

# The soft terminal: extending service intelligence from the network to the terminal

> The soft terminal is a powerful concept that enables users to access advanced network services in a user-friendly way.

## Introduction: Migration of Service Intelligence to the Edge of the Network

Even before the arrival of the Internet as we know it today, the move to push intelligence to the boundaries of the network, up to the level of the terminal, had already started. Telephone sets, for example, were enhanced by adding memory to store frequently dialed numbers and to support features such as last number redial. Later, new functions were added, such as an integrated answering machine or fax, and most recently even a web browser, e-mail clients, etc. However, this evolution does not necessarily imply that all intelligence is removed from the network. On the contrary, many new features can only be realized through a combination of intelligence in the terminal and network.

To take a simple example, Calling Line Identification Presentation requires that the network transports the calling number identification in signaling messages, while the terminal needs to be equipped with some logic and a small screen to interpret the information and display it to the user. In other words, the trend is not to "move intelligence out of the network to the terminal", but rather to "extend intelligence from the network to the terminal". End users are not interested in running applications on their terminals, but want to use

services. Hence they are not interested in the logic running on the terminal, but rather in the service offered by the combination of logic in the terminal and the network.

A similar trend can be witnessed in the IT world. Not too long ago, system developers were typically working in corporate networks centered around a large mainframe computer, with relatively dumb user terminals. With the advent of the PC, this architecture changed dramatically, and today most processing is done locally on PCs. Nevertheless, although PCs were originally designed to operate in a standalone mode, today most of them are connected to a network in which intelligence is distributed between the PCs (clients) and various servers. Note that also in this case intelligence has not been removed entirely from the network: many functions still reside on the network servers. Examples are security servers (firewalls, admission control, etc), file servers, version management of application software on the clients (e.g. automatic and remotely controlled upgrading of applications such as browsers, virus scanners, etc), and so on. The key to this evolution is "zero administration": using the power (processing, storage, etc) of the clients, under the control of the network operator, to minimize configuration and administration costs.

The Internet, in its turn, is exhibiting similar trends, but on a

larger scale. Equipment and protocols used in the core Internet Protocol (IP) network were initially designed to maximize throughput and scalability. Service functionality is typically located in the terminals (end-user domain) and servers (application service providers) outside the network. However, today more and more service functionality is being introduced into the access provider domain, in equipment such as the Network Access Server. In this paper we refer to this domain as "the edge of the network". Service elements located in the edge have the advantage that they can be closely coupled to network functionality, and become aware of the session parameters that determine the characteristics of the communication channel. Also in this case the intelligence in the terminals is complementary to that in the edge of the network.

A few examples illustrate this: typically a terminal is not permanently connected to the network, so incoming communications (e.g. e-mail) are often terminated on a network server; the user is only notified when he or she next connects to the network. In other models (e.g. instant messaging), the communication extends directly to the terminal. In this case, the terminal is considered as the last hop in the communication chain. Depending on the application and the status of the terminal (on- or off-line), the communication is either terminated in the network or in the

terminal. Also, for outgoing communications a terminal will typically connect first to an access node, referred to as a Network Access Server (NAS), in the access provider's Point of Presence (POP) to start a networking session, before contacting the actual server or terminal to which a connection is required. These POPs are evolving towards true service access platforms, as more and more service functionality is added in separate servers (e.g. portals, directory servers, etc). In this context, the terminal becomes an extension (the client side) of the service platform in the edge of the network.

### New Requirements

The above examples indicate that service intelligence on the terminal should not be thought of as separate applications, but rather as an integral part of a distributed service platform, which is largely driven by functions in the network. We refer to this terminal functionality as the "Soft Terminal". The concept is explained in more detail in the next section, but let's first have a look at the associated issues and requirements.

Increased terminal capabilities (processing power, memory, storage, etc) have given network operators and service providers the opportunity to remove some of the load from their network equipment and servers, and distribute it to the terminals. However, this evolution raises an important issue for the network and service providers: in most cases they do not control the terminal. While in the old monopoly-driven telecom world the network operators typically had full control over the telephone sets that were connected to their networks, this is no longer the case. Although the basic interactions between terminal and network are still dictated by standards, advanced terminal capabilities are beyond the control of the operator. The situation is even worse in the con-

text of Internet services, as distribution channels for PCs are completely different from those for online services on the net. Consequently, service providers are now faced with a distributed service platform over which they do not have full control.

Moreover, extending service logic to the terminal is fine, but does not always meet the end user's requirements. Although everybody knows how to use a traditional telephone set, more advanced sets are already causing problems to some users (e.g. how to program a list of telephone numbers in memory). The installation of software on PCs is even more of a problem, even with installation wizards. On the other hand, service providers do not want to send installation personnel out to every customer just to install a PC application, particularly if this application has to be frequently upgraded, adapted or extended.

To solve this problem, various software technologies have been developed that allow the full lifecycle of software components on the terminal to be controlled remotely, including installation and automatic upgrading, dynamic plug-in of additional components, etc. However, these technologies typically rely on an initial set of software components being already available on the terminal. In other words, there is a bootstrapping issue, as this initial set of software components has to be installed on the terminal somehow.

Another issue is the fact that the soft terminal will have to be "portable" in the broadest possible sense. Indeed, while the soft terminal is itself controlled by the service provider, the environment in which it runs is not. In other words, the soft terminal will have to adapt to the capabilities of the terminal on which it is installed (hardware, operating system, available protocol stacks, etc), as well as to the network environment in which this terminal is used; for example, a PC with a modem connection to an Internet Service Provider (ISP)

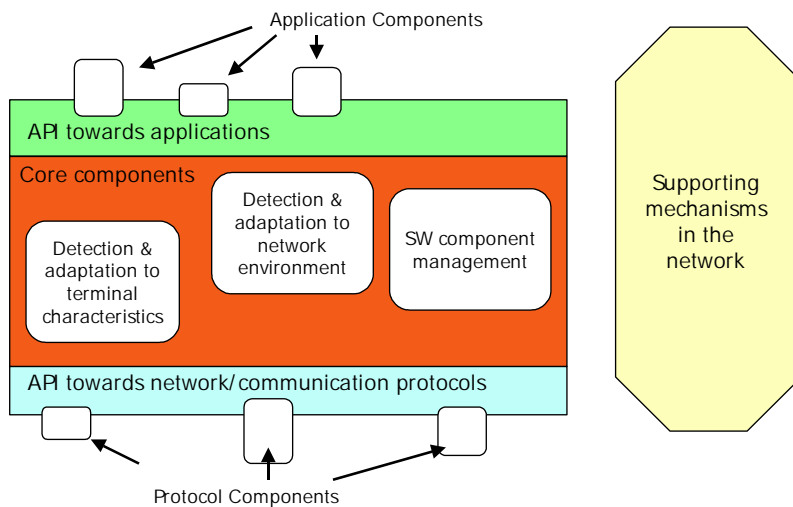
compared with a PC connected to a Local Area Network (LAN) with indirect connectivity to the Internet.

The core logic of the soft terminal should take care of the adaptation to the underlying platform and network environment, so that the service provider perceives a uniform service platform on which new service logic can be deployed, without worrying about portability issues. Note that given the rapid evolution of service and network technologies, new services may also require upgrades and extensions to the underlying communication protocols. Hence, the soft terminal should not only allow dynamic installation of application components, but also of protocol components. The latter is a complex problem as it involves replacing communication logic while the communication is active. It's like pulling away the legs underneath a chair, and replacing them with different ones while you remain seated.

Last but not least, being part of a distributed service platform, the soft terminal should offer clear Application Programming Interfaces (API) to add new modules. Which part of these APIs will be open to third parties will depend on the strategy of the service provider.

In summary, the issues and requirements associated with extending service intelligence from the network to the terminal, can be described as follows:

- Control by service provider: How can the network operator or service provider gain control over the communication functions on the terminal?
- User-friendly and cost-effective deployment: How can this be achieved without confronting the user with complex installation procedures, and while minimizing deployment costs for the service provider? In particular, how can the remote installation and management mechanism be bootstrapped?



**Figure 1 – High-level functional architecture of the soft terminal**

- Portability: How can the soft terminal adapt to the characteristics of the terminal on which it runs, as well as to its network environment?
- Flexibility: How can the soft terminal be made future safe, that is, how can it offer the flexibility to plug in or upgrade application components and even communication protocols?
- Basic platform for new services: How can the soft terminal be positioned as a platform for value-added services and applications? How can open (or closed) APIs be defined?
- can adapt to the network environment (see “portability” requirement);
- provide mechanisms to add and replace components – both application logic and protocol stacks (see “flexibility” requirement);
- offer well defined interfaces based on which new service components can be developed and plugged-in.

Seen from the user point of view, the soft terminal is the fundamental “middle-part” to turn applications into services. As mentioned earlier, end users are generally not interested in installing terminal applications; they simply want to use services. The soft terminal is designed precisely to meet this need by offering a way to turn a network terminal into a service terminal.

Figure 1 shows the high-level functional architecture of the soft terminal. The core components deal with the installation and upgrading of additional modules (software component management). They are also responsible for detecting the soft terminal’s environment, including the characteristics of the user terminal and the network environment. In other words, they are responsible for hiding the details and variety of terminals and local

network environments towards the service and application level. From these core components, two distinct (but related) APIs offer a clear framework for developing new communication protocols and new service and application logic.

Note that the flexibility towards different network protocols is an extremely complex technical problem: dynamically selecting, installing and upgrading communication protocols, while maintaining a network connection, requires leading edge telecom and software technologies. Although from the software viewpoint, these protocol components may look similar to any other software components, their telecom-specific characteristics impose additional requirements on the software architecture and technologies used. Typical issues that have to be taken into account are:

- Interaction between different layers in a protocol stack: how to dynamically compose protocol stacks from individual protocol components.
- Negotiation with the peer entity: selecting a protocol or protocol stack must take into account the capabilities and protocol stacks available at the peer.
- Dynamic replacement and modification of protocol stacks while the communication session remains active.

Last but not least, the soft terminal has to be considered as an integral part of a larger distributed service platform. Hence it cannot live without the associated intelligence in the network. This is shown in Figure 2. As it is a form of communications logic on the terminal, the soft terminal will naturally be involved in service scenarios and communication protocols with the network. It will also participate in negotiation mechanisms on how the communication functions are to be distributed between the network and terminal. At the network side, the repository of application and

### Soft Terminal Concept and Functional Architecture

The soft terminal is essentially the software equivalent of a network terminal. It consists of a number of software components, that:

- are controlled by the network operator or service provider (see “control by service provider” requirement);
- can be installed and managed remotely by the service provider (see “user-friendly and cost-effective deployment” requirement);
- can adapt to the terminal characteristics (see “portability” requirement);

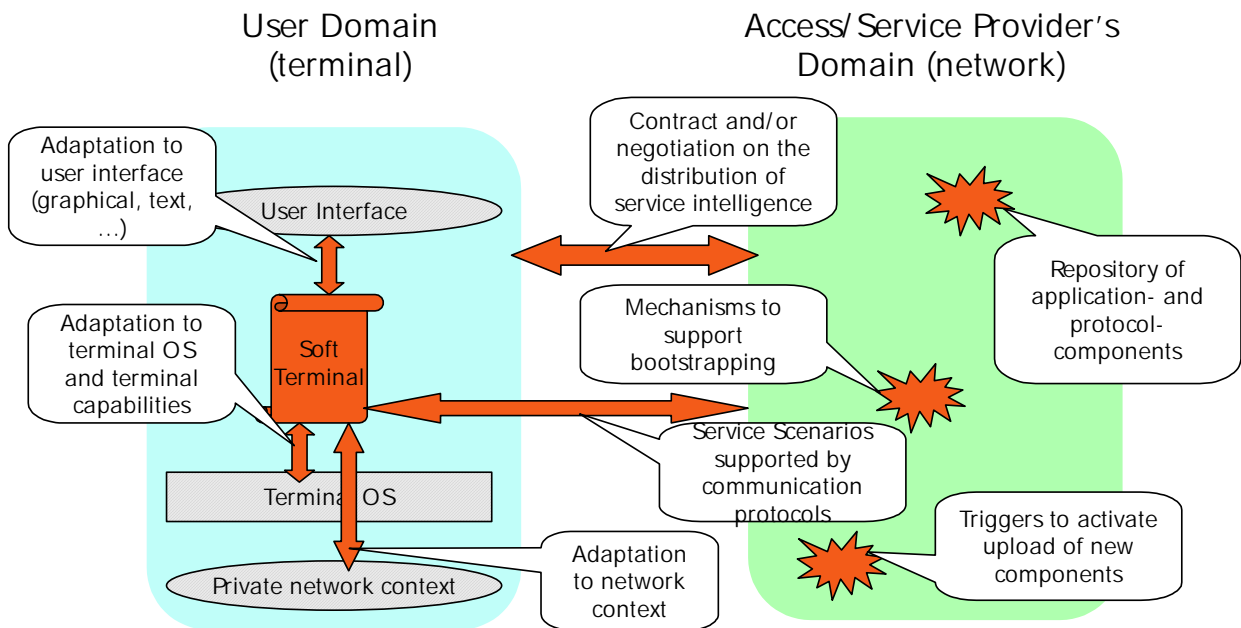


Figure 2 – The soft terminal in its environment

protocol components is the source from which all soft terminal logic can be downloaded. In addition, network elements, such as a Network Access Server, will typically provide the necessary triggers to decide when protocol or application components have to be installed or upgraded, and will also play a crucial role in the initial installation of the soft terminal's core components.

### Enabling Technologies

Java defines an attractive environment for building portable and distributed networked applications. It offers portability by defining an abstract Java Virtual Machine (JVM), which is then mapped on top of existing real architectures. In this way, applications can be written in a platform-independent way (targeting the virtual machine), while platform-dependent implementations of the virtual machine take care of the portability issues. As a result, the choice of Java as the basic environment for the soft terminal solves most of the portability problems. Nevertheless, it should be noted that some specific functions of the soft terminal, such

as the dynamic management of network protocol components, are typically not covered by the JVM. For these aspects, the portability issue has to be tackled explicitly by intelligence in the soft terminal itself.

Given the choice of Java technology, we have introduced the Java Dynamic Management Kit (JDMK) and Java Management Extensions (JMX) technology as the basic building blocks for remote management of the software components of the soft terminal. By introducing JDMK/JMX in the soft terminal, it becomes possible to monitor and control its components from a network management platform (typically in the service provider domain).

In addition, other Java-based technologies can be introduced to enhance specific features, such as the Java Embedded Server (JES) and Java Server Pages (JSP). However, an exhaustive overview of all the enabling technologies is beyond the scope of this article.

### First Prototype

Starting from the concept and requirements of the soft terminal, and building on current state-of-the-art software technologies, a first prototype of the soft terminal has been developed by the Alcatel Corporate Research Center. While the concept is applicable to various domains of the telecom world, this prototype concentrates on function-

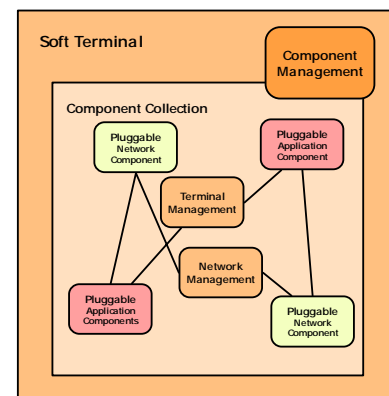


Figure 3 – Software architecture of the soft terminal

ality used in the context of remote access services, such as access to the Internet and to corporate intranets. The following paragraphs highlight a number of specific functions and mechanisms that are closely related to the requirements discussed previously.

### Software Architecture

Earlier we discussed a high-level functional architecture for the soft terminal, focusing on its role in telecom services. From a purely software perspective, the soft terminal is mainly a component manager, controlling dynamic components both at the application level and at the level of the underlying communication protocols. The core components basically define a component manager that will take care of the insertion of new components, versioning and updating of components, as well as managing the relationships between different components. From this perspective, the component manager will not care whether a given component represents an application or some network layer protocol. It is only aware of relationships between components, such as application X should run on a protocol stack defined by components Y and Z. This is represented in Figure 3.

### Solving the Installation Bootstrapping Issue

While technologies such as JDMK and JMX offer interesting features

for dynamic installation, upgrading and management of new components, they rely on a set of core components which must be available on the terminal from the start. Thus, they do not offer a solution for the bootstrapping issue.

To understand the solution implemented in the prototype, consider the network context as presented in Figure 4. The user terminal (e.g. a PC) is connected via the access network – modem connected to Public Switched Telephone Network (PSTN), Asymmetric Digital Subscriber Line (ADSL) access network, etc – to the access provider's POP. The POP is the first point in the public network where the IP layer is terminated, and where value-added services can be offered. It consists typically of a NAS combined with a number of servers, for authentication/authorization, portal-site services and applications, etc. One of these servers will be responsible for remote management of the soft terminal applications on user terminals, and will contain a repository of new service components that can be downloaded dynamically when the user subscribes to or accesses new services. Note that Figure 4 should be seen as a logical representation; the distribution of the different POP functions over multiple servers is not relevant here.

The first time the user connects to the POP, he or she can do so re-

lying on existing PC software (e.g. a classical dial-up application and web browser), and on the service scenarios and protocols used by most access providers. Typically the service scenario will connect the user to a portal site in the access provider's domain, where he or she can register as a new user. When the server detects that the user has not yet installed a soft terminal (e.g. using a cookie stored on the terminal), the user is guided to the server responsible for soft terminal installation. After obtaining the user's permission, the core components of the soft terminal are then automatically installed on the PC. The prototype uses an explicit installation wizard, but the mechanism can probably be improved by applying recently introduced features of Java applets. Once the soft terminal is installed on the user terminal, it will be the preferred way to connect to the network and to access the various services. In other words, from that moment on, the soft terminal becomes an essential component of a distributed service platform, with service logic both at the terminal and at the network side.

### Framework for Developing Different Types of Components

From a telecom point of view, the soft terminal functions as a plug-in point for both service logic components and protocol stacks. From a software viewpoint, a different component classification can be made.

On the one hand, one needs components that can be plugged in and wait for an external trigger. An example is a Virtual Private Network (VPN) login service. When activated, it offers a means of querying the user's VPN name, login name and password (e.g. through a pop-up dialog box) and then uses this information to set up a session and call and perform authentication using other plugged in components, such as a Remote Access Service (RAS) protocol stack and a security service.

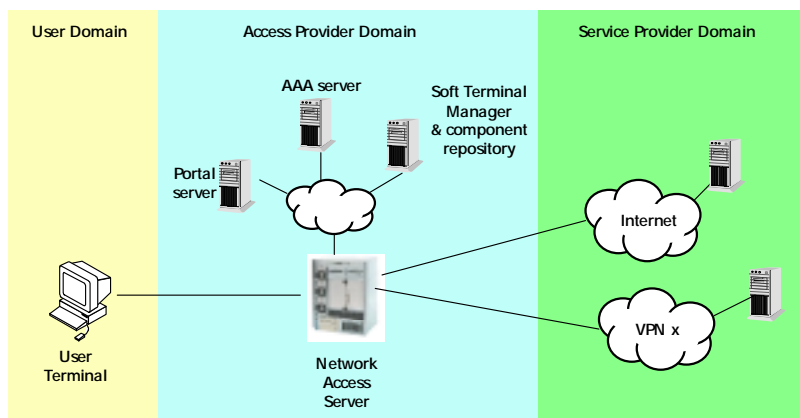


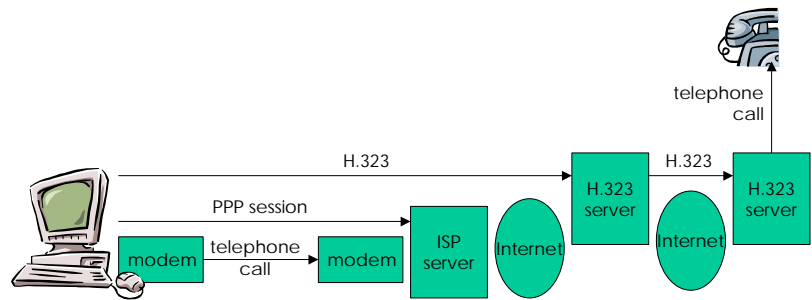
Figure 4 – Network context for the bootstrapping problem

On the other hand, one also needs components that can be plugged in and immediately activated. A messaging service component, for example, installs itself and automatically starts listening on a certain Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) port for incoming messages. It is clear that this component must run separately from other installed (and running) components. In software terms, this requires that the component should be instantiated within its own thread and start running.

The soft terminal offers a framework for the rapid development and deployment of both these types of component. In practice, this means that the soft terminal offers Java classes and Java interfaces which can be used as a basis (through classical Java mechanisms of class inheritance or interfaces) to develop new protocol components (i.e. those that are activated immediately after installation), and service logic components (i.e. those that wait for external triggers). These base classes and interfaces take care of all the common logic, enabling the developer to concentrate on the specific functions of the component itself, rather than on its interactions with other soft terminal components.

**Network Side Functionality**

The JMX framework enables a manager (e.g. service provider) to manage the components in the soft terminal and be notified about events at the terminal side. If a HyperText Transfer Protocol (HTTP) adapter is installed at the terminal side, the service provider can manage the soft terminal using a simple browser. For automated management, a Remote Method Invocation (RMI) adapter can allow the provider to achieve full control over the soft terminal from inside Java. In our prototype, a simplified interface is provided which makes it possible to install a new service onto the soft terminal and to fire remote methods on installed components.



**Figure 5 – Different "calls" in a single service**

Consider the following scenario as an example. A user wants to set up a service session with another user who does not have the necessary service components installed on his or her terminal. This is detected from inside the network, and a request can be pushed to the user, asking if the user is interested in setting up this service session with the inviting user. The user invitation is set up by calling the "invite" method on an already installed Invite Service. If the called user agrees, a server side element communicates with the soft terminal and instructs it to install the required services.

Normally, however, the user will trigger the service subscription. We have integrated the prototype of the soft terminal within a broader service platform demonstrator. In this demonstrator, the provider maintains a list of services to which a user can subscribe and a list of subscribed services. Using Java servlets and JSP technology, the user can subscribe to new services using a plain web-based interface. The servlets then communicate with the soft terminal and instruct it to install the new components.

**Visions for the Future**

The concept of the soft terminal was born in the RAS domain, more particularly from the specific issues related to client-side applications for a broadband RAS (different variants of the protocol stack, deployment costs for the access provider, etc). However, it is a

powerful concept that can undoubtedly be of great value in a much wider context.

Consider the following example of a Voice over IP (VoIP) call. There are PC applications that allow VoIP calls to be made to any telephone in the world. When one clicks the application to dial a number, the application dials in to the local ISP node. This results in a telephone call and a Point-to-Point Protocol (PPP) session. The application then connects over the Internet to the VoIP telephony provider, and sets up an H.323 (or Session Initiation Protocol) session. From there the "call" is transported over the Internet to the node nearest to the destination phone. Finally, the call is completed over the telephone network (another telephone call). At least four sessions are involved in this scenario, as illustrated in Figure 5. However, one can consider this to be only a single service: making a point-to-point voice call.

The different sessions involved in this scenario are terminated at different points in the network, but they all meet in the terminal. Consequently, an access provider or service provider, who typically controls either the modem (terminating the phone call), or the network access server (terminating the PPP session), or one of the other network elements involved, can only have a partial view of the user's session context. To obtain the complete picture, a provider needs a way of monitoring what is happening at the terminal side. The soft terminal concept provides a solution to this issue. Once installed on

the user terminal, it becomes the endpoint of all communications with the network, and can be used to trigger various advanced features in a similar way to current intelligent networks based on triggers from the call model in the local exchange.

The idea can be taken further: by controlling the terminal communication logic, the provider achieves complete control over the basic call model on which it can build value-added services. In other words, the provider is relieved of the restrictions of traditional call models, such as the one used for telephony. Thus the soft terminal opens huge perspectives for concepts such as intelligent networks, which today rely on the call model available in the Service Switching Point (SSP); the soft terminal becomes part of a distributed Service Switching Function (SSF).

## Conclusions

The soft terminal is a promising and powerful concept that can meet the varied requirements associated with the ever more rapidly evolving world of telecom networks and services. Multiple technologies are emerging that will enable this concept to be realized in the near future. A first prototype developed in the Alcatel Corporate Research Center has demonstrated the feasibility of the basic concept and features, but further research is required to investigate how new technologies can be applied to extend the concept to different domains in the world of telecommunications.

## Bibliography

- 1 D. Chantrain: "Solutions for Service Deployment in the Edge of the Network", to be presented at IEEE Symposium on Computers and Communication ISCC'2000, Paper 62, 4-6 July 2000, Antibes, France (<http://www.rennes.enst-bretagne.fr/~afifi/iscc/2000.html>).
- 2 Java Technology Homepage: <http://java.sun.com>.
- 3 Java Dynamic Management Kit (JDMK): <http://www.sun.com/software/java-dynamic/index.html>.
- 4 Java Management Extensions (JMX): <http://web2.java.sun.com/products/JavaManagement/>.
- 5 Java Embedded Server (JES): <http://www.sun.com/software/embeddedserver/index.html>.

**Dominique Chantrain** is working on the introduction of new service concepts and scenarios in the network edge as part of the Software and Services Strategic Program at the Alcatel Corporate Research Center in Antwerp, Belgium.

**Koen Handekyn** is working on the software technologies needed to realize new service concepts, such as the soft terminal, as part of the Software and Services Strategic Program at the Alcatel Corporate Research Center in Antwerp, Belgium.

**Hans Vanderstraten** is Project Manager of the Project on Services for IP and 3G Mobility which is part of the Software and Services Strategic Program at the Alcatel Corporate Research Center in Antwerp, Belgium.