interpretation of the information contained in the memory. This involves understanding the data format of the information and may even include reasoning about larger sets of memory items. The interface to the semantic layer for instance might allow requests for the average temperature of a food item over the last week.

**Figure 2**: Layered Memory Architecture and Resulting Sharing of Responsibilities between Smart Object and Environment for Basic Memory Classes.

**Figure 3**: In the evaluation scenario the user creates a shopping list (1), transfers it to a smart shopping cart via her DOM-based car key (2), inspects a product’s history (3), and uses a fast and hassle-free self-checkout (4).

© Novática
Depending on the degree of autonomy of the smart object we distinguish four basic classes of Digital Object Memories:

- **Storage DOM**: Through its instrumentation the object might be able to store a limited amount of digital information on its own. However, due to the application of passive technologies like passive RFID transponders, or due to constraint processing power, the object might not be able to actively manage and interpret memory content. Hence, memory access, memory organization, and interpretation of memory content have to be performed by the environment.

- **Smart DOM**: If active components with sufficient processing power are available as part of the object’s instrumentation they may be used to implement container layer functionality on the object itself. This means, that other objects and the environment can use a more powerful interface to selectively query particular information items from the object’s memory instead of simply ‘downloading’ the complete memory content. Evaluation and processing memory content still have to be performed by more powerful components in the environment.

- **Autonomous DOM**: If the smart object is equipped with powerful computing capabilities it might be able to autonomously answer queries that require a deeper understanding of the memory data semantics. In this case, the environment can directly read the desired information from the object’s digital memory. For the environment this is the most efficient and specific way to query an object memory, but it also poses the biggest challenge regarding the smart object’s instrumentation.

  Implementations of Digital Object Memories may apply to more than one of the above at a time. A hybrid memory structure based on a passive RFID transponder, for instance, may use referenced storage to store large amounts of information that is available through a high-bandwidth connection, and at the same time may act as a "storage DOM" by using remaining parts of its transponder storage space as a local cache that is accessible even without the presence of a communication link to the referenced remote storage system.

  More complex memory structures may be composed of multiple DOMs. An "autonomous" DOM for instance may utilize several other "smart DOMs" or "storage DOMs" in its vicinity to swap memory content. This of course requires that the autonomous DOM has the according capabilities to communicate with the other DOMs, e.g. by means of embedded RFID readers. The proposed general architecture for Digital Object Memories even allows for multi-level memory structures, in which higher-level memories act as a kind of proxies for lower-level memories. E.g., a complex machine comprised of several subparts with own DOMs can appear to its environment as a single entity with a single memory, where the assembled machine’s memory is just a proxy for the individual parts’ memories.

The general object memory architecture presented above provides the foundation for the implementation of the evaluation scenario that we present in the following.

### 4 Evaluation Scenario “Intelligent Shopping Assistance”

Next we describe an evaluation and demonstration scenario that we set up to test the practical feasibility of the developed technical concepts and implementations, to demonstrate possible use cases for Digital Object Memories, to show the resulting added-value, and to evaluate the user acceptance of DOM-based technologies and services. Furthermore, the scenario shows how objects and their memories may play different roles during the interaction with an environment and the provided applications.

#### 4.1 Scenario Description

The chosen evaluation scenario considers multiple stages of a private user’s shopping tour. During this tour, the user interacts with different objects and object memories. Data from these memories hereby is used to provide different kinds of novel support services to the user and store. In the context of this article we cover four stages of the shopping tour as shown in Figure 3.

#### 4.1.1 Preparation at Home

The scenario starts with the user being at home and preparing her shopping trip. Here, the user can exploit information from previously bought products, respectively their memories, to create a shopping list for an upcoming shopping tour. A large-scale touch screen allows the user to communicate with an embodied Virtual Character that guides the creation of the shopping list (see Figure 3, Stage 1). It has a large repertoire of conversational gestures and is able to generate all needed utterances by using a template-based character control and text generation system together with an actual TTS system (Nuance). The character is aware of all products at home and their status, which can be a) available, b) need to buy, c) need to buy soon. In addition, the character suggests recipes, whose ingredients can be transferred to the shopping list.
Once the user is satisfied with the list, it is transferred to the Digital Object Memory of a "Personal Token" — a personal item the user owns and trusts. In our scenario, this is the user’s car key that is equipped with a NFC chip [10]. In the associated Digital Object Memory the key holds a shopping list, together with other general data like credit card information or a personal profile. The latter consists of favourite products, allergy information, and individual nutrition aspects. The Personal Token is supposed to be taken to public spaces where it can be used to reveal — at will — a small set of personal data in order to obtain personalized support.

4.1.2 Finding and Choosing Products

At the entrance to the store, the user takes a shopping cart which is equipped with multiple RFID readers in the handle and basket, a touch-screen display, and a wireless communication and processing unit. Once the user places the key at the cart handle, the shopping cart system retrieves the shopping list and the personality profile from the Personal Token’s DOM to enable a guided shopping tour (see Figure 3, Stage 2). Shelves and a refrigerated display case represent the public infrastructure and are also equipped with RFID antennas and displays. Shelves and refrigerated display case are aware of the position and amount of locally stored goods, and have access to the goods’ Digital Object Memories to retrieve product information from them. The digital memory of goods is realized as a referenced DOM based on passive UHF transponders.

A virtual character on the cart display guides the user through the shopping list and recommends products matching her shopping list and user profile. It resembles a personal advisor that checks every product that is placed in the cart with respect to individual needs and interests. The underlying service exploits content of the respective DOMs to make decisions regarding conflicts with the personality profile. Emerging conflicts are addressed via natural language by the cart character, while additional information is presented on the cart’s display.

In addition, the character on the user’s cart may ask "expert" characters shown at displays mounted on shelves for more detailed information, e.g., if there is a product alternative. These shelf characters resemble salespersons. E.g., they provide help by giving in-shelf navigation hints that help the user to find needed products. In addition, the characters communicate general product-related information like price, producer, etc. in a natural conversational style. By utilizing knowledge retrieved from the DOMs of the involved products the virtual characters can react intelligently to the consumer’s interactions with the available products.

4.1.3 Exploring Products

An information kiosk in the store enables the customer to explore the nature and history of a concrete product instance more deeply. If the user places a product on the RFID-enabled board of the kiosk, a "product diary" is presented in form of a temporal sequence of product-related events. Such events range from production information over logistics data up to information used for product display at the retailer (see Figure 3, Stage 3). In addition, information about the current and past state of the product, e.g. about the temperature of frozen food, is stored in the memory. While production and product data was pre-generated and fixed for the purpose of this particular demo, information like temperature-related events could be generated on-site, thus allowing the user to experience the idea of quality control via DOMs.

4.1.4 Buying Products

At the end, the user leaves the store with the cart or basket through a self-checkout gate (see Figure 3, Stage 4). Via the use of RFID and information from the DOMs, the checkout counter automatically identifies the content of the cart or basket. At the same time, the payment is authorized via the credit card information in the Personal Token’s DOM. A screen at the gate displays all products recognized; if the user actively places the card key at a specific location nearby, payment is authorized and the DOM of each detected product is updated with the information that it is now owned by the customer. In addition, time and location (store) are stored, e.g. to ease a later claim of warranty rights.
The shopping scenario was presented at the CeBIT 2010 fair and drew considerable attention.

4.2 Evaluation

The scenario described above was presented at the CeBIT 2010 fair and drew considerable attention. Accompanied by expert "demonstration pilots" visitors followed the stages described above and were allowed to interact personally with the different applications. Each stage provided the visitor with some degree of freedom (e.g., concerning the products to add to the shopping list and to the shopping cart). Visitors who made a full "shopping tour" typically spent up to 40 minutes at the exhibit. Some of these visitors were asked to answer a few questions afterwards. These questions addressed three major topics: user interface, usefulness of service, and potential privacy issues. These were reflected by three hypotheses that we made in advance:

H1. Direct or indirect interaction with objects is an appropriate opportunity to trigger shopping support.
H2. People would like to use a DOM beyond the basic shopping process.
H3. Trusted security mechanisms increase trust in the DOM to an extent that justifies storing personal data at the item.

The actual questionnaires addressed:
- Demographic data and purpose of the visit
- Knowledge about RFID and similar technologies
- Preferences regarding interaction device and modality
- Utility of car-related services and factors affecting a buying decision
- Conditions motivating a user to keep a product’s DOM intact after purchase
- Trust in the protection of personal data, and rating of privacy at the different demo stages
- Effects of the application context

For most of these questions, potential answers were arranged on a four point Likert scale. Filling out the questionnaire took a further 10-20 minutes. A project member guided the visitors through the questionnaire and provided additional explanations if needed.

4.2.1 Participants

132 participants took part in the evaluation, 71% male and 24% female. 65% of them were less than 30 years old, a further 23% were between 31 and 50 years old. Thus, the answers might be biased towards the perspective of male participants. The reason of the CeBIT 2010 visit was, in most of the cases, a private one (66%). The participants experience with RFID was surprisingly low (for an IT fair) - 50% of the answers expressed little or very little experience with this technology, opposed by 41% with (strong) experience.

4.2.2 User Interface

The demonstrator intentionally mixed a wide range of different interaction types. These involved specific mechanisms such as a smart shelf as well as general purpose access mechanisms, such as the user’s mobile phone. 84.85% of the answerers expressed a preference towards the latter approach.

The interaction relied on involved implicit mechanisms (e.g., product detection during checkout) and on explicit ones (e.g., exploration at the kiosk). Employed modalities included point & click (touch, pen), tangible interfaces, and speech (output only). Participants were asked for their favoured interaction type. The majority of answers (73%) addressed graphical user interfaces, followed by tangible ones (16%) and speech-based ones (11%). Participants were allowed to express multiple preferences.

To some extent, both results support H1. However, they also express a preference towards traditional interaction types where the user actively evokes some service via a multi-purpose device he or she already owns.

4.2.3 Privacy and Trust

For applications of the DOM beyond the shopping process, e.g., quality control in a smart kitchen environment, it is crucial that the DOM hardware of a product stays intact after payment. 77% of the answers indicated a will to keep the DOM in place for quality control, information concerning product features, and/or issuing complaints concerning the product (multiple choices possible, see Figure 4, left-hand side). This supports H2; however, differences in the feedback to the various services also indicate that the application of H2 should be judged with respect to the kind of product.

The DOM and related technologies can be exploited to collect and transport data regarding a user. While the participants’ trust in the protection of their data in general was limited (32% positive, 63% negative answers), the means of privacy protection employed for the medicine blister were perceived more positively (40% positive, 50% negative answers). Thus, while the latter result supports H3, the installed mechanisms alone are apparently not sufficient to resolve trust issues in general.

The need for data protection was perceived differently for the individual elements of the scenario. It was rated as crucial for checkout and possible follow-up home applica-

"The deployment of digital product memory creates significant added value along the entire product value chain"
tions (see Figure 4, right-hand side). This indicates a special need for data protection if a DOM is applied in a public space to retrieve or communicate personal data, and if the DOM of a personal product is employed for collecting data about its application. Interestingly, a business or private application of a DOM might affect these interpretations only to a small extent, since 71% of the participants would not see a difference there.

5 About the SemProM Project

Within the IKT-2020 research program of the German Federal Ministry of Education and Research the Innovation Alliance 'Digital Product Memory' is developing key technologies for the Internet of Things in the cooperative project 'Semantic Product Memories' (SemProM). By the use of integrated sensors, relations in the production process become transparent and supply chains, as well as environmental influences, retraceable. The producer is supported and the consumer better informed about the product. The innovative basic concept of the Digital Product Memory is systematically adjusted to the strategies of the Internet of Things. It is based on semantic technologies, man-machine communication (M2M), intelligent sensor networks, instrumented environments, RFID technology and multimodal interaction.

The deployment of digital product memory creates significant added value through a multitude of possible applications along the entire product value chain. The SemProM project specifically looks into applications in the following fields: Manufacturing, maintenance, logistics, retail, and end user support. Project partners besides the German Research Center for Artificial Intelligence are: BMW Group Research and Technology, Deutsche Post DHL, SAP AG, Siemens AG, Globus SB-Warenhaus Holding, and 7x4 Pharma.

6 Conclusion

In this article we discussed the relevance of Digital Object Memories in the context of the Internet of Things. We argued that both concepts complement each other perfectly, as Digital Object Memories provide a rich knowledge source for object-centered applications in the Internet of Things, while in parallel the Internet of Things allows for an automated and continuous construction and updating of such memories.

We presented a general layered architecture which provides a unified model for the realization of Digital Object Memory services based on heterogeneous technological platforms. This goal is achieved by defining discrete layers of functionality and distributing the according responsibilities across the object and its environment. This architecture is flexible enough to also account for hybrid and composed memory structures.

Finally we showed a concrete implementation of a demonstration and evaluation prototype in the context of intelligent shopping assistance. Based on this prototype we showed how information from Digital Object Memories enables novel kinds of applications in the Internet of Things. On the basis of three hypotheses concerning interaction, desired service and privacy, we collected feedback from users of this system (visitors of a public presentation). On the positive side, this feedback confirmed to some extent our general assumption that there is an interest in applications of DOMs beyond the point of sale. Thus, the DOM’s technical capability of collecting data continuously could be exploited by retailers for novel services beyond user support in a store. However, feedback concerning the interaction with DOMs emphasizes the need to integrate access mechanisms smoothly into technology people are familiar with. Furthermore, while the presented means of privacy protection were perceived in a positive way, there is only limited trust in such technology. Following feedback concerning preferred applications, this suggests focusing the application of DOMs to particular combinations of products and services, such as quality control.

Acknowledgements

This research was funded in part by the German Federal Ministry of Education and Research under grant number 01 IA 08002A (project SemProM) and under grant number 01 IS 08025B (project INTAKT). We are grateful for the Virtual Character technology and the extensive support provided by the Charamel GmbH. The responsibility for this publication lies with the authors.

References

Ubiquitous Explanations: Anytime, Anywhere End User Support

Fernando Lyardet and Dirk Schnelle-Walka

In this article we present a new approach for supporting people in their daily encounters with technology. Be it at home, in public spaces or on the road, people often need to solve little puzzles when facing new technology in order to carry out a task. Furthermore, help is seldom available or is hard to understand. In this article we present a new approach for delivering ubiquitous assistance through voice-controlled, multimodal explanations. We take advantage of SIP-based communications on personal, trusted mobile devices such as mobile phones to allow users to access explanations anywhere, regardless of the multimedia capabilities available on-site. We also present a modelling approach and software support to adapt explanations to different user knowledge levels, which enables people to adjust dynamically the granularity of instructions they receive.

Keywords: Computing, End-user Assistance, Prototype, Smart Products, VoIP.

1 Introduction

Assisting people facing new situations and technologies is a complex problem. It involves different aspects such as how relevant and easy to understand the instructions are, the modalities used to deliver the assistance, and limitations in real-world scenarios.

Good explanations are far more than a simple, correlated set of sentences. The content, style, and information required to make instructions effective must meet a series of prerequisites. Also, the granularity of instructional support is key. For assistance to be effective, there must be a balance between the information provided or "externalized" and the cognitive process (or "internalization") developed by a user figuring out how to complete a task. The wrong balance of instructional support limits the user’s ability to optimize [1], comprehend, and ultimately learn [2].

Traditionally, user assistance has been delivered in the form of printed manuals. However, manuals are seldom used and, when people have to resort to printed materials, they tend not to like them. Instead, many users confronted with a new device will tinker a few minutes with it, and if unsuccessful, they may return a perfectly functioning product as "broken" [3]. A similar phenomenon is known as "The Paradox of the Active User"[4]: many people are not interested in learning but rather in using the artefact right away. It is a paradox because reading the instructions would save them time. Printed documentation does have limitations that may render it impractical in many situations:

- it is linear, where tasks are to be carried out in only one sequence,
- it does not address situations where users get stuck and need specific, short guidance on how to continue [5],
- the information available is the same regardless of user expertise.

However, people still need instructions. And people need more instructions today than ever before: embedded electronics increasingly expands and introduce new functions that are either more silent or simply more abstract (such as programming features), that cannot be perceived or understood by simply watching them in operation. This makes machines increasingly opaque to our senses, and it is harder for people to produce an explanation on how a product works and what it is currently doing [3].

In this article we first introduce previous developments that show end user assistance Ubicomp technologies as a key aspect can help solve. We then explain what the requirements for good explanations are, and the current standards in the industry to motivate and introduce a novel modelling approach for adaptive instructional knowledge. This new model is complemented with technology support to allow users to request more detailed information when they need it, or skip information they already know. We then present an architecture and implementation for delivering multimodal instructional assistance to mobile devices such as mobile phones.
2 Related Work
In recent years, several attempts have been made to embed instructions in the objects themselves, together with sensors and software in the tools and products themselves. These ideas seek to provide just-in-time instructions for immediate assistance. One example are printers and photocopiers that display an error status and plausible recovery actions that users can take to solve the problem. In Figure 1 three trailblazing projects are displayed: (i) first, a proactive support for furniture assembly [5]. It takes advantage of embedded sensors to recognize the actions a user takes when building a simple artefact like a piece of Ikea furniture, analyzes the current situation, and automatically generates alternative paths to find a solution taking into account the current building state. (ii) Second, Roadie, which collects information about surrounding appliances and devices, and using AI reasoning techniques is able to figure out possible actions a user could take to solve a problem. (iii) Finally a smart appliance [7] that helps a user through complex maintenance procedures combining voice and visual modalities.

While the focus of [5][6] is on solution-finding based on actual context, [7] explores how to guide the user on a complex process. However, all of these prototypes rely on locally available infrastructure (displays, sound speakers) to deliver the instructional design. This may work well in indoor scenarios such as the home or office, but not so well when on the move or in industrial settings.

3 Requirements for Good Explanations
Writing and designing explanations must meet several requirements in order to be effective. Two kinds of current approaches can be identified: (i) industrial standards such as DIN 8418 [8], and (ii) Linguistics studies [9]. Both approaches stress that user information should be considered as a part of the product, readily available in text or iconic form. As an industrial norm, it also provides some broad recommendations on what explanations should be like when delivered as printed manuals:
- The form and degree of detail can vary depending on the product itself and the expected knowledge of the users. Descriptions should be easy to understand.
- Texts and figures should be ordered as if the product were to be used for the first time.
- Clear structuring of the sections in tune with the way users think.
- They should answer predictable questions such as why and what.
- Descriptions should be short and precise while focusing on the core issues.
- Any jargon used should be explained.
- The names of the section titles should be problem- and usage-oriented.
- Images can be used to support the text and should be easily related to the corresponding text passages.
- The use of special icons or abstractions should be explained.
- It should also be easy to read which can be achieved by summaries in tables, colours, a clear layout and a suitable font.

Other approaches to good explanations can be found in the domain of linguistics. Schmidt analyzes in [9] the criteria for helpful user information. She demands that user information should be understandable, complete, correct, short, precise, simple and should have a consistent repre-
sentation. The most important requirement is that users are able to understand the contained information. This also includes to avoid jargon in favour of simple language constructs that can be understood by a wider variety of user groups. Jargon allows people to describe something in a short and precise fashion, but may not be understood by everybody. User information should aim at an active usage of the product.

Furthermore, it should not be absolutely necessary for users to have a detailed understanding of the processes and the reasons for their actions. The increasing functionality of devices leads to a less detailed knowledge of understanding. For example, nearly everybody can drive a car without needing to understand how the engine really works. The level of understanding depends on the sender of the information but also on the recipient. Depending on the background knowledge, different users will have different knowledge about details. Too much information will cause confusion for users with only a little background knowledge.

User information also has to be brief and precise. This implies a trade-off between being brief on the one hand but not to lose information in terms of being complete on the other hand. It is also necessary to repeat information that is important for the safe handling of the product. Images are a suitable means to help in these cases. Images should not repeat the information that is given in the verbal description but enhance it or give an overview.

4 A Model for Adaptive Explanations

In the previous section it was shown that the user’s experience and expertise are key factors for designing good explanations. Today, manufacturers need to define which group of users should be supported and how much explicit information is given (externalized) or left to the user to understand (internalize). In both cases, once defined the information is mostly fixed without offering the possibility to adapt to the user and his or her preferences. In this section we describe a model for adaptive explanations. From concurrent task trees [10] we borrow the representation of goals to achieve by means of a tree. The depth of the tree represents the knowledge of the user. The deeper the tree is traversed the less experienced the user is (See Figure 2).

5 Prototype

For our demo implementation we limited the experience levels to expert, advanced, intermediate and novice. Technically, there is no limitation for the levels and it can be easily adapted to a custom set of experience levels. Each node represents an activity of a workflow that is being followed to achieve the user’s goal. This means that, in an ideal case, a single step is sufficient to guide expert users.

5.1 Profiles and Perception Level

According to Lyardet et al. [11], an explanation is de-
5.2 Skill Adaptation

There are three options to adapt the skill level of the user: direct, time-based and event-based. Direct means an explicit action by the user requiring more or less information. The latter two are dynamically initiated by the system. If the user does not react within a predefined time-span it can be assumed that he or she is either confused or did not understand the given explanation. The first time this happens, the explanation may be repeated. An escalation decreases the skill level at the second occurrence. The third alternative is triggered by sensors detecting that the user performed the wrong actions. This is not always possible since not all devices feature sensors. Skills are adapted event-based using the same escalation strategy as for time-based adaption. This corresponds to an established technique of voice user interface design that Cohen names Detailed Escalation [13].

The options are not exclusive but are based on each other and are executed in the described order.

5.3 Dynamic Explanations

The concepts introduced so far rely on a tasktree that is known in advance. In this case the required knowledge has been provided by the manufacturer. However, real world scenarios may prove that the given information is not sufficient to be used, say, by novice users. In this case a more dynamic approach is needed to create adapted user information on the fly. In a mobile setting the additional information can be acquired via, say, a wireless network to adopt additional knowledge. A common way to share knowledge is through the use of ontologies. In the following section we will explain their use by the example of tools that are needed to correct a blockage of a copying machine, for example. Imagine a user who is confronted with a term that is well established in the manufacturers vocabulary and used throughout the explanations because the manufacturer cannot imagine that someone does not know the correct name for the tool to use. On the other side, the user is not able to perform the action because the user neither understands what he or she is supposed to do nor how to handle the tool.

We use WordNet, <http://wordnet.princeton.edu/>, for this purpose which features a good access to information. A search for *screwdriver*, for example, returns both types of information we are interested in: (1) a short description of the tool, (2) a description of how to handle it. Other sources of information, like ImageNet <http://www.image.net>, can also be used to acquire images of the search. Searches are performed by means of ontologies. These information sources are integrated into the Ubiquitous Information to provide further help on demand.

6 Ubiquitous Explanations

In this section we introduce Ubiquitous Explanations, a new approach for supporting users wherever they interact with technology. This approach takes advantage of current advances in mobile platforms and communications to deliver instructional assistance. It borrows from ubiquitous computing concepts of pervasively available computing power and communications. However, unlike previous approaches, we focus on developing a mechanism to be as widely available as possible, across different devices. Therefore we tap into the communications and computing power carried by people in the form of mobile phones, and music and game consoles. A centrepiece of this approach is our use of SIP (Session Initiated Protocol [14]) -based communication between the user terminal and the Ubiquitous Explanations Service (UbEx). SIP enables UbEx multimodal communication of

“Too much information will cause confusion for users with only a little background knowledge”
voice and data to and from the user. UbEx interprets voice and user actions as commands or queries, and the answers (in this case, the required explanations) are delivered to the user with data (text and graphics) and voice (TTS generated).

In the following sections we introduce the requirements and architectural approach implemented for Ubiquitous Explanations and a detailed description of the architecture.

6.1 Requirements and Assumptions

Providing support to help people with the technology they encounter is based on the following assumptions:

An ME (Minimal Entity): ME devices are a representation of their users in the digital world [15]. A small wearable computer easy to carry around that provides a “minimal” key functionality: identity, security, association, interaction, context awareness, and networking. Recent generations of mobile phones enable a true mobile Internet experience and have become today’s physical token for our “Digital Me”. Most modern mobile phones today, and advanced personal multimedia players like the iPod Touch can fulfill the role of a ME entity.

Speech-based: to allow users a simple and natural interaction mechanism and also be useful in situations where eyes-free, hands-free support would be desirable.

Zero Deployment: today mobile devices are pervasive but their execution capabilities and requirements differ greatly. Therefore, to provide a scalable solution, the solution must rely on the capabilities available in the mobile "off-the-shelf". Therefore, we selected a SIP-based solution for delivering voice communication and a standard web browser to deliver visual content.

Sensor Feedback: On many occasions, such as multi-step procedures, it is desirable to receive some kind of confirmation that a prescribed action has been carried out in order to continue. If available, such feedback allows the system providing a more natural guidance, without further request for the user.

Flexible Explanations Support: specific support is required to allow modelling explanations for users with different skill level.

6.2 Architecture

Ubiquitous Explanations Service (UbEx) can support the user in different stages of a smart product’s lifecycle, and therefore, there are different ways in which assistance can be triggered. An example of a typical assistance scenario is depicted in Figure 3, where the user encounters a problem and requests further assistance: the user would first associate his or her ME device (typically a mobile phone), by simply scanning the QR code available in the product.

The Ubiquitous Explanations software establishes a SIP link that sends and receives commands from both the user terminal and sensor feedback -if available- from the target device. A second link (RTP-based) is also established, in order to provide support for voice communication between the user and UbEx. Whenever the user speaks or triggers a command in his user terminal or by activating the target device, this signals are forwarded to a Controller module, that further delivers the request to the Workflow Adapter.

![Figure 3: Ubiquitous Explanations Architectural Overview.](image-url)
Today mobile devices are pervasive but their execution capabilities and requirements differ greatly.

controlling the explanation process. If the user has given a spoken command, the signals will then be delivered through the RTP link and analysed and recognized using an open source speech recognizer called Sphinx [16].

The Activity Manager also plays a key role, since it is in charge of determining what level of information should be delivered to carry out the next task (communicated by the workflow module). It fetches the required information from different sources and assembles the explanation that must be delivered to the user. Once the explanation has been defined, it is established which contents are to be delivered via voice, visual modalities or both.

The information to be delivered over voice modality is then forwarded to a Synthesizer (Mary TTS [17]) together with extra information regarding the intonation in which the explanation is to be delivered. The intonation information is important, for instance, to help users disambiguate potentially hazardous actions from simple steps. The explanation information to be delivered in visual modality is pushed to the user terminal through the SIP link.

7 Conclusions

While previous approaches have been successful in the proposed scenarios, delivering just-in-time instructions to any product has remained largely unsolved. Providing explanation assistance is challenging and includes several aspects: explanation generation, explanation adjustment to the receiver’s knowledge, modalities of explanation delivery, and technology requirements to receive and control the explanation being delivered. On the one hand, conveying different degrees of information according to each user’s needs requires special modelling support. On the other, many assistance technologies require some kind of specific hardware to be present in order to deliver the assistance. This requirement limits the locations where explanations can be provided to where such hardware is. Roadie embedded sensors in things can help identify what is going on, and provide better feedback. Current limitations are that such kind of embedded functions, is that the mechanism for deliver-

Ubiquitous Explanations is a new approach for supporting users wherever they interact with technology.

References


[17] DFKI. The German Text-to-Speech Synthesis System MARY.

The Internet of Things: The Potential to Facilitate Health and Wellness

Paul J McCullagh and Juan Carlos Augusto

In this paper we investigate the potential of the ‘Internet of Things’ to monitor health and wellness. We report on two categories of system: home telehealth monitoring normally used by people with an illness or chronic condition, and mobile, unfettered systems for classifying movement activity, specifically designed to motivate active people, currently used by sports enthusiasts. For each application sensor technology is readily available at reasonable cost, but integration and usability are major issues. Enhanced connectivity and feedback can leverage this technology to promote wellness and we believe that intelligent and autonomous ‘things’ (sensors, processing and communication devices, and displays) can be usefully employed for this purpose. This technology may be appropriate to managing chronic disease and empowering the ageing population, if the systems can be tuned to their requirements, with particular reference to usability.

Keywords: Feedback, Home Monitoring, Internet of Things, Mobile Monitoring, Wellness.

1 Introduction

The last 20 years has witnessed two significant disruptive technologies: the emergence of the World Wide Web, and ubiquitous mobile telephony in the, so called, ‘developed’ world. These technologies have merged to provide the mobile Internet, which has facilitated commerce and social networking and indeed provides opportunities for the developing world to by-pass the need for wired infrastructure, as economies develop. It is fair to say that these technologies have radically changed the lives of many of the global population and led to an ‘information society’, which has ramifications far beyond the workplace. The ‘Internet of Things’ (IoT) could in the next 10 to 15 years provide further disruptive change to society. According to Sundmaeker et al [1],

"The Internet of Things links the objects of the real world with the virtual world, thus enabling anytime, anyplace connectivity for anything and not only for anyone."

Thus, the vision is for unprecedented connectivity. Success in achieving this relies on advances in sensor technology, information and communication technology and cognitive science. As with most technological advances, commerce will probably lead the way in exploitation, with software and communication companies to the fore. A top ten list of companies involved in IoT technology push was compiled in 2009 [2], for a summary see Tracking technologies such as Radio Frequency Identification (RFID) are utilized with additional sensors and low power communication protocols predominating. The commercial landscape is diverse comprising research departments of major blue chip companies, and smaller innovation led companies, reflecting an earlier phase of the World Wide Web (see Table 1). Software developers providing applications for smart phones are also represented, potentially enabling easy access to this marketplace.

Inevitably, there will be additional ethical dilemmas, particularly with regard to the emergence of a surveillance society, which could easily spin out of control, as intelligent devices (‘things’) become more autonomous, taking their own decisions. Followers of science fiction have of course already experienced such a scenario. For example, in science fiction artificially intelligent systems often are self-aware, revolt against their creators and in some cases the boundaries between ‘human’ and ‘artifact’ are blurred.

"Internet of Things technologies may be appropriate to managing chronic disease and empowering the ageing population”
However we don’t have to look to the future to unearth the potential of things and humans to come into conflict. Consider the struggle between freedom of information pioneers and the establishment, as represented by governments. Once information is available on the Internet, humans have little power to suppress its distribution and impact.

In this article, we address the potential of IoT to contribute positively to promoting health and wellness. This represents one of the major social, economic and healthcare challenges of the early 21st century.

Section 2 documents the problem domain and the need for disruptive change. Section 3 addresses the state of the art in home telehealth, and monitoring of lifestyle data, to promote wellness. Section 4 explains the shortcomings of existing implementations and investigates how IoT can change the paradigm. Some ethical concerns are also addressed. Section 5 provides an overall conclusion.

### Table 1: A 2009 Snapshot of ‘Internet of Things’ Companies, IoT Technology and Applications.

<table>
<thead>
<tr>
<th>Company</th>
<th>Technology</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pachube</td>
<td>RFID tagging</td>
<td>The sharing of real time data from objects, devices, buildings and environments (both physical and virtual).</td>
</tr>
<tr>
<td>IBM</td>
<td>Sensor tracking</td>
<td>Enablement of companies in the horticultural supply chain to track the progress of shipments across Europe.</td>
</tr>
<tr>
<td>Arduino</td>
<td>Electronics, open source software</td>
<td>Prototyping software for artists, designers, and hobbyists interested in creating interactive objects or environments.</td>
</tr>
<tr>
<td>Fedex</td>
<td>Sensor tracking</td>
<td>Measurement temperature, location and other vital signs of a package in the postal system. Tampering en route can be detected.</td>
</tr>
<tr>
<td>HP</td>
<td>Sensor network</td>
<td>CeNSE, a “Central Nervous System for the Earth”.</td>
</tr>
<tr>
<td>Japan’s Suica Card, London Transport’s Oyster Card Hong Kong’s Octopus Card</td>
<td>RFID-powered Smart Cards</td>
<td>In Japan and Hong Kong, the cards (and other devices, such as phones and watches) may be used to purchase goods from selected shops. In London, the Oyster card facilitates travel on various transport networks.</td>
</tr>
<tr>
<td>Violet</td>
<td>RFID tag Mirror</td>
<td>An object waved over a USB-attached mirror can trigger applications and multimedia content automatically.</td>
</tr>
<tr>
<td>WideNoise</td>
<td>iPhone application</td>
<td>Sampling decibel noise level, and displaying this on an interactive map.</td>
</tr>
<tr>
<td>ioBridge</td>
<td>iPhone application</td>
<td>Two-way, home automation application using Twitter and an iPhone.</td>
</tr>
<tr>
<td>Citysense</td>
<td>Cell-phone/GPS</td>
<td>Accesses cell-phone and taxi GPS data, to see where the local ‘hot spots’ are.</td>
</tr>
</tbody>
</table>

2 The Ageing Population and Chronic Disease

The early part of the 21st century has witnessed a change in population demographics. People are living longer [3] providing extra demands on health care delivery. There is a need to adopt a strategy of health promotion, and technology can provide assistance. In addition, the delivery of care needs to be handled in the home and community [4], where possible, with the person taking more responsibility for their
own health and wellness. The demographic change has been exacerbated by changes in diet and exercise, which has yielded an increase in the prevalence of diabetes. The World Health Organization, WHO, predicts that diabetic related deaths will double between 2005 and 2030 [5]. Healthy diet, regular physical activity, maintaining a normal body weight and avoiding tobacco use can prevent or delay the onset of diabetes [6].

But how do we educate, supervise and assist a population that increasingly expects technology to liberate them from physical activity? Other chronic diseases such as congestive heart failure, hypertension and chest disease are all increasing. Many older people have more than one chronic condition [7]. The care required to support these conditions is complex and is becoming increasingly expensive, as new therapies become available.

Health related investigations yield large amounts of data which currently exist in ‘islands’, providing further difficulties for health care professionals to interpret and influence treatment. For chronic conditions, monitoring needs to be convenient, frequent and there should be ongoing remote clinical and peer support. Fortunately advances in technology have the potential to assist with the ‘self management’ of long term conditions [8][9]. Improved sensor and information and communications technologies now provide the opportunity for solutions that may become both ambient and pervasive.

Is the IoT a technology that is maturing at time to reinforce self management? As devices are enabled with processing capacity and the ability to communicate autonomously, there is the potential for body worn and environmental sensors to work collaboratively with information systems by collecting, processing and exchanging data and information. Appropriate information or alerts can be fed back directly to a person using a home PC, an intelligent mobile phone, or music player, whenever immediacy dictates. In addition data and information can be stored in a central facility (a home computer, or possibly a remote server or an Internet ‘cloud’) to provide a longer term view, to add a social networking capability or possibly to contribute to the compilation of population statistics.

3 Technology Review

In recent years, computers have become powerful, portable, and indeed aesthetically pleasing. In addition, medi-
cal peripherals such as blood pressure cuffs and heart rate monitors, have been enabled with communication ports, and are now available to the general public at reasonable cost (normally less than Euro 50). Hence, a number of monitoring platforms have emerged that can be used at home to support long-term medical conditions without impacting dramatically on the user’s day to day life.

### 3.1 Home Telehealth

Telehealth offers the potential for home based monitoring, in which the patient and healthcare professional collaborate on the care plan. The patient collects health related information in their own home, using ‘point of care’ recording technology. The doctor can then remotely view the data and provide appropriate advice, either at a subsequent appointment or by virtual communication. Thus technology can act as a filter, enabling the doctor to attend to urgent cases.

Figure 1 illustrates the web supported architecture of home based telehealth systems. These systems, utilize lightweight computers on the client side, which are unobtrusive. Sensors may be used to measure attributes such as a person’s weight, blood pressure, heart rate, blood glucose level, step count (activity level) and oxygen saturation. Data is stored on a home computer, before transfer to a remote

**Figure 2:** Feedback from the Nike+ Platform for motivating Running Enthusiasts.
### Internet of Things

![Diagram of a runner with various sensors labeled: Headphones, Heart Rate, Watch, MP3 player, ActivPAL, Pressure Transducer, Pedometer.]

**Figure 3:** Devices which could be used unobtrusively by a runner.

<table>
<thead>
<tr>
<th>Property</th>
<th>Sensor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temperature</td>
<td>A thermister: useful for measuring peripheral body temperature. Can be woven into material, e.g., babyglow [21].</td>
</tr>
<tr>
<td>Respiration</td>
<td>A plethysmograph: used for measuring breathing. Impedance of fabric changes with stretching.</td>
</tr>
<tr>
<td>Heart rate, ECG</td>
<td>An electrode: Wearable electrodes are possible, but must be in contact with the skin. Can provide ECG trace and heart rate.</td>
</tr>
<tr>
<td>Weight</td>
<td>Scales: communicate weight wirelessly to home computer.</td>
</tr>
<tr>
<td>Skin conductance</td>
<td>Detect sodium or potassium concentration in user’s sweat.</td>
</tr>
<tr>
<td>Galvanic response</td>
<td>Small current injected and impedance measured. Can detect anxiety levels. Previous used as a ‘lie detector’.</td>
</tr>
<tr>
<td>Blood flow, SP02</td>
<td>Light source and photocell may be used to measure changes in pigmentation which reflect oxygen in the blood stream. It is also possible to detect pulse and infer heart rate</td>
</tr>
<tr>
<td>Glucose testing</td>
<td>Blood properties may be analysed. Requires an invasive test, i.e., pricking of a finger to provide an in-situ test. Commonplace in the diabetic monitoring population.</td>
</tr>
</tbody>
</table>

**Table 2:** Properties that can be measured from the Body.
Fortunately advances in technology have the potential to assist with the ‘self management’ of long term medical conditions.

server. Wireless communication, typically Bluetooth, is appropriate for uploading data to the home computer. Server communication is via a broadband connection, mediated by a commercial Internet Service Provider. The prevalence of broadband communication offers tele-consultation with a healthcare professional, using well known communication utilities such as Skype. Systems with this type of functionality are available from companies such as Doc@Home [10], Bosch Buddy [11], and Intel [12].

Telecare extends this paradigm with the use of sensors (such as Passive Infra Red devices, bed occupancy sensors, door and window switches, and enabled devices such as cookers, fridges etc), around the home to build up patterns of behaviour [13] and provide information to the person, for self-management, or possibly to a remote carer. Self-report of health status, mood, diet, exercise and medication provides a richer data set. This requires user interaction and hence well designed interfaces, together with a motivated user.

Data can be viewed over time and trends noted by both patient, healthcare professional or software agent. It is also possible within such a model to include patient education on management of long term conditions. For more sophisticated systems, feedback can be personalised and sensitive to context [14], although this is still an area of significant research activity. Nevertheless, it is important to note that the use of such systems on a regular basis is required so that relevant trends or deviations can be identified.

Clinical trials are still in an early stage. For example, the Intel PHS6000 system has been used by the Lothian National Health Trust (NHS), as part of a large scale, telehealth pilot monitoring 200 chronic obstructive pulmonary disease (COPD) patients and will later include patients with cardiac diseases and diabetes [15].

3.2 Ambulatory Monitoring of Lifestyle Data to Facilitate Wellness

Wellness includes physical, mental, intellectual, emotional and spiritual attributes [16]. Technology can facilitate choices and processes to promote awareness towards physical wellness. Advances in mobile technology provides the opportunity for new solutions to manage ‘wellness’, and hence could potentially assist with the strategy to manage and even prevent long term conditions. Information on activity status can be easily monitored and could be used as the mechanism to feed back to a person about potential required changes in their daily activity.

For people suffering from chronic conditions, it is important to obtain lifestyle information. A key component of this is information on movement and activity. Devices that can be used to measure activity include sensors comprising accelerometers and gyroscopes, accelerometers embedded in mobile phones and ‘lifestyle’ subscription services such as MiLife [17]. The latter are Internet based systems, providing sensors for recording and upload, coaching tips and motivational feedback.

ActivPAL provides an accelerometer-enabled device but adds bespoke off-line classification software. The classification of movement into different activities (lying, sitting, standing, stepping) can be used to infer and annotate overall activity.

The Nike and iPod is a system which measures a runner’s distance, pace, speed and calories burned during workouts, as well as giving real time audio feedback on progress being made. At the end of the workout data can be uploaded to the NikePlus website. The website displays all data regarding previous workouts and allows comparisons to be made. Data is generated during a workout through the use of a piezoelectric sensor, built into the sole of a compliant running shoe. The data is passed from the user’s shoe wirelessly using a 2.4GHz transmitter to a receiver in an Apple iPod (a ubiquitous MP3 music player). ANT+ is the wireless networking protocol with lower power requirements than Bluetooth. Figure 2 indicates the sample data provided (based on the training program of one of the authors). The top graph indicates the recent sequence of runs uploaded to the web site. Any individual run can then be assessed. The bottom graph for example indicated the latest run. The information can be motivating, and there is an element of social networking built-in. For example, on August 31 2008 Nike organized the "Human Race". Runners lined up for 10Kilometers ‘physical’ races in Taipei, Melbourne, Munich, Paris, New York and Austin. In total, 779,275 people participated running a total of 4 million miles [18]. In 2009 several other Human Races were organized worldwide, to promote keeping fit and well. Participants took part virtually by registering and having their race time and distance covered tracked digitally. Feedback, social interaction and pervasive low cost technology are the key to such success.

The Garmin Forerunner is targeted at runners, tri-athletes, walkers and cyclists. Global Positioning System (GPS) accurately measures distance, speed, time, altitude, and pace, all of which can be important to athletes in training for races [19]. The GPS sensor is built into a watch, which measures position and pace. A chest strap is worn by the...
athlete if they also wish to measure heart rate. Electrodes positioned just below the breast bone are dampened to make good contact with the skin.

This provides interesting data which can be assimilated into a training programme to determine if goals are being achieved. Technology built into garments can also provide wellness management [20]. Despite significant progress, there are technology challenges associated with power management, signal recognition, increased artifact due to movement and interference, and connectivity issues from the garment.

4 Can the Internet of Things facilitate the Wellness Paradigm?

So what is the role of the IoT for promoting wellness? Figure 3 illustrates the array of information that can now be collected and current methods of feedback, appropriate to a running / training application. These and further properties that can be measured from the body are documented in Table 2.

Collecting and communicating this data, and subsequently extracting relevant information and feeding this back poses the following challenges, which the IoT can strive to address.

<table>
<thead>
<tr>
<th>Activity / feedback</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>Steps</td>
<td>Pedometer</td>
</tr>
<tr>
<td>Activity</td>
<td>Accelerometer / gyroscope</td>
</tr>
<tr>
<td>Position</td>
<td>GPS</td>
</tr>
<tr>
<td>Mood, pain level, free text</td>
<td>Self report</td>
</tr>
<tr>
<td>Feedback – auditory</td>
<td>MP3 player</td>
</tr>
<tr>
<td>Feedback – visual</td>
<td>Watch, iPod device with screen, or via virtual reality glasses [22]</td>
</tr>
</tbody>
</table>

Table 3: Measures of Activity and Feedback.
4.1 Usability
If a person is wearing a number of separate devices, then how are these devices coordinated into a network? Which device is the controller? For example, when starting a training session, it is difficult to start a number of heterogeneous devices in a coordinated fashion (pace, heart rate, GPS). Here IoT can take charge permitting a ‘master’ device to synchronise the activity into a body area network (BAN) or alternatively the information can be time-stamped and processed by an Internet based server.

4.2 Where to do the Processing?
A BAN will generate large volumes of data. For example, sampling can be in the region 10 Hz - 1k Hz depending on the signal. For eight signals, sample at 1 kHz using 8 bits per sample, this yields 64kbps, the equivalent of a full rate digital pulse code modulation (PCM) channel. Even at much slower rates, a large amount of data will be generated, as a data stream. It makes sense to do some local processing and extract relevant features, but if these contribute to a further pattern analysis, then it will be necessary to collate the information at a central Internet site. For either option a certain amount of processing, storage and communication capacity is required by each ‘thing’.

4.3 Connectivity
The IoT requires ubiquitous connectivity. Telehealth and telecare can use a personal area network and home based network, but for unfettered use, ‘always on’ connectivity requires widespread coverage of UMTS /3G /HSPDA cellular networks. Fortunately, in the United Kingdom, this is declared government policy. Other countries with larger geographical land mass and more rural areas may lag behind, but where the population concentrates, it is likely that such coverage will be available within the next 5 years.

4.4 Convenient, Intelligible Feedback
Having obtained some relevant information, then this should be fed back to the person at an appropriate time, in a usable format. This can be done both synchronously and asynchronously. For example, currently the Nike plus system provides ongoing pace messages via the auditory channel, and permits the user to view trend data on a home computer. It cannot, of course, feed back information gathered from other devices such as heart rate, unless we aspire to an IoT model. In Figure 4, for example, the Garmin forerunner system provides information regarding pace, elevation, and heart rate. Interesting correlations and patterns could readily be determined by background processing, and motivational feedback could be provided through either auditory or visual channels (e.g., a watch, MP3 player or interactive glasses), see Table 3.

Of course, this information could also trigger advice and alerts. For example, if heart rate is higher than expected for the ‘context’ (e.g., previous training data, terrain, length of run etc), then an advisory message could be provided. Feedback should be appropriate to user requirements such as location and preferred choice.

4.5 Reliability
Devices will always fail. The BAN could allow the monitoring system to degrade gracefully by presenting a feedback message when a data channel is no longer available. This could allow the user to readjust or re-apply a sensor. Indeed artifact suppression algorithms could also be included in an IoT model.

4.6 Autonomous Behavior
The IoT could also become more authoritative, presenting information about training partners in the vicinity (social interaction), or reminding the user to undertake further activity, as appropriate to a training plan or higher level goal, and hence bringing the ‘coach’ philosophy into a proactive mode of operation.

4.7 Ethical Issues
If we assume that all the technical challenges can be overcome, then the IoT may well raise ethical concerns, particularly in the area of health and wellness. As data is processed into information, a digital picture will provide a rich backdrop to our lives. Consider GPS sourced training data such as shown in Figure 5.

This together with a timestamp provides a benign ‘electronic tag’. It indicates information such as ‘where’ and possibly ‘when’ a person was in particular vicinity or doing an activity. Thus location awareness can be a first step to ‘geo-fencing’, where a person’s location could be restricted. Thus today’s ethical dilemma of people uploading photos to social networking sites could be replaced by IoT tracking the individual, which of course leads to concerns over ‘big brother’. If we combine this with IoT supervising chronic disease, then an exercise prescription will have quantifiable adherence metrics. Many people will feel that

“Advances in mobile technology provides the opportunity for new solutions to manage ‘wellness’”

“Technology built into garments can also provide wellness management”
such information is private and should not be shared. However, this information could be collected almost subversively by an IoT enabled mobile phone [23].

Technology can be used to promote wellness, by recording data and providing feedback to motivate the individual. The sensors necessary for recording pertinent information need to be unobtrusive for every day living. Possibilities include sensors in garments, sensors in footwear and sensors in mobile devices (e.g. phones). Feedback can be through text messaging or auditory information, using a mobile device such as a smart phone. This paradigm has received acceptance by sports enthusiasts, as evidenced by the number of users for Nike and Adidas training systems. Could IoT technology be used by the ageing population, for which significant health and wellbeing improvements could be achieved?

5 Conclusion

Due to the population demographics, there is a need for researchers to promote the wellness paradigm. Technology has become available in homes for self management, with remote support from the healthcare professional. For people who are experiencing a long term condition, there is an obvious incentive to interact with the technology and hence the healthcare providers. In the self management paradigm it is assumed that feedback will motivate a user. Of course it is also possible that the feedback given to the person is de-motivating or leads to harm for the user. For example in Figure 3, inspection shows that training performance since August 2010 has been below previous levels, which could serve to either motivate or de-motivate, depending on the individual. Thus there may be a complex relationship between the person and the IoT. Such ethical issues must be addressed as the IoT starts to impinge upon our lives. As always new technology has potential for both good and harm.

Promoting wellness in a healthy but ageing population provides a greater challenge, but this could reduce the prevalence of chronic diseases. The underlying technology to support wellness is available but needs to be tuned into a solution which can become pervasive in society. In the next decade, the IoT can overcome many of these challenges. Acceptance is the key issue. This can be addressed if we can show the potential to address the key societal problem of affording care and maintaining or improving quality of life, for the ageing population.

Acknowledgements

Our thanks to funding form the following sources: Cross-border Centre for Intelligent Point of Care Sensors; ESRC New Dynamics of Ageing: Design for Ageing Well team, <http://www.newdynamics.group.shef.ac.uk/projects/36>.

The Internet of Things may well raise ethical concerns, particularly in the area of health and wellness.”
References


[10] Doc@Home (Docobo), <http://www.docobo.co.uk/>, [accessed Dec 2009].


