The Simplicity System Architecture

N. Blefari Melazzi[(1)], S. Salsano[(1)], G. Bartolomeo[(2)], F. Martire[(2)], E.Fischer[(3)], C. Meyer[(3)], C. Niedermeier[(3)], R. Seidl[(3)], E. Rukzio[(4)], E. Koutsoloukas[(5)], J. Papanis[(5)], I. S. Venieris[(5)]

(1) DIE, Univ. of Rome "Tor Vergata", (2) Radiolabs, Rome, Italy, (3) Siemens, Munich, Germany, (4) Media Informatics Group,University of Munich, (5) National Technical University of Athens.

***Abstract*— As of today, heterogeneous services, terminals and networks create a burden of complexity on the shoulders of final users. Concepts like personalization and ease of use of ICT (Information and Communication Technologies) services are fundamental features to unleash the full potential of beyond 3G systems and paradigms such as ambient intelligence, ubiquitous connectivity, context-aware services, pervasive computing and novel access technologies.**

The aim of the Simplicity project is to ease the user interaction with devices, services and functionalities. In more details our vision is to design and deploy a "framework" able to decouple user needs and user devices, as well as services deployment and fruition, from the underlying networking and service support technologies. With this goals in mind the Simplicity System will support the effective exploitation and user acceptance of the ICT facilities. This paper provides a description of the Simplicity System Architecture.

***Index Terms*— service personalization, service portability, user profile, terminal auto-configuration.**

I. Introduction: the Simplicity Approach

The Simplicity (Secure, Internet-able, Mobile Platforms LeadIng CItizens Towards simplicity) project is a European Union program, scheduled to run for two years (January 2004 - Decmber 2005) that includes 11 major European industrial organizations, network operators, SMEs, research labs and universities [1].

The strategic goal of Simplicity is to simplify the process of using current and future "services" providing a user-friendly solution. More specifically, the project aims to design and deploy an architecture allowing:

- easy personalization of services to match user preferences and needs,
- seamless portability of distributed services, applications and sessions across heterogeneous terminals and devices,
- smooth adaptation of services to available networking and service support technologies and capabilities.

The personalization concept is based on a user profile which provides a common underlying information model for all the elements of the Simplicity architecture. This representation has been called "Simplicity User Profile" (SUP), extending the "Generic User Profile" by 3GPP [7]. The full XML definition of the SUP is included in [8].

In our view, each user will be provided with a personalized profile, giving access to different services, perhaps using heterogeneous classes of terminals (see also [2], [3]). The personalized user profile will allow automatic, transparent customization and configuration of terminals/devices and services, uniform mechanisms for recognizing, authenticating, locating and charging the user, policy-controlled selection of network interfaces and applications services. Thanks to the profile, users will also enjoy the automatic selection of services appropriate to specific locations (e.g. the home, buildings, public spaces), the automatic adaptation of information to specific terminal devices and user preferences, and the easy exploitation of different telecommunications paradigms and services.

The user profile will be either stored in a so-called Simplicity Device (SD). Though it seems natural to think of the SD as a physical device (e.g., an enhanced SIM card, a Java card, a USB stick, a sensor, etc.), the SD could also be implemented as a network location or a software agent. In some case the physical SD could store "pointers" to profile information residing in the network. If the SD is a physical device, users could personalize terminals and services by the simple act of plugging the SD into the chosen terminal.

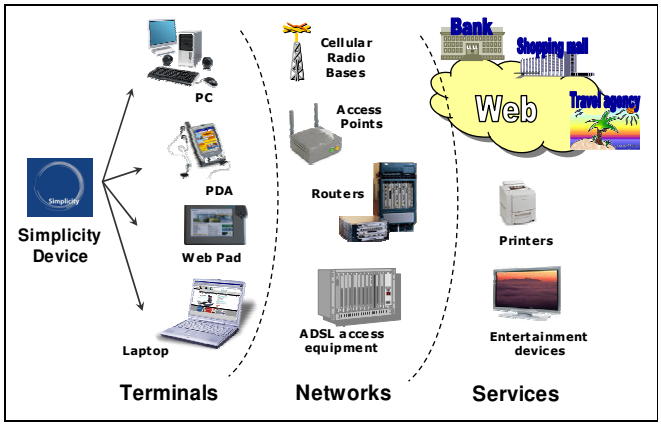

Figure 1: Overall reference scenario

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. Figure 1 shows the overall picture of the Simplicity scenario, where the SD interacts with Terminals, in order to configure and adapt the Terminals (and the Applications therein contained), the access to Networks and the access to Services. Control of personal data, security of information, and user privacy are key issues for the Simplicity approach.

The Simplicity system also encompasses a Brokerage

This work was partly supported by the EU in the framework of the project "Simplicity" IST-2004-507558

Framework. This brokerage level will use policy-based technologies (e.g. policies for mobility support, Qos, security, SW downloads) to orchestrate and adapt network capabilities, taking into account user preferences and terminal characteristics. Also it must provide adaptation capabilities to the considered context (location, time, etc) and eventually an orchestration of events, managing at the same time access of several users to the same resources, services and location.

In this paper, Section 2 will shortly discuss how "Simplicity scenarios" from the user perspective have been considered in the design phase, while Section 3 provides a description of the specification of the system architecture

II. Simplicity Scenarios from the User Perspective

In order to analyze the requirements coming from the user perspective, the Simplicity project has analyzed a large set of user scenarios. Generic functions derived from the scenarios have been considered in the definition of the Simplicity Architecture, trying to fulfill all the identified requirements. Due to space constraints the complete scenarios description and requirement analysis (using UML methodology) can be found in [4], for more condensed information see also [3], [5].

Just to name one exemplary scenario which describes how the user can profit from simplified communication spaces we mention the 'Mobile Worker and Gaming' scenario. Here, we examined in detail how the modern day worker interacts with his terminals, network technologies, applications, data and services throughout his business times and private time. Focus in this scenario was put on how Simplicity can provide a heterogeneous platform that easily integrates access to all information the user desires while requiring minimal user interaction to simplify the overall user experience.

III. Simplicity Architectural Aspects

Starting from the requirements coming from the user scenarios, the design of the architecture has been split in two stages: "high level architecture" and "detailed architecture". In the high level architecture a set of "logical" functional entities has been identified with no concern on the mapping of these entities into physical nodes and on the needed communication mechanisms. The design methodology and the high level architecture can be found in [6].

In this paper we will focus on the Simplicity "detailed architecture". At this level the architecture foresees a number of software and hardware entities that are part of the "Simplicity system" and collectively provide Simplicity services to users. The Simplicity system interacts with other "external" elements, such as user terminals, applications running on user terminals, network elements, servers, network services and so on. An overall picture of the Simplicity system is represented in Figure 2. The main components of the Simplicity system are the Simplicity Device, the Terminal Brokers (TBs), the Simplicity Personal Assistant (SPA), the Network Brokers (NBs). The interaction of the Simplicity system with existing ("legacy") application and services is depicted, as well as the interaction of the System with external applications which are designed to exploit the capability of the system (denoted as "Simplicity enabled 3rd party applications").

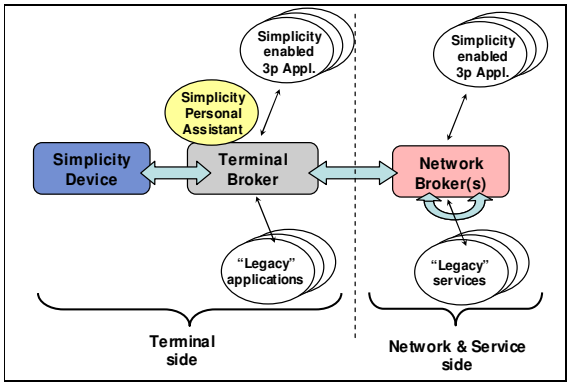

Figure 2: The Simplicity system components

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

The Terminal Brokers (TBs) manage the interaction between the information stored in the SD and the terminal in which the SD is plugged in. These SW modules enable the SD to perform actions like terminal capability discovery, adaptation to networking capabilities and to the ambient, service discovery and usage, adaptation of services to terminal features and capabilities. TBs cater also for the user interaction with the overall Simplicity system (including network technologies and capabilities).

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

The Network Brokers (NBs) have the goal to provide support for service advertisement, discovery and adaptation. Moreover, they orchestrate service operation among distributed networked objects, taking into account issues related to the simultaneous access of several users to the same resources, services, and locations. They also share/allocate available resources, and manages value-added networking functionality, such as service level differentiation and quality

of service, location-context awareness, and mobility support.

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

The interfaces between the identified entities (see Figure 2) must be clearly defined. In particular, three fundamental interfaces have been addressed: 1) the interface among the "brokers", which will be called "Simplicity Broker Communication" – SBC; 2) the interface between the brokers and the external applications willing to exploit the system, called "Simplicity Applications Interface" – SAI; 3) the interfaces between the Terminal broker and the Simplicity Device, called "SD Access Interface" – SDAI. For space constraints we cannot cover in detail the specification of these interfaces. We will rather discuss the decomposition of the architecture in "sub-systems" showing which sub-system takes care of the identified interfaces.

3.1. Detailed architecture

In order to achieve a flexible and modular specification, TBs and NBs have been de-composed in a set of separate logical components called "sub-systems" that implement the required functions. "Reusable" sub-systems implement common Simplicity functions, while specific subsystems may be defined to implement specific applications in the Simplicity system. The interaction between subsystem is defined in terms of asynchronous events exchange (specified using UML class and sequence diagrams).

TABLE I
LIST OF BROKER SUB-SYSTEMS

SBC – Simplicity Broker Communication
SAIM – Simplicity Applications Interface Manager
SDAM – Simplicity Device Access Manager
Profile Management
Capability Management
Policy Management
Policy Decision Point
Service Management
Presence
User contracts & pricing
Access Network
SDS-c - Secure Distributed Storage client
Application specific subsystems

The communication between sub-systems that are physically located in different brokers constitute the "Simplicity Broker Communication" (SBC). The SBC specifies how the brokers talk each other in the Simplicity system. Each broker include a dedicated sub-system that implements the SBC.

Table I shows the list of defined sub-systems, while Figure 3 provides a graphical representation of the Simplicity detailed architecture which shows the sub-systems and their relation to the other defined entities. Most of these subsystems are provided in two versions, one for the terminal broker and one for the network broker. In the next subsections the features of the most important sub-systems will be discussed.

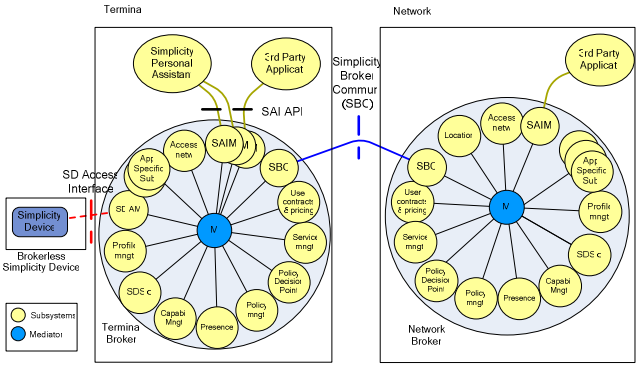

Figure 3: Overall picture of Simplicity detailed architecture

3.2. Internal Broker architecture

The internal architecture of the Terminal and Network Brokers does not need to be subject to "standardization". Different implementation are acceptable, given that they comply with the specification of inter-broker SBC interface. Nevertheless, the Simplicity project provided a "reference" specification for the internal architecture of the broker, which has been taken as input for the implementation of a demonstrator.

The brokers are defined as modular software systems that enable easy and flexible integration of different components, called sub-systems, interacting asynchronously through a central entity called Mediator. The internal architecture of brokers allows for flexible integration of new functionality and a minimal impact when already integrated functionality is removed. The sub-systems are responsible for their own tasks and need no further knowledge about the rest of the system. They communicate with one another in an asynchronous event based scheme, through the Mediator. Each subsystem, upon instantiation, registers to the Mediator for events that it is interested to receive, and publishes the events that it produces.

The Mediator is responsible for the filtering, adaptation and relaying of events between subsystems. It maintains a mapping of event types to subsystem ids so that it can route events to their intended recipients. The Mediator may cooperate with a policy decision engine that may override the normal dispatching mechanism and re-route events to other targets, filter certain types of events and reject them or resolve conflicts that may occur in complex setups.

This approach enables flexible addition and removal of subsystems, without affecting the rest of the system. It has the advantage that it allows encapsulation of new functionality within a Broker, without restricting its pre-existing functionality. Any subsystem can be plugged into any Broker, thus providing maximum flexibility for the deployment of functionalities in the Simplicity system.

A special category of subsystem is the Adaptor subsystem, which is used to introduce legacy entities into Simplicity. Adaptors communicate with the rest of the Simplicity system like an ordinary subsystem and they implement the required adaptation logic in order to interact with the legacy entity. This way it is possible to introduce legacy entities into the system without the need to change their interface or implementation.

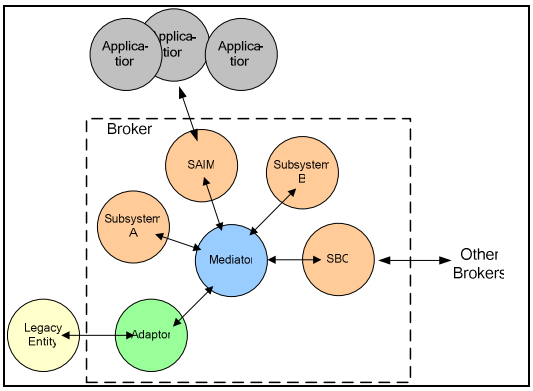

Figure 4: Internal Broker Architecture

3.3. SBC

The Simplicity Broker Communication (SBC) mechanism aims to extend the asynchronous event based intra-broker communication mechanism, in order to include other brokers as well. This mechanism acts as a transparent bridge between two remotely located mediators, handling: (i) the discovery of any required resources/subsystems on each of the remote brokers in order to decide what events need to be dispatched remotely, (ii) the required orchestration between the involved brokers so that the discovered subsystems will also be considered when an event is submitted for dispatching, and (iii) the actual transfer of events, using an appropriate XML based protocol (e.g. SOAP).

The discovery of required subsystems in a remote broker follows a "what is missing" approach. In every broker a mapping between suppliers of events and consumers of events reveals the event types that cannot be served locally.

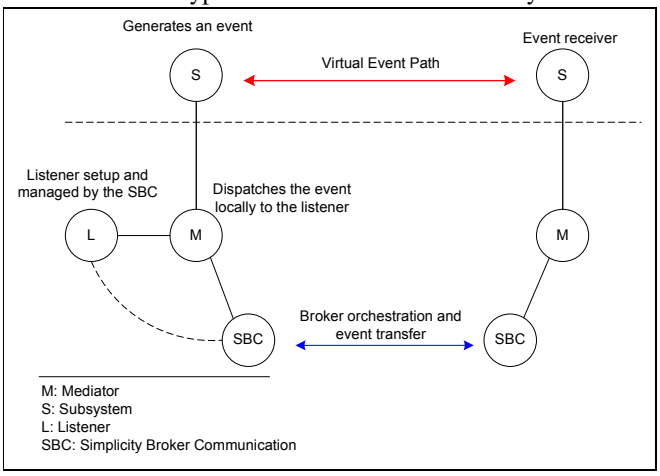

Figure 5: SBC mechanism

The involved subsystems, lying at the two edges of the communication are not aware of the networking acts between them and operate as if they were attached on the same Mediator.

In the process of the SBC specification, Simplicity will also specify the inter-SBC interface as an asynchronous XML based protocol, called Simplicity Asynchronous Event Protocol (SAEP). SAEP will describe the structure of the messages that SBCs exchange, along with the necessary exchange patterns and bindings with underlying protocols used as transport mechanism for them (such as SOAP, HTTP). This is a useful step for opening Simplicity interactions to different architectures from what has been described here. For most of its need, Simplicity Broker Communication can reuse existing protocols and communication paradigms.

3.4. SAIM

One of the principles of the Simplicity System is that it does not differentiate between native and 3rd party applications. The applications should not be conscious of the brokerage framework and event strategies that take place during operation in order to offer specific services to applications. This requirement led us to conceive and develop the Simplicity Application Interface Manager (SAIM) subsystem. The SAIM allows a 3rd party application to execute on top of the Terminal Broker and to use the functionalities of Simplicity, but without being aware of the comprehensive mechanisms of the Simplicity brokerage framework. The SAIM subsystem must be registered at the Mediator, and has the functionality to dispatch and handle events. This subsystem offers 3rd party applications a consistent interface, the Simplicity Application Interface (SAI) that makes available the underlying Simplicity mechanism transparently: SAI offers functionalities related to user interactions, service subscription, personalization of application and device, payment, user location information, and so on. The SAI cloaks the complexity of the middleware (Mediator, SBC) totally. The SAI interface is also used by the SPA to interact with the Terminal broker.

3.5. SDAM

The Simplicity Device (SD) is the "key" to the Simplicity System. Without an SD, users cannot access Simplicity. The main role of the Simplicity Device is to store the Simplicity User Profile (SUP), preferences and policies in a secure and safe way.

The ideal candidate for the SD ("ideal SD") should have an unbounded embedded secure and reliable memory space, a processing capability as high as possible, minimal physical size and minimum weight. During requirement analysis, different implementation alternatives have been investigated (flash memories, Java Card, and Bluetooth phones) but unfortunately we found that none of them offers all the aforementioned features at the same time.

Anyway, an ideal SD may be implemented using three elements: one physical SD, one or more network repository and parts of the TB. In order to answer to the necessity to guarantee integrity and confidentiality of the sensitive user's information, and to ensure an access to the simplicity system limited only to authenticated and authorized customers we may consider various solutions.

Moreover, in the case in which the SUP resides in a network repository, it is necessary to protect also the transfer of information from possible attacks and interceptions.

Some available mechanisms that resolve the aforementioned aspects are:1)Ciphering (symmetric/asymmetric), 2)digital watermark/ certificates, 3)AKA mechanism, 4)HTTPS (TLS/SSL), 5)IPsec.

Depending upon which physical SD is employed some functionality ideally residing on the SD are shifted to a TB subsystem called Simplicity Device-Access Manager (SDAM).

This subsystem will collect events targeted to the SD and will translate them into messages of the specific communication mechanism of the SD implementation. Adjusting the different SD implementations to the SD-AM requires the presence of communication controllers which will interact with the specific communication interfaces of each SD implementation. The SDAM should provide SUP data information to requesting subsystems using a standard language. The use of XML seems to be a unanimous choice; however this does not prevent the possibility to use other data format in each specific SD implementation (e.g. binary format). Therefore, the SDAM is able to convert one or more specific data format(s) into a standard XML instance document. The SDAM also provides support for privacy and controlled disclosure of information. ù

By hiding implementation details on the physical SD, the SDAM offers more freedom to the Simplicity programmer, who is able to exploit the same functionalities available from the ideal SD whatever physical SD is really owned by each Simplicity user.

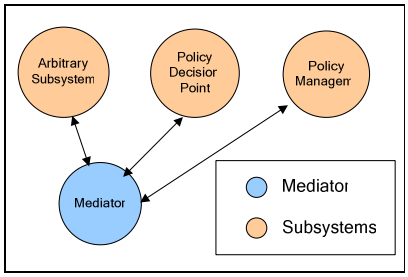

Figure 6: Policy Architecture

3.6. Policy Architecture

As already mentioned, one of the key functionalities of the Simplicity architecture is the adaptation of services, applications and terminals based on different context data like user preferences and devices capabilities. To address this, the Simplicity architecture contains two subsystems, Policy Decision Point (PDP) and Policy Management that could be part of the terminal as well as the network. These subsystems are responsible for the policy-based decision processes and the management of the different policies. Policies can be seen as sophisticated IF-THEN – statements which are interpreted by the PDP. Figure 6 illustrates the relation between the different subsystems. If an arbitrary subsystem needs information for an adaptation process it sends a corresponding request to the PDP. This subsystem requests the policies from the Policy Management subsystem and the context information from other subsystems (e.g. context management subsystem, capability management subsystem, profile management subsystem) that are needed for the current decision process. After that, the PDP sends the result back to the subsystem which asked for the information that was needed for an adaptation.

To achieve the needed flexibility, context information, the policies and the adapted services are separated, which makes it easy to change the context, to modify policies and to integrate new adaptations into the Simplicity system. Furthermore the policy architecture is not designed for a specific kind of adaptation. This allows the integration of arbitrary adaptations or decisions requested by different subsystems.

When looking on the current issues in the field of policy-based adaptive systems, the most important problems are currently conflict detection and resolution, distribution, complexity and performance. We address these problems through the usage of modules as well as domain and priority concepts.

IV. CONCLUSIONS

The Simplicity project addresses a crucial issue for future systems beyond 3G. The terminals, networks, services and applications adapt proactively to the user and not vice versa. This is very important for the acceptance and usage of new innovative services by the user. We intend to prove the advantages of this concept and to show its feasibility by implementing the presented architecture based on a set of prototypes. A key parameter to judge the outcomes of Simplicity is the user acceptability and usability of the Simplicity Device. Proof of this will be shown via a user-centered approach. This concept can also be instrumental in opening up new research directions or extensions of current ones, including e.g. user profile definition and handling, user tailored applications and API, middleware tools for high layer re-configurability or dynamic network configuration as a function of users' context.

REFERENCES

[1] IST Simplicity project: http://www.ist-simplicity.org

[2] N. Blefari-Melazzi, G. Bianchi, G. Ceneri, G. Cortese, S. Kapellaki, K. Kawamura, C. Noda, S. Salsano, I. S. Venieris: “The Simplicity project: easing the burden of using complex and heterogeneous ICT devices and services. Part I: Overall Architecture”, IST Mobile&Wireless Communications Summit 2004, June 27-30 2004, Lyon, France.

[3] N. Blefari Melazzi et al. “The Simplicity Project: Managing Complexity in a Diverse ICT World”, in Ambient Intelligence, IOS Press (in press)

[4] Simplicity Deliverable D2101: “Enhanced use cases, requirements and business models”; http://server.ist-simplicity.org/deliverables.php

[5] R. Seidl, F. Berger, S. Kapellaki, T. Frantti, E. Rukzio, J. Hamard, N. Blefari Melazzi: “User scenarios for simplified communication spaces”, IST Mobile&Wireless Communications Summit 2004, June 27-30 2004, Lyon, France.

[6] Simplicity Deliverable D2201: “Initial system architecture specification”; http://server.ist-simplicity.org/deliverables.php

[7] 3rd Generation Partnership Project. Data Description Method (DDM) - 3GPP Generic User Profile (GUP). Technical specification of Technical Specification Group Terminals, Version 6.1.0. 2004.Reference example

[8] Simplicity Deliverable D2202: “Final system architecture specification”; http://server.ist-simplicity.org/deliverables.php