

# The Simplicity project and its demonstrator: improving ease of use and personalization of ICT services

G. Bartolomeo, N. Blefari Melazzi, F. Martire, S. Salsano  
DIE, Università di Roma "Tor Vergata",  
{giovanni.bartolomeo, blefari, francesca.martire,  
stefano.salsano}@uniroma2.it

S Kapellaki, E. Koutsoloukas, G. N. Prezerakos<sup>1</sup>, N.D.  
Tselikas, I.S. Venieris  
Intelligent Communications & Broadband Networks Lab  
School of Electrical & Computer Engineering,  
National Technical University of Athens  
{sofiak, lefterisk, prezerak, ntsel}@telecom.ntua.gr,  
venieris@cs.ntua.gr

*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. The Simplicity approach is based on a personalization device, the so-called Simplicity device, which, together with a brokerage framework, offers transparent service configuration and runtime adaptation, according to user preferences and computing/networking context conditions. This paper presents the Simplicity project, briefly discusses its architecture and introduces the Simplicity demonstrator: a proof of concept implementation used to verify our approach, through diverse usage scenarios. In addition, we introduce the main aspects of a follow-up project of Simplicity, named SMS.

*Keywords-* personalization, service adaptation, pervasive, context, simplicity

## I INTRODUCTION

The Simplicity project is a European Union program, which lasted 26 months (January 2004 - February 2006), and included 11 major European industrial organizations, network operators, SMEs, research labs and universities [1], [2].

The project ended on the 10<sup>th</sup> of February with a final review that classified it as "highly successful". The project web site contains papers, public deliverables, movies, presentations and other material that describes our work in the last two years.

The aim of this paper is to present the Simplicity demonstrator, which has been recently completed. 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 systems and finally the demonstrator.

The acronym, Simplicity, 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 complete profile information, which resides 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

<sup>1</sup> G. N. Prezerakos is also with the Technological Education Institute of Piraeus

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 in their tasks. Its look, behaviour and actions are 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).

Of course, 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>). Finally, we note that the reference list is short, as compared to all the issues faced in this paper (and in the project). This is due to space limitations. However, the project produced a thorough analysis of the state of the art of related technologies, standards and works, which is available in [5]. The document [5] quotes more than 200 references.

## II THE SIMPLICITY SYSTEM

### A. The Simplicity Brokers

Simplicity Brokers are software systems 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 loosely coupled subsystems attached to it, each one offering different functionality (see Fig. 1).

This design approach has the advantage that allows flexible encapsulation of a new functionality within a Broker, without restricting pre-existing functionality. The communication between the subsystems is asynchronous and event-based. It is performed through the Mediator which has the ability to invoke the appropriate event distribution policies according to the "event context" (message type and sender) in order to filter, adapt and relay events among subsystems. A Mediator, according to the enforced policy, may reject an event or forward it to the intended receiver (with or without modification). It can also send it to a remote Broker in a transparent-to-the-subsystems way. This is done by passing the event to a specific subsystem, the Simplicity Broker Communication (SBC). In this case, the SBC is required to decide whether the event needs to be targeted to a specific Broker or broadcast to a group of Brokers. At this point it must be mentioned that although the TB version of the SBC is very similar to the NB one (same core functionality) the TB has an additional interface so that all SBC configuration data for individual users is handled using user's profile information. The SBC reconfigures itself when the user logs out and another user logs in.

### B. The Simplicity User Profile and the Simplicity Device

The Simplicity User Profile (SUP) has been designed taking into account two complementary views regarding structure and content. The first one is represented by the 3GPP Generic User Profile, in particular the Data Description Model (DDM) [3], which describes a syntax to express a "generic user profile" (XML-based). The second view was inspired by the so called Simplicity Information Model (SIM), which defines the different types of user/services/capabilities information that a profile for 3G & beyond systems must contain. It is expressed in UML diagrams and is independent of any specific application, protocol, platform, data storage and access technology and can be extended to include additional information. The SUP data are physically distributed for efficiency, but semantically represents a whole, characterizing each user. The SUP is stored in the SD in a safe and secure way. The ideal SD would have an unbounded, secure and reliable memory space for storage, powerful processing capabilities to manage data and minimal physical size. 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 [9]; ii) a Java Card SD (JCSD), which implements the SD as an applet deployed on a JavaCard [10]; 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 (see Fig. 1).

The heterogeneity of the different SD implementations was addressed by a special subsystem residing in any TB, the Simplicity Device Access Manager (SDAM). The SDAM uses 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.

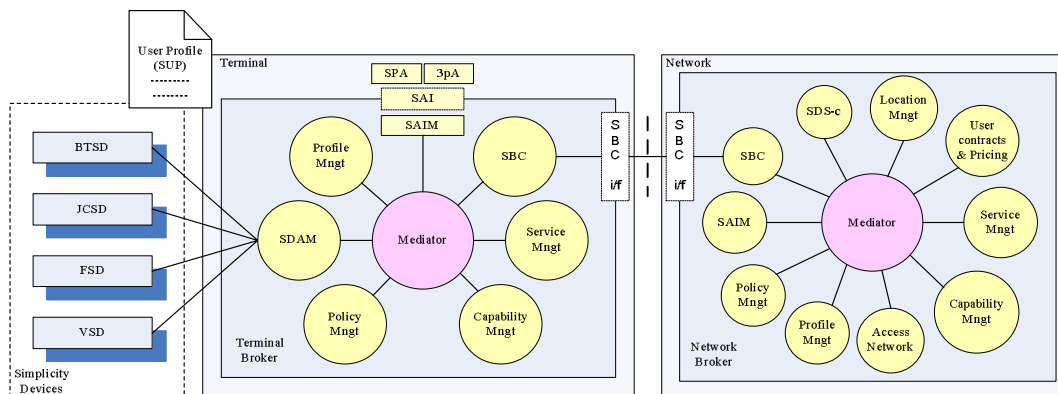


Fig. 1: The Simplicity System

The SDAM offers to other Simplicity subsystems an interface based on the XQuery/XPath language specifications [6]. 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 SDAM also enables the SD to exploit functionalities offered by the NB and TB. For example, when the SD has limited or no memory capacity at all, data are stored in a network repository called Simplicity Data Storage, SDS, and a pointer is used to link the SD to the location where data are stored.

### C. The Simplicity Personal Assistant

The Simplicity Personal Assistant (SPA) supported by a number of “core” Simplicity subsystems, offers 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/invoication, including configuration of services for used devices and session transfer across devices.

The SPA interacts with users via a User Interface and, similar to 3rd Party Applications, it is attached to the Broker System through the Simplicity Application Interfaces (SAI). The SPA includes a core user interface for the terminal and a generic user interface API to be used by other subsystems or services. The SPA User Interface combines several functionalities, e.g., a desktop manager, a task manager and a system tray. The SPA user interface API uses the capabilities of Java user interface libraries (Swing and AWT) so that applications and subsystems are able to present their own interface components to the user.

A reduced version of the SPA, named miniSPA, has also been developed for Bluetooth phones embedding a SD (BTSD). The miniSPA uses MIDP [9] UI API. One of the most important functions offered by the miniSPA 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 her/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, by analyzing name attributes and text strings shown

on a web browser and calculating the probabilities to match each field. Compared to similar existing functionalities, (e.g. AutoFill provided by Google Toolbar [8]) 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.

### D. Other core subsystems in the Simplicity framework

*Profile Management Subsystems:* Upon a subsystem request to retrieve/modify the SUP, the Profile Management Subsystem interprets the request, checks the requester credentials, checks the access rights associated to the requester and contained in the SUP and consequently allows or denies access to profile data. The profile data is then presented 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. Security and data presentation are in this way ensured.

*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 behaviour 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 that have been 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. Since cost information is very variable, 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 rules and/or parameters.

*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. 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 who 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 the required info 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.

### III SIMPLICITY APPLICATIONS

The Simplicity demonstrator comprises the Simplicity system described above, a number of Simplicity devices representing users, and a number of applications deployed by the project on top of the Simplicity system. There are four application categories; (i) applications explicitly developed for the Simplicity system; (ii) web-based applications, showing how Simplicity can complement existing web applications and enhance user experience; (iii) external applications, showcasing how existing standalone applications can be integrated with the system; (iv) operator-centric services, that shift the focus on the network operator's point of view. Fig. 2 depicts the demonstrator hardware and software environment. Key Simplicity applications are described below.

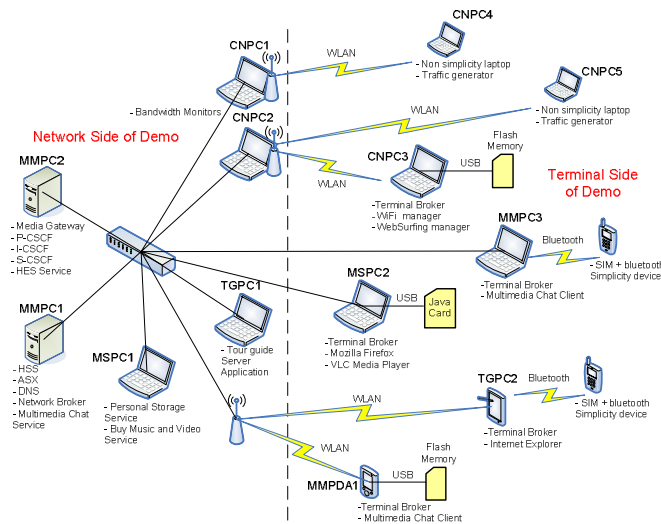


Fig. 2: Simplicity Demonstrator

#### A. Adaptive Multimedia Messaging

This is an enhanced chat application that allows users to exchange pictures and video, in addition to text messages. The application is adaptive in the sense that it exploits device and networking

capabilities, pricing information and user preferences to automatically adapt the exchanged media content in terms of quality, colour depth and size. The application uses the Simplicity framework to retrieve relevant information for each user. Information sources are the user's SUP for preferences and pricing and the Terminal Broker for device and networking capabilities. The information is collected by a dedicated "Context" subsystem at the Network Broker. The chat application was developed explicitly for Simplicity; however session management and media adaptation was provided by an external IMS solution and a proprietary Media Gateway respectively.

#### B. MyPC

MyPC is a Simplicity Application; it has the aim of assisting mobile users when they access leased devices (e.g. desktop PCs). MyPC customizes the software environment according to the user's own settings (e.g. work or home). Thanks to MyPC users find their usual settings and data wherever they go and on whatever PC. 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.

#### C. 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 colour 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 colour codes are supposed to support the user in checking the filled data before their final submission.

#### D. Home Entertainment Service

This service showcases how Simplicity can complement normal web applications and enhance users' experience with personalization features. The service consists of three parts, a web portal that sells media files, a home media streaming server and a network storage space for the purchased media files. Users interact with these services through a normal web browser, while all Simplicity specific interactions are done through an SBC specific information stream running in parallel to the normal http stream.

The web portal cooperates with Simplicity, in order to achieve features like automatic login, automatic filling of search boxes with preferred artists and credit card information forms,

personalized media proposals and automatic transfer of purchased content. All personalization and payment information is provided to the service by Simplicity.

The home media streaming server provides adaptive streaming capabilities when the user is connected to the home network. Through Simplicity, the service acquires the device capabilities of the terminal that the user is currently using and it adapts the available media files in order to meet resolution, size and quality requirements of the terminal at hand.

Finally, the network storage space service presents a web based file manager to the users, which contains the purchased media files. The network storage accepts the media files that users purchase from the web portal and it synchronizes itself with the home media streaming server.

#### E. Tour Guide

The Tour Guide is an example of how an existing legacy application, can be adapted to exploit the Simplicity vision. It is an application that offers personalized guidance for visitors of the city of Lancaster. Visitors are provided with a terminal device that runs the guide; with the guide they can receive information about attractions, navigate through a city map, create and manage a personalized tour.

In the initial application version, the personalization information had to be provided by users when the terminal devices are handed to them. The project integrated the terminal and software of the guide with Simplicity, so that personalization information can be provided automatically by the simple act of plugging in the SD. While users are interacting with the guide, the application reads information from the user's SUP and it also records visited places and user actions in order to periodically update the SUP.

#### F. Campus Network

In order to strengthen the operator interest in deploying a Simplicity system, the Campus Network application focuses on load balancing in wireless networks and is an example of how the Simplicity system can be integrated with an operator's 802.11g infrastructure, to perform access network control functions.

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.

The CNAC subsystems utilize information coming from the SUP (e.g. user role) and from bandwidth monitors installed in APs in order to interact with terminals and apply the operator's traffic policy. Interaction with terminals is done either through Wi-Fi

managers that operate directly on the user's network card or with Web Surfing managers that filter the terminal's HTTP traffic.

### IV THE SIMPLICITY SCENARIO

The Simplicity demonstrator has been shown in several public events, including conferences and meetings of standardization fora. To improve such presentations we conceived a "story", a scenario in which a user, Jan, exploits the Simplicity applications. It was this story that it was shown to the audience, before presenting specific, technical details:

*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 on the basis of Jan'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 open to everybody; but 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. 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 (the PC is also personalized with Jan's settings). 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 favourite hotel reservation service. Jan's details (e.g. last name) are automatically filled in the web form 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 is 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.

## V A FOLLOW-UP PROJECT: SMS

The work performed by the Simplicity project will be continued in a follow-up project, named Simple Mobile Services (SMS). The SMS project, co-funded by the European Union, started in June 2006 [12]. The rationale of the project is the following: a key factor in the success of the Web is that not only it is very easy to "use" it but also it is rather simple to create a web page, in an open environment. This means that a very large base of people can create Web pages and that a vast amount of imagination, culture and ingenuity is at work every day to continuously expand the contents and the services available on the Web. This is not true for current mobile services.

If mobile services are to repeat at least in part the success of the Web, they have to be simple to find, simple to use, simple to trust and *simple to set up*. These are our design goals for SMS. Like the Web, SMS will provide technology and operator-independent end-to-end connectivity. But unlike conventional Web-based services, our services will target specific locations visited by specific classes of mobile user with specific needs. Above all, they will be easy to create.

To reach these goals, SMS will combine: i) Simplicity's inheritance (especially the concept of user profile and Simplicity device); ii) localization technologies and tools; iii) customizable security, privacy and trust features; iv) new service authoring tools. Currently, in the context of SMS, we are investigating the possibility to integrate the Simplicity architecture into existing middleware solutions. This is achieved by an abstraction layer (Simplicity Middleware Independent Layer, SMILE) which models the asynchronous message exchange between the Simplicity subsystems and maps it to message facilitations provided by the concrete middleware in use. A reduced version of the Simplicity demonstrator has been already ported to two different middleware platforms, JXTA and JADE.

Additionally, in SMS, the security aspect of the framework will be further investigated, since in Simplicity security, privacy and trust were not among the project's goals.

## VI CONCLUSIONS

In this paper, we have described a proof-of-concept system realizing our Simplicity vision for existing and emerging ICT environments. First of all, the demonstrator has shown the feasibility of the universal hardware personalization device concept. The Simplicity system offers to applications adaptation and personalization features, based on an extensible, open and

standardized Simplicity User Profile (SUP), conveniently expressed as an XML construct. The SUP becomes the single source of personalization information, thus easing the spread of personalized services. Additionally, the demonstrator has shown that it is possible to support the widest range of terminals by developing different types of hardware devices.

Regarding applications, the Simplicity demonstrator has proved that the Simplicity architecture is versatile enough to support both Simplicity specific and external applications. Half of the involved applications were developed outside Simplicity and were later integrated in the project, showing that with a minimal effort it is possible to adapt any service so that it exploits the personalization features of the framework.

In addition, the demonstrator has materialized usage scenarios where users are relieved of technical knowledge requirements. In applications like the Adaptive Multimedia Chat, Simplicity transparently and silently ensures that the exchanged media files can be properly processed in any computing and networking environment. Other applications, with the Automatic Form Filling being the prominent example, relieve users of painful repetitive tasks, like filling out web forms and others, like the Tour Guide and MyPC applications, offer personalization at no cost in time or effort for the user. Last but not least, the project performed tests and measurements on the demonstrator to assess: i) the feasibility of our approach; ii) the usability, ergonomics, and human (and social) impact of the services provided by Simplicity; iii) some basic performance of the Simplicity system [11].

## ACKNOWLEDGMENT

The authors wish to acknowledge all the people involved in Simplicity, whose contributions have been essential to the success of this project. In particular, the authors wish to thank their partners from Ludwig-Maximilians Universität and DoCoMo Euro-Labs for developing the Automatic Form Filling application, DoCoMo for the development of the demo story, Lancaster University for integrating the Tour Guide application, Siemens Business Services for integrating the Home Environment Server, Siemens AG for integrating the IMS and Media Gateway servers and VTT for porting the Simplicity framework to the PDA.

## 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 et al: "The simplicity system architecture", 14th IST Summit, 19-23 June 2005, Dresden, Germany.
- [5] Simplicity Deliverable D2103 "Final Use Cases and Business Models" , <http://server.ist-simplicity.org/deliverables.php>
- [6] XQuery 1.0: An XML Query Language, W3C Working Draft 04 April 2005, <http://www.w3.org/TR/2005/WD-xquery-20050404/>
- [7] Jess Webpage, <http://herzberg.ca.sandia.gov/jess/>
- [8] AutoFill, Google Toolbar, [http://toolbar.google.com/autofill\\_help.html](http://toolbar.google.com/autofill_help.html).
- [9] J2ME Midp 2.0 specifications, <http://jcp.org/aboutJava/communityprocess/final/jsr118/index.html>
- [10] JavaCard technology, <http://java.sun.com/products/javacard/>
- [11] Simplicity Deliverable D4202 "Report on system validation and evaluation", <http://server.ist-simplicity.org/deliverables.php>
- [12] IST SMS project: <http://www.ist-sms.org/>