(3) Expansion of "MOAP" Software Platform for Mobile Terminals

Masafumi Yoshizawa, Yuichi Ichikawa and Yurin Kogetsu

*MOAP™*1, the software platform which provides common functions for developing FOMA applications, has been expanded for the 902i series of FOMA terminals. This supports new services such as PushTalk and international roaming, and improves basic functions.*

1. Introduction

The software used in today's mobile terminals is becoming increasingly advanced and complicated requiring a development effort on a massive scale. This can extend development periods and lower software quality while raising the overall cost of mobile terminal development. Against this background, planning the deployment of new services while holding down the scale of software development has become a major issue for both mobile terminal vendors and DoCoMo.

In response to this problem, DoCoMo has been developing a software platform called Mobilephone Oriented Application Platform (MOAP), that can be used in common by terminal and software vendors in the development of mobile terminal software [1]. This article overviews functional extensions to MOAP such as support for the PushTalk service launched from the 902i series of FOMA terminals.

2. Development Background and Effectiveness

Software loaded on a mobile terminal can be divided into two main functions: a communications-control section that controls radio communications with the network side and an application section that provides specific services for the user. Among these, the scale of development for the application section has been increasing explosively as services become more diversified, and to make this more efficient, we have been constructing a platform[*2] for that software.

The software making up the application section takes on a four-layer configuration consisting of device drivers[*3], Operating System (OS)[*4], middleware[*5], and applications[*6] (**Figure 1**). Here, MOAP provides common functions for the OS and middleware in a FOMA terminal (**Figure 2**). It has the following features.

1) Adoption of a General-purpose High-performance OS

With conventional Real-Time Operating System (RTOS)[*7] used in DoCoMo mobile terminals, it was difficult to control software that was becoming increasingly complicated as the need for advanced functions such as the simultaneous execution of multiple applications grew. This gradually increased the work load on software developers. In response, MOAP adopts a general-purpose high-performance OS (e.g., Symbian OS™[*8], Linux OS[*9]) that provides multi-process support and memory-protection functions.

Presently, development is progressing on MOAP(S), a platform based on the Symbian OS, and MOAP(L), a platform based on the Linux OS.

2) Provision of Common Middleware and Application Program Interface (API)[*10]

In MOAP, common and necessary functions required by a mobile terminal are provided in the form of middleware. These

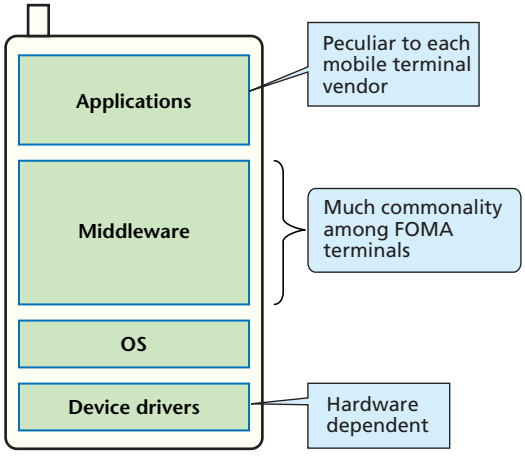


Figure 1 Layered configuration of mobile terminal software

*1 MOAP™: A trademark of NTT DoCoMo.
*2 Platform: Here, a group of software functions that can be used in common by FOMA terminals provided by various mobile terminal vendors.
*3 Device drivers: Software for controlling the various hardware components making up a mobile terminal.
*4 OS: Basic software for executing various types of software on a mobile terminal.
*5 Middleware: Software providing functions for common use by multiple applications.
*6 Application: Software having a user interface and providing the user with specific functions. Videophone, e-mail and browser are typical examples.

include the functions that constitute the application framework[*11] as well as those for communications control, system control and multimedia management.

Such middleware enables mobile terminal vendors to concentrate on unique applications, and the integrated API can lighten the load on software vendors when developing software for various mobile terminal vendors.

In addition, the provision of common communication sequences and a common framework can unify mobile terminal processes and operations among mobile terminal vendors. This can make for more efficient development test and more uniform terminal operation as seen from end users.

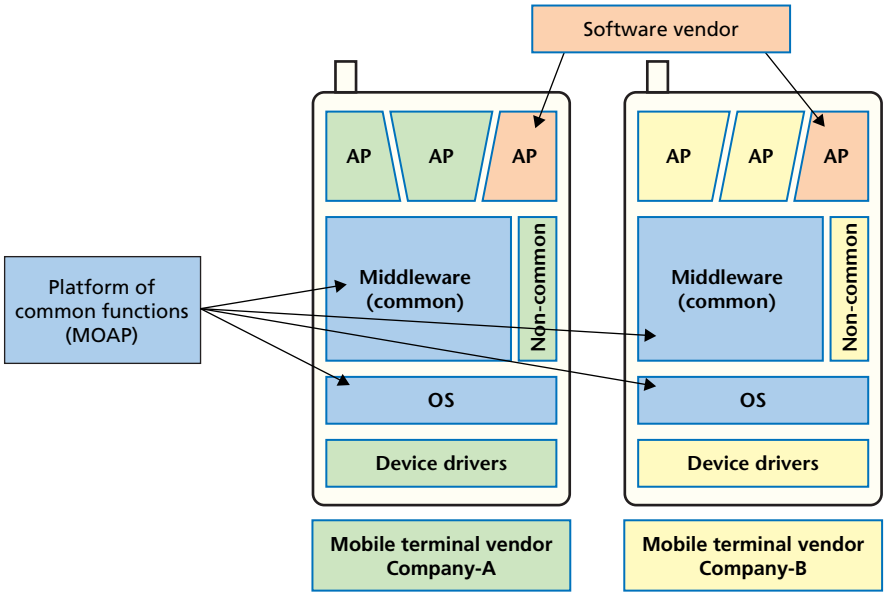


Figure 2 MOAP-based mobile terminal development

3) Provision of a Development Environment

DoCoMo provides a PC-based mobile terminal emulator[*12] for testing the operation of developed applications. This emulator enables mobile terminal vendors to proceed with software development in parallel with hardware development, and enables software vendors to develop applications independently.

3. Extension of Platform Functions

3.1 Basic Functions

The following extensions have been made in relation to basic functions and performance that should be provided by a platform independent of services.

1) OS Extensions

The MOAP(S) software platform supports Symbian OS v8.1b, the latest version of this OS incorporating a multithread real-time kernel[*13]. This guarantees real-time processing while supporting multi-thread processing. It enables testing to be performed on an emulator under a scheduler and application environment the same as that of actual mobile-terminal hardware thereby shortening the testing process.

2) Improved UI (User Interface) Framework

With MOAP, the Look-And-Feel[*14] of a mobile terminal has been enriched from the beginning of the development. The line-up of UI components provided by MOAP has a wide variety. Having functions characteristic of FOMA, these components are useful in application development.

The UI in MOAP has been recently improved by raising the extendibility of various UI components such as the standby screen and soft keys (for example, a transparent-background option and image-pasting function have been added) so as to make it easier for the features of specific mobile terminal vendors to be manifested in application development. Functions in processes associated with user operations like pressing a button (event processes) have also been enhanced, and elements associated with color, shape, arrangement, etc., have been given more flexibility to diversify mobile terminal UIs, and to give applications a greater range of visual expression. And to support a variety of mobile terminal styles, an extensive set of UI-related functions has been developed to support unique event processes such as those for a pointing device.

In addition to functions, performance has been improved. Processes have been analyzed in detail and modified for optimization purposes wherever possible. For example, suppressing unnecessary fill operations in background painting when draw-

*7 RTOS: An OS equipped with functions for performing real-time processing. It is used in embedded equipment such as Personal Digital Assistant (PDA) and home appliances that incorporate a CPU and software for specific applications.

*8 Symbian OS™: An OS for mobile terminals developed and licensed by Symbian Ltd. (UK). Symbian OS and other Symbian-related marks and logos are trademarks or registered trademarks of Symbian Ltd.

*9 Linux OS: An open-source Unix-type OS that can be freely redistributed under GNU Public License (GPL).

*10 API: An interface allowing upper-level software to use functions provided by the OS, middleware, etc.

ing a window enables the time required for switching screens to be shortened.

3) Testing Tool Extensions

We have developed a testing tool that can automatically run applications according to specific scenarios. This continuous testing tool automatically begins an emulation process for that scenario and outputting execution results and an operation log in text-file format (**Figure 3**), when inputting a test scenario consisting of key operations and an execution environment file storing attributes of the target terminal. With this tool, identical tests can be repeated efficiently, and screenshots of the emulator during scenario execution can be saved as image files.

3.2 Support to New Services

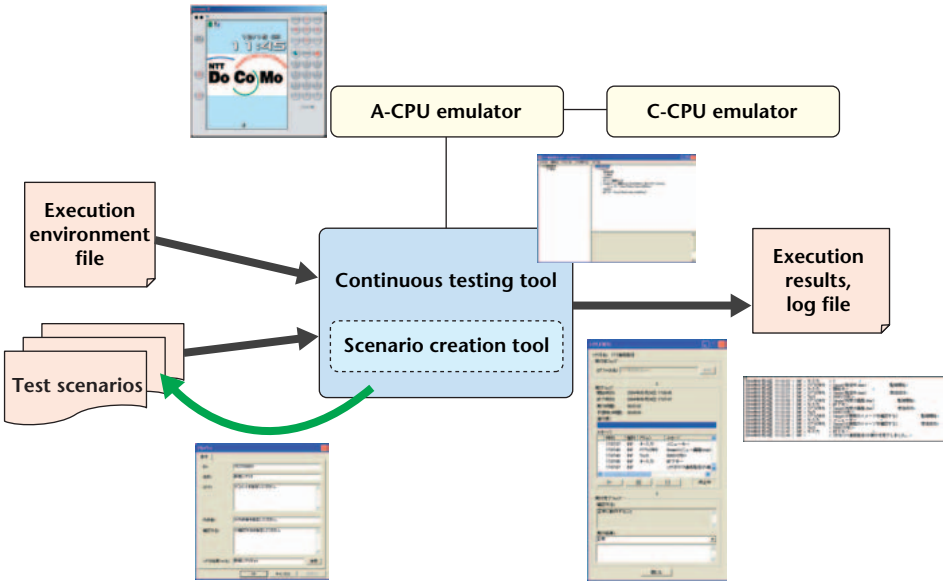
Various new services are being supported for execution on the 902i series of FOMA terminals as described below.

1) PushTalk

The 902i series of FOMA terminals is the first to provide DoCoMo's PushTalk service that enables terminals to be used like walkie-talkies among a group of users. This service is based on standards and specifications established by 3rd Generation Partnership Project (3GPP), Open Mobile Alliance (OMA)[*15] and other organizations, and is supported in MOAP. Here, we first had to support the Short Message Service (SMS)[*16] Push function in order to realize Push-to-talk over Cellular (PoC)[*17] incoming-call notification. We then added an IP-address management function and an i-mode connection process for use during POC calls, and extended the PICT[*18] display function. We also provided an interface to incorporate and operate the PoC engine.

2) i-channel

The i-channel service, which DoCoMo actually began supporting with the 701i series of terminals, enables users to receive news on favorite topics as telop text displayed on the standby screen. This service has been achieved by adding new functions to MOAP to receive and save i-channel messages from the i-mode server at the mobile terminal. Specifically, we extended the Multipurpose Internet Mail Extension (MIME)[*19] header to support i-channel messages and extended the API to save the messages to the file system and enable them to be referenced. We have also added new UI components to support the display of telop text on the standby screen, which is a feature of



Application Central Processing Unit (A-CPU): Generic name for the CPU in charge of application processing in a mobile terminal.
Communication Central Processing Unit (C-CPU): Generic name for the CPU in charge of communications and call control processing in a mobile terminal.

Figure 3 Continuous testing tool

*11 Application framework: Functions controlling application execution state such as application startup/termination procedures, linking between applications and exclusive operations, including a common user interface for use by applications and procedures for displaying that interface.

*12 Emulator: An environment for simulating the hardware and software operations of a mobile terminal on a computer to make application software running on the emulator run the same as on a mobile terminal.

*13 Multithread real-time kernel: A kernel that can perform real-time processing while executing multiple processes simultaneously. "Kernel" means the core section of an OS that runs basic OS functions. This capability is already supported in MOAP(L).

*14 Look-And-Feel: The appearance of a terminal and the sensation of operating it.

the i-channel service.

3) International Roaming

To support the GSM Subscriber Identity Module (SIM)[*20] while in chip-roaming-in mode, we have added a process to the boot sequence[*21] to set the language to be used on the mobile terminal according to information stored in the SIM card (MOAP(S) only).

4) Independent Control of Circuit-switching and Packet-switching

At times of natural disasters or other crisis, it is desirable that circuit-switching and packet-switching be separated, and that traffic volumes for each be restricted as a means of access control. To support this service, we added state-management functions for call control and system data. These include functions for displaying messages on mobile terminals for each type of restricted call, and functions for controlling the making of telephone calls and the activating of the videophone application according to the current state of access control.

5) Audiophone/Videophone in-call Switching Function

This function enables a user to switch directly to a videophone call while in the middle of an ordinary voice call without disconnecting and vice versa. In this process, MOAP first receives an application switching request generated by user actions on the terminal. It then sends an inquiry to the network via the communication-control section to verify if bearer switching is available and returns the reply to the application. Here, MOAP handles the complicated sequences required for this inquiry while upper-level interface sequences are left for the application to handle. This simplifies processing for the application.

4. Conclusion

All terminals in the FOMA 902iS series use the MOAP software platform. Compared to FOMA terminals sold in the past without MOAP, these mobile terminals feature shortened development times and a higher level of quality, demonstrating the beneficial effect of adopting MOAP. For the future, an even greater effect in the development of mobile terminal software is expected, as the use of MOAP expands to the 70x series of terminals and other models.

To support the trend toward great diversification in FOMA terminal functions and more advanced services, DoCoMo will continue in its efforts to enhance common functions, provide new functions, and improve basic performance.

REFERENCES

[1] Tsuji et al.: ""MOAP," Software Platform for FOMA Terminals," NTT DoCoMo Technical Journal, Vol. 7, No. 1, pp. 40-43, Jun. 2005.

*15 OMA: A industry forum for standardizing enabling technology for services and applications in mobile communications and for ensuring interoperability.
*16 SMS: Service for sending and receiving short text-based messages mainly between mobile terminals. It can also be used for sending and receiving control signals for mobile terminals.
*17 PoC: A calling service for mobile terminals on the IP network; standardization is progressing at OMA and other organizations.
*18 PICT: A pictogram. Icons and other forms of picture display, rather than letters. Pictograms such as i-mode and antenna symbols are used in DoCoMo's mobile terminals.
*19 MIME: A standard specifying methods for handling different types of data as used in mail and other applications (RFC2045-49).
*20 SIM: A smart card storing the telephone number and other data of a mobile phone operator subscriber. Used mainly in GSM mobile terminals. The User Identity Module (UIM) smart card used in FOMA terminals is a functional extension of SIM.
*21 Boot sequence: The software instructions that are executed after turning the mobile terminal ON up until the standby screen is displayed.