

# The Simplicity Project: Managing Complexity in a Diverse ICT World

N. Blefari Melazzi<sup>1</sup>, G. Bianchi<sup>1</sup>, G. Ceneri<sup>2</sup>, G. Cortese<sup>3</sup>, F. Davide<sup>3</sup>, N. Davies<sup>4</sup>, N. Dellas<sup>5</sup>, E. Fischer<sup>6</sup>, T. Frantti<sup>7</sup>, A. Friday<sup>4</sup>, J. Hamard<sup>8</sup>, M. Helbing<sup>9</sup>, S. Kapellaki<sup>5</sup>, K. Kawamura<sup>8</sup>, W. Kellerer<sup>8</sup>, E. Koutsoloukas<sup>5</sup>, C. Meyer<sup>6</sup>, C. Niedermeier<sup>6</sup>, C. Noda<sup>8</sup>, J. Papanis<sup>5</sup>, C. Petrioli<sup>10</sup>, E. Rukzio<sup>11</sup>, S. Salsano<sup>1</sup>, Robert Seidl<sup>6</sup>, O. Storz<sup>4</sup>, J. Urban<sup>9</sup>, I. S. Venieris<sup>5</sup>, R. Walker<sup>2</sup>

<sup>1</sup> *DIE, Università di Roma "Tor Vergata",  
{blefari, giuseppe.bianchi, stefano.salsano}@uniroma2.it*

<sup>2</sup> *Radiolabs, Consorzio Università Industria – Laboratori di Radiocomunicazioni, Roma,  
{gianni.ceneri, richard.walker}@radiolabs.it*

<sup>3</sup> *Telecom Italia Learning Services, g.cortese@computer.org*

<sup>4</sup> *Computing Department, Lancaster University,  
{nigel, adrian, oliver}@comp.lancs.ac.uk*

<sup>5</sup> *National Technical University of Athens, School of Electrical and Computer Engineering, Intelligent Communications & Broadband Networks Laboratory,  
{ndellas, sofiak, lefterisk, jopapan}@telecom.ntua.gr, ivenieri@cc.ece.ntua.gr*

<sup>6</sup> *Siemens Corporate Technologies, Otto-Hahn-Ring 6, D-81739 München, Germany,  
{Elisabeth-Anna.Fischer, Carsten.Meyer, Christoph.Niedermeier}@siemens.com*

<sup>7</sup> *Technical Research Centre of Finland (VTT), Kaitoväylä 1, 90571 Oulu, Finland,  
tapio.frantti@vtt.fi*

<sup>8</sup> *DoCoMo Communications Laboratories Europe,  
{hamard, nick, kellerer, noda}@docomolab-euro.com*

<sup>9</sup> *Siemens Mobile, Sankt-Martinstraße 76, D-81541 Munich, Germany,  
{Michael.Helbing, Josef.Urban}@siemens.com*

<sup>10</sup> *CS Department, Rome University "La Sapienza", e-mail: petrioli@di.uniroma1.it*

<sup>11</sup> *Department "Institut für Informatik", Ludwig Maximilians University, Munich, e-mail:  
Enrico.Rukzio@informatik.uni-muenchen.de*

**Keywords** - service personalization, service portability, service adaptability, service discovery, user profile definition and handling, user mobility, auto-configuration of terminals, middleware, brokerage functions, orchestration of network resources, Smart Cards, high-layer re-configurability, Bluetooth, mobile phones as general-purpose devices.

## Table of Contents

1	Introduction.....	3
2	Driving Concepts.....	4
3	Simplicity Device: short-term and long-term scenarios.....	5
4	User scenarios.....	9
4.1	A brief Description.....	9
4.1.1	Mobile Worker and Gaming.....	10
4.1.2	Car / Travel / Shopping.....	11
4.1.3	The Global Health System.....	12
4.1.4	Buy and Use a Self Learning Simplicity Device.....	13
4.2	Methodology for User Scenario Analysis.....	15
4.3	Requirements common to all scenarios.....	15
5	Requirements.....	16
5.1	Methodology for deriving requirements.....	18
5.2	Preliminary list of requirements for the SD.....	18
6	System Architecture.....	19
6.1	Horizontal view.....	19
6.2	Vertical view.....	20
7	System Functionality.....	21
7.1	Functionality of the Simplicity Device.....	21
7.2	Terminal Broker functionality.....	22
7.3	Network Broker functionality.....	23
7.3.1	Service Deployment/Advertisement.....	23
7.3.2	Service Adaptation.....	24
7.3.3	Network Side Handling.....	24
7.3.4	User Profile.....	24
8	State of the Art.....	25
8.1	Personalization and User Profiles.....	25
8.2	Simplicity Device.....	27
8.2.1	USB Devices.....	27
8.2.2	Smart Card Technology.....	28
8.2.3	Java Card Platform.....	29
8.3	Flexible Network Support.....	29
8.3.1	Policy-based brokerage frameworks.....	30
8.3.2	Mobile Agent Platforms.....	30
8.3.3	Service Discovery Frameworks.....	31
8.3.4	Simple Storage Management.....	31
8.4	Ambient Intelligence.....	32
9	Conclusions.....	33

## Abstract

As technology develops, people are using an ever broader range of heterogeneous ICT (Information and Communication Technology) devices and network-based services. New areas of research, such as pervasive computing, will further increase the diversity of the devices and services with which users have to deal. The result is an enormous burden of complexity for users, service providers and network operators. This creates obstacles to effective exploitation and acceptance of Beyond 3G systems such as ambient intelligence, context-aware services and novel access technologies. The goal of the Simplicity project 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.

## 1 Introduction

The Simplicity project is a European Union research program, scheduled to run for two years, from January 2004 to the end of 2005 [1]. Simplicity stands for Secure, Internet-able, Mobile Platforms Leading Citizens Towards Simplicity. But “simplicity”, as the acronym suggests, is also the aim of the project. The goal is to develop and evaluate a series of tools, techniques and architectures enabling users to customize and use devices and services with minimal effort.

A trans-European project, Simplicity brings together a combination of expertise from eleven major European industrial organizations, network operators, SMEs, research labs and universities [1].

The Simplicity project started from a vision: users today employ a variety of different terminals and devices to access a range of different “services” in the home, in buildings or in public spaces, for example, communications, computing capabilities, security support etc.. Some services may be as simple as remote control of an entertainment device (e.g., a television) via a wireless link, or access control to a building. Others can be very complex and may require location awareness, QoS support, message exchanges with network databases, structured interaction with remote networking devices (e.g., media gateways), etc.. The emergence of new research areas, such as pervasive computing, will further increase the diversity of the devices and services with which users have to deal.

But already today, users who attempt to exploit the services on offer have to deal with multiple procedures for configuring devices, multiple authentication mechanisms and passwords, multiple billing and payment procedures, multiple access technologies and protocols. This creates an enormous burden of complexity (as well as the physical burden of carrying different devices). This complexity is likely to limit the effective exploitation of the wide range of access, virtual reality, ambient intelligence and context-aware solutions currently under study and development. It will deepen the digital divide, making it difficult for non-technical users to benefit from new developments. And of course it will also create difficulties for network operators, who will be forced to manage the complexity of a multi-access networking environment.

The strategic goal of Simplicity is to simplify the process of using current and future “services”. More specifically, the project aims to 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.

With these goals in mind, Simplicity will:

- describe user scenarios and business models for the Simplicity approach;
- explore new brokerage mechanisms and policies in a multi-access networking environment;
- design a universal multi-application Simplicity Device providing users with a simple and uniform mechanism for customizing services and terminals;
- validate the feasibility and benefits of the Simplicity approach;
- contribute to relevant standardization bodies.

In this chapter we will present the main ideas and rationale behind the Simplicity project, outlining a number of ideas on applications scenarios and research directions developed during the initial months of the project. We will begin by illustrating the driving concepts behind the project and a number of user scenarios. These will be used to derive system requirements, which in turn will determine the system architecture and the related functional architecture. The chapter concludes with an analysis of the state of the art in technologies and standards relevant to Simplicity.

## 2 Driving Concepts

An important feature of 2G wireless systems, e.g., GSM, is the portability of user identities between different terminals (i.e., mobile phones). In what follows, we propose a generalization of this concept, allowing users to move seamlessly between different distributed applications and services using heterogeneous networking technologies and devices. This approach, we suggest, could provide a user-friendly solution to the challenges posed by a diverse service and technology environment.

The personalization concept is based on the concept of a “user profile”. Each user will be provided with a personalized profile, providing access to different services, perhaps using different classes of terminals. Creating and maintaining the user profile will involve the automatic processing of behavioral information (though the user will be able to switch off automatic storage and/or delete specific information). More refined policies on how to handle specific types of personal information will be part of the user profile and will be controlled by the user. Full control of personal data, security of information, and user privacy are key issues for the Simplicity approach.

The personalized user profile will allow: i) automatic, transparent customization and configuration of terminals/devices and services; ii) uniform mechanisms for recognizing, authenticating, locating and charging the user; iii) 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.

Depending on users’ characteristics, preferences and abilities, personal profiles could take the form of 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 high-level description language.

The user profile will be stored in a so-called Simplicity Device (SD). Alternatively storage on the Simplicity Device might be limited to a “pointer”, making it possible to download the profile from the network. Though it seems natural (from our own everyday experience of 2G systems) to think of the SD as a physical device (e.g., an enhanced SIM card, a Java card, a Java ring, a USB pen, etc.) the SD could also be implemented as 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 into the chosen terminal (see Figure 1).

One of the main novelties of the SD is that it is not tied to a single networking environment, or to a single class of user terminals. The SD will provide all the information necessary to adapt services to the characteristics of the terminal, the nature of the environment and the user's personal preferences. This means, for instance, that:

- different users plugging their SDs into the same laptop will see different working environments, file systems, software tools, connection services, etc;
- a given user who plugs the SD into different terminals will always see the same personalized working environment (adapted to the characteristics of the terminal);
- users will be able to suspend and resume running applications/sessions by simply unplugging/plugging the SD; when the user moves from one terminal to another the application/session automatically adapts to the new context.

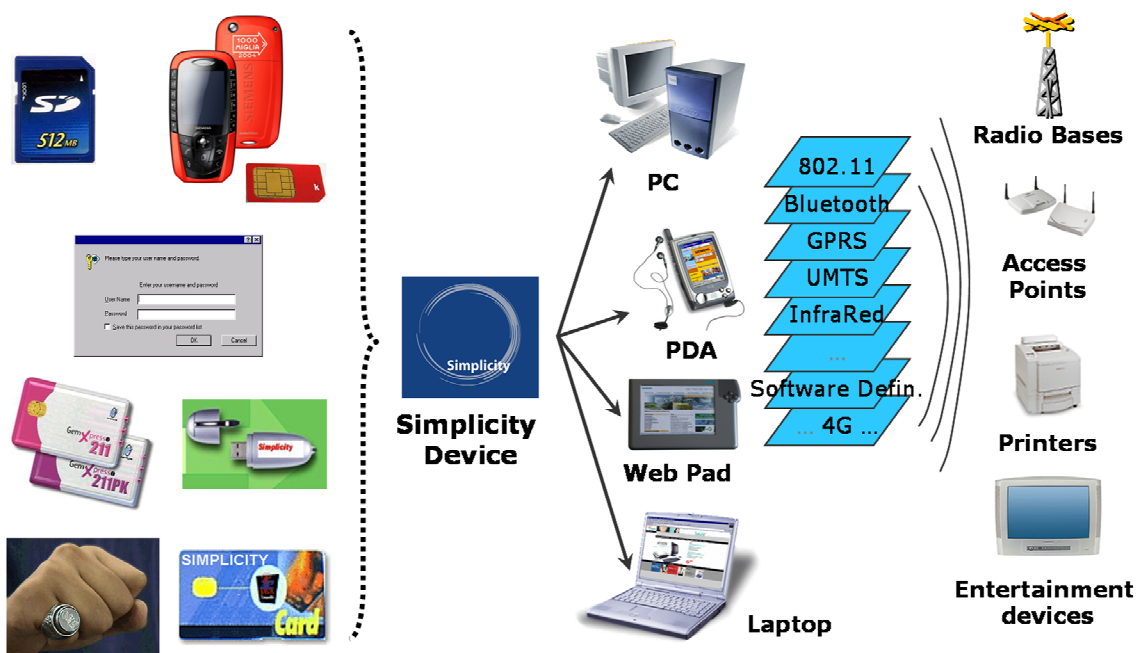


Figure 1: Overall reference scenario

In brief, the core functions of the Simplicity framework include automatic terminal and network configuration, location-based services, authentication, authorization and accounting services, service discovery, as well as user interface and content adaptation to terminal capabilities. Services and applications built on top of core Simplicity features will provide users with a simple way to perform complex tasks, allowing them to take full advantage of a multitude of terminals supporting modern communication facilities without having to perform complex configuration decisions.

### 3 Simplicity Device: short-term and long-term scenarios

The key role of the Simplicity Device is to store user profiles, preferences and policies (and possibly data), in a secure way. Potentially, it will be a “universal device”, compatible with a broad range of terminals (including both legacy and emerging devices), and with the

ability to interact with many different environments. Ideally, the SD should meet the characteristics listed in Table 1 below.

**Table 1: Ideal characteristics of the Simplicity Device**







Characteristic	Comment
<ul style="list-style-type: none"> <li>• A “universal” device, compatible with a broad range of terminals</li> </ul>	This is difficult, since we are talking of terminal devices ranging from phones to PCs to home equipments
<ul style="list-style-type: none"> <li>• Small, light-weight and low cost</li> </ul>	Not exceeding a credit card
<ul style="list-style-type: none"> <li>• High level of user acceptance</li> </ul>	Users must think of it as a familiar and trusted object
<ul style="list-style-type: none"> <li>• Include processing power</li> </ul>	Needed to i) run security algorithms; ii) start user sessions when physically plugged into terminals, without necessarily requiring actions on the terminal itself (bootstrap function); iii) complement terminal host capabilities, if needed (e.g., a display or an actuator)
<ul style="list-style-type: none"> <li>• Largest possible storage capability</li> </ul>	To store user data as well as profiles and preferences
<ul style="list-style-type: none"> <li>• Easily configurable</li> </ul>	To allow users to quickly and simply install and configure profiles, perhaps via a GUI
<ul style="list-style-type: none"> <li>• Able to become “virtual”</li> </ul>	To give users the choice of having a physical or a virtual personification of their preferences (e.g., network location or a software agent)






At first glance, Table 1 may be seen as nothing more than a wish list. It is nonetheless perfectly possible with current technology. The real issues are the following: i) will manufacturers be willing to combine all the facets of our ideal Simplicity Device into a single object? ii) will all terminal devices have slots to accept this SD? iii) will this SD enjoy widespread user acceptance?

On the one hand, we are confident that current trends go in the direction traced by this approach. On the other, we would like to find an easily implemented solution, which we can bring to market as quickly as possible. Thus, as a project, we want to follow two parallel routes: a mid/long term one and a short term one. In turn, the mid/long term solution includes two sub-options: i) assume to have a single object with all the aforementioned characteristics; ii) design a Simplicity Device concept able to be “adapted” to several instances of an idealized single solution.

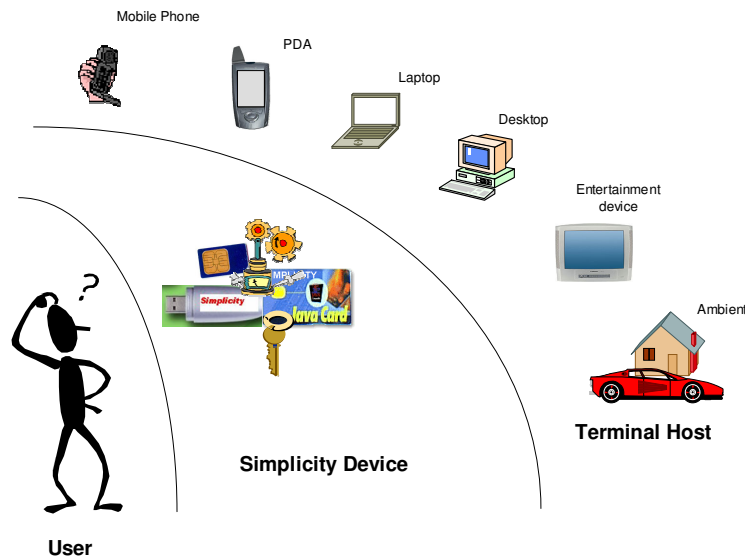
To specify the mid/long term scenario and to identify a short term solution, it is useful to analyze devices that are currently available “off the shelf”, to play the role of Simplicity Device. In this light, Table 2 summarizes pros and cons of some of these devices.

**Table 2: Pros and cons of currently available devices**

Candidate device	Pro	Cons
Smart Card 	 : it has processing power	 : limited storage; difficult to plug in into current devices; hard to say if future PCs, home equipments and phones will have a reader
USB stick 	 : huge amount of storage; easy to plug in into PCs;	 : no processing capability; some PDAs use USB, phones are more difficult (some phones have a USB “slave” port)

<p>Secure Digital memory card</p> 	<p>😊: similar to a USB stick, but perhaps easier to use in PDAs and mobile phones;</p>	<p>☹️: no processing capability; difficult for phones</p>
<p>Virtual card (or biometric identification)</p>   	<p>😊: very "light-weight"</p>	<p>☹️: no storage nor processing capability in itself; requires Internet connection; requires "smart" users; difficult to envisage the interaction with mobile phones and other relatively "dumb" devices; security issues: will users consider it a trustable object? In case of biometric identification, similar considerations apply, added to the necessity of adaptors</p>
<p>GSM SIM</p> 	<p>😊: users think of it as a familiar and trusted object;</p>	<p>☹️: not easy to plug in into current PC and laptops; if the same SIM is used for the mobile phones and for other devices, it would be necessary to plug the SIM in and out the phone; this could be annoying for users; limited storage; no processing capability</p>
<p>Sensor (e.g., Mote with TinyOS)</p> 	<p>😊: easy and powerful interaction with the environment</p>	<p>☹️: necessity of adaptors</p>

None of the solutions in Table 2, provide a genuinely convincing solution for the very short term.



**Figure 2: The Simplicity Device Dilemma**

As regards the short term scenario, to find a solution, we have to take the concept of the Simplicity Device to extremes. The original idea of the SD was to store user preferences and data in a single device (possibly smaller than a laptop...), allowing users to personalize several different kinds of terminals/devices as well. This would take us towards a storage and processing device. We could go further along this path and arrive at a virtual card solution, but in this way we would lose the concept of the SD as a physical device: a distinctive pro in many situations, and for many users. What can we find in between a completely immaterial solution and a powerful-yet-small-card with storage and processing power? What we are looking for, at least in a short term perspective, is a “trigger”: a device that can activate, prompt, and elicit reactions from more powerful components with more memory. From this point of view, the most common and widespread trigger we know is the GSM SIM: a universally accepted and trusted device.

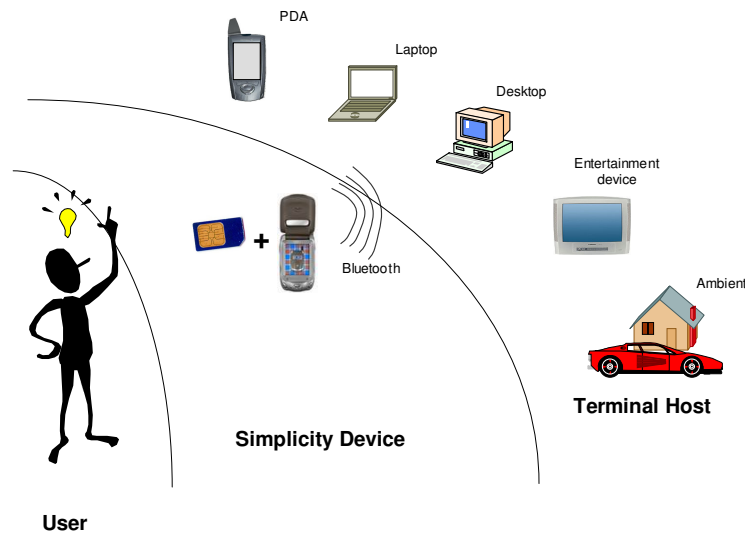
Taken in itself, the GSM SIM is rather “autistic” (or too specialized) and would need specific adaptors to reach terminals other than mobile phones. However, the SIM could use mobile phone’s capabilities to transfer data related to its role of Simplicity Device: Bluetooth, IR or even Internet connections (via GSM/GPRS/UMTS).

Thus, the short term implementation of the Simplicity Device could consist of a suitably adapted GSM SIM, installed in a phone equipped with Bluetooth/IR (though in some contexts, this solution could be replaced by/combined with other implementations). Bluetooth/IR allows the SIM to interact with other devices (e.g., PCs). In addition, a smart phone can provide additional processing power, current SIMs are Java-enabled, typical SIMs contain 64K memory and 128K SIMs are already on the market. Hence a SIM+smart phone is not very far from the ideal characteristics of the storage and processing device outlined above.

Probably, the strongest point of this solution is that it is very good from the point of view of user acceptance. Users already trust their SIM to store sensitive identity and charging data and are used to carrying their mobile phones wherever they go; there is thus no need for them to carry additional devices. Finally, mobile operators could be interested in selling or offering Simplicity in their SIMs, complementing their current services. With the SIM + Smart Phone as a SD we can also imagine complementary solutions, in which the GSM SIM provides also access to storage on a USB stick (carried by the user) or on the network. All that the user would have to do to “bootstrap” Simplicity, would be to walk



near a PC with a suitably equipped phone (and perhaps plug in a USB stick, if a large amount of personal data is needed as well).



**Figure 3: An interim solution: SIM + Bluetooth Mobile Phone**

To summarize, our proposed system will be based on a so-called Simplicity Device. The project has not yet decided a medium/long-term strategy for the implementation of the SD. In the short term, key characteristics and functions of our solution could be implemented, demonstrated and applied using an SD based on a Bluetooth-equipped GSM phone.

## 4 User scenarios

### 4.1 A brief Description

Organizations like the WWRF (Wireless World Research Forum) are currently outlining visions for systems beyond 3G [2], [3]. These visions emphasize that the successful design of future mobile networks will require a user-centered approach. The WWRF report introduces the concept of personalized communication spaces and points out the need to consider the user context, that is user goals, activities, tasks and focus of attention, users' physiological and emotional state, their location, their social and physical environment (e.g. the identity or properties of surrounding people, objects, and buildings), as well as the availability of resources (services, devices and networks). Systems beyond 3G will support many different classes of terminal and network technology. This in turn will increase the complexity of the system, bringing with it the risk of user dissatisfaction. Our Simplicity solution intends to mitigate these difficulties.

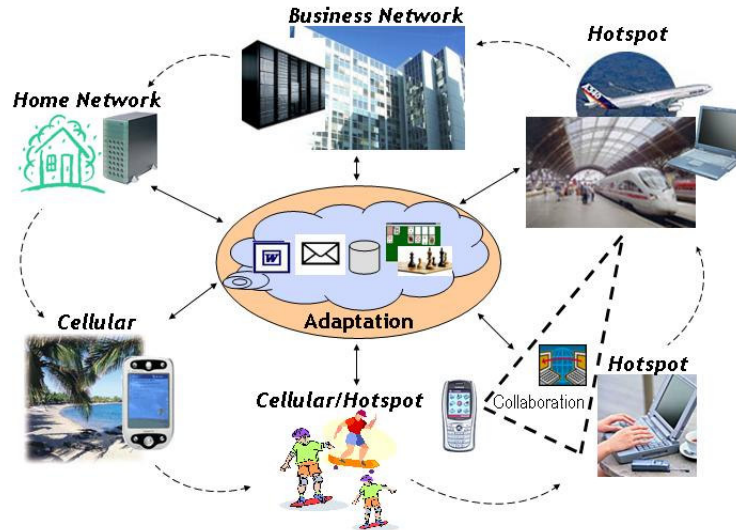
Definition of user scenarios is a common technique for illustrating user requirements and a good basis for identifying system features, discovering alternative solutions and deriving business models. During the initial phase of Simplicity, the project partners developed 25 user scenarios. Below we describe four of these, entitled, respectively:

- Mobile Worker and Gaming
- Car / Travel / Shopping
- The Global Health System
- Buy and Use a Self Learning Simplicity Device

In this section, we describe the scenarios; in section 4.2 we outline the methodology used to analyze them; in 4.3 we identify a set of basic requirements common to all.

#### 4.1.1 Mobile Worker and Gaming

The key features of this scenario are presented in Figure 4.



**Figure 4: Mobile Worker and Gaming**

- Every morning Brad plugs his Simplicity Device (SD card) into his mobile phone to access his company network. A Bluetooth connection means he can connect to a desktop PC without having to plug the device in and out. A Simplicity Personal Assistant (SPA) selects one of the available office working places according to Brad's preferences. The SPA automatically configures Brad's personalized business environment with a uniform login procedure and with access to his business applications, data storage systems and to the peripherals (e.g., printers), available at his location. The SPA adapts the services provided by the company network to the capabilities of the work environment and to the user profile.
- At noon, Brad has to attend an important customer meeting. He orders a train ticket via his office PC. The SPA uses Brad's personal travel profile to choose the best connection, taking into account his calendar and meeting information. It prepares the booking via the company travel department. Brad checks the system proposal and confirms the booking.
- When Brad leaves the office for his business trip, his SPA automatically saves the status of the applications he was just working on and performs the logout procedure.
- During his business trip, the SPA detects many different location-based services and filters them according to Brad's preferences.
- When Brad reaches the train station, his SPA notifies him that he has received a free upgrade to a first class train ticket. The SPA supports Brad in finding the station platform and his seat in the train, based on available local guidance services.
- Brad takes his seat and reads his newspaper, while the SPA detects the capabilities of the new work environment, e.g., a display in front of his seat, a WLAN hotspot and GPRS cellular access.
- Suddenly his phone rings; from the ring tone Brad can tell it is his secretary. The secretary asks him to call the customer, where he is going. When he does so, the customer suggests new ideas for his presentation.

- Brad starts up his laptop. The SPA automatically establishes a remote connection to the company network, using authentication support from the SD card. The new work environment is similar to his office and even includes additional features offered by the new location; e.g., the system proposes that Brad could view documents with the display in front of his seat. These contain critical data. Brad declines and reads the documents on his laptop.
- He works online with his colleagues in the office and with external consultants to update the presentation. During the work he uses data from central storage at the office. When he has finished the work he is convinced that he is well prepared and that the team did a good job.
- Later on, Brad asks his SPA for a connection to his private environment to continue the fun game offered last night by his Service Provider. The SPA establishes access to the game, which is stored in the Service Provider network and to Brad's game data, held on his Home Network. In this way he can continue gaming at the level he reached the previous night.
- Suddenly the system notifies him that his friend Alice is online. She invites him for an ego shooter session orchestrated by a central Gaming Server. Brad joins the session because there is one hour left before he reaches his destination and gaming with his friends is much more challenging. He asks his SPA to join the gaming environment. When his terminal is ready for the game, he sees that his friends, Andrea, who is on a sailing trip in the Caribbean, and Bob, who lives in Canada, have already joined the game. The session is established according to Brad's personal settings, which means that sound is enabled and he uses WLAN rather than GPRS. This time Brad accepts the suggestion to use the train screen, which is more comfortable than the one on his laptop.
- When the train arrives, Brad stops gaming; the SPA saves the game results and provides logout. He is now relaxed and ready to meet his customer.
- His SPA guides him to the customer's offices, using a navigation service offered by a local Network Provider. At the gate, his SPA provides automatic authorization with the Customer's Entry Service. Brad is guided to the conference room.
- Upon entering the room, his SPA detects a wireless business network and automatically configures a guest account, using the credentials he received at the gate. Using this account Brad connects to his e-mail server. Simultaneously, his SPA configures a connection to the equipment in the conference room.

#### 4.1.2 *Car / Travel / Shopping*

This scenario refers to a private environment involving a Mobile Travel Assistant (MTA), a Mobile Shopping Assistant (MSA) and a Mobile Sightseeing Assistant (MSSA), residing on a Simplicity-enhanced handset (see Figure 5).

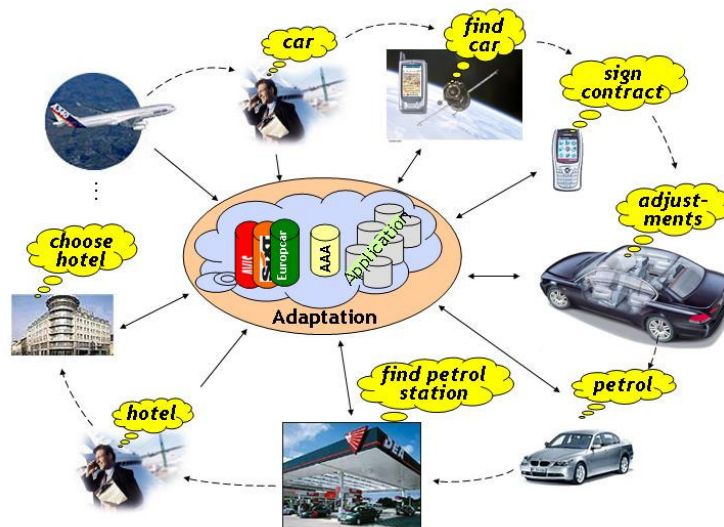


Figure 5: Car / Travel / Shopping

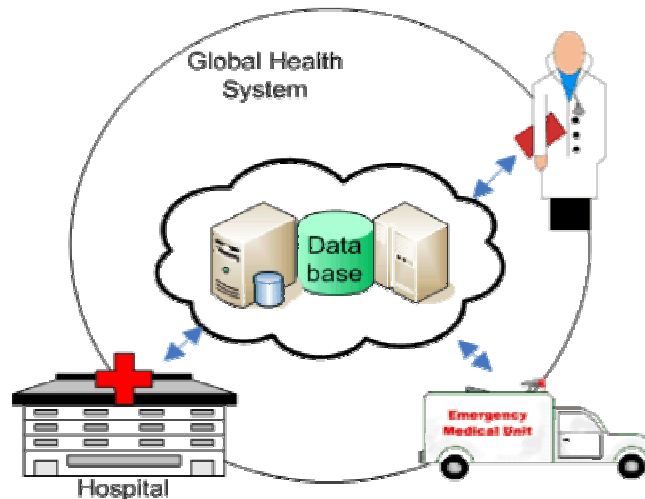
The main features of this scenario are presented below:

- Car Rental**  
 Arriving at a foreign airport, Brad asks his MTA for car rental support. After communication with Service Providers in the hotspot and consideration of Brad's travel profile, the system makes its proposals. Electronic contracting and car handling are supported by the MTA, after Brad has made his choice.
- Travel Navigation**  
 While traveling, the MTA provides information about the condition of the car, available petrol stations, local shopping centers and hotels. The navigation system may be combined with a Route Planning Service which uses data on the user's interests and preferences to propose attractive routes.
- Shopping and Sightseeing:**  
 The Mobile Shopping Assistant (MSA) can be asked to find the best offerings in a foreign shopping centre; it can also help to find the restaurants which most closely match the user's preferences. To learn about the most important places to visit, the user can activate the Mobile Sightseeing Assistant (MSSA).

In the above scenario, the MTA, MSA, MSSA are functions to be suitably defined and mapped onto the Simplicity architecture.

#### 4.1.3 The Global Health System

This scenario (see Figure 6) assumes that all hospitals are registered with a Global Health System and that doctors and patients can access medical services via their Simplicity Devices.



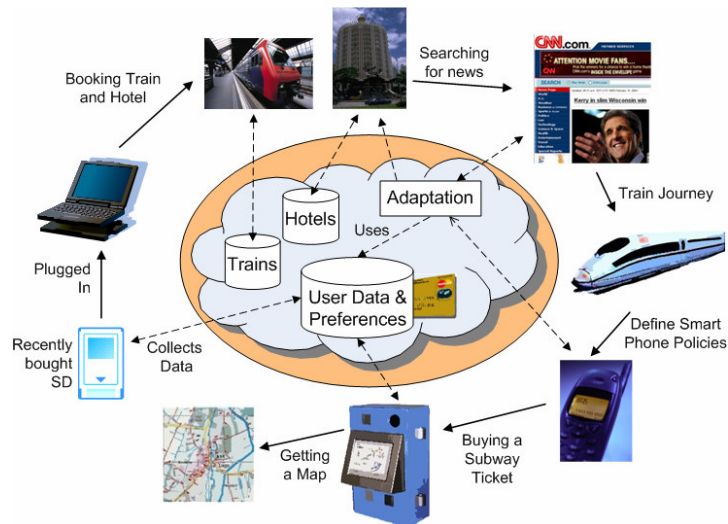
**Figure 6: The Global Health System**

- **The doctor's case.** A doctor is relaxing on her sailing boat where she receives an emergency call from one of her patients. To obtain help she inserts the Simplicity card into her notebook, implicitly activating the Doctor's Simplicity Assistant (DSA) and automatically configuring her working environment. This allows her to access the hospital's database and the data for the patient who is calling. Transfer of patient data will depend on the networks available, (e.g. GPRS on the boat). The doctor identifies this as an emergency session, thereby accelerating access the hospital server. Using the patient's past history and other information provided by the DSA she can advise the patient and provide him with information about the nearest pharmacy where he can obtain the medicine he needs. After concluding the call, the doctor updates the patient's history with the new information.
- **The patient's case:** A patient, enrolled with the Global Health System, uses a Simplicity card to store medical information for use in emergencies. In the event of an accident, a doctor in the ambulance can insert the patient's Simplicity card into a laptop to obtain useful information, such as the patient's blood type or any medication he may be taking. If further information is required, the doctor can remotely access servers at the patient's hospital. All actions performed by the ambulance crew will be documented on the patient's Simplicity card.

#### 4.1.4 *Buy and Use a Self Learning Simplicity Device*

The fourth scenario (see Figure 7) highlights some of the possible functions and services provided by the Simplicity Device (SD), such as auto-filling of forms or content and device adaptation.

Sven is a manager in a software company. He travels quite often and has recently bought a new (empty) SD which he wishes to use on his next trip, e.g. to a conference.



**Figure 7: Buy and Use a Self Learning Simplicity Device**

- Sven uses his laptop, (which is connected to the empty SD) to buy a round trip train ticket over the Internet. He types in details of his name, address, bank account, and preferences: he wants to have a window seat in a non-smoking area and a broadband internet connection.
- Next, he books a hotel room. The SD ‘remembers’ the data from his train booking, automatically fills in the name and address, and chooses a non-smoking room.
- During this time the SD analyzes the data collected by the laptop, using Bluetooth to connect to Sven’s smart phone. Via this connection, the SD accesses personal information management (PIM) data like addresses, notes, e-mails, bookmarks and appointments and configuration data (email accounts, preferred applications) for the laptop and the smart phone.
- As a next step, Sven searches for news headlines on his laptop and saves the results on his SD.
- Sven goes to the station. Once in the train, he inserts his SD into a smart phone to read the news. Given Sven’s preferences, the news is displayed without pictures.
- During the train journey Sven’s smart phone rings.
- He looks at the display and recognizes that it is his mother who is calling him. Sven pushes the mute button.
- Afterwards, Sven opens the policy application on his smart phone and creates a rule that that every private incoming call during work time should be forwarded to his mailbox. He dislikes telephoning people in the train and decides that incoming calls during train journeys should not activate the ring tone. These “policies” are automatically stored on his SD.
- Sven drives to the hotel and checks in. The hotel room contains a Simplicity-enabled alarm clock, which checks the calendar and programs an alarm one hour before the first meeting.
- On the next day, Sven wants to take the subway to the congress centre. He plugs the SD into the ticket machine. The Simplicity-enabled ticket service recognizes the meeting, the meeting place (congress centre) and the start time. Based on this information and the round trip train ticket he has already bought it pre-selects a one-day city ticket for zones 1 and 2.
- Once Sven confirms the ticket pre-selection, his bank account is charged and the ticket is issued, e.g., as e-ticket stored on the SD. In addition, an optimized route to the congress centre is transferred to the SD.

## 4.2 Methodology for User Scenario Analysis

The goal of user scenarios is to explore and illustrate the potential of the Simplicity approach and to guide the specification of the Simplicity architecture. Scenarios were prepared using the template shown in Table 3

**Table 3: User scenarios template used in the requirement analysis of the Simplicity Device**

<b>Narrative Description:</b> Describes the use case in natural language.
<b>Use case:</b> Use case identifier or reference number: unique name expressing purpose
<b>Description:</b> Describes the main goals of the use case. It should list the sources for the requirements, preceded by the keyword <i>sources</i> .
<b>Actors:</b> Primary and/or secondary actors. Primary actors have a goal requiring the assistance of the system. Secondary actors help the primary actor in achieving this goal
<b>Preconditions/Assumptions:</b> Assumptions (described in the form of a proposition evaluates to true or false)
<b>Steps:</b> The sequence of interactions necessary to achieve the goal of the use case.
<b>Variations (optional):</b> More detail about a step may be given by variations:
<b>Non-functional requirements:</b> <keyword>:<requirements in natural language or appropriate formalism>
<b>Issues:</b> List of issues awaiting resolution. Notes on possible implementation strategies or impact on other use cases.
<b>Comments from the use case:</b> In natural language.

In section 5, we will identify comprehensive set of “basic” functionalities, based on a detailed functional analysis of the user scenarios. Support for these functionalities is the main requirement we are considering during the design of the Simplicity architecture. In section 4.3 below we present a “higher level” analysis of a set of common requirements that can be extracted from the user scenarios.

## 4.3 Requirements common to all scenarios

- **System Metaphor:** Simplicity requires a system metaphor allowing users to understand the ways in which they can interact with the system. It is suggested that Simplicity can best be understood as a “Personal Assistant”. By “personal” what is meant is that Simplicity takes account of user data and user preferences, e.g. the data stored on a Simplicity Device. By “assistant” we mean that a computer based background process determines how best to respond to user requests based on user preferences, terminal and network capabilities, context information and service availability.
- **Convenient User-System Interaction:** easy user-system interaction is a major requirement for the effective use of a Personal Assistant. User context, behavior, profile and preferences should enable implicit user interactions and assist users in achieving their goals instead of requesting them to configure multiple devices or



personally search for content and services. Where necessary the system can ask for user input to disambiguate its inferences. From the output point of view, device selection, interaction modality (e.g., sound, visual), and content adaptation are major concerns.

- **Learning system:** the Personal Assistant should store a history of user behavior and use this, where appropriate, to learn implicit user preferences.
- **Interaction with the environment:** this includes communication with the user terminal and the Simplicity Device, detection and selection of access networks, detection, access and use of network services (e.g., for authentication), discovery of location-based services, ambient awareness, routing capabilities, access to private and business data stores and connectivity to other users.
- **Service discovery and handling:** the Personal Assistant should detect new services and match them against user interests. It should also support handling of new services, taking into account user capabilities and preferences.
- **System flexibility and easy integration:** Simplicity requires an iterative solution approach. User interactions with the system should be optimized step by step, based on experience; new access networks should be included, when available, and new context should easily be integrated within the system.
- **Authentication and data security:** successful deployment of Simplicity requires high quality authentication and secure data access from any location. The benefits of Simplicity (e.g., health support, remote access to critical business data or electronic banking) will only be accepted by users, if Simplicity can provide protection against unwanted intrusion.
- **Cost information:** costs will also have a major impact on the uptake of Simplicity. When Simplicity asks a user to take a decision (e.g., choosing a Network Provider, when roaming between different Providers or when downloading remote data) it should always include cost information.

Summing up, authentication and security features, service discovery and adaptation, automatic configuration of terminal equipment, adaptive user interfaces and personalization are recurrent requirements in many different scenarios. These requirements, complemented with proposals for appropriate business models, lie at the core of the Simplicity vision.

## 5 Requirements

Telecommunication systems beyond IMT2000 have two main characteristics, namely personalized communication spaces for users and interoperability of networks. In the future, end-users will be able to access different service domains using different kinds of terminals with adaptive user interfaces, without being aware of the underlying networking technologies. Simplicity will focus on these goals, introducing a personal device that simplifies the creation of a personalized communication space and a network-based brokerage framework supporting the adaptation of terminals, services and network resources.

Requirements engineering and analysis provide appropriate techniques to understand end-user desires and to define required functionalities for devices. Key steps in the process include analyzing needs, assessing feasibility, looking for a reasonable solution, specifying the solution, validating and verifying specifications and managing the translation of requirements into an operational system.



A detailed analysis of the user scenarios produced the list of functional requirements in Table 4. The table includes 18 main functionalities (listed in bold) and several sub-functionalities (reported in normal font).

**Table 4: General functionalities of the Simplicity Device as determined by the requirements analysis.**

<b>1</b>	<b>Use 3rd party services</b>	<b>8</b>	<b>Determine cost, QoS, etc. of network/bearer</b>
<b>2</b>	<b>Download application from 3rd party</b>	<b>9</b>	<b>Connection monitoring</b>
<b>3</b>	<b>Network access</b> WLAN access PAN access  Cellular access Access to fixed network	<b>10</b>	<b>Storage functionalities</b> SD Storage Card Functionalities Simplicity Network Database / Storage utilization PIM (Personal Information Manager)
		<b>11</b>	<b>Authentication &amp; Payment</b> Network authentication Service authentication SD login Identity and Payment Delegation of credentials and/or privileges
<b>4</b>	<b>Location awareness</b> Receive coordinates of target location Identify location of user Navigation	<b>12</b>	<b>Session saving / transferring</b>
		<b>13</b>	<b>Terminal capability discovery</b>
		<b>14</b>	<b>User preference gathering</b>
		<b>15</b>	<b>Tasks automation</b>
		<b>16</b>	<b>Context/environment awareness</b>
<b>5</b>	<b>Personalization based on user preferences</b> Device personalization based on user preferences Connection personalization based on user preferences Personalization of application based on user preferences Personalization of network service based on user preferences Content Adaptation	<b>17</b>	<b>Download/upload information to SD</b>
<b>6</b>	<b>Network and bearer discovery</b>	<b>18</b>	<b>SD plugability support</b>
<b>7</b>	<b>Service discovery</b>		

The requirement to **use 3<sup>rd</sup> party services** includes access to services provided by a service provider, e.g., the user connects to a car rental service and the Simplicity Device chooses and reserves a car, based on end-user preferences. In some cases, the user may need to **Download an application** to her current terminal. The Simplicity Device has also to support different underlying networking technologies (**Network access**), and to make the most appropriate choice of network, based on user preferences and/or the coverage of specific networks. The SD should be **location aware**, so that it can guide end-users to their destinations. **Personalization based on user preferences** is one of the most important requirements for the Simplicity Device. Devices, connections, application functionality etc. have to be automatically adapted to match the user preference stored on the SD. The SD should also support the discovery of different access technologies and network providers **network**, choosing the connection which best matches user preferences, for example by using a company's WLAN instead of an alternative commercial GPRS connection. Similarly the SD should also support **service discovery**. It should be able to **determine costs** for different kinds of connections and for different levels of QoS. The SD should also be able to **monitor its connection**. The **storage functionalities** required by the SD can be roughly defined to include USB-stick like memory and the ability to access database storage over a client-server network connection. **Authentication and payment** functionality includes SD login, network and service authentication, identification and payment as well as delegation of credentials and privileges. SD should also be able to **save and reactivate sessions** during and after session interruptions. One very crucial required feature for the SD is the ability to **discover terminal capability** before adapting it or

configuring connections and services. The SD should also be able to **gather user information** during use and adapt future sessions accordingly. It should be able to **automate** different kinds of predefined **task**, such as the updating of calendar information to include a flight reserved via the SD. The SD device should also be able to understand its environment, i.e., be **context-aware**. The device should support **upload/download of data** over the network connection. To ensure widespread market acceptance it should be a **plug in device**

## 5.1 Methodology for deriving requirements

To reach a final definition of the functionalities to be included in the implementation, some of the scenarios in the initial set were merged and a set of 7 critical scenarios were selected for further analysis.

In the next step in the analysis, UML (Universal Modeling Language) will be used to formalize selected scenarios via use case diagrams and first level sequence diagrams and to explore the functional relationships between them. Using this analysis we will answer the following questions:

- is each proposed functionality consistent with the overall objective?
- are the functionalities described at the proper level of abstraction?
- are all the functionalities really needed?
- are all the functionalities bounded and unambiguous?
- is there source for each functionality?
- is each functionality technically achievable?
- will each functionality be stable once it is implemented?

## 5.2 Preliminary list of requirements for the SD

The SD has to be able to interact with its environment, which may contain one or several , devices. Devices can be of sensor type (providing context information) or of actuator type, allowing the SD to perform actions. They can also be processing devices with input and output. The SD may include several hardware or software interfaces for user interaction and connection purposes. For example, the SD could use the user interface of the device to which it is connected (via a wired or wireless connection). The SD should also be able to gather and store information, discover networks, services, devices and communicate with these entities in a secure and cost effective way. This leads to the following functional requirements:

**Table 5: Functional requirements for the SD**

<b>Functionality</b>	<b>Description</b>
information gathering	profiles, contexts, services
information storage	auto-saving of: <ul style="list-style-type: none"><li>• profiles</li><li>• sessions</li><li>• files</li><li>• settings/configurations</li></ul>
device, service and network discovery	adaptivity session continuity authentication authorization
task execution	trigger actions

Another question raised by the scenario analysis is whether and how to divide functionalities between the SD and other devices. In one scenario the SD performs most of its tasks on its own; in another, devices with embedded middleware could collect information from the SD. This is a crucial issue for the definition of the system architecture and the design specification and will be tackled in more detail at a later stage in the project life cycle.

## **6 System Architecture**

The Simplicity Device will provide a uniform model for the personalization of terminals, network access and services by storing personalization rules for the individual user. However, the SD will be deeply integrated in a system framework able to decouple user needs and service deployment issues from the underlying terminal devices, networking technologies and service support platforms. The goal of the Simplicity system is to manage this complexity, while limiting end-user interaction to special decisions.

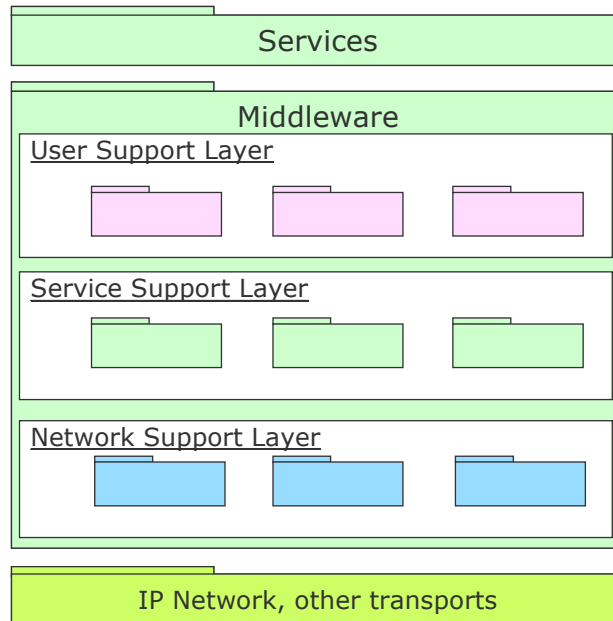
To describe the system architecture, we use a horizontal (layered) view and a vertical (user/terminal/network roles) view, as described below.

### **6.1 Horizontal view**

A first step in the architecture design is to define a logical architecture, showing layers and components for service provisioning, rather than defining the location of explicit software components in specific physical nodes (such as devices and networks).

To increase the re-usability of middleware components, the logical architecture of the Simplicity System is defined in terms of a layered approach. As shown in Figure 8, and as suggested by [4], this middleware consists of a User Support Layer, a Service Support Layer and a Network Support Layer.

The **User Support Layer** supports autonomous and proactive agent functions, providing personalization, adaptation and coordination components, which are lacking in traditional middleware. The **User Support Layer** simplifies user interactions with the system and provides user-centric services, by analyzing user contexts and user preferences.



**Figure 8: Logical architecture: layered view**

The **Service Support Layer** contains traditional middleware components, such as advertisement/discovery, profiling and security functions. A dynamic service delivery pattern enables discovery, advertisement, authentication, and authorization functions and is used to negotiate the conditions of service delivery according to users needs.

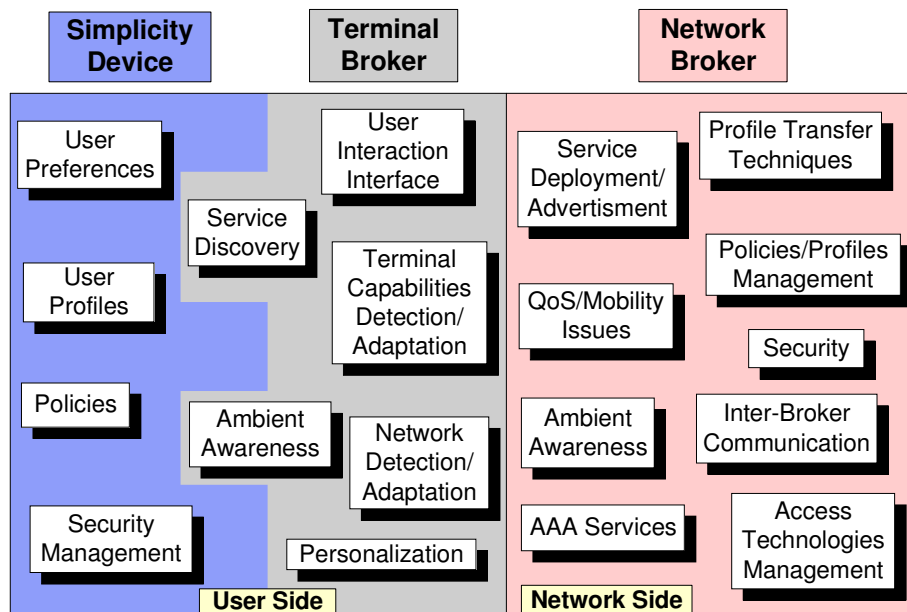
The **Network Support Layer** provides components for network communications control in heterogeneous networks, including mobility management and QoS management.

## 6.2 Vertical view

It is crucial that operations to be performed on the user side should be clearly separated from those to be performed on the network side. Since there may be a number of possible solutions, we will carry out a preliminary comparative analysis to understand which solution best suits the Simplicity system. Figure 9 depicts the basic components we expect to find in the Simplicity System, and a first proposal for the separation of functionality among them. The proposed system encompasses three main components: the Simplicity Device, the Terminal Broker and the Network Broker.

As discussed in Section 2, the role of the **Simplicity Device (SD)** is to store user's profiles, preferences and policies. It also stores and enforces user-defined mechanisms for service personalization, and for the automatic adaptation of services to terminal and network capabilities.

The **Terminal Broker** is the entity that manages the interaction between the information stored in the SD and the terminal device. The Terminal Broker enables the SD to perform terminal capability and services discovery, adapting services to network capabilities and other features of the environment, It is the Terminal Broker that handles the user interaction with the Simplicity system.



**Figure 9: Physical architecture: component view**

The **Network Broker** provides support for service description, advertisement and discovery, orchestrating interactions among distributed networked objects. It is the Network Broker that handles the issues that arise when several users simultaneously access the same resources, services, and locations. The Network Broker's role includes the sharing and allocation of available resources, and the management of value-added networking functionality, such as service level differentiation and quality of service, location-context-awareness, and mobility support.

## 7 System Functionality

In this section, we provide a more detailed view of the system functionality that the Simplicity project aims to provide. For ease of presentation, the description adopts a vertical/component view (Simplicity Device, Terminal Broker, and Network Broker). The development of the functionality described in this section will nonetheless take advantage of the horizontal (layered) view discussed in Section 6.

### 7.1 Functionality of the Simplicity Device

The key role of the Simplicity Device is to store user profiles, preferences and policies in a secure and safe way, thereby allowing dynamic, automatic discovery and registration of terminal and network capabilities. Another role for the SD is to facilitate the adaptation of services to network technologies and related capabilities. The SD is an important entity in the architecture. Users cannot access the Simplicity System and exploit its benefits without it, even if there is a Terminal Broker on the terminal and a Network Broker is available. The SD is the user's passport to the Simplicity world.

The SD can be plugged into other devices in the environment (specifically, mobile devices) or interact with them via short-range wireless networking e.g. BlueTooth. Such integration is possible both for software and hardware implementations of the SD. The latter might take the form of a Smart Card [5], [6], [7], [8] or Java card [9], [10], [11].

For this reason, user profiles, device capabilities, preferences, policies, etc. should be defined and stored in the SD using a high-level description language, such as XML. This will make it possible to provide a globally applicable, abstract functionality layer.

The SD requires memory. This can be located internally, on the SD, or externally, on devices in the environment. In the latter case, the SD can store a set of pointers to network locations (e.g., Profile Repositories, see Section 7.3). Appropriate entities in the Terminal and/or Network Broker could retrieve and process these pointers, allowing users to download essential software and data that cannot be stored in the SD (e.g. because of storage limitations). The stored information will enable dynamic, automatic discovery and registration of terminal and network capabilities, as well as the automatic adaptation of services to network technologies and terminals. Specifications for profiles will take advantage of a large amount of work which has already been carried out in standardization groups. One example of a 3GPP solution to user profiling is the Generic User Profile (GUP) [12], [13], based on XML. The most important feature of GUP is that it can be adapted to any kind of system and context, thus providing the flexibility needed by Simplicity.

## **7.2 Terminal Broker functionality**

The user terminal/device contains a so-called Terminal Broker (TB). The TB is the entity that “interfaces” the user to the network.

At this early stage, the division of functions between the SD and the TB is not yet clear. It will depend on a number of factors, including physical constraints and requirements (e.g., memory, processing power, passive/activeness support). In any case, the role of the TB is to complement the SD i) by retrieving information from the SD regarding user’s preferences, profiles and behavior, and ii) supporting specific actions depending on this data.

A key function of the TB is to provide the user with a means to read/write/modify the personalization information stored in the SD. From the terminal point of view, the user interface to services has to be adapted to fit terminal capabilities and specific contexts of use; from the services and appliances viewpoint, it might be necessary to adopt a general User Interface (UI)-description language, and a common interface to all networked devices and services. This would simplify development.

A second important function of the TB is to enable the SD to perform discovery of terminal capabilities and service adaptation to the surrounding environment. A number of XML-based Discovery Protocols (e.g., Universal Plug and Play (UPnP) [14] and JXTA [15]) have been promoted in peer-to-peer frameworks and are worthy of consideration Simplicity. The TB applies user preferences to the working environment on the terminal, perhaps by collecting contextual information (e.g., user location, user actions, surrounding devices and services) from sensors. This information will be matched to specific terminal capabilities in order to instantiate the proper service components on the terminal (as well as in the network, when functions such as remote trans-coding of downloaded information will be necessary). The TB can configure applications and network settings and can dynamically download plug-ins and applications.

Last, the TB is the entity that allows user preferences and policies, stored in the SD, to drive service adaptation to networking technologies and capabilities. Through its interface with the network, the Terminal Broker selects the most appropriate access option, and dynamically estimates current network capabilities (where necessary taking account of congestion, etc).

To conclude, the key characteristic of the TB is that its functions are adapted to the needs of the user who has to use the interfaces. An important requirement for the TB is that it should sit on top of existing and emerging terminal and network technologies. The role of the components in the Terminal Broker is to decouple Simplicity system functions from specific networking technologies and from specific terminal characteristics. Such components might, where necessary, be dynamically downloaded.

### 7.3 Network Broker functionality

As shown in Figure 10, the Network Broker (NB) acts as an intelligent edge to the IP networking domain. The NB provides a network-based platform, offering the user a unified, personalized view of available services. The platform takes account of user profiles, the characteristics of the environment, and terminal capabilities as well as the network side, handling user requests in an optimized and personalized way.

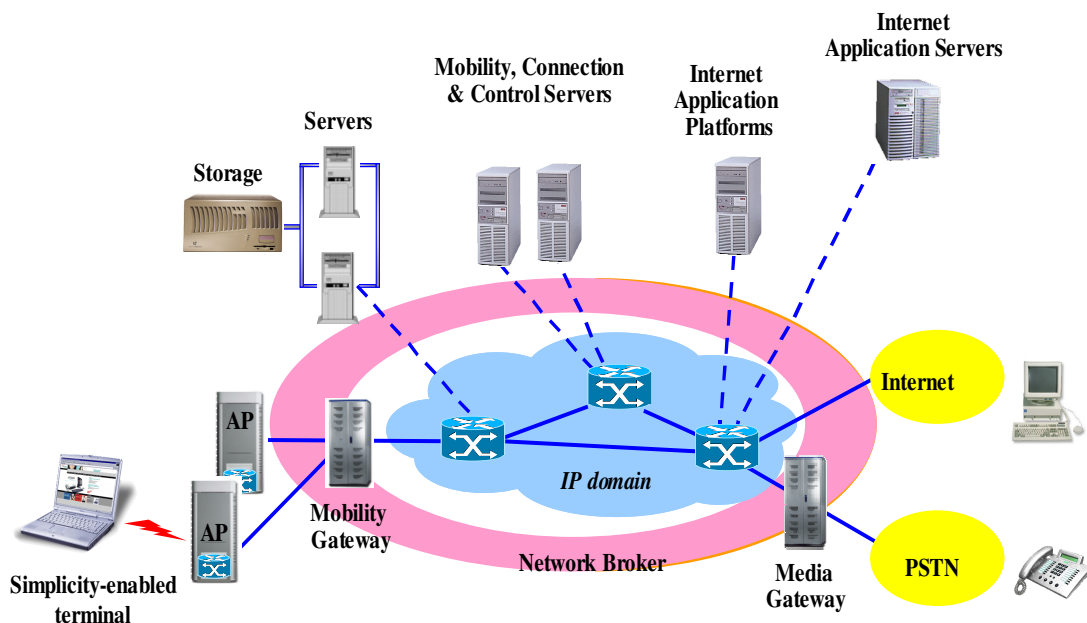


Figure 10: Network Broker

The NB architecture will enable functional enhancements without major changes to existing solutions and should be flexible enough to be used over several types of networks, e.g., business networks, banking networks and home networks. In this way, network services can be identified once and for all, provided over different kind of networks and specified only once.

#### 7.3.1 Service Deployment/Advertisement

A key issue for the NB is Service Deployment and Advertisement. Users should be able to browse available services and select those that suit them best. To this end, the NB provides users with a platform for the deployment/advertisement of application services. The NB will address the issue of the on-demand delivery of client code to user terminals. In addition, it will require the following features:

- application service providers will need to deploy client components on provisioning servers, and to associate these components with descriptive information that can be used by clients during the discovery phase;

- users need to perform discovery of software components, taking into account their preferences, contract, terminal, position, and download/install capabilities. Different methods to provide this component (e.g., MIDP OTA, JNLP) will be supported through adapters.

### 7.3.2 *Service Adaptation*

After the user selects a service, the service adaptation phase begins. The NB needs to communicate with the TB to retrieve information about the user profile and device capabilities. The Profile Transfer Techniques functional block contacts the TB and retrieves the necessary information. Another factor that plays a significant role in service adaptation is the user environment. The Ambient Awareness component interacts with the corresponding component on the TB and acquires the necessary information (e.g., location of the user, surrounding devices etc.). Services use this information to adapt to current conditions.

An attractive solution for the implementation of the NB would be to use an adaptive agent-based service platform, providing services specifically targeted to individual needs in specific environments. The platform could select these services by using information on users' current environment, their past behavior and the behavior of other users in similar situations.

### 7.3.3 *Network Side Handling*

As far as the network side is concerned, the NB combines policy-based technologies (e.g., policies for mobility support, QoS, security, software downloads), stored in a policy repository (Policies / Profiles Management Module), and a number of specific modules, handling AAA services, Security, Quality of Service / Mobility Management, and Access Technologies Management. These modules orchestrate events and available resources, the adaptation of network capabilities and the management of different access technologies and networking alternatives. The NB interacts with NBs on adjacent networks (via the Inter-Broker Communication module) to provide an optimized end-to-end service across network domains.

### 7.3.4 *User Profile*

The User Profile is of major importance within the Simplicity architecture. For this reason the design of the Profile Repository is a very significant issue. This repository should provide generic, flexible and easily updatable descriptions of compatible and available devices, services, technologies and users profiles. Such descriptions can be referred to as a Generic Profiles or Template Profiles. Generic Profiles can be divided into User Generic Profiles, Device Generic Profiles, Service Generic Profiles etc. Profiles for users, devices and services can be categorized into groups sharing common features. A standardized Generic Profile can be created for each category and stored in the repository. Examples of generic user profiles might be Business, Entertainment, Emergency etc.. Generic device profiles might include profiles for PDAs, Laptops, Mobile Phones. From the services perspective, we can categorize services based on the different service domains to which they might belong and then create the appropriate Generic Profiles. Such domains might include for example: enterprise network, home network, service provider network etc. Figure 11 presents a possible classification of Generic Profiles.

The transition from a Generic Profile to a specific User Profile will take place on the terminal. Users will be able to download any Generic Profile and modify/configure the



profile according to their preferences. Device profiles will be configured automatically with the assistance of the Terminal Capabilities Discovery module that resides within the Terminal Broker. The completed User Profile will consist of a combination of all the modified/configured generic profiles. Services selected by the user will be adapted according to the completed profile.

The User Profile will be stored within the SD or uploaded to the Profile Repository. In the former case, the User Profile will be downloaded on demand and processed by the SD. In the latter case, personal data is involved and security issues will have to be taken into consideration. The Network Broker provides a secure, location-transparent storage service that can be used for this purpose as well as for storing session and other, application-related data.

Profiles will be more than a mere static listing of features and capabilities. They will also embody dynamic representations, containing both features and logic. To this end, we might envisage a Mobile Agent approach. Mobile Agents could migrate to the SD on demand and operate in a co-operative way to formulate reconfiguration decisions that are optimal for the user's environment.

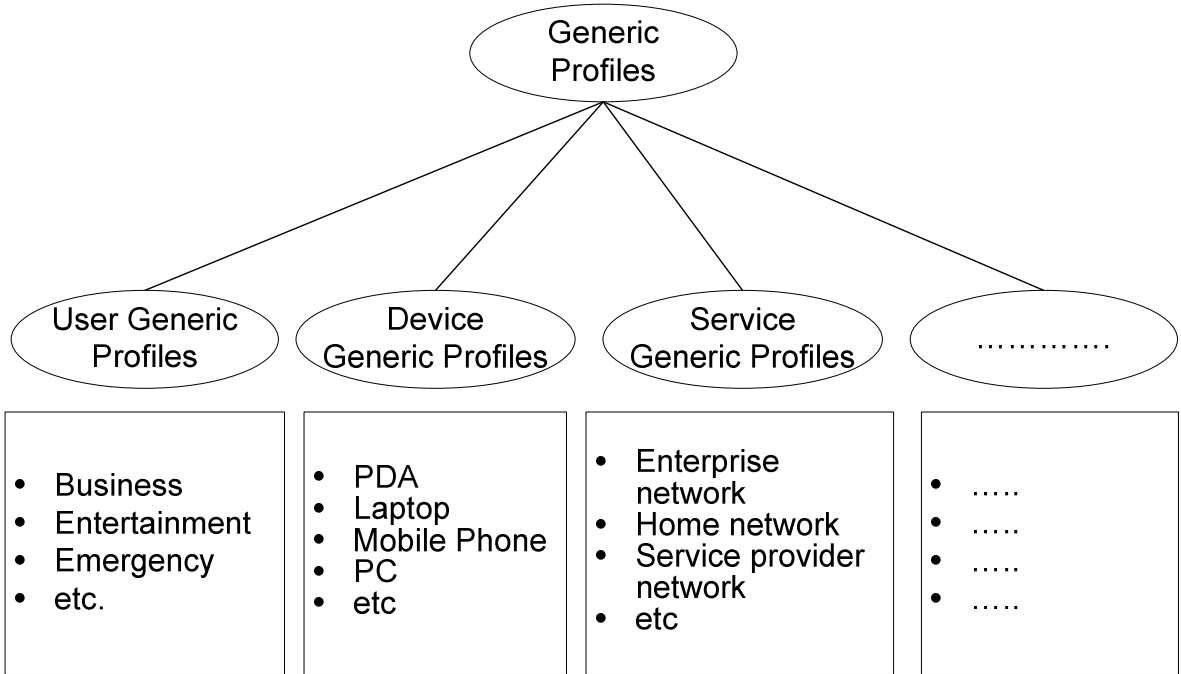


Figure 11: Categorization of Generic Profiles

## 8 State of the Art

The aim of this Section is to present a thorough analysis and qualitative evaluation of the technologies, standards and studies of relevance to the development of Simplicity.

### 8.1 Personalization and User Profiles

Simplicity creates a Personal Service Environment (PSE) which relies on user profiles for adaptation and personalization of services and terminals. In general, a personal profile is a collection of electronic information describing the user's personal characteristics and

preferences and prescribing related rules and tasks. In this section we will examine the state of the art on user profiling, service/terminal adaptation and personalization.

A 3GPP solution to user profiling, currently under standardization, is the Generic User Profile (GUP)[19] [20], based on XML. 3GPP GUP proposes a structure for the organization of data, but allows designers great flexibility in defining the content of this data. The data stored in 3GPP GUP might include, for example, data on authorized and subscribed services, general user information, user privacy control data, information about specific services and billing information. Historical/Statistical and Runtime data is not included in the GUP. In the 3GPP solution profiles are stored in and downloaded from the network. Network cooperation is thus essential. This approach can nonetheless be adapted and extended to support a user side architecture where information is stored directly in the SD. The most important aspect of GUP is that it can be adapted to any system and context, thus providing the flexibility needed by Simplicity.

Another interesting solution is the Application Configuration Access Protocol (ACAP) [17] 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 access all the information needed for automatically adapting and personalizing service. New information can be easily added without server re-configuration, thus allowing the use of standardized data as well as 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 to specify a user agent profile [16]. 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 user's preferences for the use of these capabilities.

The User Agent Profile Specification [23] 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 UAPProf specification defines so-called Capability and Preference Information (CPI), which is communicated between the WAP client, intermediate network points, and the origin server. The specification seeks to interoperate seamlessly with 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 [18], the Resource Description Format, was designed by the W3C consortium for dynamic content adaptation. It defines a mechanism for describing (Web) resources (meta-data), and thus 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.

A number of interesting projects propose the use of policy-based technologies or rule languages to achieve flexibility and generality [21] [22]. The most important rule languages are Jess, ZKB/XKB and RuleML.

Jess [25], which is entirely written in Java, is a well-established rule engine and scripting environment that is based on the CLIPS expert system shell. XKB/ZKB is a rule language included in Mandarax [26], an open source Java class library. Both Jess and XKB/ZKB allow the definition of reactive rules and facts that refer to and act on Java

objects representing for example user models, device capabilities, applications or network characteristics.

The Rule Markup Initiative has developed a semiformal XML-based language called RuleML [24] that allows Web-based rule storage, interchange, retrieval, and application. RuleML makes it possible to define integrity constraints, derivation rules and reacting rules. A number of DTDs/Schemas, engines, translators, user interfaces and rule libraries have already been developed.

Both the Simplicity Device and the terminal will incorporate policies expressing user preferences and terminal-specific information required for service-adaptation. These policies have to interact with policy-based technologies on the network side. This raises issues concerning the IETF policy framework, the Ponder framework and the Policy Description language. If policies are to be treated consistently it will be necessary to adopt a single policy technology or policy exchange language.

## 8.2 Simplicity Device

The Simplicity Device (SD) is the part of the Simplicity system that lies on the user side. All users are equipped with a SD which can be plugged into a range of terminal types and which supports automatic service discovery, AAA and policy-based configuration of the Simplicity environment. The SD combines the functionality of a hardware authentication token, a mobile storage device and a portable processing utility able to perform trivial as well as more complex tasks.

The SD could be implemented in hardware, in which case it could be a USB disk, or an enhanced Smart Card. Alternatively it could take the form of a software agent for use in special environments. A hardware implementation is, however, preferable. Since mobile code execution capabilities are desirable, the most suitable hardware is probably 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: 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 also provide tamper-resistant storage space for sensitive personal identification information. Rapid advances in processing capabilities and improvements in embedded software support those who believe in their potential as mobile general-purpose platforms for code execution [27].

The following sections review the state of the art in USB devices and Smart Card technology, provide a brief description of the Java Card architecture, one of the most important Smart Card software platforms available. On the basis of this analysis we will argue that the first SD implementation should be based on Java Card technology.

### 8.2.1 *USB Devices*

The last few months have seen the rapid spread of USB memory stick devices. This spread has been encouraged by improvements in manufacturing processes and the consequent lowering of prices, growing storage capacity and the flexibility of USB interfaces.

USB specifications [28] have gone through three steps: version 1.0, provided a bit-rate of 1.5Mbps; version 1.1 offered 12Mbps, now version 2.0 has a bit-rate of 480Mbps. Given that profile data occupies only a small number of KB, the transfer rates offered by USB interfaces are fully compliant with SD requirements. The storage capacity offered by USB memory sticks goes from 32MB up to 2GB, which is very impressive if we think that

these device are the same size as a standard key. USB devices can be easily attached to a key-ring; they thus meet the crucial Simplicity requirement for physical portability.

USB memory stick devices are well integrated with current computing and communication equipment. Most current PC/PDA operating systems provide complete support. The user simply has to plug the bar into a USB port and the service is immediately available. This is a perfect match with the SD concept. It should be noted here that many set-top boxes are also introducing support for USB devices.

The reliability of USB memory stick devices is a very important issue. Memory sticks carry information that it is very hard to retrieve from other sources. They thus have to be error-free. Nowadays, USB memory stick devices have a life-cycle of about 1.000.000 re-writes with 10 years of data retention [29]. Studies have demonstrated that the higher the number of re-write operations, the lower the retention time. When the maximum number of write operations has been reached, the retention time decreases to about 3-4 days.

There are a lot of USB memory stick devices that implement security mechanisms in order to ensure user data confidentiality. Mechanisms include PIN-PUK code, username/password and finger-print matching. An example of algorithms used to encrypt data is AES-128bit. The use of this kind of mechanism allows the protection of user data against external attacks and un-authorized copies.

A critical aspect of USB memory stick devices is that they have no computational capabilities. This is a serious problem for the SD since a device based on a USB memory would be unable to perform processing tasks on behalf of host systems.

### 8.2.2 *Smart Card Technology*

A Smart Card is a tamper-proof hardware device in which an IC (integrated circuit) chip is embedded in a plastic card. There are two sizes of Smart Cards on the market. The first meets the specifications for credit cards (as defined in ISO/IEC 7816-4 [30]) and is used primarily in banking, insurance and transportation. Telecommunications applications, on the other hand, use a 15mm\*17mm Smart Card such as the GSM SIM (Subscriber Identity Module) or the 3G UICC (Universal IC Card) standardized by 3GPP and ETSI SCP (Smart Card Platform) [31].

A typical Smart Card is equipped with an 8-bit or 16-bit processor with a clock speed of a few MHz, a few kilobytes of RAM memory, ROM memory containing built-in functionality and 32-64kb of non-volatile memory (e.g. flash memory). Recent high performance Smart Cards have incorporated attractive features such as 32-bit processors, an optional cryptographic co-processor and up to a few Mbytes of storage (combined RAM and flash memory). Smart Cards interface with terminals of various types via a card reader, also called a card acceptance device. Smart Cards conform to the ISO7816 [32] series of international standards, which define all Smart Card features, from physical characteristics to the mechanisms for interaction with the external world.

The software on Smart Cards has evolved along with the processing capabilities of their embedded ICs. [33] describes four generations of Smart Card software, ranging from the monolithic embedded operating systems of the past to today's modular, adaptable open platforms featuring secure multi-application execution environments, post-issuance application loading capabilities and object-oriented development models. Examples of such platforms include the Java Card Platform [34], a special subset of Java technology for resource-constrained devices and the Multi-Application Operating system (MultOS) [35], which provides a secure execution 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 [36] that standardize

interactions between Smart Cards and Microsoft operating systems, and the Open Card Framework [37], that standardizes Java based Smart Card solutions.

These Smart Card features, combined with their practical nature as lightweight portable electronic devices, have made Smart Cards into an important mobile platform for code execution. Their value is further enhanced by active research into applications for user mobility [38], e-commerce, personalized information services [39] [40], security [41] and interoperability using agent technology [42]. The experience gained from this research will be valuable for the implementation of the SD as a Smart Card.

### 8.2.3 *Java Card Platform*

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

Java Card Technology is a subset of Java, suitable for resource-constrained devices like Smart Cards. Java Card provides a multi-application execution environment inside the Smart Card that enforces strict separation between applications, thereby enhancing security and data integrity [43]. Java Card applications execute inside a virtual machine which runs on the card's specific operating system. The development of Java Card applications, or "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 [44] 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 [30] set of standards. For this purpose it uses a Card Acceptance Device (CAD), which interacts with the VM on the Java card and with the applets running on the VM. As well as the message passing communication model based on the exchange of APDUs, Java Card provides an alternative communication method using Java Card Remote Method Invocation (JCRMI), a subset of RMI distributed object model technology.

The features of Java Card technology make it an attractive solution for the implementation of the SD. In particular Java Card makes it possible to maintain flexibility and functionality while meeting the tight security requirements for a device which will store and process sensitive information such as credit card numbers, authentication information for online services, network access credentials and operator contract information.

## 8.3 **Flexible Network Support**

One of the main goals of the Simplicity project is to provide flexible network support for context-aware adaptation and personalization of services and terminals. The envisaged technical solution will have the following main characteristics:

- adoption of a brokerage framework using policy-based techniques to achieve optimal control and adaptation;

- a combination of 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 in a decentralized framework.
- reliable data storage as a basic service for handling distributed data, e.g. profile and context information.

The subsections below review the state of the art in these four areas.

### 8.3.1 *Policy-based brokerage frameworks*

A broker is an entity responsible for the management of resources. A broker's responsibilities are isolated from those of other entities. As a result, 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 among distributed entities. The broker concept was initially introduced by [45], where QoS was achieved through interaction between brokers residing at the end points. It was later adopted in the MASA project, which applied the concept to adaptive multimedia services in mobile contexts [46] and extended it to include additional brokers (called network brokers) residing not at the end points but in access and core networks [47], [48].

A broker is responsible for orchestrating different functions and subsystems within a domain. Management across different domains is based on negotiations between different brokers, controlled by policy-based decision mechanisms. A broker itself consists of independent but inter-operating subsystems. Again, the operation and inter-working of these subsystems is controlled and coordinated using policy rules. It is important that policies should be modular in nature, allowing the addition and elimination of policy rules without affecting other parts of the rule base. [49] has pointed out that the benefits of policy-based management in distributed systems arise from the use of proper syntax and policy management tools.

By using ambient awareness mechanisms (e.g. based on sensors), a broker can generate up-to-date context information and use this 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 [50] [51]. Support for 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 [52].

A possible Simplicity implementation of the broker concept might include a terminal broker to orchestrate user preferences, terminal capabilities and operation of local applications based on context information describing the user, the terminal, and the access network. The terminal broker would be supported by a system of network brokers which would orchestrate network features. Different types of network brokers could handle the specifics of access networks, core networks and service provider domains. Network brokers could be replicated to provide a scalable network infrastructure.

### 8.3.2 *Mobile Agent Platforms*

Mobile Agents are intelligent/autonomous software entities with the ability to migrate and execute their logic in several computational nodes. They are considered as a middleware technology enhancing distributed computing technologies such as CORBA, RMI and Web Services [53] [54] [55]. A Mobile Agent Platform (MAP) enables agents to execute on distributed nodes. A MAP consists of a set of APIs that exploit the capabilities and

mechanisms of the underlying middleware. Prominent MAPs include LEAP JADE, MicroFIPA-OS, AgentLight, JACK, Grasshopper and April.

The 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, fault-tolerance, and interoperability. With regard to Simplicity, these features are valuable for the implementation of broker coordination procedures and to meet requirements such as service discovery and dynamic code distribution. Accessing services in a visited network environment often requires support for mobility in the form of code download. JSR 24 (J2EE Client Provisioning) [64] provides a configurable and extensible framework to implement a context-aware software distribution mechanism. On the client side, standardization and research work (e.g. [63]) is currently under way to define a more flexible, robust Java-based execution platform for mobile devices, supporting full component lifecycle management (including secure download, activation and disposal).

### 8.3.3 *Service Discovery Frameworks*

The use of brokers supporting peer-to-peer (P2P) communications is one way to make a distributed system more flexible. In this model, there is no longer a central point responsible for the publication of services and information; 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 important service discovery protocols relevant to P2P communications are Universal Plug and Play (UPnP) [56] and JXTA [57]. Both consist of a set of communication protocols based on XML-encoding. In UPnP, a Simple Service Discovery Protocol (SSDP) enables devices to publish their presence and service descriptions by multicasting advertisements; to discover services, clients listen at the multicast port. Alternatively clients can search for services by multicasting requests. JXTA provides additional support for community-based activities across different P2P systems. It enables peers to create peer groups providing a common set of services. The default protocol is the Peer Discovery Protocol, which allows a peer to find advertisements from other peers or peer groups.

### 8.3.4 *Simple Storage Management*

Technologies which aim to deliver network-based reliable, secure storage services provide the ability to store and access personal data independently of user location, network point of access and terminal. In the case of Simplicity, user profile data and context data could be transparently stored and replicated on the network as an alternative to keeping the data on the SD).

Important projects in this area include OceanStore [58] (backed by IBM), Microsoft FarSite, PAST [60], and CFS [59]. In each case the service proposed by the project is built on top of a DHT routing layer. DHT middleware ([61]) 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. Smart Card mechanisms are typically used to provide encryption, to generate and verify certificates, to manage storage quotas etc. There have been early attempts (POST [62], MINO) to build email services on top of these infrastructures. These projects have shown how user metadata (folders, preferences, contact lists) can be stored in the network, while remaining accessible to users, regardless of the client they use to connect to the service.

## 8.4 Ambient Intelligence

*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 [70] 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 [65] with a focus on providing support for the elderly in their own homes.
- MIT's Project Oxygen [68], 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 [67] at Stanford University, exploring the use of collaborative, interactive workspaces.
- GAIA [69] 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 minimize explicit interactions between humans and their smart environments. Smart spaces are expected to act proactively instead of merely reacting to explicit input from users. This requires systems with the ability to collect and process rich sets of contextual data, including information about human users, physical objects and software entities.

Simplicity aims to contribute to Weiser's vision. The Simplicity Device will thus provide means to:

- store and supply contextual information about its owner, e.g. in the form of profiles
- authenticate users, either by directly using the owner's Simplicity ID or through credentials stored within a user's Simplicity device
- discover and personalize services; 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 in smart environments developed outside the project. For example, it will be necessary to discover and interact with the services provided by these environments. While Simplicity does not intend to advance the state of the art in smart environments themselves, it will provide mechanisms for easily customizing 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 should support a decoupled, asynchronous communications model making it possible to incorporate the additional infrastructure elements in the Simplicity architecture without impacting on the operation of the rest of the smart environment. Having reviewed the systems presented earlier we have decided to base our work on the iROS platform [67] developed at Stanford University, extended with features enabling it to generalize beyond the context of a meeting room for which it was originally 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 [66], 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 customizing iROS-based smart spaces.

## 9 Conclusions

The Simplicity project addresses a crucial issue for systems beyond 3G and proposes a solution to handle the increasing complexity of systems, services and technologies. The concepts developed in the project can directly impact the way citizens live and work. We intend to prove this concept by designing our proposed architecture and implementing its main concepts, thereby showing its feasibility.

### Acknowledgements

This work has been partially funded by the European Union as part of IST project 2004-507558. We gratefully acknowledge the contribution of all Simplicity partners. This support should not, however, be construed as an endorsement of the views, results and conclusions expressed in this document, for which the Authors are wholly responsible. Likewise, any errors or omissions are the sole responsibility of the Authors.

### References

- [1] Website of the Simplicity project: <http://www.ist-Simplicity.org>
- [2] Wireless World Research Forum, The book of Visions 2001 – Version 1.0, December 2001.
- [3] S.Y. Hui, K.H. Yeung, "Challenges in the Migration to 4G Mobile Systems", IEEE Communications, Dec. 2003, pp. 54-59
- [4] C. Noda, A. Tarlano, L. Strick, M. Solarski, S. Rehfeldt, H. Honjo, K. Motonaga, I. Tanaka: "Distributed Middleware for User Centric System", WWRF#9 conference, Zurich, July 2003.
- [5] Aran Ziv, Tal Segalov, "FlashDrive Performance and Reliability, White Paper", September '03
- [6] ISO/IEC 7816-4:1995, "Information technology -- Identification cards -- Integrated circuit(s) cards with contacts -- Part 4: Interindustry commands for interchange"
- [7] ETSI TS 102 221, "Smart Cards; UICC-Terminal interface; Physical and Logical characteristics", V6.0.0, 02-2003
- [8] ISO/IEC 7816, Information technology - Identification cards - Integrated circuit(s) cards with contacts, 1997-2004
- [9] 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
- [10] Java Card Technology, <http://java.sun.com/products/javacard/index.jsp>
- [11] Open Card Framework, <http://www.opencard.org>

- [12] 3GPP TS 22240-600: "3GPP GUP, Requirements, Stage 1 Release 6", March 2003. See: [www.3gpp.org](http://www.3gpp.org)
- [13] 3GPP TS 23240-110: "3GPP GUP, Architecture Specifications, Stage 2 Release 6", April 2003. See: [www.3gpp.org](http://www.3gpp.org)
- [14] UPnP, <http://www.upnp.org/>
- [15] JXTA, Project JXTA, <http://www.jxta.org>
- [16] "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/>
- [17] 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>
- [18] "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>
- [19] 3GPP TS 22240-600: "3GPP GUP, Requirements, Stage 1 Release 6", March 2003. See: [www.3gpp.org](http://www.3gpp.org)
- [20] 3GPP TS 23240-110: "3GPP GUP, Architecture Specifications, Stage 2 Release 6", April 2003. See: [www.3gpp.org](http://www.3gpp.org)
- [21] 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.
- [22] Lalitha Suryanarayana, Johan Hjelm, "Profiles for the situated web", in Proceedings of the eleventh international conference on World Wide Web, 2002, pp. 200-209
- [23] Wireless Application Group, User Agent Profile Specification, WAP Forum Approved Specification WAP-174, 10 November 1999. See: <http://www1.wapforum.org>
- [24] RuleML, <http://www.dfki.uni-kl.de/ruleml/>
- [25] Jess, <http://herzberg.ca.sandia.gov/jess/>
- [26] Mandarax <http://mandarax.sourceforge.net/>
- [27] 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
- [28] Universal Serial Bus, <http://www.usb.org>
- [29] Aran Ziv, Tal Segalov, "FlashDrive Performance and Reliability, White Paper", September '03
- [30] ISO/IEC 7816-4:1995, "Information technology -- Identification cards -- Integrated circuit(s) cards with contacts -- Part 4: Interindustry commands for interchange"
- [31] ETSI TS 102 221, "Smart Cards; UICC-Terminal interface; Physical and Logical characteristics", V6.0.0, 02-2003
- [32] ISO/IEC 7816, Information technology - Identification cards - Integrated circuit(s) cards with contacts, 1997-2004
- [33] 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
- [34] Java Card Technology, <http://java.sun.com/products/javacard/index.jsp>
- [35] Multi-Application Operating System (MULTOS), <http://www.multos.com>
- [36] PC/SC Workgroup, <http://www.pcscworkgroup.com/>
- [37] Open Card Framework, <http://www.opencard.org>
- [38] IST Project FASME, "Facilitating Administrative Services for Mobile Europeans", <http://www.fasme.org>
- [39] IST Project SM-PAYSOC, "Secure Mobile PAYments and Services On Chip", IST-2001-32526, <http://www.smpaysoc.org>
- [40] IST Project SMARTCITIES, "Multi-Application Smart Cards in Cities", IST-1999-12252
- [41] IST Project VERIFICARD, "Tool-assisted Specification and Verification of JavaCard Programmes", IST-2000-26328, <http://www.verificard.com>
- [42] IST ACTS-SCARAB, "Smart Card and Agent enabled Reliable Access"
- [43] Sun Microsystems Inc, "Java Card Platform Security", Technical White Paper pp. 6-9, <http://java.sun.com/products/javacard/JavaCardSecurityWhitePaper.pdf>
- [44] Sun Microsystems Inc, "Specification for the Java Card Platform, v2.2.1", <http://java.sun.com/products/javacard/specs.html>
- [45] Klara Nahrstedt, Jonathan M. Smith, "The QoS Broker", IEEE Multimedia, 2(1), Spring 1995.

- [46] Hannes Hartenstein, Andreas Schrader, Andreas Kessler, Michael Krautgärtner, Christoph Niedermeier, "High Quality Mobile Communication", *Kommunikation in Verteilten Systemen 2001*: 279-289.
- [47] Andreas Kessler, 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.
- [48] Andreas Kessler, 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.
- [49] 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.
- [50] 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.
- [51] 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.
- [52] Harry Chen, Tim Finin, Anupam Joshi. "An Intelligent Broker for Context-aware Systems", *Adjunct Proceedings of Ubicomp 2003*, Seattle, Washington, USA, October 2003.
- [53] P. Bellavista et al., "CORBA Solutions for Interoperability in Mobile Agents Environments", *International Symposium on Distributed Objects and Applications*, September 2000, Belgium.
- [54] J. Delgado et al., "An Architecture for Negotiation with Mobile Agents", *IFIP MATA02*, October 2002, Spain.
- [55] 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.
- [56] UPnP, <http://www.upnp.org/>
- [57] JXTA, Project JXTA, <http://www.jxta.org>
- [58] Kubiatiowicz, 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 *Proceedings of the Ninth international Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000)* (Boston, MA, November 2000), pp. 190-201.
- [59] 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).
- [60] 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).
- [61] 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
- [62] 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, Cooperative Messaging System"
- [63] Java Mobile Operation Management. See: <http://www.jcp.org/jsr/detail/232.jsp>
- [64] J2EE Client Provisioning. See <http://www.jcp.org/jsr/detail/124.jsp>
- [65] 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..
- [66] D. Gelernter: *Generative communication in Linda*. *ACM Trans. Program. Lang. Syst.*, vol. 7(1): pp. 80–112, 1985. ISSN 0164-0925.
- [67] 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.
- [68] MIT – Massachusetts Institute of Technology: *Project Oxygen*. <http://oxygen.lcs.mit.edu/>, Nov. 2002.
- [69] M. Roman and R. Campbell: *GAIA: Enabling Active Spaces*. *9th ACM SIGOPS European Workshop*. Sep. 2000. Kolding, Denmark.
- [70] M. Weiser: *The Computer for the 21st Century*. *Scientific American*, pp. 94–104, Sep. 1991.