Terminal-Centric View of Software Reconfigurable System Architecture and Enabling Components and Technologies

Nikos Georganopoulos, Tim Farnham, and Rollo Burgess, Toshiba Research Europe Ltd.
Thorsten Schöler and Juergen Sessler, Siemens Mobile AG; Paul Warr, University of Bristol
Zoran Golubicic, TTI; Fanny Platbrood, CSEM; Bertrand Souville, DoCoMo Euro-Labs
Soodesh Buljore, Motorola Labs

This work has been performed in the framework of the IST project IST-2001-34091 SCOUT, which is partly funded by the European Union. The authors would like to acknowledge the contributions of their colleagues.

ABSTRACT

Reconfigurable radio in Europe is rapidly gaining momentum and becoming a key enabler for realizing the vision of being optimally connected anywhere, anytime. At the center of this exciting technology is the reconfigurable terminal that will move across different radio access networks, adapting at every instant to an optimum mode of operation. This will require coordinated reconfiguration management support from both the terminal and the network, but the terminal will inherit a significant part of this intelligence. This article focuses on a novel reconfigurable terminal architecture that advances the state of the art and encompasses the overall protocol stack from the physical to application layer in IP-based radio access networks. The proposed architecture is composed of a terminal reconfiguration management part and enabling middleware technologies like the complementary Distributed Processing Environment and agent platforms, flexible protocol stacks that can flexibly be interchanged to support different wireless technologies and associated mechanisms, and finally, object-oriented reconfigurable RF and baseband components. The work presented in this article is conducted in the context of the IST projects SCOUT (www.ist-scout.org) and TRUST (www4.in.tum.de/~scout/trust_webpage_src/ trust_frameset.html) of the European 5th Framework Program.

INTRODUCTION

In the field of public mobile communications, there is currently a plethora of wireless access technologies with different standards, covering different geographic locations and providing different services to the users. Second-generation (2G) cellular systems provide voice and text messaging; 3G systems provide multimedia services; wireless local area networks (WLANs) provide localized Internet connectivity and terrestrial and satellite broadcasting for entertainment and news programs, to mention a few. Distinct mobile devices are currently used to access these various offered services; furthermore, multimode terminals can support different modes of operation by switching between them. One of the main goals for the future of telecommunication systems is service ubiquity (i.e., the ability for the user to transparently access any service, anywhere, anytime); multimode terminals today can be thought of as the first step in that direction. However, as the number of wireless standards and services increase, so does the design complexity of the terminals. Such complexity can be subdued with the design of a software reconfigurable terminal, with adaptive and modular components that can be switched, replaced, or configured accordingly to support single- or multistandard simultaneous operation. This would further benefit manufacturers in simplifying their equipment manufacturing process and operators in being able to cover more users with more services. Software-defined radio (SDR) [1] was known in its early stages as an enabling technology aimed at controlling by software the radio parts of wireless devices. Techniques like multiband antennas and radio frequency (RF) conversion, wideband converters, and baseband processing functions implemented by general-purpose programmable processors can be used to reconfigure a device to use different radio technologies and attach to different access networks.

0163-6804/04/$20.00 © 2004 IEEE

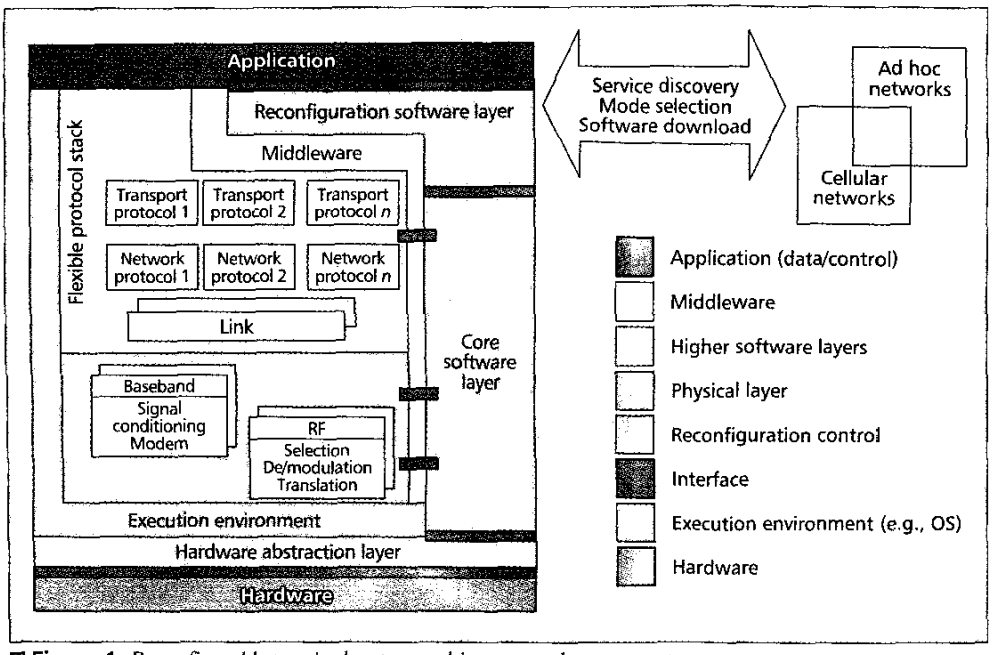


Figure 1. *Reconfigurable terminal system architecture and components.*

Extensive research in various aspects associated with multi-mode service provision and software reconfiguration was, is and will be conducted within initiatives across Europe.

Within the European research community, reconfigurable radio extends this flexibility to the higher layers, encompassing the overall protocol stacks, supporting middleware platforms, and associated services and applications. Extensive research in various aspects associated with multimode service provision and software reconfiguration was, is, and will be conducted within initiatives across Europe. In the 4th Framework Program (ESPRIT), relevant projects included M3A and SLATS, and in the 5th Framework Program (IST) CAST, MOBIVAS, WINDFLEX, WINE, PASTORAL, DRIVE, MuMoR, and others, including of course SCOUT and TRUST, are related. MVCE, a U.K.-based research program is also involved in SDR related research, which also feeds into the World Wide Research Forum (WWRF) Working Group 6 (WG6) activities dealing with reconfigurability. The general aim of all these research initiatives is to define the terminal and network supporting architecture, services and applications, mechanisms and enabling technologies that can realize such a software-defined system. The ultimate goal is making service ubiquity a reality for the wireless mobile user of the future.

Although the research area is very wide and even the extensions to the network infrastructure to support such operation are equally challenging and important, the focus of this article is on the terminal. This is undoubtedly the most central and complicated part of the whole system, and a general system architecture that incorporates different terminal components is described herein. We describe the proposed overall terminal system architecture and the general interactions between the various components. Further on, the different components are analyzed with a state-of-the-art review of available design choices followed by design recommendations and achieved or expected results. Finally, we summarize conclusions and ongoing work on the subject, and highlight future objectives.

Terminal System Architecture

Various efforts have been made to define parts of the software reconfigurable architecture. CAST proposes a three-layer (management, procedural, physical) reconfigurable architecture to provide the interface between the application and the underlying physical layer of the terminal processing platform. In SLATS, a software architecture and software library elements are designed and developed, implementing the baseband functionality of the Global System for Mobile Communications (GSM) and wideband code-division multiple access (W-CDMA) for real-time implementation on a selected digital signaling processing (DSP) platform. In WINDFLEX a reconfigurable baseband architecture is defined for an indoor high-bit-rate adaptive modem based on orthogonal frequency-division multiplexing (OFDM). The MuMoR project aims to investigate the RF front-end as well as baseband in order to find a common reconfigurable architecture that is flexible to adapt to different standards. MVCE proposes the Reconfiguration Management Architecture (RMA) [2], which consists of a configurable mobile communication network based on a distributed management structure that interacts with reconfigurable terminals. The RMA aims to support the reconfiguration process for both terminal and network. It is thus evident that efforts have been made to define parts of the terminal architecture focusing on the RF, baseband, and reconfiguration management.

However, within the SCOUT and TRUST the effort is to define a complete overall system architecture, that includes all the compo-

Due to the growing complexity of modern embedded systems, a paradigm shift to object-oriented programming can be observed, applying OOP principles for implementing protocol stacks based on class libraries tailored specially for this purpose.

nents of the software-defined reconfigurable terminal. This is shown in Fig. 1, which depicts a layered image of the system. The application layer includes software for both user and control data. The terminal protocol stack includes software instances of high-layer protocols of the IP stack (transport, network, and link) that can flexibly be interchanged to support different wireless technologies and associated mechanisms. The reconfiguration control part includes the various software modules responsible for making the (best) reconfiguration decision and then executing the reconfiguration process. This is achieved through specialized interfaces between the reconfiguration management software and the relevant terminal component. In the physical layer, RF and baseband hardware use software techniques to achieve various outputs and parameters. The middleware part provides the platform with two different tasks, one (horizontal) for external remote interaction with the network and/or other terminals for service discovery, mode selection, and software component download, and an internal one (vertical) for localized interactions, through the interfaces, between core software layer and protocol stack, RF, and baseband. The execution environment forms the unifying basis for all the various parts and operations in the terminal.

Terminal Components and Enabling Technologies

The software reconfigurable terminal thus consists of five main building blocks that are individually analyzed next, always considering their coexistence and interactions:

Flexible Protocol Stacks

Traditionally, network protocol stack software implementations are highly optimized and more or less monolithic parts of system software that cannot be dynamically reconfigured. Recent expansion in the number and complexity of radio network standards in combination with the shortage of resources in mobile terminals have led to the need to reconfigure protocol stack software within SDR terminals in order to efficiently support the multitude of standards. The following gives definitions of common terms, relevant software technologies, and a short overview of reconfigurable protocol stack architectures that may be used as a basis for reconfigurable SDR mobile terminal protocol stacks. First, reconfiguration can be addressed from two separate approaches:

- Customizable protocols are built from generic protocol modules that implement common functions, and a second part implementing customized specific extensions (i.e., object-oriented programming [OOP] inheritance) to the common protocol stack parts. A specific protocol stack can now be built by instantiating or compiling elements from these basic building blocks using a protocol stack configuration description.
- The composable (or component-based) protocol stack approach enables configuration of a protocol stack during boot or runtime from a given set of core functionalities that are supplied by a protocol stack framework and a protocol configuration description (each instance can have a different set of parameters). In this approach no customization can be performed by software extension, but different standards can be supported by adding components and changing configuration parameters.

Different design methods, component implementations, and computational models can be used to create reconfigurable protocol stacks. These are summarized below.

Design methods: Until recently, system software modules found in embedded systems were implemented following a functional programming paradigm. The functional design was preferred because of resource constraints and hard real-time requirements.

Due to the growing complexity of modern embedded systems such as mobile terminals, a paradigm shift to OOP can be observed, applying OOP principles (inheritance, generics, encapsulation, etc.) for implementing protocol stacks based on class libraries tailored specially for this purpose. One approach to reducing resource demands while supporting highly complex software is a more dynamic (runtime changeable) implementation using active interface classes and the class loader principle used by Java virtual machines.

Component implementation: In addition to dynamic linkage support, modern operating systems and middleware support a component model. However, operating systems more oriented to mobile terminals, such as Symbian or PalmOS, lack a native component model. Therefore, a component model supporting cross-platform exchange of software modules as well as providing implementation language independence (protocol stacks can be implemented in C, C++, Java, etc.) is attractive to support dynamic reconfiguration of protocol stack at runtime across different platforms.

Java 2 Mobile Edition (J2ME) enables low-footprint Java virtual machine implementations to allow operation on mobile terminals and includes a rudimentary component model in some configurations. Microsoft's .NET architecture with the Common Language Runtime (CLR) approach provides an execution environment capable of dynamic linking and supports many programming languages (Visual Basic.NET, Visual C++.NET, C#, etc.) but is limited to devices running Windows operating systems.

Computational model: Many protocols are asynchronous in nature, which means that the messages passed between protocol elements (or components) are buffered prior to protocol processing. These types of protocol suit a multithreaded computation model in which each layer is a separate thread and asynchronous message passing can be used to communicate between layers. In this case each thread instance (protocol processing element) can be reconfigured separately. On the other hand, synchronous protocols require protocol processing to be performed for each message, and a thread-per-message model is more suitable.

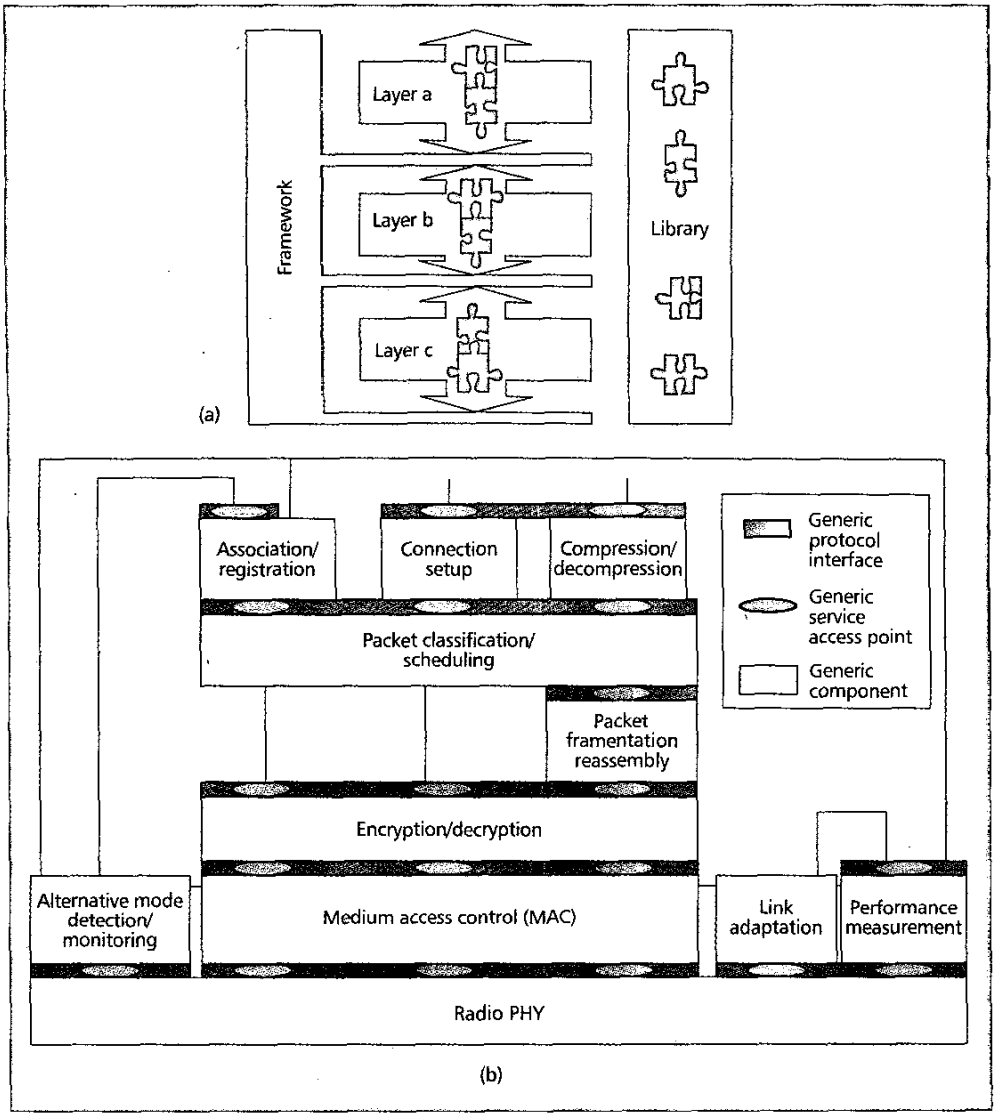


Figure 2. *a) Composable protocol stack framework; b) an example of generic stack implementation.*

A component model supporting cross-platform exchange of software modules as well as providing implementation language independence is attractive, to support dynamic reconfiguration of protocol stack at run-time across different platforms.

Frameworks — Existing frameworks are programming-language-specific and also rely on a thread-per-message computation model with configurable active programming interfaces or virtual protocol layers to provide the necessary dynamic routing of messages between protocol layers (or components). A summary and brief comparison of frameworks is as follows [3]:

- X-Kernel — C-based composable (at compile time) framework with configurable virtual protocol layers for message routing, allowing limited reconfigurability
- OPtIMA — Java-based composable and (runtime) customizable framework with configurable active programming interfaces
- DIMMA — C++-based customizable framework derived from the X-kernel framework and specially tailored for multimedia applications

An alternative method has been explored within the IST WINE project. It uses an adaptation layer allowing composition of different performance enhancements within a single protocol layer. However, this approach has limited flexibility when applied to dynamic protocol stack reconfiguration in general.

A proposed framework combines many of the benefits of the approaches mentioned above and supports one or more threads per layer in its computational model. This permits the use of multiple languages, multiple execution environments, and multiple protocol stack layer instances by virtue of this computational model. The main benefits of the proposed framework are support for runtime reconfiguration of active protocol stacks (using persistent asynchronous message queues). The assessment and comparison of the framework [4] indicate that its performance is superior to existing frameworks when asynchronous protocols are implemented in mixed language environments. Slightly lower performance can be expected when the proposed architecture is used for synchronous protocol implementations. This handicap can be over-

In essence, a reconfigurable baseband must be able to dynamically map complex signal-processing algorithms to a set of heterogeneous processors while guaranteeing to meet hard deadlines for both new and resident baseband applications. This is no easy task.

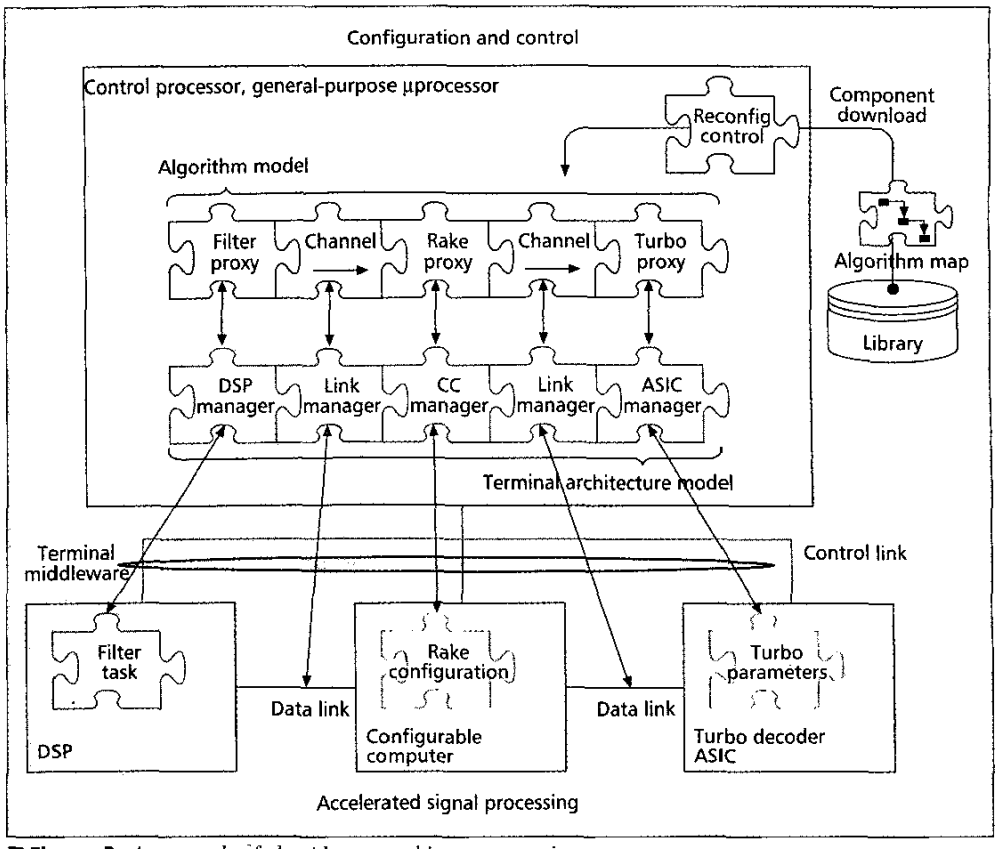


Figure 3. *An example of algorithm-to-architecture mapping.*

come by using thread-per-message implementations inside a layer instance. Using this framework, a generic protocol stack can be supported in which generic components can be bound and reconfigured to support different radio access technology modes (Fig. 2).

Given the new openness and complexity of reconfigurable protocol stack implementations combined with strong regulation requirements imposed on mobile communication equipment, new ways for software security and reliability must be found. Nowadays, software security primarily means checking integrity by verifying software on a component basis. Simulation of to-be-executed software in a virtual environment will help identify component collaboration problems in general as well as prevent malicious behavior of so-called rogue terminals. Validation of software installations on virtual execution environments (virtual prototypes) can be carried out in a multiple-stage process distributed on network-side systems and/or trusted end-user terminals.

BASEBAND

The baseband subsystem of a wireless transceiver comprises a number of signal processing functions such as filtering, channel equalization, and error correction. Many of these are compute-intensive and constrained by hard real-time deadlines. In fact, traditional implementations trade flexibility for performance, relying on a combination of hand-engineered code for DSPs and application-specific integrated circuits (ASICs). Programmable DSPs have long held the promise of extending application-level flexibility into the radio physical layer. Unfortunately, predictions show that the processing demands made by future generations of radio access technology will continue to exceed the capability of those DSPs with acceptable power consumption levels for handheld devices [5]. This leads to the conclusion that future hardware architectures will contain multiple processors, and these will be heterogeneous in nature. DSPs will of course play a part, and even ASICs will continue to be necessary where high performance and low power consumption are paramount. Moreover, a new breed of configurable computing devices that aim to combine the programmability of DSPs with the performance of ASICs will add to the diversity of processor combinations. The example system of Fig. 3 contains just such a mixture of processor types.

In essence, then, a reconfigurable baseband must be able to dynamically map complex signal processing algorithms to a set of heterogeneous processors while guaranteeing to meet hard deadlines for both new and resident baseband applications. This is no easy task. Fortunately, modern software technology provides the means to tackle this complexity. The principal tool in the software armory is the separation of concerns using well defined interfaces. As described later, the terminal software architecture provides standard interfaces (RTOS and HAL) to the

	Transmitter	Receiver
Selection	Isolate the output band at the transmit frequency and remove spurious generated in the D/A and translation stages.	Filter out unsupported operating bands and lower the dynamic range within the input band.
Amplification	Generate a signal power of the order of +30 dBm for transfer to the air interface.	Overcome the loss (and therefore noise) inherent in analog signal processing and supply appropriate signal level to the A/D process
Frequency translation	May be used to lower the absolute bandwidth and/or jitter demands on the D/A.	Translate the desired band to a frequency that can be sampled with sufficient dynamic range.

Table 1. *Analog signal functions within a practicable SDR.*

To ease the reuse and distribution of both model and accelerator processes, the use of software component technology is proposed. Software components are considered runtime equivalents of software objects.

operating environment, and for interprocess communication through terminal middleware. This principle is extended into the baseband domain by separating the configuration, runtime control, and signal processing functions. This provides the flexibility to target each function to the most appropriate processor. In the example configuration and control is best suited to the general-purpose processor at the top of the figure, while the "number-crunching" required by the modem is best suited to the accelerators arrayed along the bottom.

Another commonly used software technique is the simplification of problems through abstractions known as models. The function-to-architecture mapping problem can be dynamically solved by evaluating the interaction between models of potential baseband algorithms, and a single model of the fixed terminal hardware architecture [6]. Figure 3 illustrates this model-based approach to the configuration and control software for a simplified Universal Mobile Telecommunications System (UMTS)-style receiver. In particular, the terminal architecture model captures the essential features of the hardware resources, such as processor utilization, memory allocation, and power consumption budget. The algorithm model is constructed from proxy processes that represent the implementation of signal processing functions using one of the available accelerators. This model is used to capture the algorithm's required real-time deadlines, while the individual proxies store the actual runtime performance of their accelerator-specific implementations. Together these modeling abstractions enable the mapping of algorithm to hardware to be evaluated and optimized without disturbing the live signal processing. Indeed, optimization using the models can be performed anywhere, perhaps on a network machine. What's more, validated models can also control the distributed runtime behavior, according to process schedules and a set of execution laws together known as a concurrent model of computation. For signal processing the appropriate model is the well-known and intuitive asynchronous data flow model [7].

Finally, to ease the reuse and distribution of both model and accelerator processes, the use of software component technology is proposed. Software components (shown in the figure by a jigsaw icon) are considered runtime equivalents of software objects. In the traditional approach a developer builds a software application by hand from a collection of software objects. In contrast, we propose to automate this process by manufacturing baseband applications from a collection of software components. The developer works to a specification, usually a requirements document. Likewise, a baseband application must be manufactured according to a machine-readable algorithm specification. This specification, known as an algorithm map, is a meta-software component; it describes the data flow of the algorithm model in terms of prototype components and the connections between them, made using abstract data channels. The software component becomes the basic reusable unit of a configurable baseband system. Each type of component is implemented for a number of target processor types and therefore has widespread applicability. In the figure several types of components are shown; an algorithm map is retrieved from a library and used to build the algorithm model. The algorithm model proxy components interact with the architecture model components to form an implementation. Once a valid combination of models has been found the accelerators are initialized using appropriate executable, configuration, or parameterization components as necessary.

RF

The development of the RF aspects of an SDR should cover two areas: flexible linear RF and digital intermediate frequency (IF). The monatomic increase in the capability of signal processing platforms enables the realization of complete IF stages in the digital domain. Direct digital synthesizer (DDS) and field-programmable gate array (FPGA) circuits with clock frequencies over 2 GHz enable the design of flexible numerically controlled local oscillators and IF mixers. As a consequence, IF frequencies below 60 MHz can be processed digitally with sufficient dynamic range (bit resolution) for SDR. For example, complete air interface bands of GSM and UMTS standards can be accommodated in digital IF. Desired channel extraction and demodulation can be performed in a flexible manner for the receiver function and modulated signal construction; fine frequency tuning and signal filtering can be performed for the transmitter function. Thus, the requirement placed on

To achieve the goal of terminal reconfiguration, all involved interfaces and software layers must provide methods and mechanisms to make software modules, components, or parameters interchangeable during the process of reconfiguration.

the analog part of an SDR transceiver may be summarized according to Table 1.

These functions must be transparent (i.e., broadband) so that any standard can be supported at any frequency; low noise, to maintain the required SINAD at demodulation in the receiver; and linear, to maintain low adjacent channel power and out-of-band radiation and interference accommodation.

The power amplifier within an SDR must be able to offer appropriate gain and linearity at a programmable center frequency and band. Furthermore, the radiation from the SDR must be controlled as the system migrates from one standard to another. The more adventurous vision of an SDR includes the ability to simultaneously support diverse air interface standards, so the power amplifier must operate in an inclusive multiband mode. One approach to such an amplifying device is that of a reconfigurable wideband balanced amplifier. The two branches can operate in synchronism in single-standard mode or independently in dual-standard mode. In this system programmable linearization circuits with adaptive linearization algorithms are applied for linearity improvement and power consumption optimization. Several linearization schemes are considered in the design: feed forward, feedback, envelope elimination, and digital predistortion. It is envisaged that a subset of the available linearization applications will be chosen according to the demands of the current operating standard.

A form of flexible high-frequency filtering is vital in the SDR. Electronically tunable filters offer superior tuning speeds and smaller sizes than magnetically or mechanically tuned filters, and are thus desirable in mobile applications. Until recently, the tuning of this class of filter was done using active devices; however, this led to poor linearity. Micro-electromechanical structure (MEMS) switches and bi-stable capacitors offer very low distortion and represent the critical technology evolution for the tuning elements for a flexible microstrip filter [8]. Work is being carried out within the SCOUT project to develop an appropriate flexible filter architecture for this role [9].

Independent and simultaneous control of filter center frequency and bandwidth is being achieved through controlling the length of microstrip or stripline resonant elements alongside the coupling between them. The element resonant frequency is altered by activating periodic, grounded MEMS switches. Variable coupling is achieved through the use of periodic, bi-stable MEMS capacitors connecting adjacent resonators along their length. The discrete nature of the switching arrangement limits the resolution at higher frequencies; solving this problem by new filter architectures is key to this solution. Conversely, the physical size of the resonators constrain the low frequency limit. By utilizing a flexible transmission-line arrangement, where series or shunt connections are possible, the operating bandwidth is maximized whilst maintaining switching resolution. Tunable filters find application in several locations across a flexible transceiver. Within a receiver chain, tunable filters may be placed at the antenna port for desired operating standard selection, and at analog IF to control the input band for the variable digitization function. Within a transmitter chain, such filters may be placed to limit the input band of the power amplifier operating at a flexible center frequency and at the antenna port for out-of-band radiation control.

Where a receiver chain is subject to interfering signals, a mixer (or other frequency translation element) generates distortion products that can add in-band interference to the wanted channel. In traditional single-standard radios the operational frequency is fixed, and the out-of-band unwanted channels (blockers) can be rejected by fixed-frequency filtering. An SDR is more likely to be subjected to a number of interfering signals as, if employed at all, a flexible band selection filter will offer less out-of-band suppression than its fixed-frequency counterpart due to filter order limitations. Therefore, for the SDR receiver application, a highly linear frequency translation function is required.

The application of standard amplifier linearization schemes to mixers is badgered by noise figure issues that counteract any gain in dynamic range achieved by distortion suppression. Also, such schemes generally operate within a narrow dynamic range: the received signal power of a SDR will vary considerably as signals outside the control of the current operating standard fluctuate within the radio environment. A linearization architecture named frequency retranslation [10] is proposed to overcome these shortcomings. This technique produces an error signal generated from a comparison of the system input and the converted system output to predistort the nonlinear mixer. Practical results show a suppression of distortion products by 33 dB without reducing the signal-to-noise ratio (SNR) of the wanted signal [11].

Terminal Reconfiguration Management

A reconfigurable terminal architecture is faced on one hand with a varied composition of distributed software components, and on the other hand with the challenge of the diversity of terminal reconfiguration interfaces (TRIs). Hence, to achieve the goal of terminal reconfiguration, all involved interfaces and software layers must provide methods and mechanisms to make software modules, components, or parameters interchangeable during the process of reconfiguration. In this architecture, at least four main interfaces (terminal, core, execution environment, and hardware abstraction interfaces) with corresponding software layers (radio reconfiguration, core software, execution environment, and hardware abstraction layers) are identified as key players in the reconfiguration process (Fig. 4).

The reconfiguration layer (with the terminal interface) represents the interface to the "outside world." It should enable "outside actors" (user, service provider network operator, third parties, etc.) and user applications to interact with the mobile device. So, for example, a user may start an application in view of synchronizing his/her device with a network-server, a service provider could push location-based messages about new facilities, or a mode change could be

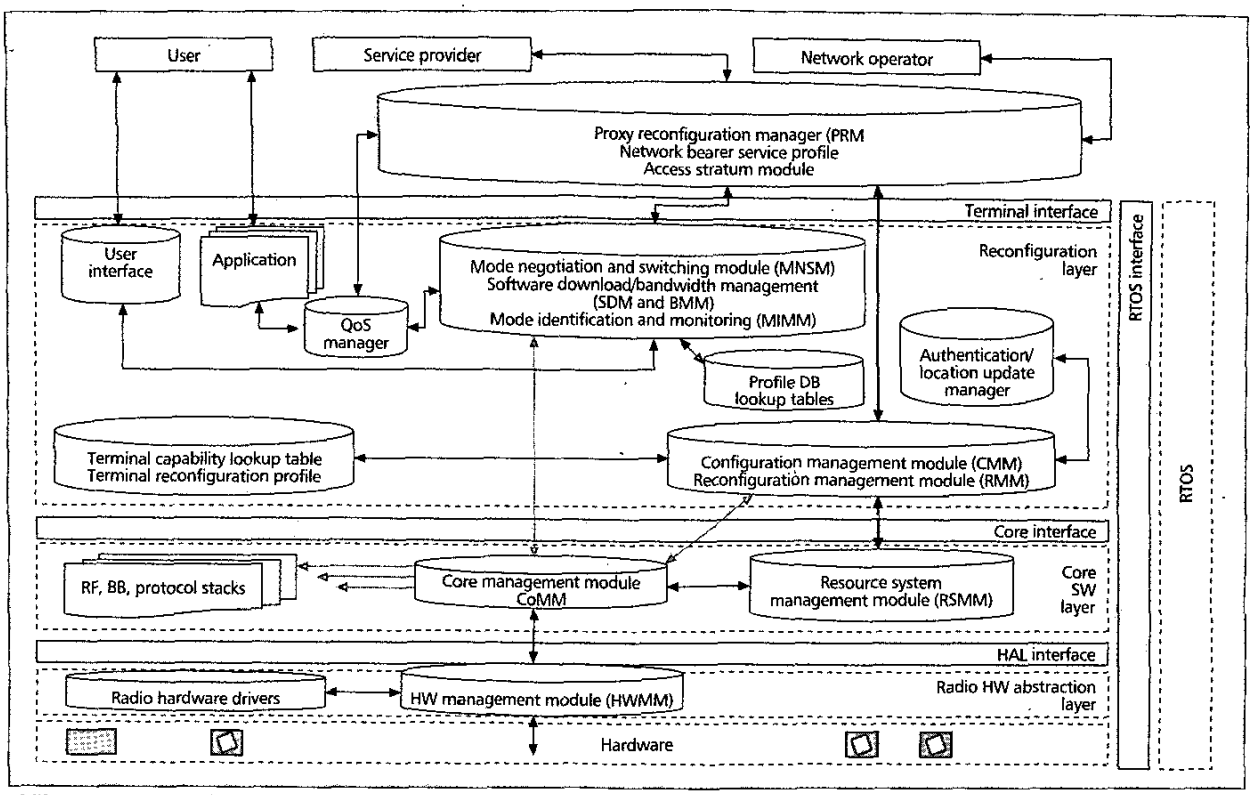


Figure 4. *Terminal reconfiguration management architecture.*

proposed by the network operator to support device capabilities necessary for a streaming job. The reconfiguration layer should support tasks like discovery and mode detection, negotiation management, QoS management, software download and synchronization, and software management for profiles and data.

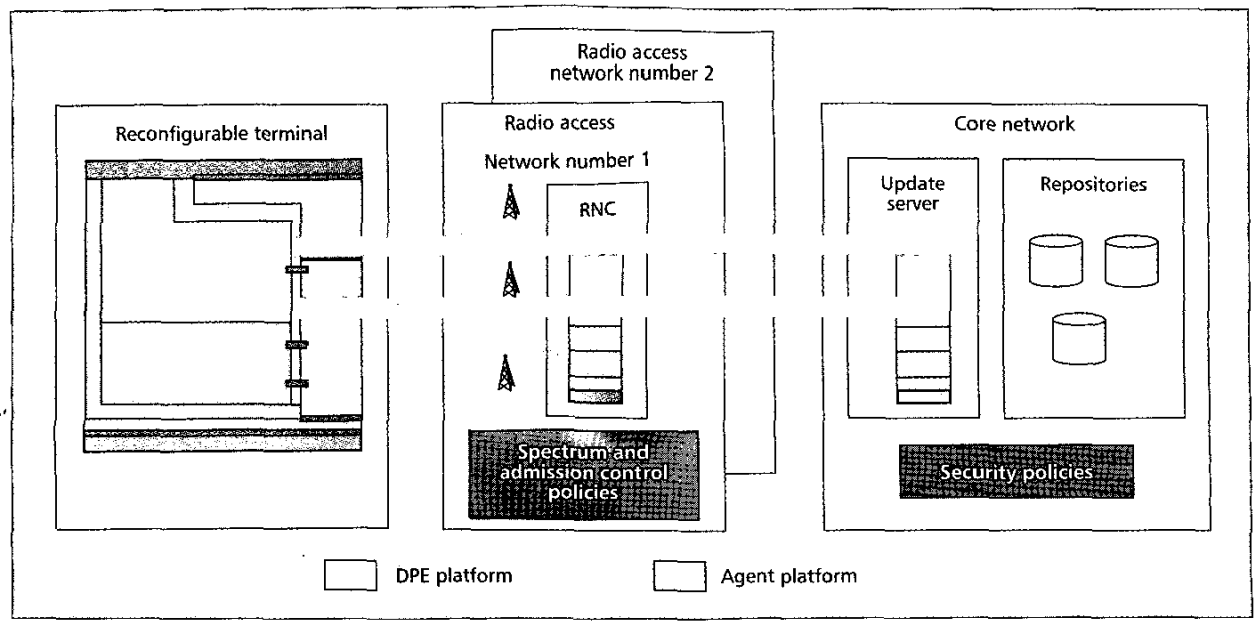
The core software layer is at the interface of the radio configuration and radio hardware abstraction layers, and supports all such tasks as configuration management and system resource management. The configuration management module is responsible for instantiating, monitoring, and controlling the core software layer, which contains the core radio software components for the current radio configuration (e.g., BB, RF, and the protocol stacks). Furthermore, the resource system management module is responsible to provide the radio configuration layer with resource availability. This core software layer is responsible for providing the hardware abstraction layer with the interface profiles that will be used to enable the component drivers.

The execution environment includes operating system and related extensions (e.g., virtual machines for execution of downloaded software modules or applications). Further tasks are support for operating system (OS) services (e.g., communication interfaces, I/O interfaces) and kernel functionalities (memory management, multithreading, task scheduling, etc.), OS controllers, and drivers that translate OS commands into machine code. One important aspect of mode reconfiguration is the system status report. Insufficient availability of capability resources can swing the mode decision. Further support of dynamic exchange of software modules is essential for the whole reconfiguration process to go through without risk of disconnection. Therefore, a software reconfigurable execution environment must meet criteria of multitasking.

The radio hardware abstraction layer includes all the radio software that depends directly on the underlying hardware. It interacts with the core software and OS layers, allowing the OS to make an abstraction of the hardware and run independent of the hardware platform. This abstraction layer is of primary importance in the deployment of the future reconfigurable systems for easing the change and reconfiguration of the hardware independent of the hardware. It can be accessed by software applications via the OS or directly to allow for optimum speed and power performance.

How can terminal reconfiguration take place? At first one actor has to initiate a mode change because of a deficit in the actual device configuration. In most cases, algorithms for mode decision require cooperation between network and terminal components in consideration of available resources. As soon as all essential parameters are analyzed, the mode decision will be made and the reconfiguration process will start. As an example, for this architecture the following steps constitute the reconfiguration process:

1) The MNSM requests of the service discovery and mode detection component available network services and network entities. Further analysis is made about bandwidth requirements,



□ **Figure 5.** *Middleware and policies supporting reconfigurable terminals.*

terminal/network preferences, and aspects of software download and management.

2) After a mode switching decision has finally been made, the MNSM requests the CMM to initiate all steps to be taken for execution of reconfiguration process.

3) The CMM sends a request to RF/BB protocol stacks, asking for all actual and potential protocol stacks to check completion of qualifications for current, newly selected, and possible alternative modes.

4) The CMM sends a request to the RSMM, getting a snapshot of the dynamic terminal resources and the ability to support both reconfiguration and the new selected mode.

5) The CMM now starts and monitors the reconfiguration process, including integration of downloaded software (insert, modify, convert, delete, etc. software modules), updating hardware components by using the HWMM. The functionality of the hardware abstraction layer interface will enable flexible implementation of all necessary communication and reconfiguration processes.

6) As the reconfiguration process is finished, a status message will be sent to the MNSM containing information about the new terminal configuration.

Middleware for Software Reconfigurable Terminals

Middleware is the connectivity software that handles communication between heterogeneous platforms. The inclusion of this technology in the software architecture is primarily to provide an abstraction of the communication between the waveform components and the core control software components. The waveform components interact and control the behavior of heterogeneous hardware resources. They can be instantiated in different ways and replaced at runtime via updates so that the mode of operation and configuration of the terminal can be changed. Indeed, the potential of SDR terminals will be greatly increased if the software updates are performed at runtime, allowing the concept of service ubiquity to become reality: users will be able to communicate and access their services irrespective of the location or access technology employed.

The infrastructure required for dynamic software updates is based on the Distributed Processing Environment (DPE) platform following the generic trends in the field of online software upgrading and evolutionary systems [12]. International bodies such as the Object Management Group work on open standards for online upgrades and software radio architectures. Nevertheless, the integration of these solutions in commercial terminals is rather challenging since commercial terminals are characterized by low capabilities and low power consumption requirements. Therefore, further investigations are needed.

Another task of the DPE platform is to provide transactional properties for the coordination of interactions between the reconfiguration software components and remote components. These remote components can be deployed in the network or in other terminals, and can be supported by different hardware platforms. In such a distributed environment, reconfiguration failures must be addressed. The following nonexhaustive list of reconfiguration failures can be identified:

- Terminals may be disconnected or no longer reachable during the reconfiguration. This is particularly so in ad hoc networks.
- Inappropriate or incompatible updates are installed on the terminal.
- Validation of the new configuration of the terminal fails.

Efficient commitment protocols and algorithms [13] can ensure consistent reconfigurations. Malfunctioning terminals might harm the network, and fallback procedures to the old configuration might be appropriate for recovery procedures.

The agent platform is a valuable complementary approach to the DPE platform for reconfigurability. Reconfiguration decisions rely on a number of parameters collected from the context environment, the current configuration of the terminal, and user expectations. Therefore, specific tasks such as negotiations can be delegated to intelligent software agents that move from the terminal to the network. The benefits of this approach are manifold:

- Software agents support asynchronous communication models and can be federated to form a multi-agent system. This means that agents can communicate, negotiate, and cooperate with other agents while the terminals are disconnected from the network.
- The computation resources and intelligence required for an efficient reconfiguration decision are distributed in the terminal and in the network, relieving the terminal resources.
- Software agents are coordinated by the underlying DPE platform and can adapt their behaviors based on the context and specific policies.

Figure 5 illustrates this approach. In the case of dynamic spectrum sharing context, the terminal might trigger a reconfiguration in order to improve the user-perceived quality of service. Therefore, the software agent moves to different radio resource control entities (e.g., RNC in IMT-2000 networks) in order to negotiate possible reallocation of spectrum. Clearly, spectrum policies set by regulators should be enforced in order to limit and control interference with other systems (e.g., public safety systems). Policies can be seen as constraints on the behavior of the agent. For instance, the agent may decide to select another mode (i.e., radio access technology) because of high amounts of interference. In such a scenario, admission control policies set by network operators should be enforced to control the network load. Eventually, the agent moves to the update server. It performs the required capability negotiations based on the terminal configuration and selected mode so that the update server can push the relevant software components to the terminal. Security policies can also be set by network operators to authorize access to the repositories.

The use of mobile intelligent agents and policies is quite promising in the context of reconfigurability. The signaling overhead needs to be quantified through further study.

Conclusion and Future Work

The idea of software reconfigurable terminals in telecommunications is very attractive, putting the user in the center of the telecommunication model by providing a platform where terminal logic can by switched on the fly and adapt to a specific system and service, depending on parameters such as geographic location, user profile, and network coverage. However, future reconfigurable systems are not only about SDR hardware, but encompass the whole protocol stack and all communication layers. This article presents such a system view of a software-defined reconfigurable terminal as part of ongoing European research activities in the context of reconfigurable software systems. The overall terminal architecture and functional model of operation are described, including the various involved components. These components are further analyzed, and their interactions and enabling technologies described. The terminal is divided in the reconfigurable data part, with the flexible protocol stack, the baseband and RF parts, and the control part with the terminal reconfiguration management layers and specific interfaces. Reconfiguration middleware supports the whole process by enabling interactions inside the terminal and with remote processes in the network. Work on all these parts is ongoing, and technical advances in all areas are still needed in order to design, implement, and test a complete software reconfigurable terminal.

The use of mobile intelligent agents and policies is quite promising in the context of reconfigurability. The signaling overhead needs to be quantified through further study.

References

[1] J. Mitola, *Software Radio Architecture*, Wiley-Interscience, 2000.

[2] K. Moessner and R. Tafazolli, "The RMA — A Framework for Reconfiguration of SDR Equipment," *IECE Trans. Commun.*, vol. E85-B, no 12, Dec. 2000, pp. 2573–80.

[3] G. Clemo *et al.*, "Flexible Protocols and Middleware to Support Terminal Reconfiguration," Proc. 1st SCOUT Workshop, available at www.ist-scout.org, Paris, France, Sept. 2003.

[4] T Farnham and T Schöler, "A Flexible Protocol Stack Framework: Design, Validation and Performance," *Proc. SDR Forum Tech. Conf. '03*, SY3-001, Orlando, FL, Nov. 2003.

[5] H. Blume *et al.*, "Model-based Exploration of the Design Space for Heterogeneous Systems on Chip," *Proc. Wksp. Heterogeneous Reconfig. Sys. on Chip '02*, Hamburg, Germany, Apr. 2002, pp. 29–40.

[6] R. Burgess, "A Software Framework for Model-Driven Configuration and Control of Wireless Baseband Systems," *Proc. SDR Forum Tech. Conf. '03*, SW-1-001, Orlando, FL, Nov. 2003.

[7] E. Lee and D. Messerschmitt, "Synchronous Data Flow," *Proc. IEEE*, vol. 75, no. 9, Sept. 1987, pp. 1235–45.

[8] D. Hyman *et al.*, "GaAs-Compatible Surface-Micromachined RF MEMS Switches," *Electronics Letters*, vol. 35, no. 3, Feb. 199, pp. 224–26.

[9] B. Carey-Smith *et al.*, "A MEMS-ready Wide Tuning Range Planar Resonator with Application to Microwave Flexible Filters," *Proc. APMC '03*, Seoul, Korea, Nov. 2003.

[10] T. Nesimoglu *et al.*, "Linearised Mixer Using Frequency Retranslation," *IEE Elect. Lett.*, vol. 37, no. 25, Dec. 2001, pp. 1473–74.

[11] T. Nesimoglu *et al.*, "Mixer Linearization for Software Defined Radio Applications," *Proc. VTC Fall '02*, Vancouver, Canada, Sept. 2002, pp. 534–38.

[12] M. E. Segal, "Online Software Upgrading: New Research Directions and Practical Considerations," *Proc. COMPSAC '02*, Oxford, UK, Aug. 200, pp. 977–81.

[13] C. Prehofer and B. Souville, "Synchronised Reconfiguration of a Group of Mobile Nodes in Ad-hoc Networks," *Proc. ICT '03*, French Polynesia, Feb. 2003, pp. 400–05.

Biographies

NIKOS GEORGANOPOULOS [M] (n.g@ieee.org) obtained his M.Sc. in telecommunications research in 1997 and his Ph.D. in telecommunications in 2003 both from Kings College, University of London. In January 2000 he joined the Center for Telecommunications Research (CTR), King's College London as a research associate and worked on the IST projects BRAIN and MIND, on architecture and protocols for future IP-based access networks. In October 2002 he joined the Toshiba European Telecommunications Research

Future reconfigurable systems are not only about software defined radio hardware, but encompass the whole protocol stack and all communication layers.

Laboratory (TREL), Bristol, United Kingdom, where he is currently a senior research engineer in the software and protocol research group. His research interests include flexible reconfigurable IP terminal architectures and protocols. He is a member of the IEE.

TIM FARNHAM (tim.farnham@toshiba-trel.com) is chief research fellow for software and protocol research activities within the Toshiba Telecommunications Research Laboratory (TRL) in Bristol. He obtained his B.Eng. degree in electrical and electronic engineering in 1991 from the University of Manchester Institute of Science and Technology (UMIST) and his Ph.D. in mathematics and computer science in 2000 from DeMontfort University Leicester. He began his research career in 1992 with the U.K. Defense Research Agency (now known as QuinetiQ) researching networking and network management aspects of tactical communication systems. In 1995 he joined the Hewlett-Packard research laboratories to perform research in wireless communication systems for the office and home. In 1999 he joined Logica-Telecom, a provider of service platforms for the mobile telecom industry. In 2000 he joined TRL where he is conducting research into reconfigurable terminals and in particular protocol stack reconfiguration, which has included involvement in European collaborative projects TRUST, SCOUT, and E2R.

THORSTEN SCHOELER (schoeler@sra.uni-hannover.de) earned his Dipl.-Ing. (FH) degree from the University of Applied Sciences Kempten, Germany, in 1999 and his M.Eng. electronic systems degree from the University of Ulster, Northern Ireland in 2000. He joined Siemens Mobile in 2000, designing and implementing J2ME and MMI mobile phone system software. Later he joined the software technology group of Technology and Innovation at Siemens Mobile as a software architect. He is a member of the European research project IST-SCOUT for software radios. His main research interest is SDR terminal software reconfiguration management and validation, especially focusing on protocol stack software. He is a Ph.D. student at the Institute of System and Computer Architecture, University of Hannover.

ROLLO BURGESS (rollo.burgess@toshiba-trel.com) received his B.Sc. degree in physics with optoelectronics in 1994 from the University of Surrey, Guildford, United Kingdom. He joined the Telecommunications Research Laboratory (TRL) of Toshiba Research Europe Ltd. (TREL, Bristol, in November 2000 as a research engineer. Research interests include the application of advanced high-level software engineering techniques to wireless digital signal processing and software-defined radio.

PAUL WARR (paul.a.warr@bristol.ac.uk) received his Ph.D. in 2001 from the University of Bristol for his work in octave-band linear receiver amplifiers, his M.Sc. in communications systems and signal processing, also from Bristol, in 1994, and his B.Eng. in electronics and communications from the University of Bath, United Kingdom, in 1994. He is currently a lecturer in radio frequency engineering at the University of Bristol, where his research covers the front-end aspects of software (reconfigurable) radio and diversity-exploiting communication systems; responsive linear amplifiers, flexible filters and linear frequency translation. Funding sources for this research have included U.K. DTI/EPSRC alongside CEC ACTS and IST programs. Prior appointments have included the Marconi Company where he worked in secure high-redundancy cross-platform communications switching.

ZORAN GOBULICIC (golubicic@ttinorte.es) received his B.S.E.E. degree in electrical engineering from the University of Belgrade, Yugoslavia, in 1982 and joined the Institute of Applied Physics in Belgrade working on a project related to nonlinear RF components. From 1991 to 1999 he worked at the Institute of Microwave Techniques and Electronics on RF wideband nonlinear components intended for spread spectrum communication systems. His work includes adaptive systems and algorithms for RF chain linearity control. He is currently a research and development engineer at TTI Norte, Santander, Spain. His research interests include modeling and control of nonlinear microwave devices and systems.

JUERGEN SESSLER (juergen.sessler@siemens.com) received his Dipl.-Ing. FH in 1990 from the University of Applied Sciences, Rosenheim, Germany. In 1991 he joined the organization and information department of Siemens and Matsushita Components. Later he changed to different departments at Siemens AG and Siemens Business Services with a focus on software development and client-server applications. Since 2001 he has worked as a software architect at Siemens Mobile Group Technology and Innovation. From 2002 on he has been SubWorkPackage leader on the European research project IST-SCOUT.

FANNY PLATBROOD (fanny.platbrood@csem.ch) received her Dipl.Eng. (M.Sc.EE) degree in electrical engineering from the Faculté Polytechnique (Mons-Belgium) in 1996 after doing some research at the University of Rochester, New York, and the Swiss Federal Institute of Technology. She worked as an ASIC designer in the VLSI Design Department of Alcatel Bell, Belgium, until the end of 1998. She was more particularly responsible for ASIC development and testing for ADSL. In fall 1998, she joined CSEM to work on research and development as an expert in wireless communications. She is presently a project manager, and her areas of expertise are ASIC design, and digital and mobile communications (ADSL, H/2, WCDMA, baseband reconfigurability, and space-time processing techniques for multiple antennas systems).

BERTRAND SOUVILLE (souville@docomolab-euro.com) received in 2001 an engineering degree from the Ecole Nationale Suprieure d'Electronique et de Radiolectricité de Grenoble, France, and a Diplom-Ing. degree from the University of Karlsruhe, Germany. In October 2001 he joined the Future Networking Laboratory at DoCoMo Euro-Labs. He has been engaged in the IST-SCOUT project and follow-up OMG activities, and attended SDR Forum meetings. His main research interests are software-defined radio and mobile middleware technologies.

SOODESH BULJORE (soodesh.buljore@motorola.com) received his Ph.D. dégree in electrical engineering in 1996 from Ecole Doctorale Sciences pour l'Ingnieur, Ecole Centrale de Nantes. He held a post-doctoral fellowship from 1996 to 1998 in the Department of Electrical and Computer Engineering, University of California at San Diego. He joined the European Communications Research Labs Paris, Motorola Labs, in February 1998, where he is currently a senior staff research engineer He has been involved in the design and specifications of UMTS W-CDMA FDD in new modulation and transmit diversity schemes. He is currently technical manager of the IST project SCOUT, besides other internal research programs. His research interests include air interfaces and evolution beyond 3G; end-to-end architectures, protocol stacks, and enabling technologies for future reconfigurable networks and terminals.