Development Reports

Software Update System Using Wireless Communication

Seiji Hoshi, Akihiro Ichinose, Yasuhiro Nose,
Atsushi Hosokawa, Masato Takeichi and Eiji Yano

A software update system using wireless communication had been developed for the purpose of improving user convenience in cases where bugs occur in the software running on a mobile terminal. This system is adopted from the PDC 252i series, which was released in September 2003.
This article discusses the configuration of this system as well as implementation technologies for the server system, MSR, and mobile terminals equipped with this function.

1. Introduction

Lately, the number of cases where software bugs are detected is increasing in new models after the launch in commercial market, primarily due to the expanding size of software caused by the increasing number of functions implemented in them [1]. For this reason, DoCoMo developed and deployed a system that allows updating software at customer service desks such as DoCoMo shops in parallel with efforts to improve the software quality. This system, however, requires customers to go through the trouble of visiting the customer service desks and thus lacks convenience. To improve the convenience for the users, we have developed a mobile terminal software update system that allows remote downloading of software by means of wireless communication.

This system allows the users themselves to update the software as needed. It also has the advantage that the cost involved in replacing mobile terminals borne by both mobile terminal manufacturers and DoCoMo can be reduced.

This system supports both the Personal Digital Cellular (PDC) and Freedom Of Mobile multimedia Access (FOMA) systems. In terms of usage, it addresses the needs of not only customers using i-mode services, but also customers using DoCoMo's Internet Service Provider (ISP) connection services (services connecting i-mode compatible terminals to ISPs via a packet-based network).

This article describes the requirements related to implementation of this system, and technologies realized in mobile terminals with software update system and Mobile terminal Software Remote distributing system (MSR).

2. Technologies Implementation

2.1 Requirements

The basic purpose of this software update system is to update the mobile terminal software by taking advantage of wireless communication in case a bug occurs. The following requirements can be defined for the system.

1) Minimum expansion of existing network facilities
2) Minimum cost increase of the mobile terminals
3) Independence of the mobile terminal architecture
4) Earlier implementation of the system

2.2 System Overview

Figure 1 shows an overview of the system. If bugs are detected in the mobile terminal software, update data to fix the bugs is created (① in Fig.1) and is delivered to the MSR (② in Fig.1). When the mobile terminal accesses the MSR (③ in Fig. 1), the appropriate update data is downloaded directly into the mobile terminal if an update is necessary (④ in Fig.1). The mobile terminal rewrites the software using the downloaded update data (⑤ in Fig.1). After the software is updated, the mobile terminal functions properly.

2.3 Interfaces between the Mobile Terminal and MSR

HyperText Transfer Protocol (HTTP), which works within i-mode as well, is adopted as the communication protocol between the mobile terminal and MSR. No changes are made to existing networks; in order to shorten the development period, the only system components that have been developed anew are the MSR and the update function in the mobile terminals.

As shown in **Figure 2**, the GET method in HTTP is used to transfer (download) update data. Moreover, the POST method in HTTP is used if it is necessary to send information from the mobile terminal to the MSR. For example, when inquiring whether an update is necessary in Fig.2, the manufacturer name, model name, version information etc. are placed in the Request body of the POST method. In response to this request, the MSR indicates the presence of any update data along with the URL of the update data, if any update data is present, in the Response body of the POST method returned to the mobile terminal.

There are ten interfaces between the mobile terminal and MSR; in addition to inquiry whether an update is necessary, issues such as notification of the completion of update, notification of the interruption of update, reservation setting/changes and cancellation are handled through these interfaces. The MSR controls the update data transfer to the mobile terminal and per-

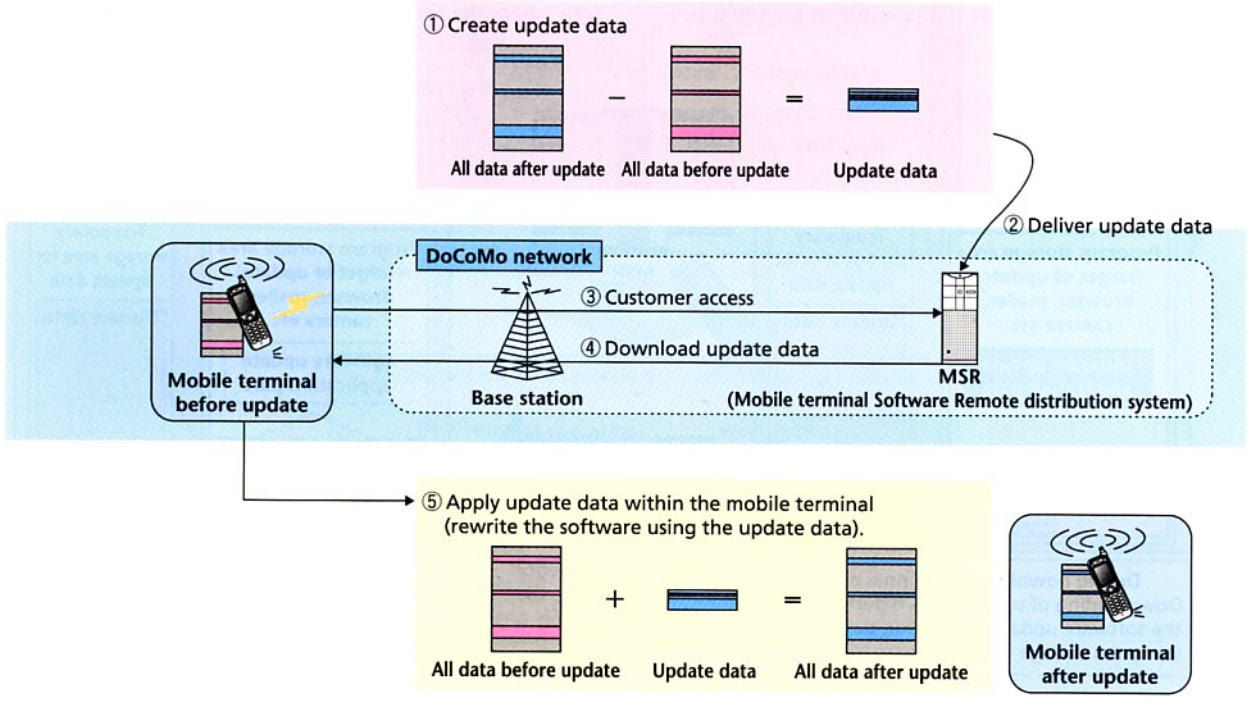

Figure 1 Software update system using wireless communication

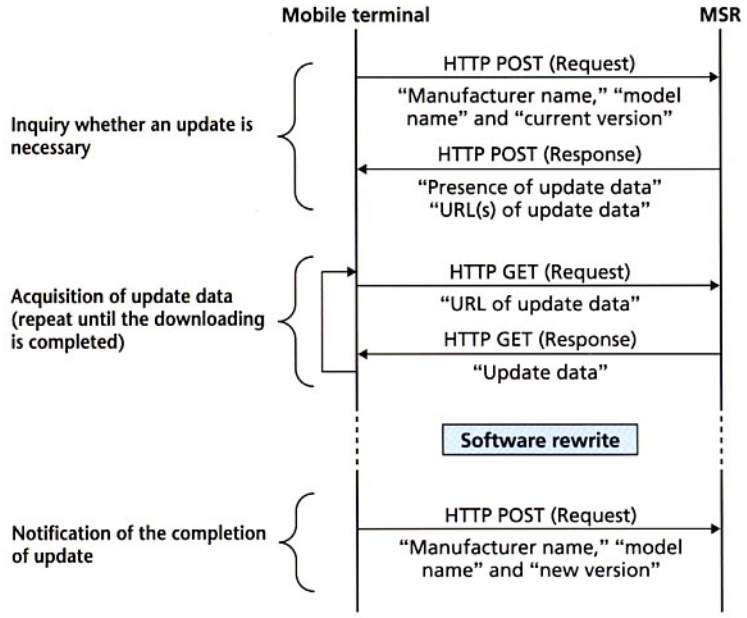

Figure 2 Interfaces specification between the MSR and mobile terminal (excerpt)

forms procedures related to update reservation via the interfaces [2]-[4].

2.4 Implementation Method on Mobile Terminal without Requiring Additional Memory

Since this function is not used on a daily basis, it is necessary to implement it by using a method that does not require additional memory, which would lead to an increase of the mobile terminal cost [2], [3].

When updating software, it is not possible to rewrite a program in memory while the program is being executed. It was thus considered to duplicate all or part of the update target, but this method requires additional memory. For this reason, we have divided the software update function into two parts: a part that handles the display of the mobile terminal and downloading of update data (application part) and a part that performs rewrite of the software (rewrite engine part). As shown in **Figure 3**, this system rewrites the program storage area of the mobile terminal. The application part is stored in the program area and the rewrite engine part is stored in a special area used for starting up the mobile terminal (boot area). The mobile terminal is operated by the application part until downloading of the update file is completed (normal mode). After downloading is completed, the mobile terminal is switched to a mode where it is operated by the rewrite engine part (rewrite mode), and the software is updated. This procedure allows rewriting the target software directly without duplicating code.

In order to update the software, it is required to acquire an area where the downloaded update data can be stored. For this

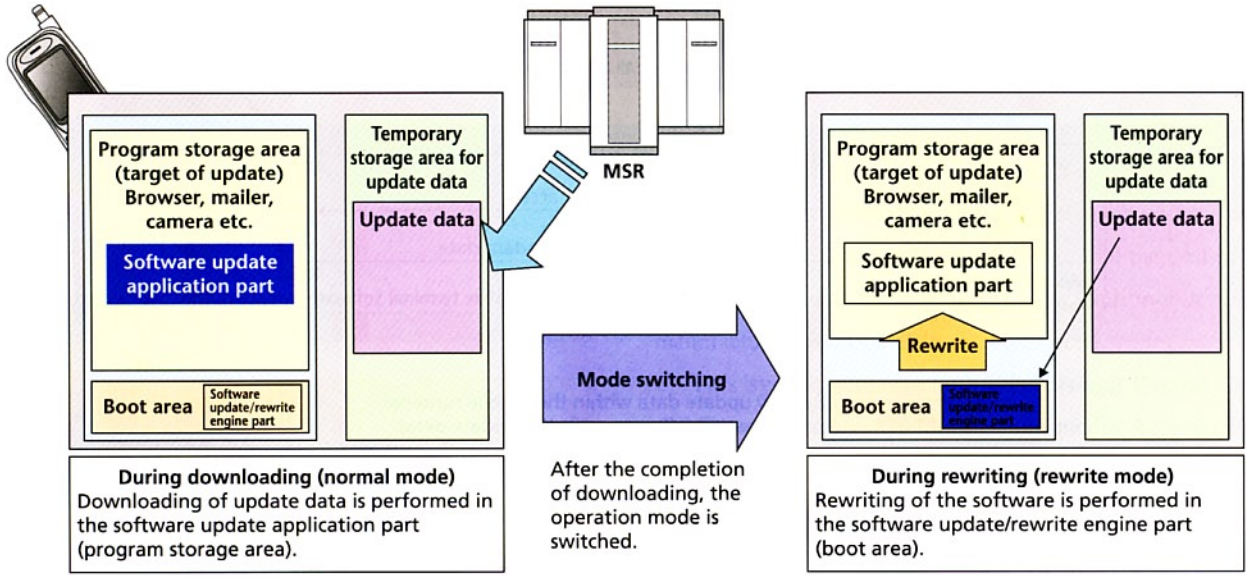

Figure 3 Division into the application and rewrite engine

purpose, we decided to allocate the work memory, which is normally used by applications to store data during it's execution. When another function is in use, for instance Java*, the work memory will generally be occupied for it's execution. To activate the software update function, the operation of those application functions is regulated and the storage area for update data is secured within the work memory (**Figure 4**).

If the power supply is interrupted while the mobile terminal is in the rewrite mode, the software rewrite will fail. Even in this case, the failure can be recovered through the conventional software update system at the customer service desks.

Furthermore, in order to implement this update function, we also reused existing functions as much as possible and tried to reduce the size of the function itself. By doing so, we have managed to implement the software update function without requiring additional memory.

2.5 Elimination of Dependency on Mobile Terminal Architecture

Because the baud rate of wireless communication is lower than that of broadband cable communication and the scale of mobile terminal software is generally increasing, wireless communication is not sufficient to transfer such data as is. For this reason, it is necessary to set the size of update data to be transferred properly. In order to set the proper size of data to be transferred, methods are adopted such as extracting and compressing the difference of data before and after updating the mobile terminal software [5]-[8]. In this case, we took the flexibility of implementation on the mobile terminal into consideration and made the design independent of a specific architecture and its components [2], [3]. Specifically, we set the specifications only for the method of downloading update data and the maximum data size that can be transferred, without defining the update data format and update algorithm. This allows implementation of different data formats and update algorithms according to the mobile terminal architecture, which varies depending on the manufacturer and model. Moreover, this system can be used even if the mobile terminal architecture may evolve in the future.

2.6 Update Data Transfer Control Method of the MSR

It is expected that the number of connections from mobile terminals to a network for the purpose of software update is much smaller compared to the normal number of network con-

* Java: An object-oriented development environment specialized for network programming, proposed by Sun Microsystems in the U.S.

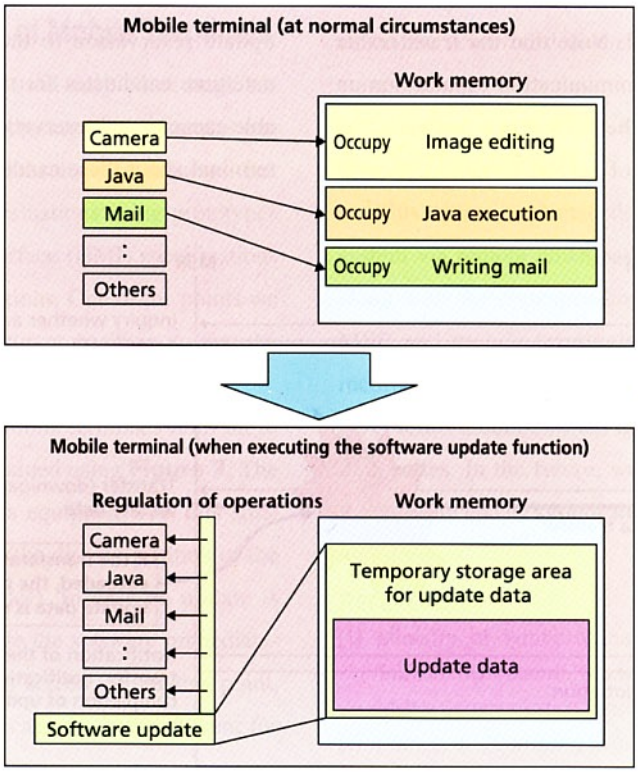

Figure 4 Securing temporary storage area for update data using the work memory

nections made by the entire population of subscribers. On the other hand, depending on the nature of the software bugs to be fixed, the size of update data may become larger than the contents viewed by i-mode browsers. This means that the factor that must be considered in order to avoid exceeding existing network traffic capacity limits, is the amount of data to be transferred rather than the number of connections.

The amount of data to be transferred can be calculated from the number of connection requests and the size of the update data. Thus, it is possible for the MSR to control the amount of data to be transferred by calculating the amount of data flowing into the entire network when a mobile terminal requests a connection to the MSR and allowing data transfer to the mobile terminal only when the amount does not exceed a predetermined transferable capacity.

How to calculate the amount of data flowing into the entire network and the transferable capacity is explained below. The amount of data flowing into the entire network is calculated as follows. The MSR can learn the size of the update data and the average throughput of the communication system (PDC or FOMA) from the type of mobile terminal, which is obtained when a mobile terminal requests a connection. Based on these two pieces of information, the MSR calculates the amount of data to be transferred and the hold time for one mobile terminal per unit time, which in turn can be multiplied by the number of mobile terminals to be connected. Note that the transferable capacity is defined as the free communication capacity on an existing network, obtained as the difference between the installed capacity and the amount of normal communication of the existing network, which changes as time passes. To obtain an estimate amount of normal communication, we acquired and used traffic data in the past to create an approximate model of the amount of communication.

Figure 5 shows an illustration of the update data transfer control method. The MSR stores the transferable capacity along with the update data and related information. The related information includes the size of the update data. When a mobile terminal makes an inquiry whether an update is necessary to the MSR, the MSR checks whether it is necessary and, if that is the case, transfers update data while controlling the amount of data to be transferred [3], [4].

2.7 Update Reservations

Due to the data transfer control discussed in the previous section, downloading of update data may not be permitted immediately when a customer requests software update. Moreover, since a mobile terminal restricts some of the normally used functions while executing the update function, the convenience for the customer may be impaired if it takes too much time to update the software. For this reason, we equipped the system with an update reservation function that allows updating the software automatically on a date and time desired by the customers.

When a mobile terminal notifies that it wishes to make an update reservation to the MSR, the MSR extracts a list of date/time candidates for the reservation based on the transferable capacity and reservation conditions, and notifies the mobile terminal about these candidates. When the customer has select-

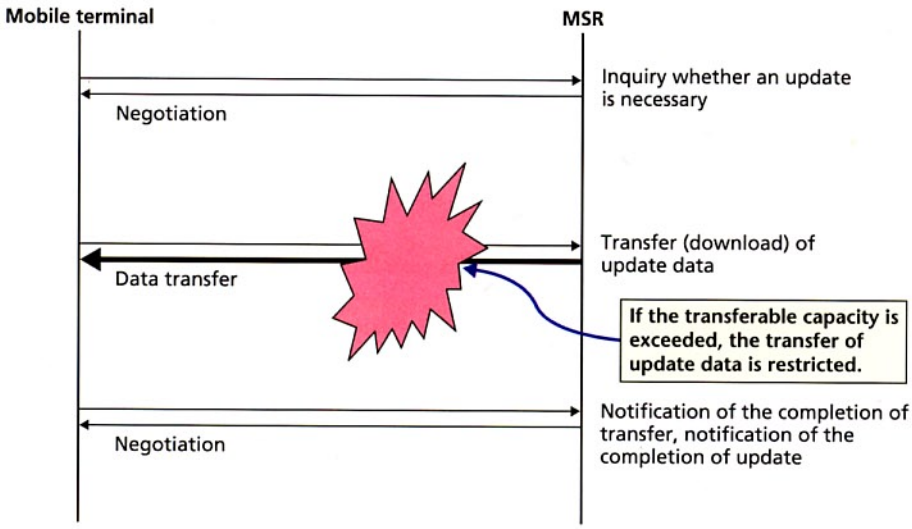

Figure 5 Update data transfer control method of the MSR

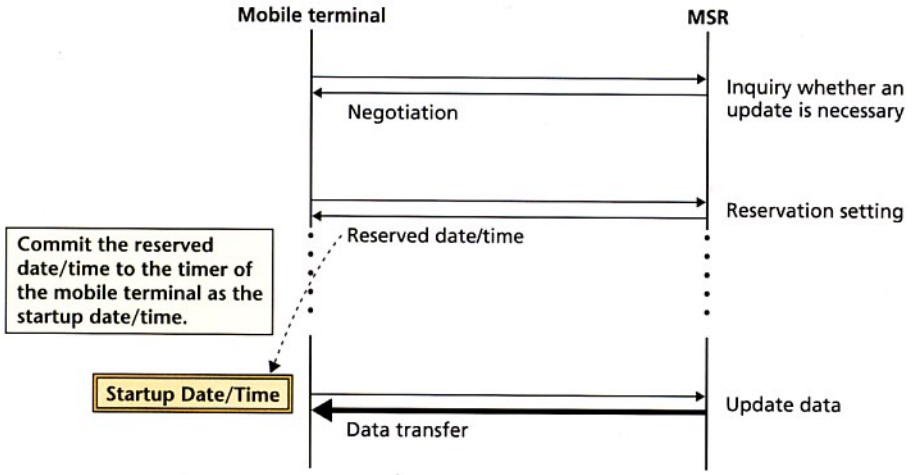

Figure 6 Software update via reservation

ed a date and time from the notified list of reservation candidates, the mobile terminal commits this information to the timer inside the mobile terminal, and starts requesting downloading of update data at the date and time of the reservation (**Figure 6**).

This function not only improves the convenience for the customers, but also allows using existing network facilities effectively by leading customers, who were not allowed to download update data because the transferable capacity was insufficient at the time of request, to time slots where the traffic is light [3], [4].

3. Operation Specifications of Mobile Terminals

The software update function is used only when bugs are detected in the software. It is thus necessary that customers can perform software update easily without fail when required. For this reason, we conducted subject evaluations using prototypes and formulated Human Machine Interface (HMI) specifications that reflect the results of these evaluations. One of the points we reflected was to specify the default cursor position, so that the update can be performed via operation of one specific key.

The operation specifications of mobile terminals relevant to the software update function are explained using **Figure 7**. The menu screen of the mobile terminals equipped with this software has a "Software Update" menu (the display position of the menu varies depending on the model). In case an update is required, screen ③ appears. To update the software immediately, the user selects "Now Update" in screen ③. At this point, screen ⑧ appears if the transferable capacity is insufficient for an immediate update.

If the user selects "Reserve" in screen ③ or ⑧, the list of reservation date/time candidates is displayed. Setting the update reservation is completed by selecting either one of the candidates. If the user selects "Others", the user can select any date and time slot s/he prefers. The desired date can be set to up to several days ahead, but dates/time slots where the transferable capacity is estimated to be insufficient cannot be selected. The availability of capacity in different time slots is displayed as "○: Available," "△: Almost full" or "×: Full." The user can confirm the reserved date and time, or "Change" or "Cancel" the reservation by executing the software update function again after setting the reservation. Downloading is automatically started on the reserved date and time, and the software update is completed.

4. Conclusion

This article explained the overview of the software update system via remote downloading using wireless communication along with the corresponding technologies implemented in the MSR and mobile terminals. This function is planned to be incorporated in each of FOMA models as well as subsequent PDC series models, which had been already adopted from PDC 252i series. In the future, we will study further applications of the software update system and its implementation technologies.

REFERENCES

[1] Ministry of Public Management, Home Affairs, Posts and Telecommunications: "Study Group on Safety and Reliability of 3G Mobile Communication Systems," Technical Report, 2001. [In Japanese]

[2] Hoshi et al.: "Correcting Bugs in Mobile Terminal Software Using Radio Communication," IEICE General Conference, B-6-208, 2003. [In Japanese]

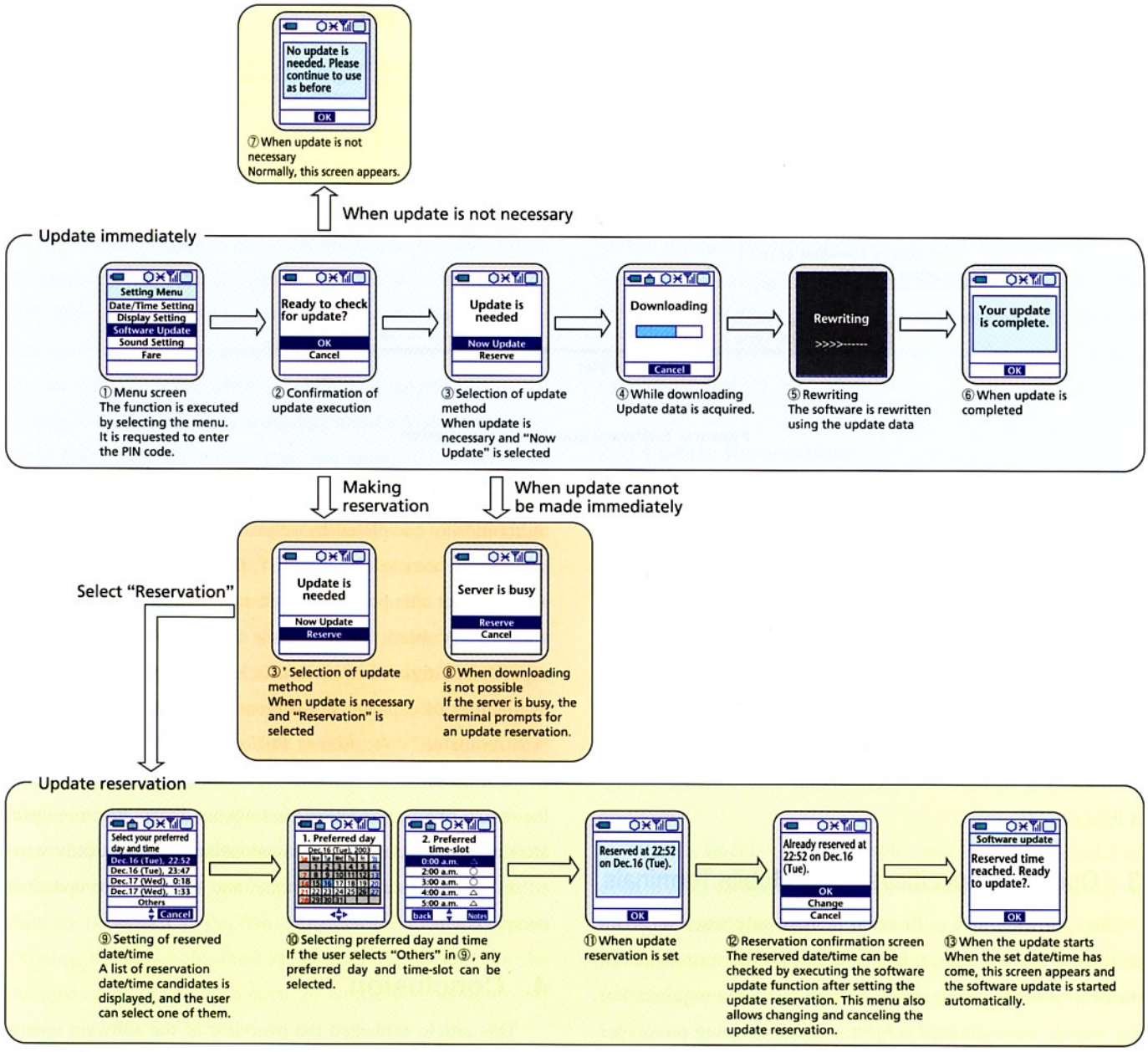

Figure 7 Operation Specifications of Mobile Terminals Relevant to the Software Update Function

[3] Takeichi et al.: "Correcting Bugs in Mobile Terminal Software Using Radio Communication," IEICE Technical Report, Vol. 102, No. 705, 2003 (Technical Committee on Telecommunication Management). [In Japanese]

[4] M. Takeichi, A. Hosokawa, K. Nasu, S. Hoshi, K. Moriyama, T. Takami and K. Terunuma: "Bug Fix of Mobile Terminal Software using Download OTA," The Asia-Pacific Network Operations and Management Symposium (APNOMS) 2003.

[5] A. Tridgell and P. Mackerras: "The rsync algorithm," The Australian National University TR-CS-96-05, 1996.

[6] J. J. Hunt, et al: "Delta Algorithms: An Empirical Analysis," ACM Transactions on Software Engineering and Methodology, 1998.

[7] Kiyohara et al.: "Examination of Update of Mobile Terminal Software," Information Processing Society of Japan MBL22-13, pp. 93–100, 2002. [In Japanese]

[8] Kurihara et al.: "Examination of Data Difference between Different Versions of Mobile Phone Terminal Software," Information Processing Society of Japan DICOMO 2003, 4A-074. [In Japanese]

ABBREVIATIONS

FOMA: Freedom Of Mobile multimedia Access
HMI: Human Machine Interface
HTTP: HyperText Transfer Protocol
ISP: Internet Service Provider
MSR: Mobile terminal Software Remote distributing system
PDC: Personal Digital Cellular