The Simplicity Project: easing the burden of using complex and heterogeneous ICT devices and services

Part II: State of the Art of Related Tecnologies

N. Blefari Melazzi[1], G. Ceneri[2], G. Cortese[3], N. Davies[4], N. Dellas[6], A. Friday[4], J. Hamard[5], E. Koutsoloukas[6], C. Niedermeier[7], C. Noda[5], J. Papanis[6], C. Petrioli[9], E. Rukzio[10], O. Storz[4], J. Urban[8]

[1] DIE, University of Roma "Tor Vergata", e-mail: blefari@uniroma2.it

[2] Radiolabs, email: gianni.ceneri@radiolabs.it

[3] Telecom Italia Learning Services, e-mail: g.cortese@computer.org

[4] Computing Department, Lancaster University, e-mail: {nigel,adrian,oliver}@comp.lancs.ac.uk

[5] DoCoMo Communications Laboratories Europe, e-mail: {hamard,noda}@docomolab-euro.com

[6] , School of Electrical and Computer Engineering ,National Technical University of Athens, e-mail: {lefterisk, jopapan, ndellas}@telecom.ntua.gr

[7] Siemens Corporate Technologies, e-mail: Christoph.Niedermeier@siemens.com

[8] Siemens Mobile, e-mail: Josef.Urban@siemens.com

[9] CS Department, Rome University "La Sapienza", e-mail: petrioli@di.uniroma1.it

[10] "Institut für Informatik", Ludwig Maximilians University, e-mail: Enrico.Rukzio@informatik.uni-muenchen.de

ABSTRACT

As of today, to exploit the variety of different "services", users need to configure each of their devices by using different procedures and need to explicitly select among heterogeneous access technologies and protocols. In addition to that, users are authenticated and charged by different means. The lack of implicit human computer interaction, context-awareness and standardisation places an enormous burden of complexity on the shoulders of the final users. The IST-**Simplicity** project aims at leveraging such problems by: i) automatically creating and customizing a user communication space; ii) adapting services to user terminal characteristics and to users preferences; iii) orchestrating network capabilities. The aim of this paper is to present the technical framework of the IST-**Simplicity** project. This paper is a thorough analysis and qualitative evaluation of the different technologies, standards and works presented in the literature related to the Simplicity system to be developed.

I. SIMPLICITY FRAMEWORK

Our vision is that of a user surrounded by different devices, providing him with access to several "services" and functionalities (e.g. access control to a building, location aware services, ...). As of today, to use these services, the user has to access the network through heterogeneous technologies and protocols, must have different devices, must configure each of them by using different procedures, must be recognized and authenticated in different ways and must be charged with different means. Consequently a complexity burden lies on the shoulders of the user who doesn't know how to "choose" between such possibilities, and can't carry a large number of different devices at the same time.

The aim of the IST-Simplicity project is to ease the user interaction with devices and the use of services and functionalities. In more details, the project goal is to design and deploy a "brokerage" level able to decouple user needs and user devices, as well as service deployment and fruition, from the underlying networking and service

support technologies. In our view, each user should be ideally endowed with a personalized profile to be used for different services/transactions, eventually based on different classes of terminal. Such profile should ideally allow an automatic, transparent personalization and configuration of terminals/devices, and should provide a simple and uniform way to be recognized, authenticated, located and charged. Thanks to this profile, users could also enjoy the automatic selection of services appropriate to specific locations (the home, buildings, public spaces), the automatic triggering of home/building/public-space functionalities, and the easy exploitation of different telecommunications paradigms and services. Depending on user's characteristics, preferences and abilities, the profile could take the form of e.g.,: i) a standard profile defined by a Service Provider; ii) a pre-defined template whose parameters can be configured by the user; iii) an open profile designed by the user using a GUI or a high-level description language. The user profile is stored in a so called Simplicity Device (SD) or a network location or a software agent. If the SD is a physical device, users could personalize terminals and services by the simple act of plugging the SD in the chosen terminal (see Fig. 1).

Fig. 1: The reference scenario

The **Simplicity** system (see Fig. 1) thus encompasses the Simplicity Device and a Brokerage Framework. The Brokerage Framework will use policy-based technologies (e.g., policies for mobility support, QoS, security, SW downloads) to orchestrate and adapt network capabilities, taking into account user preferences and terminal characteristics. The Brokerage Framework will encompass a Terminal Broker module, which is primarily used to allow the interaction of the SD with both the terminal and the network, and a Network Broker module. Since the SD has the goal to allow a uniform and personalized user view of services, there must be a way to describe, and advertise such services, to allow the user to browse and select them. Subsequently, there is the need to coordinate services and share/allocate the available resources. The Network Broker is responsible to perform the aforementioned tasks, by providing a platform for service deployment, advertisement, personalization, etc. The brokerage level must provide adaptation capabilities to the considered context (location, time, etc) and eventually an orchestration of events, managing also simultaneous access of several users to the same resources, services, and locations.

In the next sections we will discuss the technologies, standards and solutions currently available for each of the Simplicity system components.

II. PERSONALIZATION ISSUES AND USER PROFILES

The Simplicity system creates a Personal Service Environment (PSE) which relies on users profiles for adaptation and personalization of services and terminals. In general, a personal profile is a collection of information electronically representing the user such as personal characteristics, preferences, rules, and tasks. In this section we will detail the SoA regarding user profiling, service/terminal adaptation and personalization.

A 3GPP solution to user profiling under standardization is the Generic User Profile (GUP)[4] [5], based on XML. 3GPP GUP proposes a structure according to which data have to be organized, but leave great flexibility on the content of the data themselves. For example, 3GPP GUP may store data like authorized and subscribed services, general user information, user privacy control data, information about specific services and billing information. Historical/Statistical and Runtime data are not included in the GUP. The 3GPP solution envisions network cooperation as profiles are stored and downloaded from the network, but this approach can be adapted and extended to support a user side architecture where information are stored directly in the SD. The most important aspect of GUP is that it could be adapted to every system and context, thus providing the flexibility needed by the Simplicity project.

Another interesting solution is the Application Configuration Access Protocol (ACAP) [2] that is designed to support remote storage and access to customization, configuration and preference information. The data storage model is designed to allow a client to simply access all the information needed for automatically adapting and personalizing the service. New information can be easily added without server re-configuration thus allowing the use of both standardized data and custom or proprietary data.

In the field of terminal capabilities description technologies, the Composite Capabilities/Preferences Profile (CC/PP) framework is an important standardization effort, which defines how a user agent profile can be specified [1]. The goal of the CC/PP framework is to specify how client devices express their capabilities and preferences (the user agent profile) to the server that originates content (the origin server). The origin server uses the 'user agent profile' to produce and deliver content appropriate to the client device. In addition to computer-based client devices, particular attention is being paid to other kinds of devices such as mobile phones. The framework describes a standardized set of CC/PP attributes

that can be used to express a user agent profile in terms of capabilities, and the users preferences for the use of these capabilities.

The User Agent Profile Specification [8] is a specification, which extends the WAP v1.1 standard to enable the end-to-end flow of a user agent profile in mobile environments. The UAProf specification defines in this respect so-called Capability and Preference Information (CPI), which is communicated between the WAP client, the intermediate network points, an the origin server. The specification seeks to interoperate seamlessly with the emerging standards for Composite Capability/Preference Profile (CC/PP) distribution over the Internet. It uses the CC/PP model to define a robust, extensible framework for describing and transmitting CPI about the client, user, and network. The specification defines a set of components and attributes that WAP-enabled devices may convey within the CPI.

RDF [3], the Resource Description Format, was designed by the W3C consortium for dynamic content adaptation. It defines a mechanism for describing (Web) resources (meta-data), to enable "automated" processing of these resources. It provides a model for representing these meta-data, and proposes XML as the syntax for this model. No assumption is made about a particular application domain.

Some interesting projects propose the use of policy based technologies or rule languages for personalization aspects to achieve flexibility and generality [6] [7]. The most important rule languages in this context are Jess, ZKB/XKB and RuleML.

Jess [10] is a well-established rule engine and scripting environment that is based on the CLIPS expert system shell and that is entirely written in Java. XKB/ZKB is the rule language which is included in the open source java class library Mandarax [11]. Both these two projects allow to express reactive rules and facts that refer to and act on Java objects representing for example user models, device capabilities, applications or network aspects.

The Rule Markup Initiative develops a semiformal XML-based language called RuleML [9] that permits Web-based rule storage, interchange, retrieval, and application. It's possible to define integrity constraints, derivation rules and reacting rules. There already exist some corresponding DTDs/Schemas, engines, translators, user interfaces and rule libraries.

Policies will be included in the simplicity device to express preferences of the user as well as in the terminal to define terminal specific adaptation aspects. These policies have to interact with the policies based technologies on the network side which are related to the IETF policy framework, the Ponder framework and the Policy Description language. Furthermore the same policy technology or a policy exchange language should be used to get a consistent policy treatment.

III. SIMPLICITY DEVICE

The Simplicity Device (SD) is the part of the Simplicity system that lies in the user side. Each user is equipped with an SD pluggable to a multitude of terminal types that allows the user to participate to policy-based configuration, automatic service discovery and AAA mechanisms of his Simplicity environment. It may be perceived as a component that combines the functionality of a hardware authentication token, a mobile storage device and a portable processing utility able to perform trivial and somewhat more complex tasks.

The SD could be realized in hardware, in which case it could be a USB disk, an enhanced smart card, or it could even be realized as a software agent for use in special environments. A hardware implementation is however preferable, and since mobile code execution capabilities are desirable, the hardware realization that currently best facilitates the abstract functionalities of the SD is a smart card. Even though USB disks provide storage space ranging from several hundred Mbytes up to a few Gbytes and connectivity with most computing equipment, they lack processing capabilities which is a desirable feature for the SD. Smart Cards on the other hand are pluggable to any terminal type that provides connectivity to some sort of card reader equipment, they provide tamper-resistant storage space for sensitive personal identification information and their much anticipated view as general mobile code executing platforms [12] has recently been realized through the fast paced advances in their processing capabilities and the evolution of the embedded software that supports them.

The following sections describe the state of the art in USB devices and Smart Card technology, provide an overview of Java Card architecture, one of the most important smart card software platforms available today and argue in favor of Java Card as the implementation solution for the SD.

A. USB Devices

In the last few months, a high diffusion of USB memory-bar devices has taken place. The reason stands in the lowering of prices due to the high progress done in the manufacturing process of memory modules, in the growing capacity that these devices offer and in the high flexibility provided by USB interfaces

USB specifications [13] have gone through three steps: ver. 1.0, that provides a bit-rate of 1.5Mbps, ver. 1.1, that provides a bit-rate of 12Mbps and ver. 2.0 with a high bit-rate of 480Mbps. Data transfer speed offered by USB interfaces is fully compliant with SD requirements as profile data occupies only a small amount of KB. The storage capacity offered by USB memory-bar goes from 32MB up to 2GB which is very impressive if we think that such a device has dimensions like a standard key. In addition, it strikes the portability requirements of the SD as, for example, we can bring it attached to a key-ring.

Integration of USB memory-bar devices with current computing and communication equipment is very good. Most of the current PC/PDA operating systems provide the complete support for this kind of devices. The user has just to plug the bar into a USB port of his equipment and the service is immediately available. It aligns perfectly with

the concept of SD. Moreover, many set-top boxes are introducing support for USB devices.

Reliability of USB memory-bar devices is a very important issue. Data stored into a memory-bar have to be error-free as it carries information that is very hard to retrieve from another source. Nowadays, USB memory-bar devices have a life-cycle of about 1.000.000 re-writes with 10 years of data retention [14]. Some studies have demonstrated that the higher the number of re-write operations, the lower the retention time. When maximum number of write operations is reached, the retention time decreases at about 3-4 days.

There are a lot of USB memory-bar devices that implement security mechanisms in order to assure user data confidentiality. Some sample mechanisms are PIN-PUK code, username/password and finger-print matching. An example of algorithms used to encrypt data is AES-128bit. These features provide the user data to be protected against external attacks and un-authorized copies.

A critical aspect of USB memory-bar devices is that they have no computational capabilities. It is very limitative for the implementation of the SD since it cannot perform any processing tasks required by the host systems.

B. Smart Card Technology

Smart Cards are often defined as an IC (integrated circuit) chip embedded in a plastic card as a tamper-proof hardware. In the market, there are two sizes of smart cards. One is the same size with a credit card specified by ISO/IEC 7816-4 [15], especially in the field of banking, insurance and transportation. In the telecommunication field, a different size of smart cards, 15mm*17mm, is used, often called GSM SIM (Subscriber Identity Module) cards or 3G UICC (Universal IC Card) standardized by 3GPP and ETSI SCP (Smart Card Platform) [16].

A typical smart card is equipped with an 8-bit or 16-bit processor clocked at the speed of a few MHz, a few kilobytes of RAM memory, ROM memory with built-in functionality and 32-64kb of non-volatile memory (e.g. flash memory). Recently, high performance smart cards have become available with attractive features, such as 32-bit processors with an optional cryptographic co-processor and up to a few Mbytes of storage (combined RAM and flash memory). Smart Cards rely on special equipment, a card reader, also called a card acceptance device, to interface with terminals of various types. These interfaces are governed by the series of international standards ISO7816 [17] that rule all smart card features, from physical characteristics to interaction mechanisms with an external world.

Smart Card software has evolved along with the processing capabilities of their embedded ICs. Four generations of smart card software are described in [18], spanning from monolithic embedded operating systems to today's modular, adaptable open platforms featuring secure multi-application executing environments, post-issuance application loading capabilities and object-oriented development models. Examples of such platforms are the Java Card Platform [19], a special subset of Java technology for resource constrained devices and the Multi-Application Operating system (MultOS) [20], which provides a secure executing environment for multiple applications on the same card. Such platforms rely on open standards that ensure interoperability with operating systems, the most important being the Microsoft PC/SC Specifications [21] that standardize interaction of smart cards with Microsoft operating systems, and the Open Card Framework [22], that standardizes Java based smart card solutions.

These smart card features, combined with their practical nature as lightweight portable electronic devices, deem smart cards as significant mobile code execution platforms. Their value is further enhanced by active research on their applications concerning user mobility [23], e-commerce and personalized information services [24] [25], security [26] and interoperability with agent technology [27]. The experience on smart cards gained from these research projects will be valuable for the implementation of the SD as a smart card.

C. Java Card Platform

The Java Card Platform is an attractive choice for the implementation of the SD, as it introduces the proven value and quality of Java technology to the embedded software scene, with features such as code portability, enhanced security and object-oriented development techniques. Java Card technology is widely supported in the smart card industry and it is constantly evolving to take advantage of the hardware advances of smart cards.

Java Card Technology is a subset of the Java technology, suitable for resource constrained devices like smart cards. Java Card provides a multi-application executing environment inside the smart card that enforces strict separation rules between applications, thus enhancing security and integrity of data [28]. The Java Card applications execute inside a virtual machine which in turn executes on the card's specific operating system. The development of Java Card applications, which are called applets, follows an object-oriented methodology. Applets are portable to cards from different manufacturers and can be loaded after the card has been issued, a feature which facilitates software updates and the development of new services for Java Card users.

The Java Card Platform Specification [29] consists of three parts; the specification of the Virtual Machine and the Java language subset, the specification of the runtime environment for applets, and the Java card API, the framework for developing applets. A typical Java Card application consists of a back-end information system that interacts with a reader-side host application. The host application exchanges commands and responses packed into Application Protocol Data Units (APDU), which are defined in the ISO7816-4 [15] set of standards, with the Card Acceptance Device (CAD), and the CAD interacts with the VM executing inside the Java card along with a number of active applets. Besides the message passing

communication model that exchanges APDUs, Java Card provides an alternative communication method using Java Card Remote Method Invocation (JCRMI), a subset of the RMI distributed object model technology.

The aforementioned Java Card technology features make it an attractive solution for the realization of the SD. First of all, Java Card meets the increased security requirements of the SD which will store and process sensitive information such as credit card numbers, authentication information for online services, network access credentials and operator contract information. The strict security requirements should not, on the other hand, deprive from the SD features such as flexibility and rich functionality, since its duties include more than mere authentication. Java Card provides the required extensible functionality with a sound security mechanism.

IV. FLEXIBLE NETWORK SUPPORT

Flexible network support for context aware adaptation and personalization of services and terminals is one of the main goals of the Simplicity project. The envisaged technical solution shows the following main characteristics:

- Adoption of a brokerage framework that employs policy-based techniques for achieving an overall control and adaptation platform
- The combination with flexible agent-based technologies supporting the distribution and execution of code across a variety of different terminals
- A distributed solution for service discovery as a key element for a decentralized framework.
- Reliable data storage as a basic service for handling distributed data, e.g. profile and context information.

The next subsections reviews in more detail the state of the art regarding the four characteristics mentioned above.

A. Policy based brokerage framework

A broker is an entity that undertakes management action on resources. A broker insulates his area of responsibility from other entities so that all administrative actions are performed through requests to the appropriate broker. Overall administration of resources is achieved through broker cooperation and coordination, with the aid of an enabling technology that facilitates interaction of distributed entities. The broker concept was initially introduced by [30], where QoS was achieved through interaction between brokers residing at the end points, and was adopted in the MASA project applied on adaptive multimedia services in mobile contexts [31] and extended to include additional brokers (called network brokers) residing not at the end points but in access and core networks [32], [33].

According to this concept, a broker is responsible for orchestrating different functions and subsystems within one domain. The coordination of management efforts across different domains happens by negotiations between different brokers that are controlled by policy based decision mechanisms. A broker itself consists of independent but interworking subsystems. Again, the operation and inter-working of these subsystems is controlled and coordinated using policy rules. Of particular importance is the modular nature of policies that allows addition and elimination of policy rules without affecting other parts of the rule base. The benefits of policy based management of distributed systems arising from usage of proper syntax and policy management tools have been pointed out in [34].

By using ambient awareness mechanisms (e.g. based on sensors), a broker can generate up-to-date context information as a basis for negotiations with other brokers. Context information in combination with policy based decision mechanisms facilitates flexible adaptive end-to-end management of services [35] [36]. Support of context aware systems in smart spaces can be provided by Context Brokers that employ common ontologies, a shared context model and a common policy language [37].

A possible implementation of the broker concept within Simplicity includes a terminal broker responsible for orchestration of user preferences, terminal capabilities and operation of locally running applications based on context information regarding the user, the terminal, and the access network. The terminal broker is supported by a system of network brokers that are responsible for orchestration of all network features. Different types of network brokers may be introduced to account for specifics of different network domains as access networks, core networks and service provider domains. Network brokers may be replicated to provide a scalable network infrastructure.

B. Mobile Agents Platforms

Mobile Agents are intelligent/autonomous software entities able to migrate and execute their logic in several computational nodes. They are considered as middleware oriented technology enhancing distributed computing technologies such as CORBA, RMI and Web Services paradigm [38] [39] [40]. A Mobile Agent Platform (MAP) enables the agents' execution to distributed nodes. A MAP consists of a set of APIs that exploit the underlying middleware capabilities and mechanisms. Prominent MAPs are the LEAP JADE, MicroFIPA-OS, AgentLight, JACK, Grasshopper and April.

Benefits of mobile agents are communication and execution state transparency, autonomous and intelligent execution, programming and communication flexibility, adaptability to specific conditions, life cycle management, robustness and fault-tolerance, and interoperability. With regard to Simplicity, these features are valuable for the implementation of broker coordination procedures and requirements such as service discovery and dynamic code distribution. Accessing services in a visited network environment requires often support of mobility in the form of code download. JSR 24 (J2EE Client Provisioning) [49] provides a configurable and extensible framework to implement a context aware software distribution mechanism. On the client side, standardization and research work (e.g. [48]) is ongoing to define a more

flexible, robust Java-based execution platform for mobile devices, supporting full component lifecycle management (including secure download, activation and disposal).

C. Service Discovery Frameworks

Focusing on design of brokers that support a peer-to-peer (P2P) communication paradigm is one of the directions enabling a distributed system to be more flexible. In this model, there is no longer central point to publish services and information, and all brokers can transparently share information in a global space.

Service discovery frameworks are conceived as a method to discover available services and resources in a network. The most emerging service discovery protocols relevant to P2P communications are Universal Plug and Play (UPnP) [41] and JXTA [42]. Both are a set of communication protocols based on XML-encoding. At UPnP, Simple Service Discovery Protocol (SSDP) enables devices to publish their presence and service descriptions by multicasting advertisements and clients to listen at the multicast port to discover services, or alternatively clients to search services by multicasting requests. JXTA further supports community base activities across different P2P systems. It enables peers to create peer groups providing a common set of services. Peer Discovery Protocol is the default protocol for all peers to support, allowing a peer to find advertisements from other peers or peer groups.

D. Simple Storage Management

Technologies which aim at delivering network-based reliable, secure storage services provide the ability of storing and accessing personal data independent of user location, network point of access and terminal. In case of Simplicity user profile data, context data should be possibly stored or replicated transparently to the user in the network (as an alternative to keep such data in the SD).

Relevant projects in this area include OceanStore [43] (backed by IBM), Microsoft FarSite, PAST [45], CFS.[44]. All of them are built on top of a DHT routing layer. DHT middleware ([46]) provide an application-level routing layer which can be exploited by higher level middleware services and applications (such as event notification, multicast, storage and file systems, and naming systems). Since user data is distributed in the network, security and integrity are primary concerns in these systems. Smart Card mechanisms are typically used for this purpose (e.g. to provide encryption, to generate and verify certificates, to manage storage quotas etc.).Early attempts exist to build on top of such infrastructures email services (POST [47], MINO). These projects show how user metadata (folders, preferences, contact lists) can be stored in the network so that they can be available to the user independently from the client attaching to the service.

V. FLEXIBLE NETWORK SUPPORT

Sal awakens: she smells coffee. A few minutes ago her alarm clock, alerted by her restless rolling before waking, had quietly asked "coffee?", and she had mumbled "yes." "Yes" and "no" are the only words it knows...

These are the opening sentences of a powerful scenario [55] that Mark Weiser used in 1991 to outline his vision of a futuristic, computer-assisted world. His revolutionary thoughts and ideas soon began to inspire researchers all over the world and provided a foundation for emerging areas of research, i.e. ubiquitous computing and ambient intelligence. Weiser envisioned a future where computational power and intelligence would be embedded into our everyday world in a seamless fashion. Hundreds, possibly thousands of computational devices, sensors and actuators would turn every physical space into a smart, intelligent space. Doing so would create a world that had the possibility to assist humans in their activities.

Examples of current state-of-the-art Ambient Intelligence and Ubiquitous Computing projects include, but are not limited to:

- Georgia Tech's Aware Home [50] with a focus on providing support for the elderly in their own homes.
- MIT's Project Oxygen [53], trying to create smart environments by using a variety of embedded or handheld devices and adaptive networking technologies. Particular highlights include new means of human-computer interaction, e.g. via natural language and gestures.
- The Interactive Workspaces Project [52] at Stanford University, exploring the use of collaborative, interactive workspaces.
- GAIA [54] at UIUC with a strong emphasis on mobility support for people, devices and applications.

A key element in Weiser's vision is the desire to minimise explicit interactions between humans and their smart environments. Smart spaces are expected to act proactively instead of merely reacting to explicit input from users. Systems are therefore required to be able to obtain and process rich sets of contextual data, including information about human users, physical objects and software entities.

Within this context, Simplicity will provide means for:

- storing and providing contextual information about its owner, e.g. in the form of profiles
- authenticating users, either by directly using the owner's Simplicity ID or through credentials stored within a user's Simplicity device
- discovering and personalising services. For example, Simplicity's intelligent brokering framework will be able to discover services that are relevant to the user's context, preferences and objectives.

Simplicity will need to operate within the context of smart environments developed outside the project. For example, services within these environments will have to be discovered and interacted with. More specifically, we do not expect Simplicity to advance the state of the art in smart environments themselves, rather we expect the project to provide a mechanism for easily customising these environments. As a result we plan to base our Simplicity prototypes on an existing smart environment

platform. Our requirements for this platform are that it supports a decoupled, asynchronous communications model in order that we can easily incorporate the additional infrastructure elements of the Simplicity architecture without impacting on the operation of the remainder of the smart environment. Having reviewed the systems presented earlier we have decided to base our work on the iROS platform [52] developed at Stanford University, extended with features to enable it to generalise beyond the context of a meeting room for which it was developed.

The Interactive Room Operating System (iROS) is part of the Interactive Workspaces Project. It comprises three main subsystems: iCrafter (a framework for service discovery and the dynamic composition of user interfaces), the Data Heap (a shared data space with support for transcoding) and the Event Heap. The Event Heap represents the core component of the iROS system. Extending the classic tuple-space paradigm [51], the Event Heap provides an asynchronous, event-based communication framework for interconnecting components in distributed systems. It is suitable for building loosely coupled applications, thereby catering for important aspects of mobile and ubiquitous computing systems such as fault-tolerance and support for mobility and temporary disconnections. Furthermore, a loose coupling of components facilitates the introduction of new entities into existing systems, making iROS a suitable platform for prototyping and research. It is therefore our aim to investigate possible ways of using Simplicity for customising iROS-based smart spaces.

REFERENCES

[1] "Composite Capabilities/Preference Profiles: Requirements and Architecture", Mikael Nilsson et al. (eds). W3C Working Draft, 21 July 2000. See: http://www.w3.org/Mobile/CCPP/

[2] Newman, C., et al., "ACAP – Application Configuration Access Protocol, Internet Request for Comments", RFC 2244, November 1997. See: http://www.ietf.org/rfc/rfc2244.txt

[3] "Resource Description Framework (RDF) Model and Syntax Specification",. Ora Lassila, et al., (eds.). W3C Recommendation 22 February 1999. See: http://www.w3.org/TR/REC-rdf-syntax

[4] 3GPP TS 22240-600: "3GPP GUP, Requirements, Stage 1 Release 6", March 2003. See: www.3gpp.org

[5] 3GPP TS 23240-110: "3GPP GUP, Architecture Specifications, Stage 2 Release 6", April 2003. See: www.3gpp.org

[6] Patricia Lago, "A Policy-based Approach to Personalization of Communication over Converged Networks", 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY'02), 2002.

[7] Lalitha Suryanarayana, Johan Hjelm, "Profiles for the situated web", in Proceedings of the eleventh international conference on World Wide Web, 2002, pp. 200-209

[8] Wireless Application Group, User Agent Profile Specification, WAP Forum Approved Specification WAP-174, 10 November 1999. See: http://www1.wapforum.org

[9] RuleML, http://www.dfki.uni-kl.de/ruleml/

[10] Jess, http://herzberg.ca.sandia.gov/jess/

[11] Mandarax http://mandarax.sourceforge.net/

[12] Roger Kehr, Michael Rohs, Harald Vogt, "Mobile Code as an Enabling Technology for Service-oriented Smartcard Middleware", 2nd IEEE International Symposium on Distributed Objects and Applications, Antwerp, Belgium, 2000, p.2

[13] Universal Serial Bus, http://www.usb.org

[14] Aran Ziv, Tal Segalov, "FlashDrive Performance and Reliability, White Paper", September '03

[15] ISO/IEC 7816-4:1995, "Information technology -- Identification cards -- Integrated circuit(s) cards with contacts -- Part 4: Interindustry commands for interchange"

[16] ETSI TS 102 221, "Smart Cards; UICC-Terminal interface; Physical and Logical characteristics", V6.0.0, 02-2003

[17] ISO/IEC 7816, Information technology - Identification cards - Integrated circuit(s) cards with contacts, 1997-2004

[18] Damien Deville, Antoine Galland, Gilles Grimaud, Sebastien Jean, "Smart Card Operating Systems: Past, Present and Future", The 5th USENIX/NordU Conference, 2003, pp. 2-4

[19] Java Card Technology, http://java.sun.com/products/javacard/index.jsp

[20] Multi-Application Operating System (MULTOS), http://www.multos.com

[21] PC/SC Workgroup, http://www.pcscworkgroup.com/

[22] Open Card Framework, http://www.opencard.org

[23] IST Project FASME, "Facilitating Administrative Services for Mobile Europeans", http://www.fasme.org

[24] IST Project SM-PAYSOC, "Secure Mobile PAYments and Services On Chip", IST-2001-32526, http://www.smpaysoc.org

[25] IST Project SMARTCITIES, "Multi-Application Smart Cards in Cities", IST-1999-12252

[26] IST Project VERIFICARD, "Tool-assisted Specification and Verification of JavaCard Programmes", IST-2000-26328, http://www.verificard.com

[27] IST ACTS-SCARAB, "Smart Card and Agent enabled Reliable Access"

[28] Sun Microsystems Inc, "Java Card Platform Security", Technical White Paper pp. 6-9, http://java.sun.com/products/javacard/JavaCardSecurityWhitePaper.pdf

[29] Sun Microsystems Inc, " Specification for the Java Card Platform, v2.2.1", http://java.sun.com/products/javacard/specs.html
[30] Klara Nahrstedt, Jonathan M. Smith, "The QoS Broker", IEEE Multimedia, 2(1), Spring 1995.
[31] Hannes Hartenstein, Andreas Schrader, Andreas Kassler, Michael Krautgärtner, Christoph Niedermeier, "High Quality Mobile Communication", Kommunikation in Verteilten Systemen 2001: 279-289.
[32] Andreas Kassler, Andreas Schorr, Christoph Niedermeier, Reiner Schmid, Andreas Schrader: "MASA - A scalable QoS Framework", Proceedings of Internet and Multimedia Systems and Applications (IMSA) 2003, Honolulu, USA, August 2003.
[33] Andreas Kassler, Andreas Schorr, Lingang Chen, Christoph Niedermeier, Carsten Meyer, Michael Helbing, Michal Talanda: "Multimedia Communication in Policy based Heterogeneous Wireless Networks", IEEE Vehicular Technology Conference VTC2004-Spring, Milan, Italy, May 2004.
[34] Damian Marriott, Morris Sloman, "Management Policy Service for Distributed Systems", IEEE 3rd Int. Workshop on Services in Distributed and Networked Environments (SDNE'96), Macau, June 1996.
[35] M. E. Anagnostou, A. Juhola, E. D. Sykas. "Context Aware services as a step to pervasive computing", Lobster Workshop on Location based Services for accelerating the European-wide deployment of Services for the Mobile User and Worker, Mykonos, Greece, 4-5 October, 2002.
[36] John Keeney, Vinny Cahill: "Chisel: A Policy-Driven, Context-Aware, Dynamic Adaptation Framework", Proceedings of the Fourth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2003), Lake Como, Italy, June 2003.
[37] Harry Chen, Tim Finin, Anupam Joshi. "An Intelligent Broker for Context Aware Systems", Adjunct Proceedings of Ubicomp 2003, Seattle, Washington, USA, October 2003.
[38] P. Bellavista et al., "CORBA Solutions for Interoperability in Mobile Agents Environments", International Symposium on Distributed Objects and Applications, September 2000, Belgium.
[39] J. Delgado et al., "An Architecture for Negotiation with Mobile Agents", IFIP MATA02, October 2002, Spain.
[40] I. Foukarakis, A. I. Kostaridis, C. G. Biniaris, D. I. Kaklamani and I. S. Venieris, "Implementation of a Mobile Agent Platform based on Web Services", MATA, Marrakech, Morocco, October 8-10, 2003.
[41] UPnP, http://www.upnp.org/
[42] JXTA, Project JXTA, http://www.jxta.org
[43] Kubiatowicz, J., Bindel, D., Chen, Y., Czerwinski, S., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, H., Weimer, W., Wells, C., and Zhao, B. OceanStore: An architecture for global-scale persistent storage. In Proceeedings of the Ninth international Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000) (Boston, MA, November 2000), pp. 190-201.
[44] Dabek, F., Kaashoek, M. F., Karger, D., Morris, R., and Stoica, I. Wide-area cooperative storage with CFS. In Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01) (Oct. 2001).
[45] Rowstron, A., and Druschel, P. Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. In Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01) (Oct. 2001).
[46] Hari Balakrishnan, M. Frans Kaashoek, David Karger, Robert Morris, Ion Stoica 'Looking Up Data in P2P Systems' Communications of the ACM, Vol. 46, No. 2, February 2003, pp. 43-48
[47] Alan Mislove, Ansley Post, Charles Reis, Paul Willmann, Peter Druschel, Dan S. Wallach, Xavier Bonnaire, Pierre Sens, Jean-Michel Busca, and Luciana Arantes-Bezerra, "POST: A Secure, Resilient, Cooperativse Messaging System"
[48] Java Mobile Operation Management. See: http://www.jcp.org/jsr/detail/232.jsp
[49] J2EE Client Provisioning. See http://www.jcp.org/jsr/detail/124.jsp
[50] G. ABOWD, A. BOBICK, I. ESSA, E. MYNATT AND W. ROGERS: The Aware Home: Developing Technologies for Successful Aging. Proceedings of AAAI Workshop and Automation as a Care Giver – held in conjunction with American Association of Artificial Intelligence (AAAI) Conference. July. 2002..
[51] D. GELERNTER: Generative communication in Linda. ACM Trans. Program. Lang. Syst., vol. 7(1): pp. 80–112, 1985. ISSN 0164-0925.
[52] B. JOHANSON, A. FOX AND T. WINOGRAD: The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms. IEEE Pervasive Computing Magazine, vol. 1(2), Apr. 2002.
[53] MIT – MASSACHUSETTS INSTITUTE OF TECHNOLOGY: Project Oxygen. http://oxygen.lcs.mit.edu/, Nov. 2002.
[54] M. ROMAN AND R. CAMPBELL: GAIA: Enabling Active Spaces. 9th ACM SIGOPS European Workshop. Sep. 2000. Kolding, Denmark.
[55] M. WEISER: The Computer for the 21st Century. Scientific American, pp. 94–104, Sep. 1991.
[56] Website of the Simplicity project: http://www.ist-simplicity.org