

# Open application environments in mobile devices: Focus on JME and Ericsson Mobile Platforms

Angana Ghosh, Magnus Olsson and Patrik Persson

Advances in hardware in mobile handsets are rapidly overcoming computational constraints. Similarly, handset interfaces are improving in terms of quality and usability. At the same time, wireless operators are offering increasingly faster transmission rates for data communication. The table is thus being set, as it were, to provide a feast of innovative and interesting end-user services.

Not all these services are installed in end-user handsets at the time of purchase, but thanks to the adoption of open application environments, more and more handsets can download and upgrade services. Handsets with Java application environments, for example, can download games and other applications over the air.

The authors provide an overview of available open application environments. In particular, they focus on the Java for Micro Edition (JME) standard, a highly popular environment being developed through the Java Community Process (JCP). They also introduce Ericsson Mobile Platform's (EMP) middleware architecture and Open Platform API (OPA), putting emphasis on how OPA and JME provide a complete product offering to Ericsson's customers, including a compelling open application environment that meets wireless operator requirements.

## Introduction

The mobile communications industry is becoming increasingly aware of the importance and benefits of open application environments for mobile handsets. In this article, the term *open application environment* is used to depict environments that allow end users to download third-party applications to their handsets; the underlying operating system (OS) may be proprietary or open (for example, Linux).

Wireless operators want to make use of open application environments because they

facilitate the development of a consistent suite of applications that can be customized for a diverse community of end users. The integration of an open application environment on top of a consistent architecture further minimizes the efforts required to port applications between classes of devices. Indeed, to realize a platform for seamless end-user experience that spans across devices from different vendors in a global market, it must be possible to combine a plethora of connectivity features with a network-agnostic application program interface (API). Doing so allows end users to download even more applications and content, resulting in a fully personalized experience that can be extended with new features over time and generations of handsets.

## Introduction to EMP

### What is EMP?

Established in September 2001, Ericsson Mobile Platforms (EMP) is based on the research and development (R&D) group that developed the core technology for Ericsson's mobile phones throughout the 1990s. On January 1, 2005, EMP became a Business Unit within Ericsson. EMP provides mobile terminal technology to customers who want to develop and produce mobile phones for the GPRS, EDGE and WCDMA standards. The company's mission is very simple: "Help customers sell more phones."

By the second quarter of 2005, EMP had signed license and development agreements with sixteen customers worldwide. Several top handset manufacturers use mobile platforms from EMP. Other customers include major original design manufacturers (ODM) and several large manufacturers in Asia.

EMP develops extensive core technology in the form of integrated circuit design, platform software, complete reference handsets, and test software. Its business model is to license this core technology and sell consultancy support services.

### EMP system architecture

EMP offers a complete platform that consists of all the necessary integrated circuits and software needed to build a GPRS, EDGE or WCDMA handset. EMP's platforms must pass type approval, be proven to work in every major network in the world, and provide all necessary system functionality (network access, data communications,

## BOX A, TERMS AND ABBREVIATIONS

3G	Third generation mobile system	MIDP	Mobile information device profile
API	Application program interface	MMS	Multimedia messaging service
CDC	Connected device configuration	MSA	Mobile service architecture
CLDC	Connected limited device configuration	OAF	Open Application Framework
COM	Component object model	ODM	Original design manufacturer
EC	Executive committee	OEM	Original equipment manufacturer
ECM	Ericsson component model	OMTP	Open Mobile Terminal Platform
EDGE	Enhanced data for GSM evolution	OPA	Open Platform API
EMP	Ericsson Mobile Platforms	OSI	Open systems interconnection
GPRS	General packet radio service	OSS	Operations support system
GUI	Graphical user interface	OSS/J	OSS through Java Initiative
IMS	IP Multimedia Subsystem	PDA	Personal digital assistant
JCP	Java Community Process	R&D	Research and development
JEE	Java for Enterprise Edition	RAM	Random access memory
JME	Java for Micro Edition	RI	Reference implementation
JSE	Java for Standard Edition	RTOS	Real-time operating system
JSR	Java specification request	TCK	Technology compatibility kit
JTWI	Java Technology for the Wireless Industry	VSCL	Vodafone-specific class library
JVM	Java virtual machine	WAP	Wireless application protocol
		WCDMA	Wideband code-division multiple access

and multimedia services). Therefore the platforms give customers a very fast and safe way of bringing new 2.5G (GPRS) and 3G (EDGE, WCDMA) products to market.

The EMP platforms are based on common system architecture. Ericsson has evolved the system design using detailed use-case analyses identified together with customers, key operators and network vendors. The resultant architecture features layered service stacks similar to the OSI reference model (Figure 1). A common abstraction layer supports hardware, and a layer of common middleware services supports application software. Certain components are used for individual platform products and can be differentiated, for example, to match a specific mobile communications standard. In summary, a uniform application environment enables customers to derive more from their investments in application software.

#### EMP OPA

The EMP story described the platform components (Figure 1).<sup>1</sup> In this article, we describe the platform's middleware services, which provide mechanisms for managing and executing customer and third-party applications. Among other things, they include

- a Java execution environment with an extensible Java virtual machine (JVM);
- an open application framework for application security and management;
- a user interface toolkit; and
- a comprehensive API called Open Platform API (OPA).

Successful application development calls for an API that is rich in functionality, offers superior performance, and is well organized and documented. EMP designed OPA with these traits in mind. EMP's JME environment supports a rich set of standardized APIs giving customers the option to add standard or proprietary extensions.

The OPA functionality is divided into categories, where each category represents an aspect of handset functionality, such as data communication or security. Each category is further divided into subcategories, making it easy for programmers to locate specific functionality.

For optimum performance, OPA keeps time-critical data flows at the core of the platform. That is, the OPA functionality allows applications to control all essential details (such as resolution, encoding, or other parameters) of the data flows. This approach minimizes buffering and other overhead while giving applications full control of the data flow.

However, there is more to a successful API than functionality and performance. The associated interfaces, for example, continue to evolve as existing functionality is refined and new functionality becomes available. Great care must thus be taken to ensure that applications are aware of changes in functionality. Otherwise, application maintenance can become a nightmare. OPA has been designed to minimize application maintenance. The Ericsson component model (ECM, based on the component object model, COM) manages interface evolution, presenting interfaces to applications in a fully versioned, implementation-independent fashion. The mechanisms in OPA thus allow applications to take advantage of the evolving platform in a controlled manner.

#### Extension to OPA – support for Smartphones

A third-party application processor can be added if the handset manufacturer needs to run another OS, such as Symbian. In this case the application processor handles all applications and most multimedia functionality and the EMP platform handles a reduced

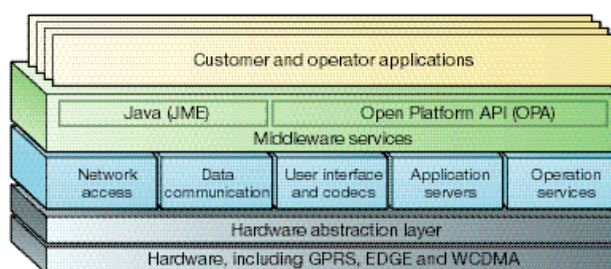


Figure 1  
System architecture of all EMP products.

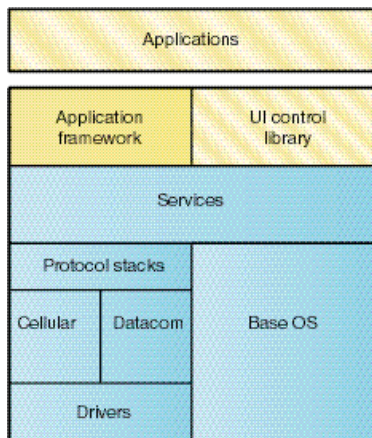


Figure 2  
Generic application environment.

set of functionality including all mobile telecommunications functionality. EMP has developed an interface mechanism that enables applications on the application processor to access (via OPA) every necessary function in the EMP platform as if the applications resided directly on the same processor.

## Open application environments in mobile handsets

Mobile handsets have traditionally been voice-centric devices that use proprietary operating systems to handle communication tasks. The operating systems were either developed by third parties or handset vendors. Ordinarily, the programming interfaces in these handsets were not made available to third-party developers. As a consequence, end users were dependent on handset manufacturers for applications.

However, software and platform companies such as Symbian, Nokia, Microsoft and Ericsson Mobile Platforms have changed the landscape by introducing open application environments. The phone functionality in the platform is exposed through open APIs. The “openness” of these interfaces makes them available to developers, which allows end users to obtain applications from a variety of sources. This, in turn, makes the device more interesting in the wireless value chain, because it opens the door to new revenue models from which every party stands to benefit.

Large-volume production of handsets requires that software and platform vendors carefully balance performance against cost. Given a relatively large footprint (and unit cost), the handset OS from SavaJe, for example, exposes a set of JSE APIs. Parts of the operating system are written in Java and parts are written in native language. A set of applications, such as the telephony and phone book applications, may thus be written in Java, and can be replaced and upgraded over the air. The all-Java platform primarily targets future high-end Smartphones.

Wireless operators like open application environments because they provide a common set of APIs for use in different devices from different vendors. In other words, they preserve operator control and ownership of the environment and its economics.

## Examples of open application environments

Although they differ greatly in degree of completeness, several application platforms and operating systems have been developed for the mobile environment. For the sake of comparison, we use a simplified graphical model of a generic application environment (Figure 2). A handset’s software stacks are built on a base OS (for example, OSE, or the Linux kernel) and a set of drivers for interfacing radio hardware, storage, multimedia and the user interface.

The communication capabilities of the device are provided to the application domain via a set of protocol stacks for the radio and data communications domains. The radio stacks are usually very tightly coupled to device hardware.

Other device resources are made available to the application domain via a services layer, for example, a wireless application protocol (WAP) stack or database server.

An application framework allows the system to control application life cycles and inter-communication.

The user interface control library handles the graphical user interface (GUI) in a systematic way.

The applications reside on top of this architecture, allowing end users to interact with the handset and network services.

## EMP OPA and JME

EMP is the only supplier to provide a fully integrated terminal platform complete with radio hardware, via drivers and stacks, and a mature, open application environment. EMP’s tightly integrated reference designs can be extended and adapted to yield custom designs.

Apart from a highly optimized and versatile JME environment, which includes a comprehensive set of standard Java specification requests (JSR), the Open Application Framework (OAF) and OPA support and provide extensive functionality for applications.

## Symbian

The Symbian OS is currently the most popular operating system for Smartphones. By itself, however, the OS does not provide an application framework or user interface that is particularly suitable for handsets. Additional software, such as the Symbian UIQ graphical library, must be added on top of it.

The main strengths of the Symbian Smartphone platform are that major hand-

set vendors including Nokia and Sony Ericsson are using it, and the UIQ framework is at least moderately customizable for a different look and feel according to the brand identity of the operator or vendor.

### **SavaJe**

SavaJe is a Java-based operating system and application environment for Smartphones. Every API is exposed via Java. The underlying functionality is implemented in either Java or native code. SavaJe aims to bring JSE functionality to mobile handsets as provided by a collection of proprietary Java APIs and existing Java API standards. Its core features are Java-based user interfaces, separation of application logic from look and feel, and the security features inherited from JSE.

Memory and CPU requirements for an all-Java OS platform imply that only high-end Smartphones are a realistic choice for SavaJe for the foreseeable future.

### **Windows Mobile**

Windows Mobile is Microsoft's product for mobile handsets. There are at least two variants of Windows Mobile, each of which adequately supports an application environment, including user interface controls, and an application framework:

- Windows Pocket PC, for stylus-input devices; and
- Windows Smart Phone, for keyed-input devices.

The Windows Mobile application environment, which is based on the .NET Compact Framework, supports third-party applications written in Visual C++, Visual C# or Visual Basic.

### **Linux-based OS platform**

Linux is a version of the traditional UNIX kernel that is available as open source. Several companies offer embedded Linux variants tailored for use in mobile devices. As with Symbian OS, an application framework, such as Qtopia UI from Trolltech, has to run on top of the Linux OS. Generally speaking, the applications are realized using a form of native user interface environment (QTopia) or they are based on a Java virtual machine.

The Linux kernel was originally designed for desktop computers and servers, not embedded devices. As a consequence, despite recent preemption enhancements in the Linux 2.6 kernel, it does not deliver the same real-time performance as the kernels of a

dedicated real-time operating system (RTOS). Several initiatives, including Mobilinux, aim to offer a more complete environment for mobile devices.

Where original equipment manufacturers (OEM) and wireless operators are concerned, the Linux OS scores high in terms of openness and low unit cost but it does not provide an application environment.

### **Market trends**

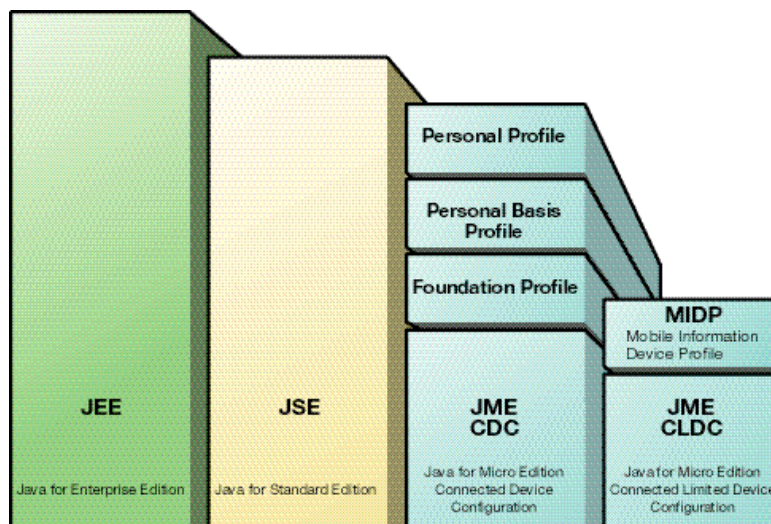
Proprietary and open operating systems for mobile handsets will continue to evolve and reside alongside one another. To date, handsets with open OSes have not introduced new device features to market more quickly than handsets with proprietary OSes. In fact, if anything, the opposite is true. And Ericsson believes this trend will continue. In the short term, Symbian will maintain its position as the dominant operating system in the Smartphone segment, but Microsoft's Windows Mobile is a noteworthy contender.

However, where mass-market phones are concerned the picture is quite different. Regardless of whether the handsets have a proprietary or open OS there will be continued strong growth in devices that expose programming interfaces through open APIs. These phones will continue to implement functions optimized in native code and exposed through open APIs, such as Java. A perfect example is the EMP platform family, with OPA exposed for native applications, and JSRs implemented for Java applications.

Zelus Group, which provides predictive analysis, ranks Linux "very strong" as the preferred operating system for connected devices, thanks especially to its openness and low cost.<sup>2</sup> Notwithstanding, diversity in embedded application environments and weak "hard" real-time performance might keep Linux from becoming the operating system of choice in the immediate future. Several Linux-based devices are currently available and the Linux stronghold is growing, particularly in Asia.<sup>3</sup>

At the JavaOne 2005 conference, Sun Microsystems stated that the Java economy yields about USD 120 billion annually. Being an open standard, JME is poised to continue its dominance as a popular application environment provided standards development keeps pace with the needs of the market. Ericsson believes Java will emerge as a strong technology for Smartphones and mass-market phones.

Figure 3  
JME platform compared with JSE and JEE.



## Introduction to JME

### JME architecture

Sun Microsystems introduced Java in 1995. Thanks to its ability to leverage the power of networks and to run without consideration for operating system or hardware compatibility, Java can today be found almost everywhere. It is in different kinds of computers, consumer gadgets, automobiles, medical devices, smart cards, toys and games.

But because one size does not fit all, the Java technology has been grouped into three categories or editions (Figure 3):

- Java for Standard Edition (JSE), for personal computers;
- Java for Enterprise Edition (JEE), for enterprise applications; and
- Java for Micro Edition (JME), for consumer and embedded devices, including mobile handsets.

JME, in turn, has been divided into two configurations: connected limited device configuration (CLDC), which is used in mobile handsets, and connected device configuration (CDC), which is typically used in personal digital assistants (PDA) and Smartphones. In time, given declining costs of memory and computation power, this differentiation might fade.

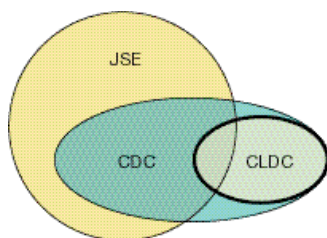
From the outset, CLDC developers have paid careful attention to footprint (RAM and Flash), thereby allowing handsets to include the technology early on. Moreover, the CLDC evolution has kept pace with hand-

set capabilities and market requirements, providing an increasing set of features for the development of interesting applications. Today, CLDC represents an excellent trade-off between functionality and footprint for mobile handsets.

The JME configurations (CDC and CLDC) are complemented by JME profiles that specify functionality for use by applications. In other words, whereas the configurations match handset capabilities, the profiles match the applications that run on the handset (Figure 4).

CLDC supports the mobile information device profile (MIDP), which provides APIs for the MIDlet application model, GUI, connectivity, multimedia, gaming, persistent storage, and more. Like CLDC, MIDP has evolved to keep pace with changing market needs. MIDP version 2 represents the *de facto* standard for mobile Java environments. In addition, MIDP 2.0 has gained momentum behind the CLDC/MIDP environment, thanks to developers and implementers who continue to identify functionality that makes it even more compelling. JCP is currently in the process of defining MIDP version 3 (JSR-271), which will overcome many of the limitations in MIDP 2.0. For example, MIDP 3.0 will enable concurrent MIDlets in a virtual machine (MIDP 2.0 allows only one application to run at a time). The specification will also allow inter-MIDlet communication, automatic launching of MIDlets, shared libraries and enhanced security. In addition, it will spec-

Figure 4  
Relationship between JSE and JME configurations.



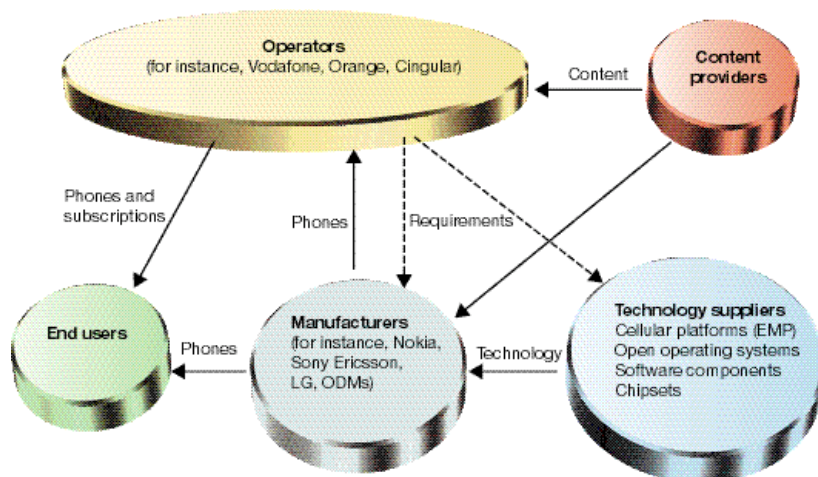


Figure 5  
The wireless value chain.

ify the behavior of MIDlets for CDC, giving MIDlets the potential to use enhanced services of the CDC environment. EMP is a member of the JSR-271 expert group that is helping define the new specification.

#### Standards organization: JCP

JCP is a Java standards development program created and managed by Sun Microsystems with participation by members of the industry, academia and individual users. Java specification requests (JSR) represent descriptions of proposed and final specifications of the Java platform. JSRs are developed in JCP. After JSRs have been finalized, they are implemented as additional APIs in handsets.

EMP is a highly committed member of JCP, actively contributing to various JME expert groups (Table 1). EMP is also a member of the JCP executive committee (EC), the governing body of JCP. In addition, EMP monitors the development of specifications in the mobile service architecture (MSA) expert groups.

EMP believes that the success of Java can be attributed in part to the efforts of JCP in enforcing strong compatibility requirements for Java implementations. The JCP community is organized to develop

- high-quality specifications;
- reference implementations (RI) that provide proof of concept; and
- technology compatibility kits (TCK) that determine whether or not implementations comply with the specifications.

This trio of aids ensures that developers write applications to a specification and not to an implementation.

### JME value proposition

Although Java did not initially live up to the promise of its early catch phrase *Write once, run anywhere*, its platform independence, interoperability, compatibility, and freedom from vendor lock-in have been key attributes contributing to its adoption in a variety of environments, including mobile handsets.

#### Entire value chain

Figure 5 describes the wireless value chain. At present, the myriad of business relationships that are necessary to take a phone from conception to product and to provide content and services for it is often mind-boggling. For the technology for open ap-

TABLE 1: LIST OF JAVA SPECIFICATION REQUESTS (JSR) IN WHICH ERICSSON AND EMP PARTICIPATE

JSR Number	JSR Name
JSR-211	Content Handling API
JSR-226	Scalable 2D Vector Graphics
JSR-232	Mobile Operational Management API
JSR-234	Advanced Multimedia Supplements
JSR-239	Java Bindings for OpenGL ES
JSR-246	Device Management API
JSR-253	Mobile Telephony API
JSR-258	Mobile User Interface Customization API
JSR-271	Mobile Information Device Profile (MIDP), version 3.0
JSR-281	IMS Services API

Figure 6  
Third-party Java implementation on customer handsets (EMP platform) vs. EMP Java.

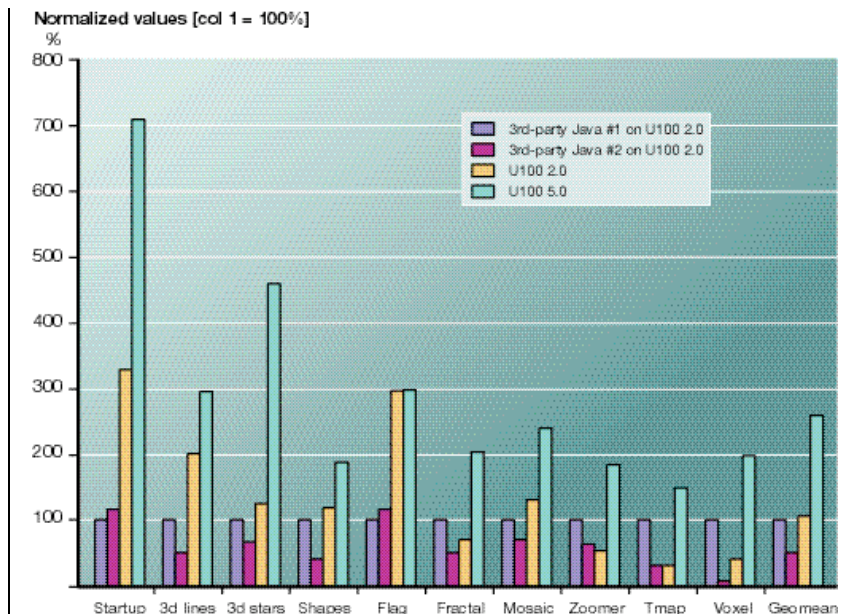
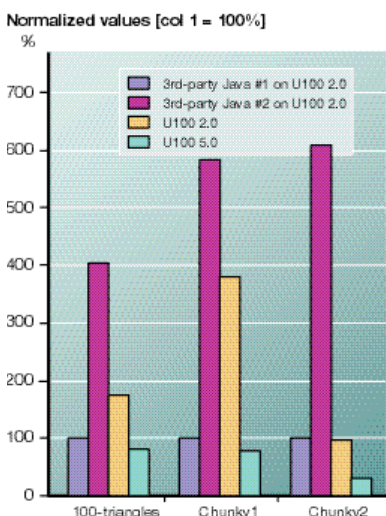


Figure 7  
Third-party Java implementation on customer handsets (EMP platform) vs. EMP Java.



plication environments to thrive in this market, every stakeholder must adopt it, providing clear paths for revenue models. Moreover, the technology needs to be employed in an end-to-end manner starting from network servers down to mobile devices.

The market for Java on mobile handsets can best be described as an upward spiral. In September 2005, the GSM Association reported that the number of users of mobile handsets had exceeded two billion.<sup>4</sup> And Sun Microsystems states that there are currently more than 700 million Java-enabled handsets.<sup>5</sup>

**Content provider benefits**

The sheer size of the market gives purveyors of content an excellent opportunity to develop and market their applications.

**Operator benefits**

Thanks to the presence of a lively Java developer community, applications can be developed and deployed quickly and in a cost-effective manner. Operators can thus bring interesting services to market in less time, yielding a quick return on investments. What is more, Java's built-in security features promise peace of mind because each application must pass through a stringent

verification process before it can run on a device. Using value- and event-based charging models, JME applications can give operators and aggregators an additional and consistent source of revenue.

**End-user benefits**

Java brings PC-like openness to mobile handsets, giving end users the ability to download the applications of their choice from a variety of sources. This is good, because end users do not like having to depend on an operator or device manufacturer for applications. The ability of Java to dynamically download applications transforms mobile handsets into a device for personal entertainment and productivity.

**Device manufacturer benefits**

Java arms manufacturers with new features with which to sell their devices. It facilitates device customization and serves as a differentiating factor through optimized implementation of standard and proprietary features. The various editions of Java (JSE, JEE and JME) position it to provide the end-to-end service delivery architecture. The OSS through Java (OSS/J) initiative, for example, develops APIs for end-to-end telecommunication solutions. Ericsson is a participant of this initiative.<sup>6</sup>

EMP Platforms	U100 5.0	U250 1.0	U250 2.0	U250 M1.0
Amark Geomean (FPS)	22	~40	40-50	60+
Triangles/s in Java (JSR 184)	16,010	46,466	52,000	100,000
Triangles/s in software	20,000-100,000	40,000-200,000	40,000-200,000	100,000-500,000
Mpixels/s (Display output)	1-2	5	5	72

**Figure 8**  
Java performance roadmap: Higher numbers in the EMP Platform names indicate more recent versions.

### Ericsson and EMP

Ericsson and EMP believe in and support Java. EMP's mission is to help its customers sell more phones. To succeed, EMP must do more than merely supply the right functionality in its platforms – it must also make this functionality available to application developers, enabling them to use the EMP platforms. JME is an effective means of doing this because it complements the native EMP OPA for situations where platform independence is important. The optimization of telephony, communications, and application functionality in the platform helps EMP's customers to ensure that applications written by third-party developers have been optimized for speed and low power consumption.

Ericsson is the largest supplier of mobile systems. Its customers include the ten largest mobile operators in the world; 40% of all mobile calls are made through systems from Ericsson. The company's technology leadership, which includes end-to-end capabilities, gives customers first-mover advantage. Ericsson is working to ensure that network services are available for rapid adoption in handsets. Thanks to a strong developer base, JME and its partner editions in the server space (JSE and JEE) provide an excellent mechanism for delivering and further enhancing end-to-end services, such as multimedia messaging service (MMS). In time, the IP Multimedia Subsystem (IMS) will also find its way into JME. Ericsson and BenQ Mobile are leading the standardization of IMS in JCP through JSR-281 (IMS Services API).

By adopting JME, Ericsson and EMP continue to encourage the Java developer community to develop services that take full advantage of Ericsson technology and EMP devices.

Ericsson and EMP's interest in JME is backed by stakeholder interest in the technology – competitors, content providers and wireless operators whose collective efforts

are making the wireless data ecosystem more profitable and sustainable.

### JME in EMP

#### JME implementation in EMP platform

JME continues to play an integral part in EMP platforms. At the heart of the EMP JME environment is the Java virtual machine, whose advanced techniques, such as adaptive dynamic compilation, achieve performance comparable to that of native applications. The just-in-time compiler is carefully tuned to guarantee a smooth, uninterrupted user experience.

Amark 1.3 benchmark scores show that the EMP JME implementation outperforms other single-CPU phones, several Smartphones, and most PDAs with a dedicated application processor.<sup>7</sup>

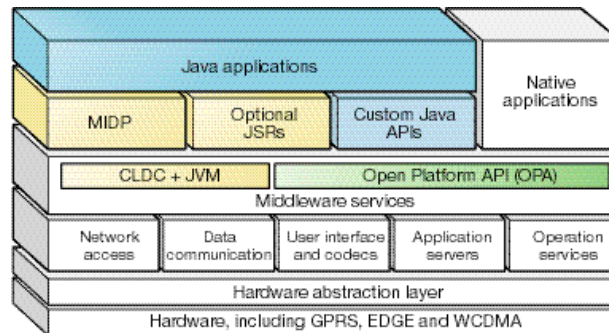
Java performance varies with choice of software implementation and platform hardware. Moreover, the performance of implementations by different vendors on equivalent hardware varies significantly. Certainly, it is possible to deploy a third-party Java implementation on an EMP platform, but in all likelihood the performance will be inferior to that implemented by EMP on its own platform (EMP Java). Figures 6-9 compare the results of EMP Java with third-party implementations in customer handsets built on EMP platforms.

The EMP JME environment includes version 2.0 of the MIDP profile, which allows operators, manufacturers and end users to

	EMP Java	3rd-party Java
Integration time	2 man months	6-12 man months
Technical risk	Minimal	Medium
Performance	100%	30-80%

**Figure 9**  
Time to market, integration cost and risk.

Figure 10  
JSR implementation in EMP platform.



#### TRADEMARKS

- Linux is a registered trademark of Linus Torvalds
- Mobilinux is a registered trademark of MontaVista Software Inc.
- OMTP Limited is a registered trademark.
- OSE is a registered trademark of Enea Embedded Technology, a subsidiary of Enea AB
- OSS through Java Initiative and OSS/J are collectively owned by the members of the OSS through Java Initiative
- Qtopia is a registered trademark of Trolltech in Norway, the United States and other countries
- SavaJe is a registered trademark, and SavaJe OS is a trademark, of SavaJe Technologies, Inc.
- Sun Microsystems, Java, JEE, JME, JSE, and Java Community Process are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries
- Symbian and Symbian-based trademarks are brands of Symbian Software Ltd and as such are valuable assets to Symbian Software Ltd.
- UNIX is a registered trademark in the United States and other countries, exclusively licensed through X/Open Company, Ltd.
- Visual Basic, Visual C++, Visual C# and Windows Mobile are trademarks or registered trademarks of Microsoft Corporation.
- Vodafone is a trademark of Vodafone Group Plc.

download and install MIDlet applications. The implementation complies with initiative JSR-185 of the Java Technology for the Wireless Industry (JTWI). EMP's role in JCP ensures compliance with future standardization. The EMP platform also contains many other JSRs including

- JSR-82 (Java APIs for Bluetooth);
- JSR-177 (Security and Trust Services API); and
- JSR-184 (Mobile 3D Graphics API).

In addition, EMP has implemented explicit operator APIs, such as the Vodafone-specific class library (VSCL).

EMP was first in the industry to implement a multitasking CLDC environment based on Sun's CLDC HotSpot Implementation.<sup>9</sup> The multitasking CLDC environment allows multiple Java applications to run concurrently, even in the background. This paves the way for more diverse applications, such as productivity tools, e-commerce, and enterprise applications.

The Java environment includes a wide range of APIs (JSRs) that tap into an abundance of standard features in the EMP platform. The extensible Java virtual machine supports additional Java APIs, such as JSRs and operator-specific class libraries. Developers who implement custom Java APIs stand to benefit from the rich functionality provided by OPA.

Figure 10 shows how the JME implementation is integrated into the EMP platform architecture. Java applications exist

alongside native applications using the functionality of the platform and OPA via the JSRs delivered with the platform or via customer JSRs. The result is a high-performance, low-footprint Java platform that is comprehensive enough to be used directly but can also be adapted to new customer and operator requirements.

#### JME and EMP – a winning combination

EMP is convinced that OPA and JME make a winning combination. To satisfy end-user expectations, mobile handsets must be stable, feature-rich and consistent. EMP's line of platform offerings beats the competition in component count, cost and performance. What is more, EMP exposes rich and optimized platform features through OPA.

EMP believes in Java and is confident that JME will continue to evolve at a satisfactory pace. JME in EMP platforms helps ensure platform independence: optimized mobile terminal functionality is open for innovation to the Java developer community, and a secure download and execution environment can accommodate applications written by third-party developers. This translates into peace of mind. End users can rest assured that core applications have been fully tested and will always be available. Moreover, they will be able to securely download applications from a variety of sources. And operators gain a flexible, secure, and optimized implementation of Java features that draw on the EMP platform implementation.

To maximize the end-user experience, EMP platforms provide a single, shared, user interface toolkit that supports a consistent look and feel across native and Java application environments.

EMP is committed to enable its customers to enter the 3G market safely and in a timely fashion. Third-generation mobile communication brings the promise of faster data networks and innovative services. An optimized JME implementation for the EMP platform eliminates customer worry about Java, leaving customers free to focus their energies on obtaining greater differentiation and market share.

Shorter development time means less cost and shorter time to market. EMP is helping to shorten its customers' development time every step of the way. EMP's implementation of the JME environment facilitates customer extensions by means of additional JSRs or proprietary extensions. The EMP platform comes with the JSRs required by leading operators.

Handset manufacturers want to preserve and reuse investments in application software. EMP's JME is built on a common programming environment (OPA) to provide consistency across platforms. Customers may thus safely reuse their JME-based implementations in multiple products.

Device manufacturers continue to focus on reducing device costs in memory and processors, component count and physical die area as well software licensing. The complete EMP platform with JME gives manufacturers an important advantage in meeting this challenge.

EMP works closely with wireless operators to implement their requirements during the early development phases of the platform. Moreover, given that EMP's platform is being used in every third WCDMA handset, EMP is well positioned to address fragmentation through a compatible and consistent Java implementation.

Toward the end of 2004, several major wireless operators formed the Open Mobile Terminal Platform (OMTP) forum, whose aim is to stipulate unified operator requirements regarding the end-user experience, device customization, and application interfaces. EMP actively participates in OMTP projects to ensure that customers receive an OMTP-ready platform in a timely fashion.

## Conclusion

Ericsson embraces Java as a strategic choice for open application environments. As more and more end users experience mobility they will become increasingly interested in more compelling services, with the expectation that these services will make the most of advanced features in mobile devices and networks.

A burgeoning number of services and device features means that cost of deployment must be kept to a minimum without affecting time to market. A consistent application environment is thus needed to help reduce complexity and allow developers to concentrate on providing an optimized user-experience.

The Java community is motivated and well aligned to take on this challenge – it provides a consistent and comprehensible set of Java standards in time for a fast-paced mobile industry that is eager to adopt and bring new innovative services to market.

EMP's highly integrated Java environment provides optimized access to EMP platform features with best-of-class performance. This helps customers to differentiate their Java offering. The complete EMP platform with JME provides an important advantage in meeting the price challenge that device manufacturers must face.

Empowered with the strength of Java, EMP's Open Platform API (OPA) provides the best of both worlds: EMP OPA enables phone manufacturers to develop high-performance applications that take early advantage of new network functionality. At the same time, EMP JME technology enables operators, and a large and vibrant Java developer community, to add applications with features that can effectively target any group of users.

EMP's platform is already proven and widely deployed – it is used in every third WCDMA handset. A major strength of the EMP platform is its common programming environment (OPA), which provides consistency across network technology and product lines. This means EMP is positioned to address destructive fragmentation of technology and services (a key operator concern) through a fully compliant and consistent Java implementation. The combination of OPA and the JME environment thus makes it possible to bring truly interesting services to mass-market phones.

## REFERENCES

1. Kornby, M.: The EMP story. Ericsson Review, Vol. 82(2005):1, pp. 32-43
2. [www.zelosgroup.com](http://www.zelosgroup.com)
3. [www.linuxdevices.com/articles/AT9423084269.html](http://www.linuxdevices.com/articles/AT9423084269.html)
4. [www.gsmworld.com/news/press\\_2005/press05\\_21.shtml](http://www.gsmworld.com/news/press_2005/press05_21.shtml)
5. [www.sun.com/smi/Press/sunflash/2005-06/sunflash.20050627.9.html](http://www.sun.com/smi/Press/sunflash/2005-06/sunflash.20050627.9.html)
6. [www.ossj.org/index.shtml](http://www.ossj.org/index.shtml)
7. [www.anfyteam.com](http://www.anfyteam.com)
8. [www.ericsson.com/press/20050627-104418.html](http://www.ericsson.com/press/20050627-104418.html)