THE SIMPLICITY PROJECT: INITIAL SYSTEM ARCHITECTURE SPECIFICATION

N. Blefari Melazzi[(1)], S. Salsano[(1)], R. Seidl[(2)]
[(1)]DIE, Università di Roma "Tor Vergata", [(2)]Siemens AG, Munich, Germany
E-mail: blefari@uniroma2.it, stefano.salsano@uniroma2.it, seidl.robert@siemens.com

Abstract

The research community is working towards a new all-encompassing vision of the Internet. This vision certainly includes powerful transport and switching technologies, wireless networks easy to deploy and to manage, stimulating concepts such as the disappearing computer and smart spaces, sensor networks, ubiquitous connectivity, on demand access with QoS guarantees, ever-evolving entertainment and office appliances, miniaturized devices, and so on and so forth. However, all these facets of the new Internet are probably not sufficient by themselves to unleash its full potential. To reach this goal, personalization and ease of use of ICT (Information and Communication Technology) services are a fundamental feature.

In fact, as technology develops, people are using an ever broader and heterogeneous range of ICT devices and network-based services. The result is an enormous burden of complexity on the shoulders of users, service providers and network operators. Excessive complexity, in turn, creates obstacles to effective exploitation and acceptance of beyond 3G systems and paradigms such as ambient intelligence, context-aware services, pervasive computing and novel access technologies. The goal of the Simplicity project, supported by the European Union, is to reduce this complexity by: i) providing automatic customization of user access to services and the network; ii) automatically adapting services to terminal characteristics and user preferences; iii) orchestrating network capabilities.

In this paper, after a short introduction on the Simplicity project, the Simplicity approach is presented in Section 2. Section 3 reports an example of a "Simplicity scenario" from the user perspective and Section 4 provides some details on the initial specification of the system architecture.
.

Keywords: service personalization, service portability, service adaptability, service discovery, user profile definition and handling, auto-configuration of terminals.

1. INTRODUCTION

The Simplicity project is a European Union 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. The project acronym intends to convey the very aim of the project: 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 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". 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 to provide users with a simple and uniform mechanism for customizing services and terminals;
- Validate feasibility and benefits of the Simplicity approach within a test-bed.

2. THE SIMPLICITY APPROACH

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 could provide a user-friendly solution to the challenges posed by a diverse service and technology environment.

The personalization concept is based on a "user profile". In our view, each user will be provided with a personalized profile, providing access to different services, perhaps using different classes of terminals (see also [2], [3]). Creating and maintaining the user profile will involve also 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. 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 by using a high-level description language.

The user profile will be either stored in a so-called Simplicity Device (SD) or 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 stick, a sensor, 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.

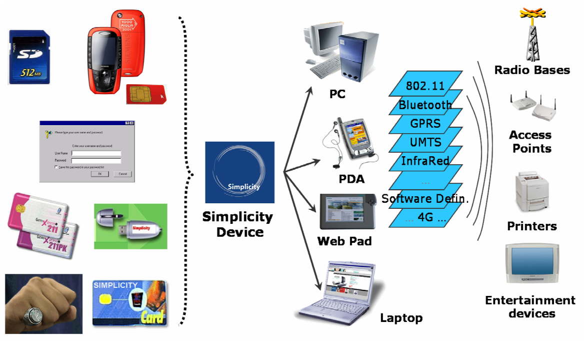

Figure 1: Overall reference scenario

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 and plugging the SD; when the user moves from one terminal to another the application/session automatically adapts to the new context.

Figure 1 presents the overall reference scenario, including a set of "candidate" Simplicity Devices that can be used to configure terminals and services.

3. MOBILE WORKER AND GAMING: A SAMPLE SCENARIO FROM THE USER PERSPECTIVE.

The Simplicity project began with a requirement analysis process. In the first phase of the process, the project partners created twenty-seven narrative descriptions of user scenarios, expressed in natural language. In a second phase, we used these scenarios to identify the general functionalities to be provided by the system. A subset of these scenarios were formalized as use case and sequence diagrams written in the Universal Modelling Language (for details see [3], [4] and [5]). One of them, named "Mobile Worker" is depicted in Figure 2 and briefly described below, as an example.

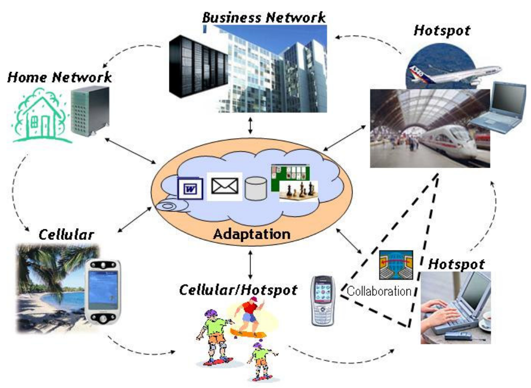

Figure 2: Mobile Worker and Gaming

In the "Mobile Worker" scenario a software assistant installed in the network or in the SD card provides easy and convenient communication between user and system. A brief narrative description is the following. 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. 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, his 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. 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. The session is established according to Brad's personal settings, which means that sound is enabled and he uses WLAN rather than GPRS.

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, where 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. SIMPLICITY ARCHITECTURAL ASPECTS

A key attribute of Simplicity is re-configurability, at various levels. Re-configurability is typically understood as operating at lower layers (e.g., software defined radio, which indeed may be an "add on" to our solution). However, to integrate different paradigms from the user point of view, it is necessary to break, as much as possible, logical wires that still tie mobile users to networks and services, also at upper layers. This way, heterogeneous and mobile access networks can be really integrated, as IP has glued heterogeneous networks. To enable this full-spectrum re-configurability, our system encompasses three main components: the Simplicity Device, the Terminal Broker and the Network Broker (see Figure 3).

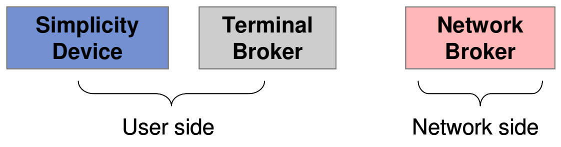

Figure 3: System components

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 Broker is the entity that manages the interaction between the information stored in the SD and the terminal in which the SD is plugged in. The Terminal Broker enables 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. The Terminal Broker caters also for the user interaction with the overall Simplicity system (including network technologies and capabilities).

The Network Broker has the goal to provide support for service advertisement, discovery and adaptation. Moreover, it orchestrates service operation among distributed networked objects, taking into account issues related to the simultaneous access of several users to the same resources, services, and locations. It also shares/allocates available resources, and manages value-added networking functionality, such as service level differentiation and quality of service, location-context awareness, and mobility support.

4.1. DETAILED ARCHITECTURE

An overall picture of the Simplicity architecture is represented in Figure 4. This high-level concept provides an abstract explanation of the "Simplicity Environment". The Simplicity Environment includes the following entities:

- Simplicity Personal Assistants (SPAs),
- 3rd Party Applications
- Simplicity Brokers
- Legacy entities
- Simplicity Device

A distinction is made between the Simplicity System and the Simplicity Environment. The Simplicity System is a set of software and hardware components, based on Simplicity specifications that collectively provide Simplicity services to users. The Simplicity environment has a broader scope than the Simplicity System. The Simplicity Environment includes the Simplicity System and all elements that interact with the Simplicity System, including user terminals, applications running on these terminals, network elements, servers, network services and so on.

A Simplicity Personal Assistants (SPA) is the component in the Simplicity System that handles context evaluation, learning, rule generation, personalization and service subscription/orchestration/ invocation. SPAs interact with users via a convenient User Interface. Their look, behavior and actions are strongly adapted to user preferences and needs.

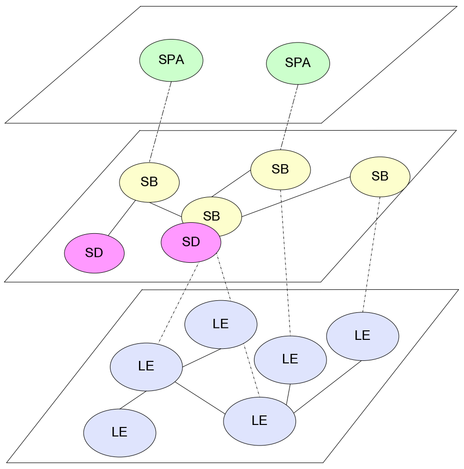

Figure 4:Overall picture of Simplicity architecture

3rd Party Applications run on the user terminal and on other legacy entities in the Simplicity System. 3rd Party Applications use features provided by the Simplicity System.

Simplicity Brokers (SB) are software subsystems that enable easy and flexible integration of different functionalities. SBs are used to provide functionalities to the Simplicity Environment. These functionalities may include (but are not limited to): collection and combination of information from other entities, provision of the resulting information (context, preferences, policies) to other entities, management of network-related functionalities (e.g. advanced mobility management), and provisioning of services. Simplicity Brokers will be instantiated in User Terminals and in network servers. Simplicity defines communication among Simplicity Brokers. In what follows we will refer to this communication as Simplicity Broker Communication (SBC). As far as possible, Simplicity Broker Communication will reuse existing protocols and communication paradigms.

Legacy Entities include all entities that are not Simplicity-enabled and Simplicity-configurable, but are accessible and configurable through a non-Simplicity- interface. Examples of Legacy Entities are legacy services, sensors etc. Legacy Entities may provide services to Simplicity users and to Simplicity elements, but do not use Simplicity Broker Communication.

SPAs, Brokers and Simplicity Devices are components of the Simplicity System; 3rd Party Applications and Legacy Entities are not. Legacy entities are connected to the Simplicity System via an adaptor.

SPAs operate on the very "top layer" (see Figure 4), interacting with the user and with other elements of the Simplicity system. SPAs are meant to provide as much support as possible to the user. They act autonomously whenever they can, requiring only minimal input from the user.

In the "middle layer", distributed Simplicity Brokers provide functionalities to the Simplicity Environment. There are several different Simplicity Brokers that offer different services and data. Due to Simplicity's distributed, non-centralized, approach, these Brokers need to communicate with each other across network domains. To meet this need, Simplicity introduces a special communication mechanism, called Simplicity Broker Communication (SBC). In this layer, Simplicity (Terminal) Brokers interact with Simplicity Devices. The way in which this interaction is handled will depend on the implementation of the Simplicity Device. A "powerful" Simplicity Device may include a Broker of its own; in a "dumb" Device, the Broker will use an adaptor, the same mechanism used with Legacy Entities.

The "lower layer" is dedicated to Legacy Entities. These entities are not Simplicity-enabled. They can only be accessed via an adaptor. This adaptor enables the seamless integration of Legacy Entities into the Simplicity Environment, enhancing LE functions with Simplicity-specific features. The category of Legacy Entities also includes any newly developed service that interacts with the Simplicity System without using Simplicity Broker Communication.

4.2. BROKER ARCHITECTURE

Brokers provide an architecture that allows the flexible integration of subsystems at run-time (see Figure 5). To achieve this goal, functionality is encapsulated in separate subsystems each of which is responsible for its own tasks and needs no further knowledge about the rest of the system. Interaction between these subsystems is enabled by a Mediator which relays messages between them.

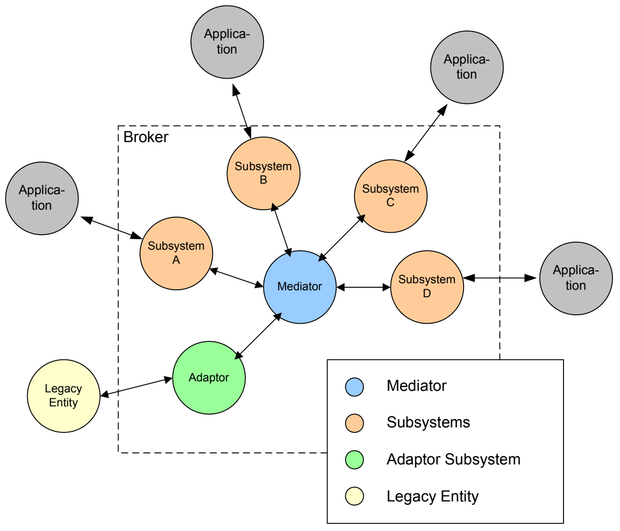

Figure 5: General Broker Architecture

This approach enables flexible addition and deletion of subsystems, without affecting the rest of the system. It has the advantage that it allows encapsulation of new functionality within a Broker, without restricting its pre-existing functionality. Any subsystem can be plugged into any Broker, thus providing maximum flexibility for the deployment of functionalities throughout the Simplicity Environment. The Simplicity Broker architecture is a generic integration architecture, which consists of a Mediator and at least two subsystems. The two subsystems do not have to know about each others' functionalities. Whenever a new subsystem is introduced into a Broker, the subsystem has to register with the Mediator. The registration procedure includes publishing of all events that are sent by this subsystem, and the registration for all events that can be processed by the subsystem.

Adaptor subsystems allow the introduction of legacy entities. These subsystems represent an interface to the legacy service. All information flows between Simplicity and the Legacy Entity pass via the adaptor subsystem, which acts like an ordinary Simplicity subsystem. The complete adaptation logic is implemented in the adaptor subsystem avoiding the need to change the legacy entity. All Simplicity-related information is stored, managed and processed within the adaptor. The Mediator is responsible for the filtering, adaptation and relaying of events between subsystems. There is no direct communication between subsystems; all communications are based on events. Subsystems normally function autonomously. When the performance of a task by a subsystem requires assistance from other subsystems, it is the Mediator that provides the intelligence to coordinate their work.

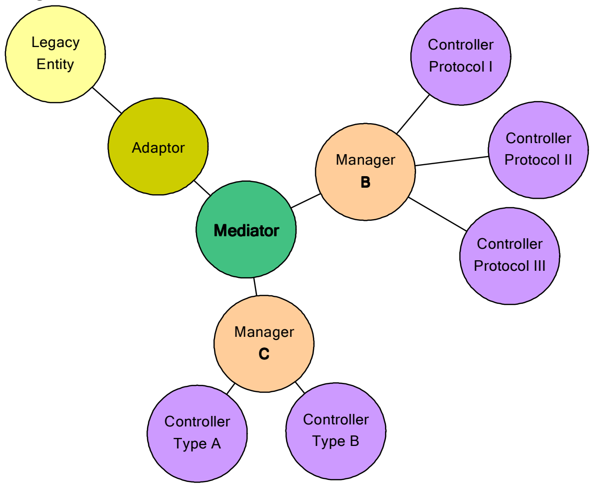

Figure 6: Broker Architecture with Managers and Controllers and Legacy Entities

Each autonomous subsystem takes care of its own sphere of interest. A policy decision engine reconciles conflicting requests. In specific cases, defined by its configuration settings, the Mediator allows the decision engine to interfere with messages and to reject requests. The rejection is immediately notified to the sender of the message. This interception can take place before the Mediator relays a message to other subsystems and/or after receiving responses from these subsystems. A further possible scenario is a direct message to the decision engine that is not sent to other Managers.

5. CONCLUSIONS

The Simplicity project addresses a crucial issue for future systems beyond 3G (and for the Internet at large). The concept developed in the project can directly impact the way citizens live and work. We intend to prove this concept by implementing our proposed architecture, to show its feasibility. 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-centred approach. This concept can also be instrumental in opening up new research directions (or extensions of current ones), including:

- User Profile definition and handling;
- User (and SD) tailored applications and API;
- Middleware tools for high layer re-configurability;
- Network planning and dimensioning as a function of customers profiles (stored in the SD);
- Services and resources (including access networks) discovery and selection, as a function of SD profiles and of current network status;
- Dynamic network (auto)configuration and resource allocation as a function of users' profiles, collected users' statistics, observed current users' behaviour, users' location, and current network status;
- Flexible authentication mechanisms;
- Seamless service accessibility through heterogeneous and independently-owned network infrastructures;
- Simplicity aware network management, and related policies (proactive and reactive).

REFERENCES

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

[2] N. Blefari-Melazzi et al.: "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: "Use cases, requirements and business models"; http://server.ist-simplicity.org/deliverables.php

[5] R. Seidl et al.: "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