WIRELESS WORLD
RESEARCH FORUM

The Simplicity System and its Demonstrator

G. Bartolomeo[(1)], N. Blefari Melazzi[(1)], J. Hamard[(2)], C. Noda[(2)], S. Salsano[(1)]

[(1)] University of Rome, Tor Vergata, [(2)] DoCoMo Communications Laboratories Europe

***Abstract*— As technology develops, people are using an ever broader and heterogeneous range of ICT devices and network-based services. The result is an enormous burden of complexity on the shoulders of users, service providers and network operators. The goal of the Simplicity project, supported by the European Union, is to reduce this complexity by: i) providing automatic customization of user access to services and the network; ii) automatically adapting services to terminal characteristics and user preferences; iii) orchestrating network capabilities.**

This paper presents the Simplicity project, briefly discusses its architecture and introduces its demonstrator and related applications.

***Index Terms*—service personalization, service portability, service adaptability, service discovery, user profile definition and handling, user mobility, auto-configuration of terminals, middleware, Smart Cards, Bluetooth.**

INTRODUCTION

The Simplicity project is a European Union program, scheduled to run for two years (January 2004 - December 2005) that includes 11 major European industrial organizations, network operators, SMEs, research labs and universities [1], [2].

The aim of this paper is to present the Simplicity demonstrator, which has been recently completed and is currently undergoing tests and measurements. Before presenting the demonstrator, we need to introduce the main aspects of Simplicity and of its architecture, which we will do in this introduction. Then, we will present key Simplicity components and the demonstrator.

The project acronym intends to convey the very aim of the project: design and deploy a brokerage level allowing i) easy personalization of services to match user preferences and needs, ii) seamless portability of services, applications and sessions across heterogeneous terminals and devices, and iii) smooth adaptation of services to available networking and service support technologies and capabilities.

The personalization concept is based on a user profile. In our view, each user will be provided with a personalized profile, giving access to different services, and using heterogeneous classes of terminals. The personalized user profile will allow automatic, transparent customization and configuration of terminals/devices and services, uniform mechanisms for recognizing, authenticating, locating and charging the user, policy-controlled selection of network interfaces and applications services. Thanks to the profile, users will also enjoy the automatic selection of services appropriate to specific locations (e.g. the home, buildings, public spaces), the automatic adaptation of information to specific terminal devices and user preferences, and the easy exploitation of different telecommunications paradigms and services.

The user profile will be stored in a so-called Simplicity Device (SD). Though it seems natural to think of the SD as a physical device (e.g., an enhanced SIM card, a Java card, a USB stick, a sensor, etc.), the SD could also be implemented as a network location or a software agent. In some case the physical SD could store “pointers” to

profile information residing in the network. If the SD is a physical device, users could personalize terminals and services by the simple act of plugging the SD into the chosen terminal.

The Simplicity system also encompasses a Brokerage Framework. This brokerage level 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 main components of the Simplicity system are the SD, the Terminal Brokers (TBs), the Simplicity Personal Assistant (SPA) and the Network Brokers (NBs). The Simplicity system can interact with existing ("legacy") application and services and with external applications which are designed to exploit the capability of the system (denoted as "Simplicity enabled 3rd party applications").

The role of the SD, as discussed above, is to store user's profiles, preferences and policies. It also stores and allows the enforcement of user-personalized mechanisms to exploit service fruition, to drive automatic adaptation to terminal capabilities, and to facilitate service adaptation to various network technologies and related capabilities.

The TB manages the interaction between the information stored in the SD and the terminal in which the SD is plugged in. The SD enables the TB to perform actions like adaptation to networking capabilities and to the ambient, service discovery and usage, adaptation of services to terminal features and capabilities. The TB caters also for the user interaction with the overall Simplicity system (including network technologies and capabilities).

The Simplicity Personal Assistant (SPA) represents the interface of the Simplicity systems towards the end-user. The SPA interacts with users via a convenient User Interface, assisting users towards completing their tasks. Its look, behavior and actions are strongly adapted to user preferences and needs. SPAs are meant to provide as much support as possible to the user. The SPA acts autonomously whenever it can, requiring only minimal input from the user. This entity also provides uniform access to the Simplicity System, and to the services it provides. More specifically the SPA is involved in many tasks, which include user authentication, management of user's preferences and also application related functionalities like session management, service subscription, adaptation (personalization) and invocation.

The NB has the goal to provide support for service advertisement, discovery and adaptation. Moreover, it orchestrates service operation among distributed networked objects, taking into account issues related to the simultaneous access of several users to the same resources, services, and locations. Other functions of NBs include sharing/allocating available resources, and managing value-added networking functionality, such as service level differentiation and quality of service, location-context awareness, and mobility support.

3rd Party Applications run on the user terminal and on other network-side entities. 3rd Party Applications use features provided by the Simplicity system through a specific interface, called Simplicity Applications Interface (SAI).

The interfaces between the identified entities must be clearly defined. In particular, three fundamental interfaces have been addressed: 1) the interface among the brokers, handled by means of a "Simplicity Asynchronous Event Protocol" – SAEP; 2) the interface between the brokers and the external applications willing to exploit the system, called "Simplicity Applications Interface" – SAI; 3) the interfaces between the TB and the SD, called "SD Access Interface" – SDAI. In addition to the interfaces, we need to specify the representation of the user profile information, which provides a common underlying information model for all the elements of the Simplicity architecture. This representation has been called "Simplicity User Profile" (SUP), extending the "Generic User Profile" by 3GPP [3]. More details on the Simplicity architecture can be found in [4] and in Simplicity deliverables (http://server.ist-simplicity.org/deliverables.php).

The core of the Simplicity System

The Simplicity Brokers

The Simplicity System is a set of software and hardware components that collectively provide users services. Simplicity Brokers (SBs) are software systems that enable easy and flexible integration of different functionalities. SBs are instantiated in user terminals (Terminal Brokers, TBs) and in

network servers (Network Brokers, NBs). Each broker consists of a central entity called Mediator and a number of subsystems attached to it. This approach enables flexible addition and deletion of subsystems, without affecting the rest of the system. It has the advantage that it allows encapsulation of new functionality within a Broker, without restricting its pre-existing functionality. Any subsystem can be plugged into any Broker, thus providing maximum flexibility for the deployment of functionalities.

Subsystems normally operate autonomously by utilizing a policy framework. When the execution of a task of a subsystem requires assistance from other subsystems, the Mediator provides the intelligence to coordinate their work. The Mediator does not contain any information about the state of subsystems and does not communicate to the external world. Whenever a new subsystem is added to a Broker, the subsystem has to register with the Mediator. The registration procedure consists in publishing all events that are sent and received by this subsystem. The Mediator is responsible for the filtering, adaptation and relaying of events between subsystems in form of asynchronous messages. There is no direct communication between subsystems; all communication is based on events. The Mediator uses a set of rules to determine which messages are to be forwarded to which receivers. The forwarding rule appropriate for a particular message is selected depending on the type of message and sender. A key feature of the Mediator is the ability to invoke policies for messages. “Event Distribution Policies” are used by the Mediator to decide how to process an event generated by a subsystem. Events always contain an “event context” which provides the information needed for a policy to evaluate the event. A Mediator can take several different actions in response to an event: the event may be rejected without further relaying or it may be forwarded to the intended receiver (with or without modification). It can also be sent to a remote Broker. This is done by passing the event to a specific subsystem, the Simplicity Broker Communication subsystem (SBC).

The Simplicity inter-broker communications

This mechanism is completely transparent to subsystems. Achieving this kind of transparency and loose coupling between subsystems requires two carefully planned policy mechanisms: one, allowing the Mediator to decide whether an event needs to be sent remotely, and another allowing the SBC to decide whether an event needs to be targeted to a specific Broker or broadcast to a group of Brokers. These policies are employed especially in two contexts: the discovery of a subsystem listening for a particular event, and batch discovery of all unmatched events in the Mediator.

The specific TB version of the SBC subsystem is very similar to the NB one, since both require the same core functionalities. The TB version however requires more flexibility in the configuration phase, since a TB is able to host different users, and each user owns a different profile concerning communication choices. Therefore, the SBC on the TB has an additional interface, so that all SBC configuration data for individual users is done using user’s profile information. The SBC reconfigures itself when the user logs out and another user logs in.

The Simplicity User Profile and the Simplicity Device

The Simplicity User Profile (SUP) has been designed taking into account two complementary views. The first one is represented by the 3GPP’s specifications on the Generic User Profile, in particular the Data Description Model (DDM) [3] which describes a syntax to express a “generic user profile”, meaning that it doesn’t mandate anything on the profile’s contents but gives some directives about its format (effectively XML-based). The second view was inspired by the so called Simplicity Information Model, which represents an attempt to address specific contents of profile information for 3G & beyond systems. The model, expressed in a number of UML class diagrams, is therefore independent of any specific application, protocol, platform, data storage and access technology; it addresses user, services, capabilities profiles and may be extended and customized, including also other information. Finally, a process of distribution has been applied to the result of this activity to obtain an entity, the SUP, whose data are physically distributed for efficiency, but semantically represents a whole, characterizing each user.

An obvious question is: “Where is the

profile stored?". Simplicity has developed a solution based on the so called Simplicity Device (SD), which is the "key" to Simplicity. The main role of the SD is to store user profiles, preferences and policies in a safe and secure way.

The ideal SD would have an unbounded, secure and reliable memory space to store profile data, powerful processing capabilities to manage it, minimal physical size and minimum weight to be accepted by the user. Simplicity didn't design a particular physical device to embed the ideal SD and let the implementer a wide range of freedom. The project has developed four prototypes using devices very familiar to end-users: i) a Bluetooth phone SD (BTSD), which exploits the memory, connectivity and processing capabilities of J2ME and Bluetooth-enabled phones [8]; ii) a Java Card SD (JCSD), which implements the SD as an applet deployed on a JavaCard [9]; iii) a Flash Memory SD (FSD) which uses memory cards or memory sticks and iv) a Virtual SD (VSD), consisting in a pure software implementation .

If the physical device does not have all the required capabilities, the SD can exploit functionalities provided by the TB and NB. This is possible through a subsystem called Simplicity Device Access Manager (SDAM) which exists in any TB. For example, when an implementation of the SD does not have enough storage capabilities or does not have memory at all (i.e. the VSD), data are stored in a network repository called Simplicity Data Storage, SDS. In this case, a pointer is used to link the SD to the location where the data are stored. The SDS, based on the Pastry technology [10], offers mechanisms through which it is possible to store persistent data, providing privacy and confidentiality guarantees, create replicas, taking care of data consistency, and always have the data available, regardless of the specific devices and access networks being used.

A very flexible mechanism supporting the distribution of profile data information between any SD and the SDS has been implemented and may be used to complement any implementation of the SD.

The SDAM uses so called controllers that provide a unique interface for developers, regardless of the particular SD implementation. A single TB may host more than one controller, allowing the use of different kinds of SD with the same TB. Depending on the nature of the SD, communication with the SD itself may be based on asynchronous messages over Bluetooth connections, exchanges of Java Card APDUs, or other mechanisms.

The SDAM offers to other Simplicity subsystems an interface based on the XQuery/XPath language specifications [5]. XQuery is both a data definition language and a data manipulation language; this allows a great degree of freedom on the kind of operations which is possible to perform on the XML representation of the SUP.

The Simplicity Personal Assistant

The Simplicity Personal Assistant (SPA) consists of mechanisms to proactively assist users in their interaction with the system. It supports personalization and adaptation for services and user interfaces, as well as automatic service subscription/invocation including configuration of services for used devices and session transfer across devices. The SPA allows 3rd parties to provide personalized user interfaces and applications and makes it possible to adapt user interfaces and applications to the current state of the environment and profile information stored in the SD.

The SPA interacts with users via a User Interface. The SPA, as well as 3rd Party Applications, is attached to the Broker System through the Simplicity Application Interfaces (SAI).

In the actual demo, the SPA User Interface comprises a core user interface for the terminal and a generic user interface library useable by any Simplicity subsystem or service.

The SPA class libraries provide a simple, clean and extensible user interface APIs supporting the implementation of Simplicity User Interfaces. A mechanism is also provided for integrating these user interfaces into the SPA. The SPA User Interface combines several functionalities, e.g., a desktop manager, a task manager and a system tray, to name only a few. SPA User Interface API uses the capabilities of Java user interface libraries (Swing and AWT).

A reduced version of the SPA, named miniSPA, has also been developed for Bluetooth phones embedding a SD (BTSD). The miniSPA uses MIDP [8] UI API.

The miniSPA provides various functions. One of the most important of these is the regeneration of a working XML instance representing the SUP information stored on the mobile phone (skipping data stored on

the SDS), thus allowing users to view and edit his data, even when there is no connection between the BTSD and the TB. Data changes are committed as soon as the BTSD connects to a terminal.

The SPA and the miniSPA provide also a functionality to automatically fill out input fields of downloaded web pages with user profile information. Based on analysis of name attributes and text strings shown on a web browser for input fields and order, the SPA calculates the probability of a matching filling for every input field and chooses the item with the highest probability to match that field. Compared to similar existing functionalities, (e.g. AutoFill provided by Google Toolbar [7]) it has two advantages: it runs also on a mobile device by supporting a light algorithm to fill out user data with the highest probability, and it does not require any action to service providers; in fact, it is just a plug-in application.

In order to assist users in their interaction with the system, the SPA uses a number of "core" Simplicity subsystems, which will be described in the following.

Other core subsystems in the Simplicity framework

Profile Management Subsystems

The Profile Management Subsystems are the way to access the Simplicity User Profile: they enable other Simplicity subsystems to retrieve, edit and modify user profile data.

In fact, though the SUP is stored on different data repositories (SDs and the SDS), it cannot be accessed directly by other Simplicity subsystem for two reasons: data presentation and security.

The Profile Management Subsystem interprets the requests coming from other subsystems and presents data in a suitable format which may be, depending upon the request, plain text, xml, or even custom structured data. In order to do this, it exploits the aforementioned interfaces exposed by the SDAM.

Another task of the Profile Management Subsystem consists of checking access rights. Each request coming from every subsystem contains the credential of the requester. The Profile Management Subsystem first checks these credentials, then checks the access rights associated to the requester and contained in the SUP; on the basis of these checks, it allows or denies accesses to profile data (this function is not yet implemented in the current demo).

There are many Profile Management Subsystems; one in the NB and in each TB . The one in the NB mirrors the Profile Manager in the TB and allows the network side systems to make queries inside the NB, without overloading the TB and its communication channel.

Policy Subsystems

One important goal of Simplicity is to build a flexible adaptive system. This can be achieved by "Subsystem Policies" which define choices in the behavior of a system, using context information.

Simplicity provides two core subsystems dealing with policies. The policy subsystem, representing the policy decision point, makes decisions based on a specific subset of context information and policies that are provided by other Simplicity subsystems.

The policy management subsystem is in charge of administrating all policies that are used inside the whole Simplicity environment.

Service Manager Subsystems

The functionality of the Service Subsystem on the TB is to request services from the Service Manager on the NB based on user's profile and on the needs of the moment. The Service Manager looks for the services from service registries and provides the Service Subsystem with the necessary information from the services found. Finally, if the user chooses to subscribe to a service, the SUP is suitably updated.

User contracts and pricing manager

This subsystem is responsible for providing information about costs of services and network accesses. As cost information are a very variable information, costs are not stored in the SUP, but in a database which is updated every time a service or a network provider decides to change the charging.

Capability Manager

The Capability Manager stores and provides the hardware and software configuration of the user terminal system. The complete terminal configuration is stored in an xml file.

Access Network Subsystems

The purpose of the Access Network Subsystem on the TB (ANS-TB) is to auto-

detect network parameters (IP address, subnet mask, gateway IP, etc.) of every terminal's network interface.

Inside the Simplicity User Profile (SUP), there is a section that stores information on every network connection subscribed by the user. After the user logs in, the ANS-TB, starts an "IP packet sniffer" and tries to find matches between the most probable network parameters as guessed by traffic sniffing, and the ones stored in the SUP. If it succeeds, the ANS-TB commands the underlying operating system to set the matched parameter for the corresponding network interface; otherwise it asks the user for further information. When the user logs out or removes his SD from the reader, it stops all sniffing activities and goes into a stand-by state. In the current demo, the ANS-TB runs only on Windows machine.

The Access Network Subsystem on the Network Broker (ANS-NB) is instead responsible to provide information about a given user's current connection with the system. The ANS-NB records users which log in and out from the system, and associate to each user an entry keeping connection information. It also may provide information about a given user's subscribed network connection, taking them from the user's SUP.

Location Manager

The Location Manager subsystem uses RFID technologies to provide information about a user's position. In order to use this service, laptops or PDAs have to be connected to RFID reader devices.

The Demonstrator Applications

This section describes some applications running in the Simplicity demonstrator. They are of four kinds: Simplicity Applications, explicitly developed for the Simplicity system; Web based Application, showing how Simplicity can be interfaced and complemented with web services and other technologies; External Applications, showcasing how an existing standalone application can be integrated with the system; Operator Centric Services, shifting the focus on the network operator's point of view.

Multimedia Messaging

Multimedia Messaging is a Simplicity Application, a simple instant messaging service. It allows the user to exchange text and pictures with remote users. Furthermore, it is possible to adapt the transferred pictures to the configuration of the terminal receiving the picture (size, color depth, etc.).

Multimedia messaging runs both on PCs and PDAs. User terminals allowed to transfer data between each other are said to be in a session. Each session has a unique identifier. When a user wants to join the session, he uses the session's identifier. Also, a user may invite another to join a session by selecting his contact. User's contacts are stored in the SUP as well. On the network side, a subsystem is in charge of deciding what kind of multimedia services should be offered (i.e. what data transfers between different terminal brokers are allowed) by collecting and calculating settings information for all users participating in a session.

The Multimedia Messaging service uses an IP Multimedia System (IMS) to transfer pictures and text from one terminal to another. Adaptation is done by a special part of the IMS named "Media Gateway".

MyPC

MyPC is a Simplicity Application and focuses on the ability of mobile users to access leased devices (e.g. desktop PCs). MyPC customizes the software environment according to the user's own settings (e.g. work or home).

The demonstrator presents the technical features of the personalization capabilities that can be implemented through the SD, working in combination with a network-based service supporting storage of privacy-sensitive user data. We considered representative application settings such as Microsoft Outlook Express' address book and Microsoft Explorer/Mozilla Firefox's favourites as well as OS customizable features, like desktop's wallpapers; MyPC acts on these applications and services to perform software environment reconfiguration by taking information from the user's SUP.

Home Entertainment Service

The Home Entertainment Service (HES) is a Web based Application and runs on a OSGi fully compliant gateway. It consists of three services: Media Streaming, Buy Media File and Personal Storage.

The web based Media Streaming service runs in a standard web browser. The service

is started from the SPA. The start site of the streaming service presents all media files, which pertain to the user, in a table with three columns: File, Action and Converted File. Possible actions are:

- Information: it presents to the user a media specific information page for a file, including a description of the media file, the recording date, the file size, etc.
- File conversion: it converts the original file in the wanted format, size and quality suitable for the user and the used device.
- Streaming: it starts the http streaming of the device dependent converted file.
- Store: it allows the user to store the file directly on the terminal device.
- Delete: it deletes the access to the file.

The SUP and the capability profile of the terminal are transferred over an http connection in an xml format to the Media Streaming Service.

The Buy Media File service provides to Simplicity users the capability to interact with a web portal to browse and acquire media files. In this scenario Simplicity operates as a middleware enhancing the users experience with the portal, by automating functions like auto-form filling, automatic login, automatic payment and transfer of media files to the user's personal storage space.

Users interact with the service through a web browser launched by the SPA when the user selects the corresponding menu. In parallel to the normal http based interaction between users and the service, a second Simplicity specific information exchange is established between the TB and the service, through the NB service specific subsystems.

The SUP is used by the service implementation in order to present personalized suggestions to the user and automatically fill in search boxes during the purchase. Finally, the files that the user selects for purchasing are automatically transferred to his personal storage service (described hereafter), during a process orchestrated by Simplicity.

The third feature of the HES is the Personal Storage Service. It provides a network storage space for user's files that are accessible through a simple web interface. This service has a specific buffering role in the overall HES scenario; it serves as the intermediate storage space for media files that users buy, before they are synchronized with their offline storage space of the Home Environment Server. The personal storage space retrieves the media files that are made available through the Buy Media File service and it notifies the home streaming server that new files are available. Then, the streaming server can implement a synchronization policy: retrieve the new files immediately or connect to the personal storage (for example, once a day) and retrieve the new media files.

Auto-form filling

The auto-form filling is a Web based Application that automatically fills in input fields displayed on Web pages (e.g. hotel reservation). It is based on the retrieval of user data (e.g. first name) from the SD or the SUP and is implemented as a plug-in application on a SPA-supported browser.

When the user accesses a HTML/XML file containing input fields, these are filled out with the user data according to certain probabilities based on several parameters (e.g. position of the input fields).

Three different color codes are used for expressing on the screen the level of uncertainty with regards to the probabilities of the input fields filling (i.e., higher probability in green, middle probability in yellow and lower probability in red). These color codes are supposed to support the user in checking the filled data before their final submission.

Tour Guide

Tour Guide is an external application (developed at Lancaster University). This context-aware application can be automatically personalized using the Simplicity framework. Tour Guide is an example of an application used by individuals who are not necessarily familiar with common ICT devices. Like MyPC, this application showcases also the use of the Simplicity framework to design and develop personalized services, which involve the use of equipment not owned by the user (in this case, the tourist guide devices). Unlike MyPC, the Tour Guide application was not originally developed for use with Simplicity. Thus, it is used in the Simplicity demonstrator to show how the Simplicity framework can handle such "external" applications.

Therefore, the Simplicity Tour Guide demonstrator has been developed in two phases. The first one saw the development of the core Tour Guide application. This initial version was customizable by directly soliciting personal preferences from users at start-up time. During this phase the

implementation was carried out without taking into account how this application might, later on, be integrated within the Simplicity framework. The second phase saw the interfacing and integration of the application with the Simplicity framework.

When the application is started, the user is presented with a number of options on how to explore a visited spot. The users may retrieve information about attractions, navigate through a city map, create and follow a tour.

There are currently two ways of personalizing the Simplicity Tour Guide demonstrator: i) by filling out a series of forms soliciting information about personal preferences from users; ii) by connecting a SD to the Tour Guide terminal device.

In the latter case, the application detects the presence of the SD on the terminal and, after being authorized from the user, it retrieves information from the user's SUP.

The Tour Guide application records various pieces of information while a user is employing the system, including information about visited places and the tours the user has followed. This information is periodically automatically synchronized with the SUP.

Campus Network

The Campus Network is an Operator Centric Service and shows the feasibility of a Simplicity-based access network control procedure in an 802.11b environment (e.g. an university campus). With this application, we move the focus towards the management of access network resources (operator side). To accomplish the access network control, a specific subsystem, named CNAC (Campus Network Access Control), has been developed and implemented on both TB and NB.

The goal of the access network control process is threefold. Firstly, it allows the operator to perform network access differentiation and load balancing, based on the user role. In more detail, so called "high priority" users (e.g. professors) have full access to all access points (APs), whereas low priority users (e.g. students) have access to some "restricted" APs and only if their congestion level is low. In addition, professors can always browse the web in a "normal mode", whereas students can be forced to use a "limited mode" in case of high traffic load. Second, the access network control service provides users with an automatic mechanism able to manage the network connection without requiring any effort from the user. The third advantage given to Simplicity users is that high priority users perceive a better service in case of network congestion.

In the current demonstrator, the user is made aware, through a panel of the SPA, of the following information: the AP his or her terminal is currently attached to and the current web surfing mode. The service starts automatically when the user logs in, and it stops when the user logs out.

To provide a user and network aware access control, the CNAC subsystems need to interface with a number of processes running in background, in the terminals and in the network, namely: the Wi-Fi manager and Web Surfing manager in the terminals and the Bandwidth monitor in 802.11b APs. The Wi-Fi manager interfaces and drives the terminal's network card driver. The Web Surfing manager is in charge to filter HTTP requests. Finally, the BW monitor has the goal to measure the amount of wireless bandwidth currently consumed at each AP.

The CNAC in the TB is in charge to retrieve user side information from the SUP and access network context (i.e., surrounding APs) by interacting with the Wi-Fi manager. Also, it is in charge to pilot, through the Wi-Fi manager, the network card towards the target AP.

The CNAC on the NB is instead in charge to retrieve: i) the access network context from controlled APs (i.e. the state of wireless network resources), ii) the user side information, iii) the terminal side network context from the CNAC on the TB. On the basis of this set of information, it takes decisions about target AP and web surfing mode and communicates them to the CNAC in the TB, which applies them through the Wi-Fi manager and Web Surfing manager.

The Demonstrator Scenario

Chat

Jan plugs his SD into his Laptop and enters his password. Once accessing the SPA environment, Jan launches the "Chat" application from the Services menu and starts a chat session with Clara and Bernat. Jan sends them a picture from his last vacation in Athens. Clara and Bernat receive a picture fitting their terminal screens.
Later on, Jan quits the "Chat" application.

Get AV

Jan launches the "Get AV" application from the Services menu in order to purchase a movie. The search field is automatically filled in based on the user's favourite artists and genres stored in his preferences. Jan selects the movie "Permanent vacation" from his favourite filmmaker Jim Jarmush. Jan is required to confirm his Credit Card details and then the movie is automatically downloaded to his Personal Storage Server.
Jan removes the SD from his laptop, which automatically logs him off.

Media Streaming

Arriving at home, Jan plugs his SD into his desktop computer and enters his password.
Jan opens the shortcut folder corresponding to his Personal Storage Server and accesses the movie "Permanent vacation" by streaming it through a media player.
Jan removes the SD from his desktop computer; which automatically logs him off.

Campus Network

Jan goes to the University Campus in order to access the Web and prepare a report. The Campus Network offers two wireless network accesses A and B. The Network A is opened to everybody; however, in case of network congestion, students and guest users can browse the Web in limited mode only (e.g. no pictures). Professors have always full access. The Network B is also opened to everybody; however, in case of network congestion its use is allowed only to professors.
Jan borrows a public computer to which he connects his SD, enters his password and gets automatically logged as a guest. At this time, the network A is the fastest one; and it is therefore automatically selected. Jan starts surfing on the Web but soon the network A gets more and more busy. Therefore Jan's connection is automatically and transparently switched to the network B. Jan goes on surfing the web.
Some hours later, Jan finally gets all the information he needs and simply removes the SD from the public computer, which automatically logs Jan off.

Hotel Reservation

On the way back home, Jan makes a hotel reservation for Edinburgh using his mobile phone. He launches his mobile browser and opens the bookmark of his favorite hotel reservation service. Jan's details (e.g. last name) are automatically filled in the webform displayed during the transaction. Jan briefly checks the correctness of the data and confirms the booking.

Tour Guide

Some months later, Jan is visiting the city of Edinburgh. He picks up a Tour Guide (TG) unit from the tourist information centre and connects his SD into it. Jan enters his password and gets automatically logged.
The user interface of the TG unit and the services it provides are tailored to Jan's preferences stored in his SD. According to these preferences, larger buttons are displayed on the screen and information about popular shopping areas are omitted. Jan starts a personalized tour through the city (i.e. avoiding shopping areas) but unfortunately, he does not have enough time to finish it. The TG system periodically updates Jan's personal profile with information about the places he has already visited so that Jan can easily resume his tour on the following day.
Before Jan returns the TG terminal to the tourist information centre, he simply unplugs the SD, causing the TG application to erase all Jan's personal data and to return to its default state.

ACKNOWLEDGEMENT

At this time of the project, near the end, the authors wish to acknowledge managers, designers, developers, technicians and all the people involved in Simplicity, whose contributions have been essential to the success of this project.

REFERENCES

[1] IST Simplicity project: http://www.ist-simplicity.org
[2] N. Blefari Melazzi et al. "The Simplicity Project: Managing Complexity in a Diverse ICT World", in Ambient Intelligence, IOS Press.
[3] 3rd Generation Partnership Project. Data Description Method (DDM) - 3GPP Generic User Profile (GUP). Technical specification of Technical Specification Group Terminals, Version 6.1.0. 2004.Reference example
[4] N. Blefari Melazzi, S. Salsano, G. Bartolomeo, F. Martire, E.Fischer, C. Meyer, C. Niedermeier, R. Seidl, E. Rukzio, E. Koutsoloukas, J. Papanis, I. S. Venieris: "The Simplicity System Architecture", 14th IST Summit, 19-23 June 2005, Dresden, Germany.
[5] XQuery 1.0: An XML Query Language, W3C Working Draft 04 April 2005, http://www.w3.org/TR/2005/WD-xquery-20050404/
[6] Jess Webpage, http://herzberg.ca.sandia.gov/jess/
[7] AutoFill, Google Toolbar, http://toolbar.google.com/autofill_help.html.
[8] J2ME Midp 2.0 specifications, http://jcp.org/aboutJava/communityprocess/final/jsr118/index.html
[9] JavaCard technology, http://java.sun.com/products/javacard/
[10] FreePastry Technology HomePage (http://freepastry.rice.edu/).

Appendix: images from the demonstrator

In this appendix, we present some pictures taken from the demonstrator.

Fig. 1. The Simplicity device in three different "flavors": Bluetooth phone+SIM card, memory card, Java card.

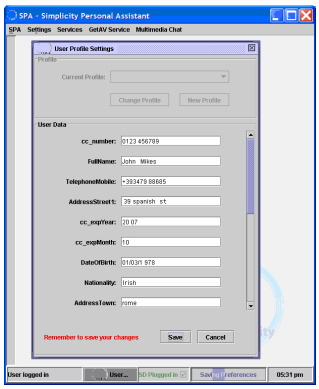

Fig. 2. Editing profile data using the SPA (running in a PC and in a mobile phone, respectively)

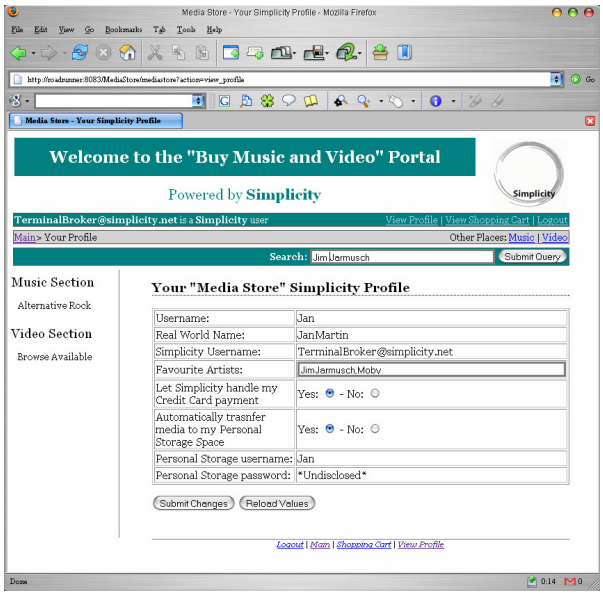

Fig. 3. Screenshots from Demonstrator Applications: the Home Entertainment Service