Mass Customization | Peripheral Linking | Mobile Terminal Management

Technology Reports

Proposal for User-customizable Mobile Terminal

In an effort to meet the diversifying needs of customers, we have created a prototype mobile terminal which allows user customization of hardware functionality. The prototype incorporates a function to physically join new hardware devices to the mobile terminal, allowing users to customize the device in an intuitive and easy to understand way.

Research Laboratories
Akira Kinno
Masaji Katagiri
Kazuhisa Sekine
Satoshi Muto
Takashi Yoshikawa
Takayuki Tamura
DOCOMO Communications Laboratories Europe GmbH
Makoto Hamatsu

1. Introduction

The needs of mobile phone users continue to diversify rapidly, and technology is advancing quickly in response to this. As a result, mobile terminals are providing many different functions and becoming an important part of users' daily lives.

For a given user, the ideal mobile terminal has the correct balance of various elements, including required functions and no more, portability, ease of use, price and design. Most current mobile terminals have features that most users want, together with excellent usability and pricing in an all-in-one type.

However, as lifestyles and values diversify, users' needs are becoming more fragmented. In this sort of market environment, it is important to study ways to provide the ideal mobile terminal to individual users in the future.

Functions to meet the diversifying needs of users at the individual level have so far been classed as so-called long tail[*1] products. Providing all of this functionality built into an all-in-one mobile terminal is not practical from various perspectives, such as price. One way to address this is to introduce individual concept models in a variety of segments through the small-volume production of many products, but to use those functions, users would have to purchase additional mobile terminals.

In fact, users are already purchasing additional all-in-one-type terminals just to use a specific feature. Examples of functions prompting additional purchases are Global Positioning System (GPS), One Seg and high-resolution cameras. Purchasing a new all-in-one terminal for a new function is good, but many terminals that are still-usable are no longer being used, which presents an issue from an ecological perspective.

Considering these issues, there is a need for a new mobile terminal concept able to provide flexibly for diversifying individual needs, distinct from the current all-in-one type. Accordingly, the authors have devised a mobile terminal concept and developed a prototype terminal that allows users to combine hardware functionality according to their wishes.

This article describes the concept of a mobile terminal that can be physically joined to other hardware devices, and describes technical issues that arose during development of a prototype and user opinions obtained through demonstration of the prototype.

*1 **Long tail**: A category of product with low demand and low sales.

2. Concept

The idea of extending functionality exists in the PC field, but users must manage extended functionality themselves. This management includes various process required to use the extended functionality, such as installing and configuring applications, installing device drivers[*2] and resolving conflicts with other device drivers. However, mobile terminals are devices with a broad user base and anyone must be able to use them, without requiring specialized knowledge. Thus, to realize a mobile terminal with extendable functionality, we concentrated on making extended functions intuitive and easy to use.

2.1 Extended Functionality with the Mobile Terminal as the Core Module

A common approach used to develop devices requiring a communications function has been to provide the mobile communications function in a module that is able to connect to various types of device [1]. A multi-purpose mobile terminal capable of meeting users' needs would be possible if such a communications module could be combined flexibly with a variety of hardware.

However, simply providing the communications module to users would present significant issues for most users, requiring them to perform all procedures related to hardware configuration (e.g., downloading required software, configuration, etc.).

So, rather than adding a communications function to other hardware, we adopted the concept of joining extended-function hardware to the mobile terminal itself. The mobile terminal can then provide a software platform able to support other software in addition to providing the communications function. Also, its ability to link to the network can allow it to perform configuration procedures for the hardware so that easy-to-use extended functionality can be provided all users, with no need for specialized knowledge.

2.2 Extending Functionality through Hardware

Even with earlier mobile terminals, it has been possible to extend functionality through software. Compared to hardware extensions, software extensions are less limited by the device, and it is much easier to manage maintenance of these extensions. On the other hand, if the existing mobile terminal hardware is inadequate, software is not an option. Examples include providing a larger screen for viewing video, or new, analog input methods, such as with a musical instrument.

Furthermore, extending functionality through hardware provides a concrete expression of the new function, so it can be more intuitive and easy-to-use than new software functions, which may involve selecting new functions from a menu on a small screen.

For these reasons, we focused on the concept of adding hardware to provide extended functionality that anyone can use easily.

2.3 Physically Joining Components

As with extending functionality on a PC, hardware devices can be connected using a wireless connection such as Bluetooth®[*3], a wired connection such as Universal Serial Bus (USB), or a physical connection such as a PC card[*4].

For wireless connections, no connector on the core module is needed, which is an important benefit for embedded devices that must save space. On the other hand, it presents several usability issues including complex configuration to connect the device, and once connected, the connection cannot be confirmed by looking at it directly, which is non-intuitive. There are also hardware issues, such as the fact that both devices will require a battery.

In contrast to wireless connections, cable connections require no configuration, can be confirmed visually, and can also provide power to the device. However, using cables with mobile terminals can be inconvenient, as cables tend to get tangled. Operating the terminal with the extension device connected can also occupy both hands, which is inconvenient.

If the devices are physically joined, the type of connector is restricting, but as with cables, a connection that is

*2 **Device driver**: A software component which provides an abstract interface for control of a particular type of hardware on a mobile terminal, used by the application software.

*3 **Bluetooth®**: A short-range wireless communication specification for wireless connection of mobile terminals, notebook computers, PDAs, and other portable terminals. Bluetooth is a registered trademark of Bluetooth SIG Inc. in the United States.

*4 **PC card**: An extended function card for PCs.

more-intuitive and easier-to-use than wireless is possible. Having the core module and hardware device joined into a single unit also makes for easier and more convenient mobile use.

For these reasons, we placed a priority on intuitive ease-of-use, and settled on the concept of physically joining components.

3. Prototype Development

In order to identify technical issues in a mobile terminal with interchangeable hardware functions, we developed a prototype based on the three concept points described in the preceding chapter. The prototype consists of a base unit that is essentially a mobile terminal, and extension units that can be combined with the base unit (**Figure 1**). The base unit can be combined with various extension units and can link with various network services related to the extended functionality.

3.1 Base Unit

1) Hardware Functionality

Comparing the base unit with a conventional mobile terminal, it must provide a connector for extension units. We also wanted to ensure the devices were convenient for mobile use, which was seen as a benefit of physically joining them together. Thus, a major feature of the prototype is that it provides two connectors, one on each of the top and bottom (**Photo 1**). Providing two connectors, rather than one, yields even more expandability by allowing for combinations of extension units. It can also increase convenience of the extended device for mobile use.

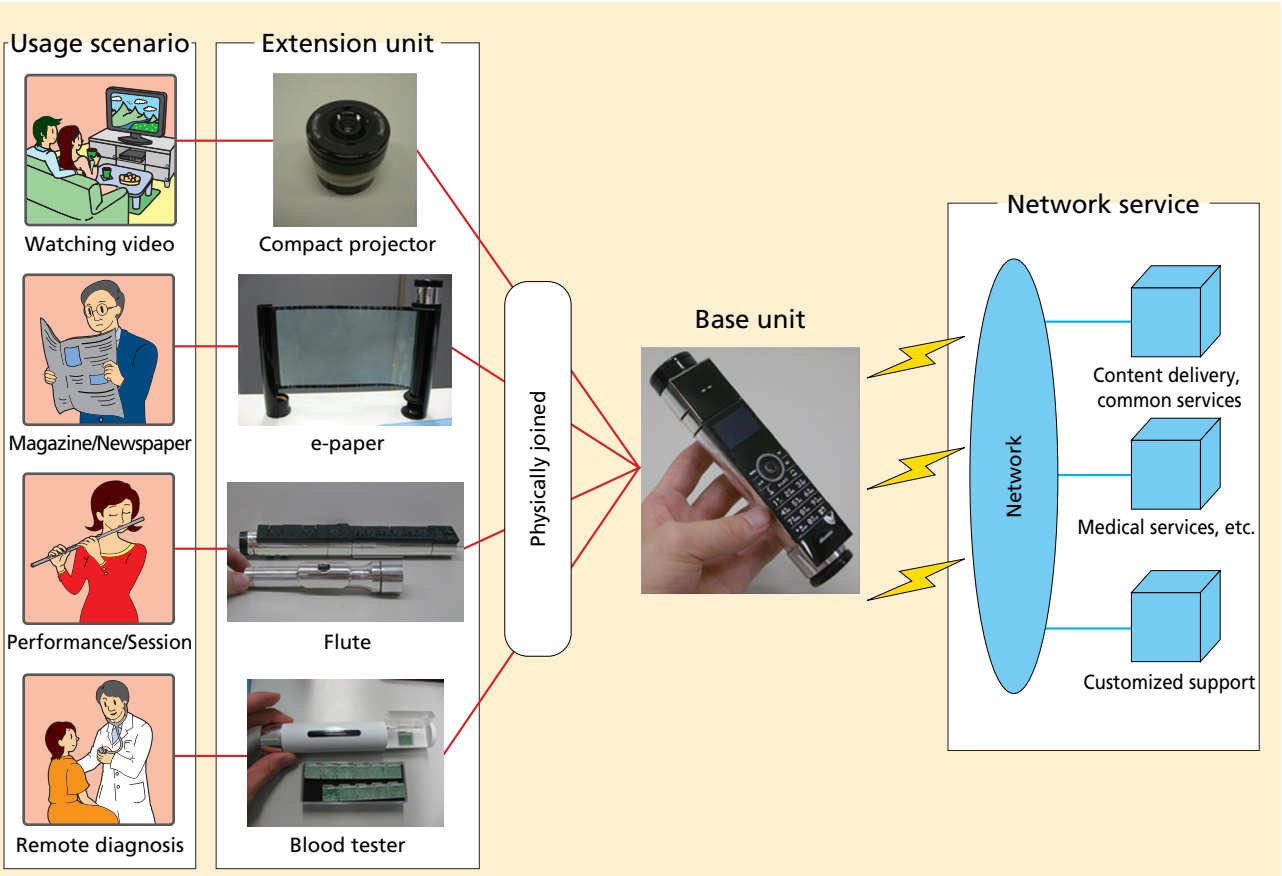
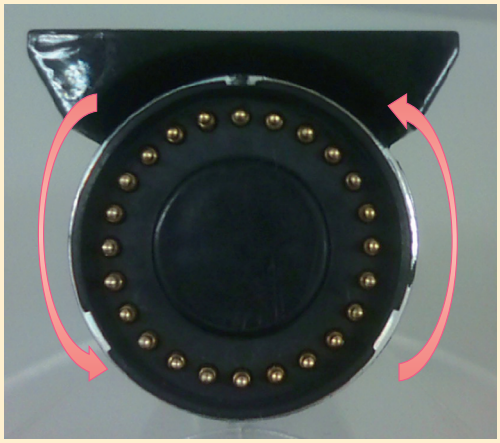


Figure 1 Prototype mobile terminal with customizable hardware functions

(a) Normal state

(b) With large display



(c) Connector

(d) Connected to extension unit (Flute unit)

Photo 1 Base unit

The connectors share a common format, so either can accommodate any of the extension units. Each connector has 21 pins, including audio, display, serial communication, power supply, and USB and each extension unit provides the pins required for its function. The USB pins can be used to identify the type of each extension unit. The connector also provides a screw-in mechanism to maintain good contact with pin connections while the unit is connected.

Also, the base unit was provided with a large screen on the expectation it will be used in scenarios similar to current mobile terminals (Web browsing, e-mail viewing and editing, etc.). Also, the screen can be folded away so the entire unit is not too large when not in use (Photo 1(b)).

2) Software Functionality

It was assumed that the base unit would not be able to have general device drivers pre-installed. For PC Operating Systems (OSs), the OS has access to use large amounts of storage on the PC and can provide many pre-installed general device drivers, so that when a new device is connected to it, one of the general device drivers can be used to provide access to the device. However, a mobile terminal has limited storage space, so the OS cannot provide such general device drivers. When a new extension unit is connected, the base unit must obtain a device driver from an external source.

Possible sources for the mobile ter-

minal to obtain the required device driver include the network, or the extension unit itself. If the driver is obtained from the extension unit, the extension unit must provide drivers for all supported mobile terminal types, since drivers are OS dependant. This suggests that obtaining the driver from the network is more desirable.

Another issue when installing device drivers is that any conflicts with other hardware devices must be resolved. For example, the Windows®*5 XP SP2 device manager displays all hardware devices installed in the PC, allowing them to be enabled, disabled and otherwise configured. The mobile terminal will require a management function similar to this, and this management must be done without imposing on the user. As an example, a device management technology called Open Mobile Alliance (OMA)*6-Device Management*7 and being standardized by the OMA provides a management server on the network and can check the status of software installed on the mobile terminal and perform various configuration tasks. Applying this technology would allow the mobile terminal to handle device-manager functions through its link to the network.

It is desirable that applications suited to the connected extension unit be launched automatically, and that the mobile terminal maintains this association between extension units and applications. However, when connecting general hardware functions such as a display or projector, there may be several possible relevant applications. In such cases, the possibilities should be presented as a list, prompting the user to select the desired application.

Considering these requirements, we included the automatic-launching part of the self-configuration function in the prototype. The base unit is built on the base of a Linux platform and provides an external-device control module to control connection and communication with the various hardware devices, a software control module to controls installation, configuring and launching of applications related to installed extension units, and a communications module which the software control module uses to connect to the network and download the appropriate applications (**Figure 2**). These elements are used to provide a function that identifies the type of extension unit and automatically launches the appropriate applications.

3.2 Extension Units

Various model extension units enabling new usage scenarios were created, by either improving the performance of functions already in the base unit, or by providing entirely new functionality. Of the former, a compact projector unit and an e-paper unit were created, and of the latter, a simple blood-testing unit for remote health monitoring and a musical instrument unit for performance over the network were created (Fig. 1). Each extension unit maintains extension-type information, and has a function to transmit the

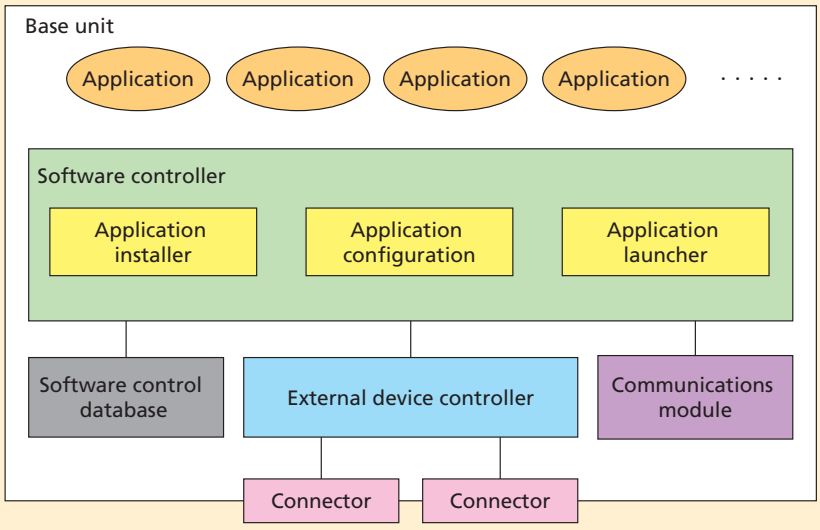


Figure 2 Base unit software architecture

*5 **Windows®**: A registered trademark of Microsoft Corp. in the United States and other countries.
*6 **OMA**: An industry standardization organization that aims to standardize service and application technology and achieve interoperability in mobile communications.
*7 **OMA-Device Management**: An OMA specification regarding a method for remotely managing clients (mainly mobile terminals) from a server.

relevant information to the base unit through the USB pins of the connector.

4. Evaluation of the Prototype

In order to achieve intuitive ease-of-use with the extended functions, the prototype provides a connector which physically joins the extension unit to the main unit, and when the extension unit is connected, the appropriate application starts automatically. To identify technical issues from a perspective of ease-of-use, the authors performed a qualitative evaluation of the prototype using a fixed framework [2]. The results are shown in **Table 1**. The benefits of physically joining the components showed in the positive results for learnability and memorability, but several issues related to efficiency arose, as described below.

The first issue was that connecting extension units consumed too much time. The connector used on the prototype involved holding the connector pins together tightly while operating a screw-in mechanism with one hand. Connection also required a specific direction due to the specialized pin positioning. For these reasons, extra time was required to tighten the screw while ensuring the components were positioned properly.

The second issue was that the design and size of the extension units was limited by the base unit. The prototype was implemented so that the base unit could be used by itself in conventional mobile terminal scenarios, with a display and key pad and of appropriate size. Because we had to consider hardware size and design in terms of ease-of-use while extension units were connected. In order to realize a variety of scenarios using extension units, extensive study of the base unit from various perspectives will be necessary to determine the best size, design, and hardware functionality.

5. Issues and Discussion Related to Physically Joining Components

5.1 Hardware Issues

Technical issues related to hardware and software are shown in **Table 2**. In terms of ease-of-use, it is important that connecting and releasing the connector extension units be easy, but it is also important to show intuitively, whether an extension unit can be connected or not based on the current state of the base unit. With the prototype, multiple extension units could be combined, but some combinations were not appropri-

Table 1 Prototype usability evaluation results

Item	Result	Summary
Learnability	○	The hardware need only be connected. It is very easy to learn how to use it, up to the point where the application has launched.
Usability (efficiency)	×	The screw-in mechanism must be operated while checking pin-alignment when connecting an extension unit. Also, two seconds elapsed before the application started, even though only application launching was implemented.
Memorability	○	The hardware need only be connected. It is easy to remember how to use it, up to the point where the application has launched.
Error infrequency	×	Pin position must be checked, so if a mistake was made, the required application does not start.
Subjective satisfaction	△	Effectiveness and subjective satisfaction were low due to the number of errors. Also, extension units are limited by the size of the base unit.

Table 2 Current issues

Issue type		Summary
Hardware	Connecter	Ease of connecting/releasing, maintaining a good contact, intuitive indication of whether connection is possible, etc.
	Unit	Appropriate size and weight, maintaining an appropriate weight balance, managing supply of power to extension units, etc.
Software	Self-configuration	Speeding up expansion processing (automatic application/device driver installation, configuration, auto-launch of application), presenting messages to users with appropriate timing and frequency, conflict resolution when installing applications and device drivers, etc.
Security	Handling in case of loss	Remote initialization, discovery, etc. for extension units to prevent unauthorized use.

ate (incompatible or not useful), and a way of indicating this to the user is needed. Further, it would be preferable to notify the user that a connection is inappropriate before the user attempts to connect the device. One possibility for identifying the type of extension unit before connecting it to the base unit would be to use wireless technology near the connector to obtain unit-type information. In particular, passive radio IC tags[*8] are hardware devices requiring no battery and are a very promising option.

5.2 Software Issues

As we described in Section 3.1 Software Functionality, one of the requirements of the base unit was that it be self-configuring, so that when an extension unit is connected, all processes required to prepare related applications for use are done automatically, including installing and configuring device drivers and applications and launching the applications. Of all of the self-configuration steps, only the automatic launching step was implemented on the prototype, and automatic launching of applications required about two seconds from when an extension unit was connected. Self-configuration also includes automatic installation and configuration, so processing time will likely be longer in total. Thus, it will be necessary to study ways to reduce automatic configuration time. For example, self-configuration processing could begin even before the device is actually connected if the base unit could detect that an extension unit could be connected soon, such as by using passive short-range radio communication.

5.3 Security Issues

In the past, security services have been offered, such as one able to clear and initialize a mobile terminal via the network if it was lost. These services dealt with mobile terminals, and could not be applied to extension units, which may also handle sensitive private information, such as with the health management unit. Assuming that extension units do handle such sensitive information, being able to respond appropriately if the device is lost is a significant issue. One possible solution would be to tie the extension unit to a single mobile terminal (or a user) so that it cannot be operated using other terminals (or terminals used by other users), and so it initializes itself if it is connected to a third-party terminal.

5.4 Issues Arising from User Opinions

Many users stated that current mobile terminals have functions that are too complex, functions that they do not use, and that they did not like services that are dependent on specific terminal models (**Table 3**). This suggests that meeting needs at an individual level is an important issue in the mobile market, and there is a need for mobile terminals with extendable functions. On the other hand, some users indicated that having to connect extension units to the base unit was inconvenient. As mentioned above, there is a need for further study of how to enable rearranging components with as little intervention as possible from the user.

Table 3 Main user comments from demonstrations

Attribute	Opinion type	Opinion summary
Positive	Meeting diversifying needs through customization	Current mobile terminals have many unused functions. This dissatisfaction can be resolved by allowing rearrangements.
		Current dissatisfaction with device-dependant services can be eliminated.
	Extension through hardware	Finding software is difficult. "Just working" when connected is easy and good.
Negative	Core module is a mobile terminal	It would be better if the devices controlled the mobile terminal functions, rather than the other way around.
	Extension through hardware	Current mobile terminals are already difficult. Cannot keep up with extensions.
		Seems like the overall price will increase.
	Physical joining	Having to physically join devices is inconvenient.
		Wireless would be unconstrained by hardware.

*8 **Passive radio IC tag**: One type of radio IC tag with ID information embedded in the IC chip, which doesn't have internal power to operate, but uses power from the reader/writer device signal.

6. Conclusion

In order to meet the diversifying needs of mobile terminal users, we have conceived a mobile terminal with customizable hardware functionality and created a prototype. By allowing customizable hardware, selection of functions and selection of the terminal are completely decoupled, and users no longer need to select a terminal based on functionality. Also, users no longer need to buy a new terminal just to use a particular feature, so terminals can be used for longer periods of time. Finally, new functionality can open up use of mobile terminals in new fields where they have not been used before.

For the prototype, we used a concept involving a mobile terminal as the core module, and hardware extensions that are physically joined to the mobile terminal. This provided an intuitive and easy-to-use format for extending functionality. Building the prototype revealed several technical issues, including possible improvements for the connector, the need for a high-speed, highly-reliable self-configuration function, and the need for security measures for hardware devices.

In the future, we will study solutions to some of these technical issues. Also, towards creating a practical product, we will solve technical issues, gain cooperation from hardware vendors, standardize the connector interface, and create a framework to encourage development of a variety of extension units, as well as studying business issues related to such a product.

REFERENCES

[1] I. Kurosawa: "A Linux-based mobile phone using a W-SIM PHS Communications module," Interface, pp. 121-128, Dec. 2006 (In Japanese).

[2] M. Kurosu, M. Ito and N. Tokitsu: "Introduction to user engineering—A practical approach to ISO13407 Usability considerations," Kyouritsu Publishing, 1999 (In Japanese).