OSMOSIS - Semantic Work-Spaces for Smart Environments

Luís Veiga Paulo Ferreira

INESC-ID/IST
Rua Alves Redol N^o 9, 1000-029 Lisboa, Portugal
{luis.veiga, paulo.ferreira}@inesc-id.pt

Abstract

Recent advances in device miniaturization have clearly contributed to make Weiser's vision of ubiquitous computing closer to reality. However, such vision is still far from being accomplished because some difficult problems remain to be solved.

It is still hard to develop applications that take advantage of the disappearance of the real/virtual barrier, and to support such applications in a way that can be easily integrated with current environments.

We propose a middleware approach (OSMOSIS) that employs pragmatic approach to solve the previous drawbacks so that ubiquitous computing can become a reality. It provides a file-system abstraction in order to make real objects virtual so that each real-world object has a virtual counterpart in the form of a file. Thus, the topology of a workplace, or a house, is reflected in the directories and subdirectories organization.

The OSMOSIS middleware combines the use of RFID, Location and Context Management and a Policy Engine in order to provide to applications a context-aware file-system enriched with semantic-information.

1. Introduction

Recent advances in device miniaturization (CPU, WiFi, sensors, RFID tags, etc.) have clearly contributed to make Weiser's vision of ubiquitous computing closer to reality [37, 3, 36]. However, the merging of virtual and real worlds, one of the vision's aspects, is still far from being accomplished because some difficult problems remain to be solved.

In particular, it is still hard: i) to develop applications that take advantage of the disappearance of the real/virtual barrier, and ii) to support such applications in a way that can be easily integrated with current environments while ensuring users' privacy. As a matter of fact, current ubiquitous applications are mostly developed with no clear standard application programming interface that can be used by all application programmers, many solutions are cumbersome imposing a heavy burden on the user as is the case with virtual or augmented reality, and many systems are very expensive for the normal user (e.g. requiring many large flat displays, attaching sensors and actuators to all objects, etc.) while exposing many aspects of users' private life [28, 25].

Contribution and Paper Road-map: In OSMOSIS, we propose a pragmatic approach to solve the previous drawbacks so that ubiquitous computing can become a reality right now, at our homes and offices, at a reasonable cost. The idea is to make real objects virtual so that each real-world object has a virtual counterpart in the form of a file.

Thus, the topology of a workplace, or a house, is reflected in the directories and subdirectories organization. Symbolic links can be used to express cases in which a room is reachable from several others. Objects currently in a room appear as files inside the corresponding directory (virtual counterpart of a room).

This view is intuitive to most users, even for those that are not computer experts (the main target of OSMOSIS). As a matter of fact, most users know what a file and a directory is; listing the contents of a directory shows the names (or thumbnails) of the objects currently in the corresponding room.

The rest of the paper is organized as follows. The next Section presents an overview of OSMOSIS, its goals, features and usage scenario. Section 3 presents the OSMOSIS architecture focusing on communication and middleware modules. In Section 4, we briefly describe a number of proposed applications to run on top of OSMOSIS, while Section 5 presents the main implementation issues. Section 6 is dedicated to the vast related work in the fields of ubiquitous computing with context-awareness and use of semantic information. In Section 7, we finish with some conclusions.

2. OSMOSIS

The main objective of OSMOSIS is to design, implement, test and demonstrate a ubiquitous middleware system that can be used at home or at the office by non-computer experts, on a daily basis, providing a large number of functionalities to improve the quality of life for many.

In particular, such a middleware can provide answers and notifications to users concerning real-world objects and situations as those illustrated in previous sections. In addition, the system provides a file-based API that can be easily used by programmers as this is a simple and widely known programming interface. More precisely, OSMOSIS prescribes:

- Insert real objects into the virtual world by attaching them passive RFID tags allowing us to obtain their identification and location.
- Provide a context-aware file system, offering traditional file-system API to develop applications, supporting context-information (e.g. object's location and history) associated to such virtual objects.
- Allow the specification of complex history-based security policies thus ensuring users' privacy concerning their real-world objects.
- Offer a simple (as invisible as possible) user interface so that non-experts users can interact with OSMOSIS applications in a non-disruptive way.

Making real objects virtual, thus creating a file as a counterpart of a real-world object has obvious advantages as it allows extending features and operations available in the virtual world to real objects (even those most "dumb"). We envisage several scenarios in which such an extension is useful as it allow us to answer common everyday questions as well as being notified of certain situations:

- Where is the object that I brought from my last Summer vacations?
- What was the present that Joe gave me on my last birthday?
- Warn me if a child is close to some dangerous object *x*.
- Warn be if object *x* and *y* get close to each other.
- Remember me that I should take object *x* whenever I take object *y*, etc.

These examples reveal common situations that can be handled once we extend common operations performed on virtual objects (i.e. files) to real-world objects, and consider the context information associated with them.

In particular, i) a search operation can be performed on a desktop's disk including real-world objects that have been previously incorporated into the virtual world, ii) it is possible to provide the user with the information concerning

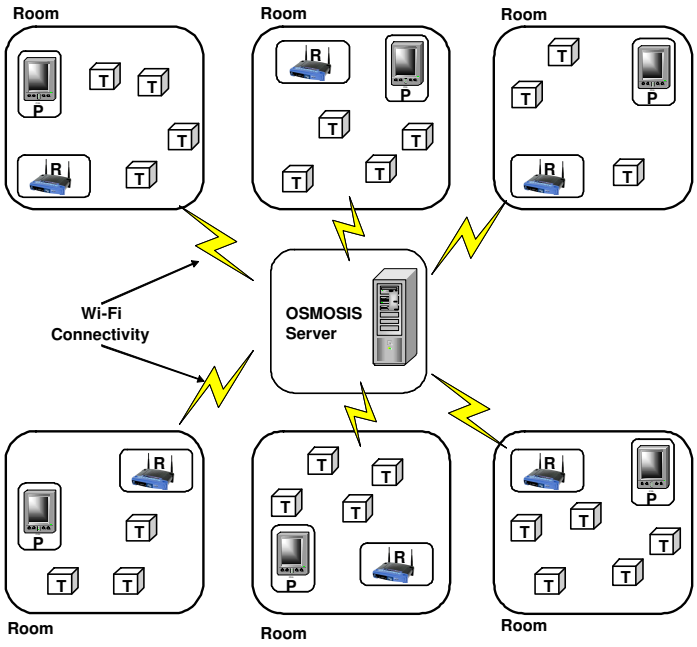

Figure 1. OSMOSIS Network Architecture.

which real-world objects should be hoarded together (once such objects are included in the virtual world), etc.

Thus, the main requirements addressed in OSMOSIS are the following:

- The design and implementation of a context-aware file system that, while keeping the well-known traditional file-system API, is capable of storing the context information related to each real-world object.
- Ensuring that users' information is really kept private while providing the tools to specify and enforce complex security policies.
- Provide a simple (and as much as possible invisible) interface to the system.

3. Architecture

In this section, we describe the main architectural aspects of OSMOSIS. Initially, we present the network architecture of OSMOSIS describing how the the various devices and computers interact. Next, we present the software architecture of the OSMOSIS middleware.

3.1. Network Architecture

The physical entities involved in a system employing the OSMOSIS middleware are portrayed in a prototypical situ-

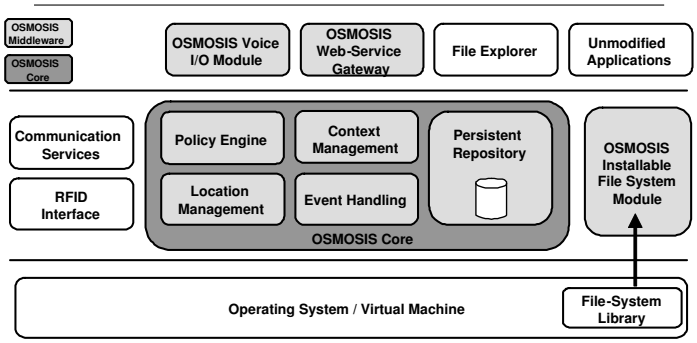

Figure 2. OSMOSIS Middleware Components.

ation depicted in Figure 1. We envisage a house with a number of rooms, naturally connected via doors.

There is a central OSMOSIS server running on a desktop machine. In each room, there is a PDA that connects, via Wi-Fi, with the OSMOSIS server. The PDA in each room makes use of a RFID reader to be aware of which objects are in the room, and when objects enter and leave the room. Naturally, all relevant objects are

3.2. Middleware Components

Thus, OSMOSIS will be comprised of a set of middleware components depicted in Figure 2. A number of them are mandatory and constitute the OSMOSIS core. The other ones depend on this core to provide extend functionality to applications, namely a file-based API and other forms of interaction. The OSMOSIS core is comprised of the following modules:

RFID Location Management: The RFID location subsystem is responsible for detecting (tagged) real objects and communicating their location to the OSMOSIS server. It comprehends a set of passive RFID tags attached to real-world objects, a set of fixed RFID readers strategically located at doorways, closets, etc. (so that the entrance and exit of tagged objects is detected), and a set of mobile non-obtrusive readers (e.g. bracelets [5]). We expect that passive RFID tags and readers cost will continue to drop while the accuracy and performance increase [29] thus making OSMOSIS a widely accepted reality in the near future.

Context Management: The context information can be stored by the file system using several techniques (e.g. semantic file systems, context aware systems, relational or XML databases, etc.) with consequences on the namespace organization, file contents and file properties [14, 17, 1, 18]. Although interesting for the OSMOSIS project, these approaches are mostly targeted at finding files (with a specific content) with no relation to real-world objects. Consequently, the requirements raised by OSMOSIS are different, thus requiring a different approach. In particular, the location of directories and files are tied to their real counterpart physical location; additionally, deleting a directory or a file is not acceptable in OSMOSIS (even if you are the legitimate owner) as this would be inconsistent with the fact that the corresponding real-world objects still exist.

Policy Engine: Understandably, the issue of privacy is vital for users as it is not acceptable that tagged objects location and characteristics are accessed freely. To ensure privacy and security in general, OSMOSIS extends the file system access control mechanisms with a security policy specification language (SPL) and the corresponding policy enforcer [8]. SPL supports a wide variety of security policies, including more complex ones such as those based on history (e.g. chinese-wall) and obligations. Using SPL it is possible to, for example, specify and enforce the following scenarios:

- Only the owner of object *x*, Joe, is allowed to be aware of *x*'s location,
- Only Joe and Mary are allowed to know that a certain object *y* does exist,
- Notify Joe whenever Mary enters room 421 for the third time during the last two weeks,
- Notify Joe if Mary and object *x* are moved from room 123 at the same time, etc.

Note that OSMOSIS does not provide a means to enforce access control to real-world objects as all policy specifications and enforcement is applied to the real objects virtual counterparts.

The rest of the OSMOSIS middleware is comprised by the following modules:

Context-aware Installable File-System: The context-aware file system is provided by the OSMOSIS middleware and is the preponderant mediation between applications and the OSMOSIS core. It looks to applications like a standard file system. Applications perform file-system system-calls that are redirected to and ultimately serviced by the OSMOSIS IFS.

The OSMOSIS IFS has the ability to store real-world objects context information; this information consists on the current (and past) location(s) of each tagged object, along with a corresponding narrative description (e.g. stating where and when the object was bought, what is used for) and any other relevant characteristic. This context information is particularly useful for (context-based) search operations performed on virtual objects on behalf of real objects as illustrated by the applications we intend to develop (described in Section 4). Although this module is not part of the OSMOSIS core, it is the one that provides the most basic abstraction to applications and should, therefore, be prevalent.

Web-service Gateway: The OSMOSIS Web-Service Gateway provides for applications running on remote computers to interact with the OSMOSIS core, residing in an OSMOSIS server, without the need to set-up a complex distributed file-system infrastructure. Furthermore, it can be used to integrate OSMOSIS with virtually any application running on top of any architecture.

Voice Input/Output Interface: An important aspect of ubiquitous applications is the interface provided to users. Ideally, it should be invisible [20]. For this purpose we will allow users to instruct the system (for the most usual operations) using voice commands in Portuguese and English; for example, the user should be capable of asking "where is the book that Joe offered me last year" ? Accordingly, OSMOSIS will provide an answer in spoken English and Portuguese as well.

4. Proposed Applications

We intend to develop a set of applications on top of the context-aware file system supporting the scenarios previously illustrated. Among others, the following applications seem of most value to users: *notificator*, *finder* and *hoarder*.

- The *notificator* is responsible for notifying the user of some particular situation such as Mary being close to a dangerous object *x*. Notifications can be sent as email messages or mobile text messages.
- The *finder* supports complex searches of objects based on contextual information; for example, where is the object that Joe offered me last week? show me all the objects I brought from last vacations. The *finder* is able to provide not only the current location of an object but also its past locations along with any other details that have been attached to it; virtual directories are automatically created containing related objects.
- The *hoarder* is responsible for hoarding files thus suggesting the set of real-world objects that the user should pack together. The set of hoarded objects results from the context information related to each one along with the action that the user will perform. As an example, when going for a meeting or on vacations, the *hoarder* aggregates all files related to such activities (based on the associated context information a virtual directory is created) and suggests the set of real-world objects that should be taken by the user.

Note that, since real world objects can be seen as files, the above mentioned applications (as other applications running on top of OSMOSIS) are very simple to develop. They can simply use the file-based API and are built upon all other already existing legacy (file-based) applications.

For example, detecting that Mary and object *x* are close to each other is performed by detecting that the virtual counterpart of Mary and the virtual counterpart of real-world object *x* are co-located in the same directory (virtual counterpart of a particular place).

Naturally, the *hoarder* can also take advantage of existing solutions and improve on them [23].

5. Implementation Issues

OSMOSIS is targeted to desktop PCs and PocketPCs, equipped with .Net Framework and .Net Compact Frameworks and implemented in C#. We employ RFID passive tags on objects and RFID readers on top of each room door. The Persistent Repository is implemented as a SQLServer database.

Context Management and the Policy Engine extend previous work in policy-based adaptive middleware [34]. Hoarding is performed as described in our previous described in [16]. The OSMOSIS context-aware FS is developed on top of previous work regarding file-system extension in mobile constrained devices [35].

6. Related Work

Much work has been done in the area of ubiquitous computing since Weiser's seminal work [37] and several projects have addressed the specific issue of breaking the frontier between real and virtual worlds.

Many projects [22, 7, 14, 13] focus on providing location awareness technologies to support people and real-world presence in the web; e.g., CoolTown's main goal (which shares some similarities with MIT's Oxygen [4]) is to enable nomadic computing such that computing resources follow the user and customize human computing interaction based on the local human environment.

Some other projects focus on people location and study the adequacy of several technologies such as WiFi, Bluetooth, RFIDs tags, etc. for such purpose [9, 11, 24, 21]. All these systems are mostly concerned with the specific location aspects and technology, i.e. they are not concerned with the usage of such location information.

The Smart-Its project [19] developed a platform for augmentation of everyday objects. A Smart-It is a small computing active device (similar to a label) that can be attached unobtrusively to physical objects so that these become empowered with processing, context-awareness and communication features. This kind of approach is interesting but it increases significantly the cost of everyday (traditionally "dumb") objects.

Romer [26] presents two frameworks, one based on Jini and other based on web-services to support the development of ubiquitous computing applications. Real-world objects

are identified with passive RFID tags as proposed in OSMOSIS. The system was used by a set of applications that illustrated the usefulness of inserting real-world objects into the virtual world. However, when compared to OSMOSIS, it is much less ambitious as it does not consider neither security, nor interface issues and, differently from our proposal, it does not explore the advantages of the well-known file-system paradigm and API.

Some projects propose the creation of interactive workspaces [2, 31, 15] whose aim is to provide an infrastructure for interactive rooms equipped with large displays and other wireless devices for interaction; thus, they focus on user interaction and group work in augmented rooms. This is the kind of approach that it is currently very costly and mostly oriented towards professional environments activities such as presentations, meetings, etc.

Other projects [10, 12, 6, 33, 38] share the goal of providing an infrastructure to support augmented environments. They address the issues of what are the basic abstractions and mechanisms for coping with the dynamics and device heterogeneity of pervasive computing. These approaches, as many others, have some points in common with OSMOSIS but their main goal is clearly different.

Savant [30] (developed at MIT Auto-ID Center) has the goal of replacing the traditional barcodes with passive RFID tags. For this purpose, the project investigates a large number of issues such as low-level protocols for the communication between tags and readers, XML-based language to exchange information about products, etc. The results from this initiative as well as the result of other similar projects [29] provide valuable input for OSMOSIS as they tend to make RFID technology cheaper and more precise.

Specifically concerning context-aware file systems, the ParcTab project [27] was one of the first addressing the integration of context with file access. Although they only considered location in their file system, this work demonstrated the relevance of context in data access and application adaptation. Since then, and for several purposes, many other systems addressed the issue of context-awareness in file systems [14, 17, 1, 18] and distributed applications [32]. Their work provides valuable insights for OSMOSIS. However, none has addressed the privacy issues resulting from the fact that directories and files are the counterparts of real world objects. In addition, the OSMOSIS context-aware file system must take into account specific issues such as those resulting from the unrestricted context information that can be attached to virtual counterparts of real-world objects (e.g. historical data) as well as from the needs of the applications finder, *notificator* and *hoarder*, among others.

OSMOSIS differentiates from all these projects as it addresses, in a single system, all the most fundamental issues that are vital for a prototype to be useful: it goes from the operating system level (context-aware file system) up to the (invisible) interface while taking privacy into account from the beginning. In addition, it targets everyday (possibly dumb) objects, including those whose cost does not allow adding computing and communication features (e.g. clothes, toys, CDs, books).

7. Conclusion

In this paper, we presented OSMOSIS, a novel middleware architecture that employs a pragmatic approach to ease application development in ubiquitous computing. These applications will be more and more prevalent with the disappearance of the real/virtual barrier occurring today. We also overview a thorough set of related work and projects in the areas of context-awareness and management of semantic and location information.

OSMOSIS aims at making real-objects virtual so that each real-world object has a virtual counterpart represented as a file. This provides an intuitive view for most users as well as the ability for programmers to use the well-established file-system API.

The OSMOSIS middleware provides a persistent repository enriched with context-awareness, location management, and policy enforcement, integrated with RFID, providing a file-system abstraction, encompassing the possibility to employ other interfaces as well (e.g., web-services, voice input).

References

[1] N. B. The prospero file system: a global file system based on the virtual system model. comp sys 5(4):407- 432, 1992.

[2] A. F. B. Johanson and T. Winograd. The interactive workspaces project: Experiences with ubiquitous computing rooms. ieee pervasive computing, 1(2):71-78, apr. 2002.

[3] G. Borrielo. Rfid: tagging the world. communications of the acm, september 2005, vol. 48, n. 9.

[4] M. L. Dertouzos. The future of computing. scientific american, aug. 1999.

[5] J. S. el at. Rfid-based techniques for human-activity detection. communications of the acm, september 2005, vol. 48, n. 9.

[6] B. B. et al. Easy living: Technologies for intelligent environments. huc 2000, bristol (uk), sept. 2000.

[7] B. S. et al. Bootstrapping the location-enhanced world wide web. 5th international conf. on ubiquitous computing, oct. 2003.

[8] C. N. R. et al. Spl: An access control language for security policies with complex constraints. proceedings of the network and distributed system security symposium, san diego, california, feb 2001.

[9] L. M. et al. Landmarc: Indoor location sensing using active rfid. 1st ieee conference on pervasive computing and communications (percom03), mar. 2003.

[10] M. R. et al. Gaia: A middleware infrastructure to enable active spaces. acm sigmobile mobile computing and communications review,vol. 6(4), oct. 2002.

[11] N. B. P. et al. The cricket location-support system. 6th international conf. on mobile computing and networking. aug. 2000.

[12] R. G. et al. System support for pervasive applications. transactions on computer systems, 22(4):421-486, november 2004.

[13] R. W. et al. The active badge location system. acm trans. information systems, vol. 10(1), jan. 1992.

[14] R. W. et al. Bridging physical and virtual worlds with electronic tags. computer-human interaction conference, 370-377,1999.

[15] T. P. et al. Spontaneous marriages of mobile devices and interactive spaces. communications of the acm, september 2005, vol. 48, n. 9.

[16] J. C. Garcia, L. M. A. Veiga, and P. J. P. Ferreira. Context awareness: an experiment with hoarding. October 2006.

[17] M. U. Gopal B. Integrating content-based access mechanisms with hierarchical file systems. proceedings of the 3rd symposium on operating systems design and implementation, new orleans, la, february 1999.

[18] C. Hess and R.Campbell. An application of a context-aware file system. pers. ubiquitous computing, 7: 339-352, 2003.

[19] e. a. Holmquist, L.E. Smart-its friends: A technique for users to easily establish connections between smart artefacts. ubicomp 2001, atlanta, usa, oct. 2001.

[20] H. Ishii and B. Ullmer. Tangible bits: Towards seamless interfaces between people, bits and atoms. proceedings of chi '97, 234-241, march 22-27, 1997, acm.

[21] G. B. J. Hightower. Location systems for ubiquitous computing. computer, vol. 34 (8), aug. 2001.

[22] T. Kindberg. Implementing physical hyperlinks using ubiquitous identifier resolution. 11th conference on www. acm press, 2002.

[23] G. H. Kuenning and G. J. Popek. Automated hoarding for mobile computers. proceedings of the 16th acm symposium on operating systems principles, st. malo, france, october, 1997.

[24] M. T. M. Spreitzer. Providing location information in a ubiquitous computing environment. 14th acm sigops conf., asheville, nc, december 1993.

[25] M. Ohkubo. Rfid privacy issues and technical challenges. communications of the acm, september 2005, vol. 48, n. 9.

[26] K. e. a. Rmer. Smart identification frameworks for ubiquitous computing applications. kluwer/acm wireless networks (winet), vol. 10 no. 6, pp. 689-700, december 2004.

[27] A. N. Schilit BN and W. R. Context-aware computing applications. proceedings of the ieee workshop on mobile computing systems and applications, santa cruz, february 1994.

[28] F. Stajano. Security for ubiquitous computing. john wiley and sons. chicester, uk, 2002.

[29] T. P. I.-C. System. wwwus2. semiconductors.philips.com/identification/products/icode.

[30] O. Systems and M. A.-I. Center. The savant. technical report mit-autoid-tm-003, mit auto-id center, may 2002.

[31] P. Tandler. Software infrastructure for ubiquitous computing environments: Supporting synchronous collaboration with heterogeneous devices. ubicomp 2001, atlanta, usa, sept. 2001.

[32] T. Urnes, A. Hatlen, P. Malm, and Ø. Myhre. Building Distributed Context-Aware Applications. *Personal and Ubiquitous Computing*, 5(1):38–41, 2001.

[33] K. Van Laerhoven and K. Aidoo. Teaching Context to Applications. *Personal and Ubiquitous Computing*, 5(1):46–49, 2001.

[34] L. Veiga and P. Ferreira. Poliper : Policies for mobile and pervasive environments. In *3rd Workshop on Reflective and Adaptive Middleware. In 6th ACM International Middleware Conference*, Toronto, Canada, October 2004.

[35] L. Veiga and P. Ferreira. Semantic-Chunks a middleware for ubiquitous cooperative work. *Proceedings of the 4th workshop on Reflective and adaptive middleware systems*, pages 1–6, 2005.

[36] R. Want. Rfid: A key to automating everything. scientific american, 56-65, jan. 2004.

[37] M. Weiser. Some computer science issues in ubiquitous computing. communications of the acm, 36(7), 1993, pp. 74-84.

[38] T. Zimmer and M. Beigl. AwareOffice: Integrating Modular Context-Aware Applications. *Proceedings of the 26th IEEE International ConferenceWorkshops on Distributed Computing Systems*, 2006.