● Development Reports ●

"MOAP," Software Platform for FOMA Terminals

Hiroyuki Tsuji, Keisuke Ohno and Tetsu Saito

*We have developed "MOAP," a software platform for Linux OS and Symbian OS[*1] which can be used for FOMA terminal software development. This platform makes it possible to improve the quality of the software, reduce costs and shorten the development period.*

1. Introduction

Since the i-mode service was introduced in 1999, the mobile terminal services have migrated gradually from voice communication services to data communication services, such as browsing Web sites, sending/receiving e-mails and downloading contents such as ringtone and Java[*2] applications. Moreover, in Freedom Of Mobile multimedia Access (FOMA) terminals, which were introduced in 2001, video/visual services such as video phone and i-motion have been launched, which take advantage of high-speed data communication.

In order to achieve such diversified and sophisticated services, mobile terminal software has become increasingly complicated and the development scale has increased explosively. This complexity of mobile terminal software and increased development scale are placing significant burdens on mobile terminal vendors and DoCoMo in terms of resources devoted to the analysis and development period, as well as for securing software quality.

As a solution to these problems, DoCoMo has developed Mobilephone Oriented Application Platform (MOAP), a software platform that can be used commonly by multiple mobile terminal vendors and software vendors when developing mobile terminal software.

This article provides an overview of the development of MOAP, introduced in November 2004.

*1 Symbian OS and all trademarks and logos related to Symbian are trademarks or registered trademarks of Symbian Ltd.

*2 Java: Object oriented development environment specifically designed for networking advocated by Sun Microsystems in the US.

2. Issues in Software Development for Mobile Terminals

In conventional development, mobile terminal vendors developed all parts of the mobile terminal software by themselves, including applications, middleware, Operating System (OS) and device drivers. As the services to be supported increase and become more diversified, however, it is becoming increasingly difficult for a single company to develop all the necessary pieces of software within a limited development period. For this reason, most mobile terminal vendors have begun to purchase parts of the software from software vendors and install it in the terminals.

However, when implementing applications created by software vendors, it becomes necessary to develop additional special interfaces to middleware already implemented by the mobile terminal vendors. This represents a large burden for the software vendors. Moreover, as applications become more and more sophisticated, the middleware software itself becomes increasingly complicated, and the development scale grows; the burden of the accompanying maintenance tasks cannot be taken lightly anymore either (**Figure 1**). Under these conditions, it is necessary to make mobile terminal software development substantially more efficient in order to be able to develop and provide advanced services in the future as well.

Given these factors, it is clear that the development scale of mobile terminal software can be reduced by sharing the infrastructure software including OS and middleware between mobile terminal vendors, or at least reusing it for several mobile terminal models, as shown in **Figure 2**. This eliminates the need for development of interfaces between applications created by software vendors and specific middleware required for individual mobile terminal vendors, allowing for an improvement of Quality, Cost and Delivery (QCD) in mobile terminal development.

We thus developed “MOAP,” a software platform that can be used commonly by multiple mobile terminal vendors and software vendors in order to solve the problems in mobile terminal software development outlined above.

3. Software Platform

3.1 Advanced OS

When considering enrichment of functions in mobile terminal software, a serious issue is the functions provided by the OS, its base. Conventionally, Real Time Operating Systems (RTOS) such as μITRON (μ Industrial TRON) have been used in mobile terminals because they allowed developing highly responsive applications by emphasizing the real time performance and efficiently utilizing small, low-power Central Processing Units (CPU).

However, RTOSs lack support for stable execution of multiple applications and are not equipped with sufficient utilities; hence, special knowledge and skills are required in development, which is a drawback.

For this reason, taking improvement of development efficiency and support for future advanced hardware into consider-

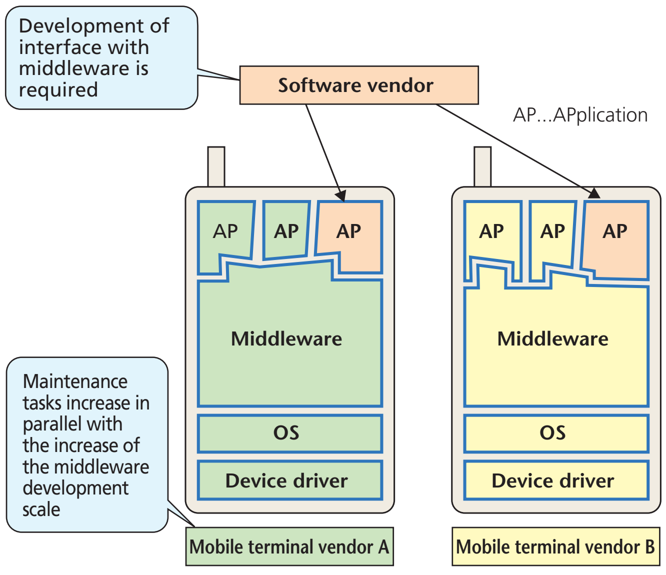

Development of interface with middleware is required
Software vendor
AP...APplication
AP
AP
AP
Middleware
OS
Device driver
AP
AP
AP
Middleware
OS
Device driver
Maintenance tasks increase in parallel with the increase of the middleware development scale
Mobile terminal vendor A
Mobile terminal vendor B

Figure 1 Conventional development of mobile terminals

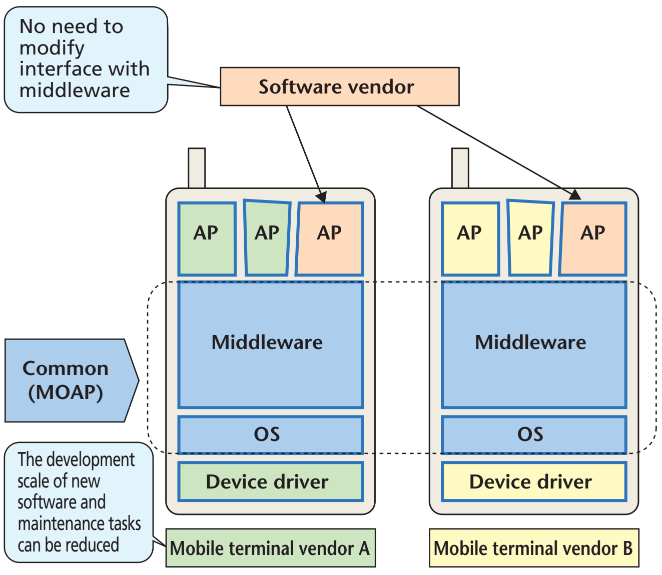


Figure 2 Development of mobile terminals with MOAP

ation, we decided to adopt the sophisticated OSs "Linux OS" and "Symbian OS," which are highly expandable and offer support for various software management functions as standard such as multi-threading and memory protection, allowing safe execution of multiple applications, as OSs for MOAP. We thus built "MOAP (L)," a platform based on Linux OS, and "MOAP (S)," a platform based on Symbian OS.

3.2 Middleware

In MOAP, the following middleware was implemented based on Linux and Symbian OS mentioned above.

1) User Interface (UI)

In addition to general UI functions including operations via displayed components such as windows, buttons, list boxes and the keyboard, MOAP provides UI functions unique to mobile terminals such as keeping a button pressed in order to achieve sufficient operability solely with the numeric pad and cursor keys. MOAP allows development of applications supporting operations unique to mobile terminals by using these functions.

MOAP (L) supports such functions through customized versions of X-Window and Gimp ToolKit+ (GTK+) widely used in Linux OS. MOAP (S) supports these functions by building unique object oriented libraries on top of the UI toolkits provided by the OS.

2) Application Management

In mobile terminals, it is necessary to coordinate the operations of multiple applications, such as phone book and mail applications. For this reason, MOAP is equipped with functions for executing application startup and close processing triggered by an operation of the keys and reception of a call, communication functions for exchanging data among applications, exclusive management functions used when multiple applications take turns using the Liquid Crystal Display (LCD) screen and speakers and sequence management functions used during startup and shutdown of the mobile terminal. In addition, it has a function that monitors the operation status of applications at all times, which detects failure promptly and records any failure information if an abnormal condition occurs.

3) Storage Management

As the resolution of cameras and the amounts of video and music contents handled by the mobile terminals grow, the amounts of data that must be processed by the mobile terminal applications are ballooning. Accordingly, the selection of storage devices mounted in mobile terminals is becoming increasingly diversified as SD Memory Card and Memory Stick are adopted in addition to the built-in flash memory.

MOAP provide unified functions for writing, reading and deleting data for this wide array of storage devices.

4) Device Status Management

Mobile terminal applications are required to recognize the open/close status of a folding-type mobile terminal, radio signal strength and remaining battery, silent mode, terminal lock, etc., and to take appropriate actions accordingly to this information.

MOAP provides a device status management function that manages such information centrally so that each application does not need to manage it individually. Moreover, changes to any status, e.g., open/close of a mobile terminal, are automatically notified to applications requiring such information.

5) Communication Functions Using Mobile Communication Networks

Communication using mobile communication networks is divided into several types including circuit switching-based communication such as voice and video phone communication and packet switching-based communication such as i-mode, and each type requires special control and processing.

MOAP provides control functions necessary for each communication type. For example, in circuit switching-based communication, MOAP provides setting functions for additional services, including answering machine and drive modes. For packet switching-based communication, a function that controls opening and closing of packet communication channels is provided, among others.

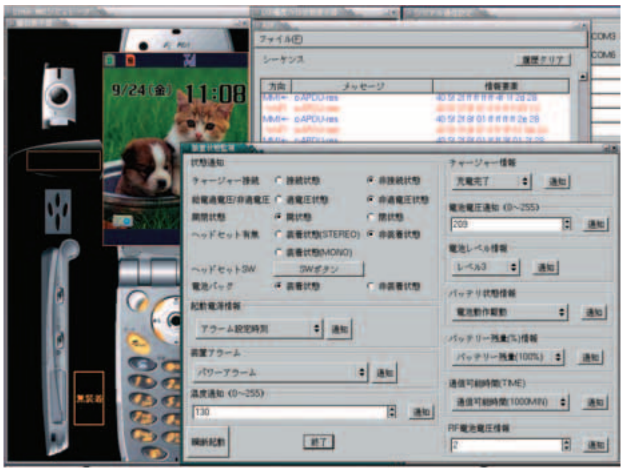
6) Icon Management

Various icons are displayed on the LCD screen in order to notify the status of the mobile terminal, such as radio field strength, remaining battery, mounting of external memory, reception of mails, unread mails, Secure Sockets Layer (SSL) and infrared communication.

In order to implement this mechanism in an integrated manner, MOAP is equipped with a function for displaying and updating icons automatically according to the status changes.

3.3 Development Environment

In order to help mobile terminal vendors and software vendors exploit the enriched OS and middleware environment discussed above in their application development, MOAP provides a development environment for software in addition to the mobile terminal software itself. By making use of this develop-



*This function is provided in Japanese only.

Photo 1 MOAP (L) mobile terminal emulator

ment environment, it is possible to carry out efficient application development on PCs.

MOAP (L) allows using editors, debuggers, configuration management tools, etc. that are available on Linux OS. On MOAP (S), Visual C++®*3, Microsoft's integrated development environment, and related products can be used. Moreover, each platform provides a mobile terminal emulator so that the applications can be tested in a PC environment, allowing for easy confirmation of the operations of the software on the mobile terminals. **Photo 1** shows the mobile terminal emulator of MOAP (L).

4. Conclusion

This article provided an overview of MOAP, which was developed in order to accommodate the increasing complexity and growing development scale of mobile terminal software. MOAP improves the development efficiency by providing common functions required for mobile terminal software development and is already being applied in the FOMA 901i series.

It will become important to enrich MOAP's common functions further and make sure to provide it to mobile terminal vendors and software vendors in a timely manner in order to promote a more efficient development of DoCoMo's mobile terminals, which are becoming more and more sophisticated and complicated.

In the future development of this software platform, we intend to examine how to perform abstraction of the hardware layer and reinforcement of multi-platform compatibility by hiding the differences of hardware and OS using middleware, as well as pursue international rollout by supporting multiple languages and overseas specifications using middleware.

*3 Visual C++® is a registered trademark or trademark of Microsoft Corporation in the US or other countries.

ABBREVIATIONS

CPU:	Central Processing Unit
FOMA:	Freedom Of Mobile multimedia Access
GTK+:	Gimp ToolKit+
LCD:	Liquid Crystal Display
MOAP:	Mobilephone Oriented Application Platform
μITRON:	μ Industrial TRON
OS:	Operating System
QCD:	Quality, Cost and Delivery
RTOS:	Real Time Operating System
SSL:	Secure Sockets Layer
UI:	User Interface