Open communication devices using the EPOC operating system

Anders Waesterlid

To fully utilize the capability of new technologies, GPRS, Edge, UMTS, and other third-generation cellular standards need a whole new class of communication devices. In the last couple of years, Ericsson has investigated the software requirements for the advanced third-generation of cellular devices. This research involved evaluating in-house solutions as well as several non-Ericsson software solutions. Ericsson chose the EPOC operating system.

The author describes the architecture and key characteristics of EPOC, the client-server framework, and the user-interface design, which together offer an unprecedented level of openness in Ericsson's new range of communication devices.

BOX A, ABBREVIATIONS

API	Application program interface
DFRD	Device family reference design
Edge	Enhanced data rates for GSM and TDMA/136 evolution
GPRS	General packet radio services
HTML	Hypertext markup language
ISV	Independent software vendors
JVM	Java virtual machine
OS	Operating system
PIM	Personal information manager
TCP/IP	Transmission control protocol/Internet protocol
UMTS	Universal mobile telecommunications system
VGA	Video graphics array
WAP	Wireless application protocol

The third-generation device: a challenge

Third generation cellular devices will have excellent data rates (up to 384 kbit/s and maybe even higher in the local network) and considerably larger displays than the current generation of phones. Also, in accordance with Moore's law, they will offer much more memory and better execution speed. Together, these factors will allow us to design really usable applications other than telephony. Besides communication applications and general information-management applications, such as calendars and to-do-lists, the phones will be able to satisfy a wider range of personal or company-specific needs. However, if these types of more vertical applications are to become a reality, the devices used must satisfy new requirements for openness to third-party software. In fact, the degree of openness displayed by the software platform will determine how quickly third-party vendors develop new software applications.

Today, plain voice telephony is the completely dominating application for cellular phones. We believe that voice will still be important to third-generation devices, but so will other modes of communication.

What we foresee is the emergence of a category of smart phones and communicators characterized by their openness and their high-quality, well-integrated communication capabilities. The key applications for these devices will be

- enhanced telephony;
- unified messaging;
- contact management.

We call this set of applications the *communications suite.* All applications that play a key role in facilitating communications between people and enhancing mobility can be included in the communications suite.

A *smart phone* is a voice-centric device. However, it also offers good unified messaging as well as applications for managing personal information (PIM) and accessing information on the Web.

A *communicator* excels in unified messaging but still retains good voice capabilities. In addition, a communicator offers a richer set of applications for managing personal information and accessing information on the Web, thanks to its greater "real estate" of memory, display size, and so on. A communicator may be an integrated device or a large-display phone companion that communicates with a phone via a Bluetooth link.

Although smart phones and communicators will be multi-purpose computing devices, they will still be phones, which puts special requirements on software performance and behavior. Specifically, these devices must provide

- a phone-like feel—for example, one-handed use, no visible file structure, and the ability to handle voice as a data format. In essence, a device that is as easy to use as a home appliance, as opposed to a complex computer;
- excellent real-time capabilities; for example, short software start-up time and the ability to handle high-priority events, such as incoming phone calls;

Figure 1
Smart phones and communicators are "in between" ordinary voice phones and handheld PCs.

- capacity for handling limited resources of power, memory or execution capability;
- extreme software reliability and robustness with respect to third-party software;
- support for internationalization and wireless standards, such as Bluetooth and the wireless application protocol (WAP).

Software structure

The multiple application properties of smart phones and communicators also imply a shift in software structure. Today's ordinary phones are essentially a combination of core software components involving one or more cellular stacks, a real-time operating system, and device drivers. The user manipulates this core software through a man-machine interface.

With the introduction of multiple applications, an application framework will be needed which contains components that can be shared across applications. Some examples of components in the application framework are graphics libraries, databases, servers and data stacks (Figure 2).

Much of the core software and application framework that makes up the software platform will have to be common to many vendors before an open platform can be achieved. EPOC will provide this software for our devices and allow us to meet the phone-type requirements outlined above. The fact that Nokia, Motorola and other vendors have also chosen EPOC ensures the necessary industry standardization. In addition, EPOC provides us with a standard set of applications. Thus, by using EPOC, we will be able to focus on our key software areas, namely, the communications suite, the cellular stacks, and selected drivers.

EPOC

EPOC is Symbian's third-generation operating system (OS). During the past decade, the present EPOC32 OS has evolved from a rather small OS for less complex organizers into today's fully pre-emptive, multitasking, 32-bit operating system. The design goals for EPOC have been similar to those for cellular phones, with the focus on power conservation, a small static and dynamic footprint, and speed of execution. Ten years of experience has also allowed the OS to mature and stabilize.

EPOC has a fully object-oriented design using C++, which allows for different configurations to fit the footprints of a low-end smart phone as well as a full-featured communicator. EPOC can be viewed as four separate layers (Figure 3):
- The application layer.
- The system and server layers, which cor-

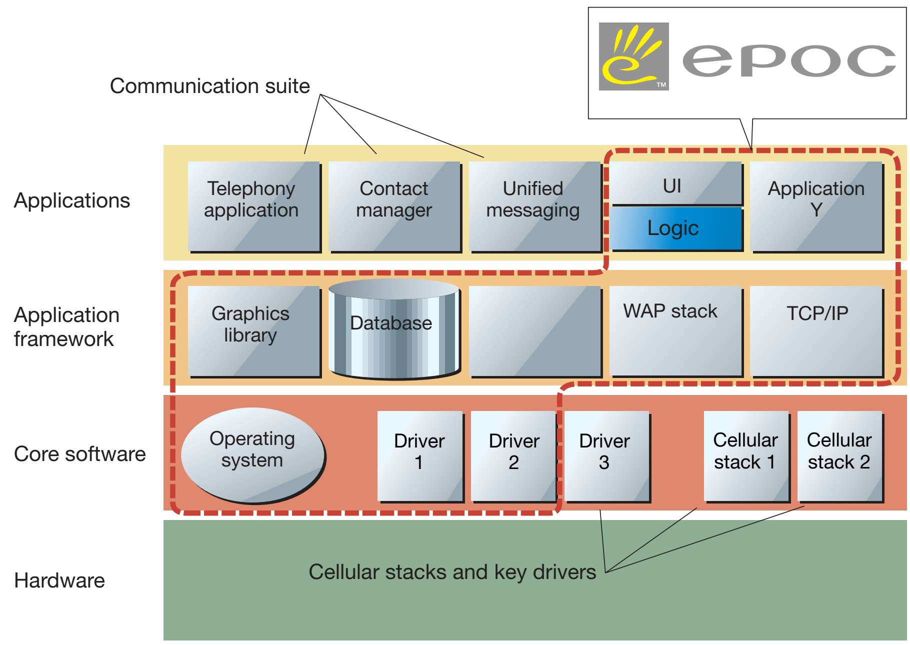


Figure 2
Ericsson software structure.

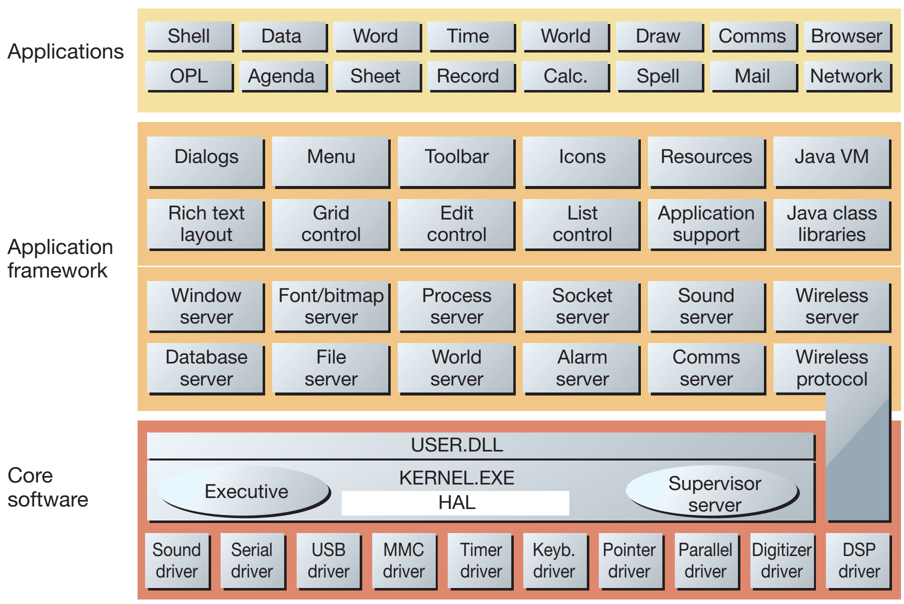


Figure 3
The EPOC operating system.

respond to the application framework in Figure 2.
- The base layer, which corresponds to the general part of the core software in Figure 2.

Capabilities of the base layer

Besides managing system resources, the key feature of the base layer is its capacity for handling real-time events, conserving power and minimizing memory use, maximizing speed, providing basic mechanisms for robustness, and enabling easy porting to new hardware. A few examples of how this is achieved follow below.

Portability

EPOC was designed to be portable. Because the kernel and device drivers must interface with the underlying hardware platform, some of their components are platform-dependent. However, hardware dependency is minimized by using a hardware abstraction layer.

Power conservation

A thread is the unit of execution in EPOC. High-priority threads are run first. The lowest-priority thread, called a null thread, only runs in the absence of other threads: the null thread immediately quiesces the CPU electronics in order to save power.

Software reliability

Memory is protected by using processes. Memory owned by one process cannot be accessed by another (unless explicit permission is granted). Another example of EPOC's approach to mission-critical software is the way it handles exceptions. When a program encounters an exception, such as an out-of-memory state, it recovers without any loss of data and, if the exception occurred part-way through an operation that had begun to allocate other resources, then those resources are freed immediately.

Lean and simple

Even though the EPOC executive supports mechanisms, such as pre-emption, semaphores, and mutexes to handle parallel, real-time execution of multiple threads, the use of pre-emptive multi-threading is discouraged, in that it is prone to faults. EPOC provides a simpler solution that permits most of its native applications and servers to run as a single thread—a solution based on the concept of *active objects*. Active objects are used to represent event sources, such as an application requesting a certain resource through a server. The server is then implemented as an *active scheduler* running a single thread. By limiting the time a call for a particular function is allowed to take, the need for pre-emptory action is eliminated. Using active objects is also a much leaner approach than using multiple threads, and reduces the need of dynamic memory.

Real-time performance

EPOC allows for very fast contextswitching from a device driver to a thread. This ensures that low and defined interruption latencies can be achieved without compromising the system architecture or the protection and isolation of the user process.

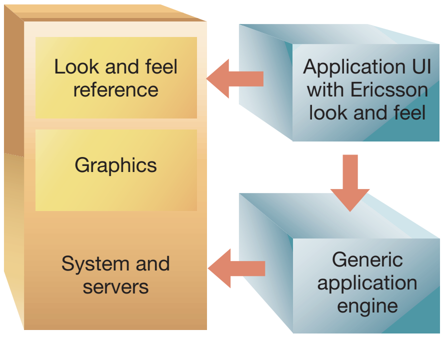


Figure 4
Ericsson can reuse the logic of an EPOC application while still achieving the Ericsson "look and feel."

Client-server framework

Servers simplify both EPOC operation and application programming. Usually, servers have a thread priority that is higher than any of their clients' thread priorities. This ensures that requests sent to the servers will not be blocked by their clients. EPOC has several servers:

- The file server owns all file, volume, and drive resources, and loads installable file systems. It allows clients to share these resources in an orderly manner.
- The window server owns the screen, keyboard and pen. Requests from a client to draw in a window, for example, will be attached to that window's visible extent. If the window server detects that part of a window has become invalid, it notifies the client.
- The alarm server keeps track of all requested alarms, and allows them to be either canceled or reviewed. If an alarm set by an application goes off, the alarm server sends a confirming signal to the application in question so that it can actively trigger another alarm, if one is warranted.

User interface design

The system layer in EPOC contains much of the user interface-related functionality. EPOC is designed to be used by a range of licensees on a variety of devices. To encourage openness and maintain interest among independent software vendors (ISV), several device families have been defined according to their display size and basic input metaphor:

- 640 · 240 pixels (1/2 VGA) landscape, keyboard expected;
- 240 · 320 pixels (1/4 VGA) portrait, pointer device expected;
- 320 · 120 pixels (1/8 VGA) landscape, pointer device expected;
- 200 · 200 pixels, keypad expected.

The first two are meant to be used in communicators, while the latter two are for smart phones. Device family reference designs (DFRD) will be provided. EPOC licensees will have to provide binary compatibility with these designs to minimize the need for testing on the part of independent software vendors. Nevertheless, the DFRDs allow Ericsson a considerable amount of differentiation:

- Applications are divided into an engine part, which contain the logic for manipulating application data, and a user interface part. Because of this division, the engine can be shared by device families, while Ericsson can still implement its own user interface.

Figure 5
New, open, communication devices will allow for more vertical applications. The application below, for example, supports navigation.

- Each DFRD contains a reference "look-and-feel" library. Ericsson can replace components in the library and still maintain binary compatibility. The DFRD also has options for adapting to different hardware configurations.
- Ericsson can add applications to the standard EPOC applications.
- Ericsson can replace standard EPOC applications, provided that application program interface (API) compatibility is maintained.

Wireless Java API

A Java virtual machine (JVM) is already available on EPOC. In future releases of EPOC, Java will be reinforced as the preferred language for third-party software development.

A wireless Java API will be provided to open up the full capability of the software platform to independent application design (Figure 6). The wireless Java API will be harmonized with current international and cross-platform standardization activities, which will make the platform even more open.

A new degree of openness

In terms of their software platforms, the new range of Ericsson communication devices based on EPOC will have an unprecedented level of openness. Ericsson has taken the lead in opening up several dimensions of this openness, which consists of the following areas:

Software platform openness

EPOC is already secured as the industry standard for smart phones and communicators. The concept of open APIs and DFRDs will ensure the development of a wide range of third-party software.

Cross-platform openness

The wireless Java API will allow applications designed for EPOC to run on other platforms as well, besides allowing applications designed for other platforms to run on EPOC.

Internal platform openness

EPOC's modular architecture enables Ericsson to add new hardware or software modules without degrading system characteristics.

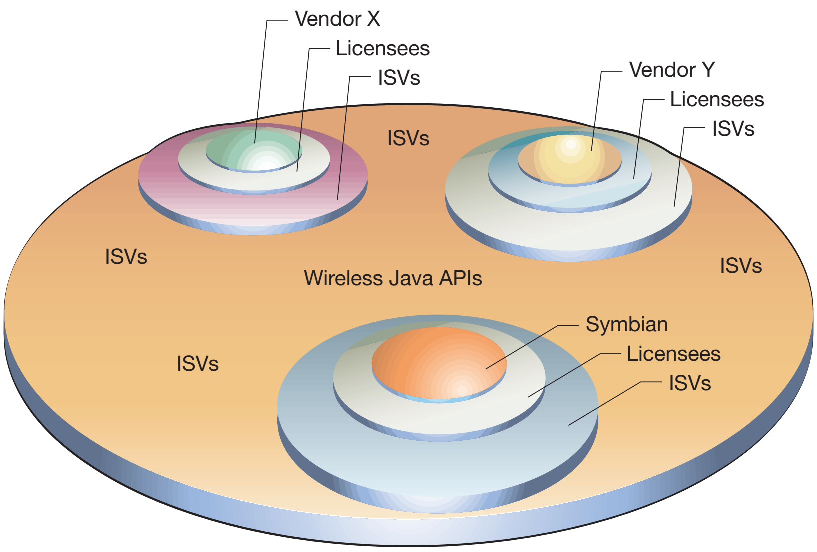
Software technology openness

The use of mainstream software technologies such as C++ and Java makes it easy to find tools for commercial development and the expertise required to use them.

Internet openness

EPOC standard applications include the

Figure 6
Wireless Java application program interface.



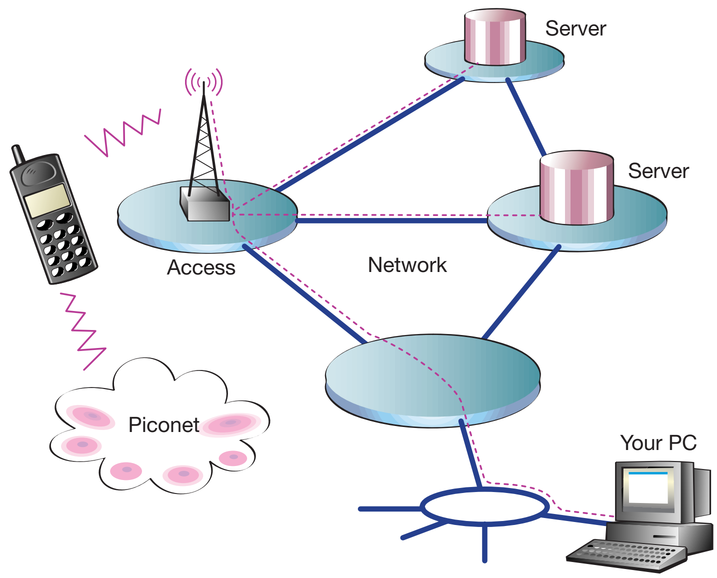


Figure 7
A new paradigm for communication device software.

WAP application and an HTML Web browser.

Piconet openness

EPOC will be Bluetooth-enabled, which means that all EPOC devices will be able to establish piconets in their local environments in order to utilize local resources outside of the phones.

Openness creates new opportunities

The new level of openness will give rise to a new paradigm for wireless device software (Figure 7). As a result, the third generation of cellular devices will no longer be limited by their own ability to store information, or by memory or processing capability. Instead, additional resources in the local environment or on the Internet will be available to the user.

Some services will best reside in the phone while others will be resident in the network. Services in the phone will be able to maximize usability, by applying detailed knowledge of the actual phone platform (hardware and software) and personal data in the phone. Network-based services can benefit from unlimited access to content, memory and execution capacity.

Users will be able to synchronize data between devices locally in the pico-network or remotely over the network.

Users will always have access to personal data, current messages, an up-to-date calendar, and so on. The smart phone or communicator will be so easy and efficient to use that it could easily become the primary tool for communicating at work or home, for managing personal data and for accessing information.

Conclusion

Ericsson's new, open, smart phones and communicators powered by the EPOC operating system will

- ensure improved usability and a new level of integration for communication applications;
- promote improved personal productivity by providing communication applications, personal information management and WAP/Internet access in a single device;
- establish a whole new, open-application environment—for the devices as well as the network—that satisfies many personal or company-specific application needs;
- preserve Ericsson's core software values: power conservation, high software reliability, and memory efficiency;
- capitalize on the high bit rates offered by third-generation networks.