Development of New/Previous Hybrid System with MegaOakIBARSII

MANO Makoto, KIKUCHI Hirohito, YANAGIHARA Keisuke, KURAMOCHI Atsushi

Abstract

MegaOakIBARSII is medical coding package software for use in large as well as in small hospitals. It has been developed as the successor to MegaOakIBARS that is currently being run at about 800 medical institutions. It is indispensable that transition from the previous product to the new one is done smoothly and this paper introduces a method for hybridizing the new and previous systems as a solution to this issue.

Keywords

medical coding, .NET, .NET Framework

1. Introduction

Recent computerization in the field of medicine has resulted in IT systematization of almost 100% of medical coding jobs. These include acceptance of patients, management of medical records, calculation of insurance points and compilation of the receipts (medical insurance reports) to be submitted to the medical insurance examination/payment organizations.

NEC has introduced the medical coding package software "MegaOakIBARS" to about 800 medical institutions in Japan in order to enable them to deal securely with modifications of medical fees and to reform their medical systems.

However, now that 11 years have elapsed since the development of MegaOakIBARS, a large number of revisions are needed following modifications to the medical fee systems. Meeting user requests has increased the number of modules compiled using COBOL, Visual Basic (R) , C and ACCESS. This has tended to result in issues such as an increase in the amount of labor required for carrying out maintenance and installation works.

Therefore, to solve the above issues, enhance those functions that are attractive to users and facilitate linkages with the electronic medical record system MegaOakHR, we have commercialized MegaOakIBARSII (Second) as the next-generation medical coding system running on the .NET Framework.

We proceeded to the development of the new system by setting the mandatory conditions to inherit all of the necessary functions of the previous system in order to smooth the transition of about 800 users of the previous system to the new system. Although we are planning to port all of the functions of the previous system to the NET, we have developed the first version of the new system as a hybrid system utilizing the masters, databases (DB) and modules of the previous system in order to meet the mandatory conditions described above.

In the following sections, we will describe the hybridization that is one of the big features of the new system.

2. Hybridization of Masters and DBs between New/Previous MegaOaks

MegaOakIBARSII uses the database table configuration used with MegaOakIBARS without alteration. It deals thus

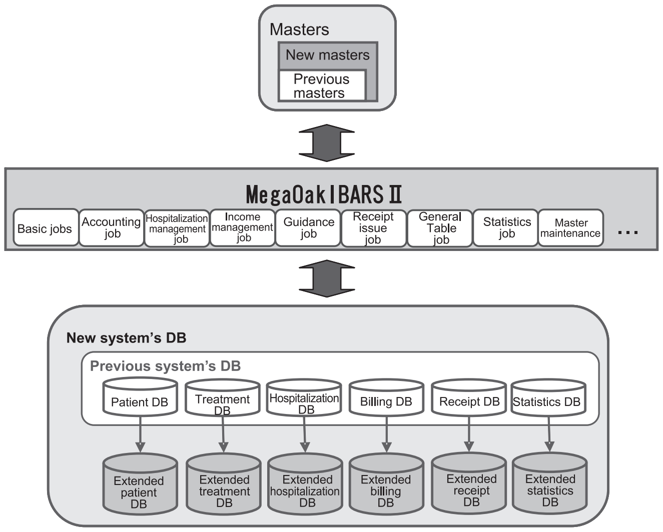

Fig. 1 Image of the Master/DB configuration of MegaOakIBARSII.

with the information required for the new system by preparing extension tables linked to the corresponding tables. Also, with regard to the masters, all we needed to do was to apply partial column extensions and additions to the masters of the previous system.

This procedure has facilitated a smooth transition from the previous system to the new system as well as enabling the utilization of the modules of the previous system.

Fig. 1 shows an image of the master and DB configuration of MegaOakIBARSII.

3. Hybridization of Inter–Job Linkage between New/ Previous MegaOaks

3.1 MegaOakIBARS

(1) Task Management and Display Control

With MegaOakIBARS, each job AP had its appropriate display, which could be activated from a launcher called the task controller.

The job management module manages the activation and exiting of notifications from the job APs and stores information on them in the shared memory.

This architecture makes it possible to implement the function that exits job APs at the end of a control task and that for displaying a specific job in the front position.

(2) Inter-Job Communications

The inter-task communications of MegaOakIBARS is handled mainly by the job management module, which reserves the paths of communications between jobs.

A function for activating a communication destination in case it does not exist is also retained.

When for example, job A wants to send a notification to job B, the following procedure is executed (see **Fig. 2**).

1) The communication source, job A, sends a notification (Destination: Job B. Content: X) to the job management module.

2) If the communication destination, job B is not activated, the job management module re-activates it.

3) The job management module stores the notification information (Destination: Job B. Content: X) in the shared memory.

4) The job management module sends notification to job B.

5) The communication destination, job B receives the notification and acquires its content from the job management module.

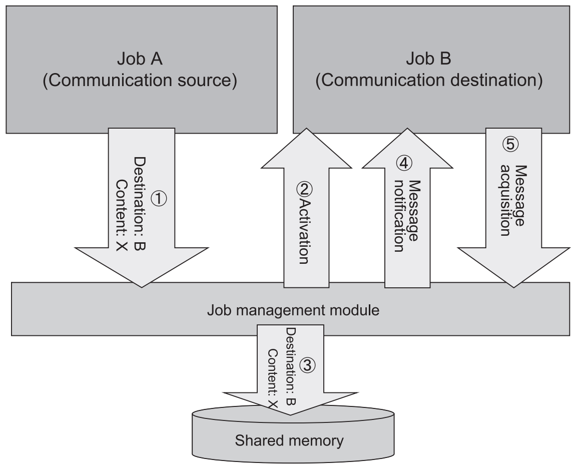

Fig. 2 Inter-job communications of MegaOakIBARS.

3.2 MegaOakIBARSII

(1) Job Management and Display Control

MegaOakIBARSII adopts the MDI (Multiple Document Interface) that shows the job window in the main window (main form). This means that the measures for displaying the currently-used jobs should be shown in the main window (main form).

The current jobs are individual EXE compiled in Visual Basic $^{(R)}$ 6, and those of MegaOak IBARSII are compiled in .NET Framework.

As the main window (main form) has been developed with .NET, it cannot be displayed as it is using MDI.

Therefore we converted the displays of the current jobs into an MDI window using the following method.

1) We prepared the WindowsForm class (hereafter called the proxy form) for use as an MDI sub-window with .NET.

2) The Proxy form 1) manager is used to manage the activation, display and to exit a current job process.

The proxy form should manage, monitor and arrange the following items of information.

- Job window of current job (WindowHandle)
- Process status of current job
- Display status of current job

The above method has allowed the MegaOakIBARSII main format to handle the current jobs of MegaOakIBARS in the same way as newly created job forms.

As a result, the MDI window is possible with minimum modification of the current jobs (see **Fig. 3**).

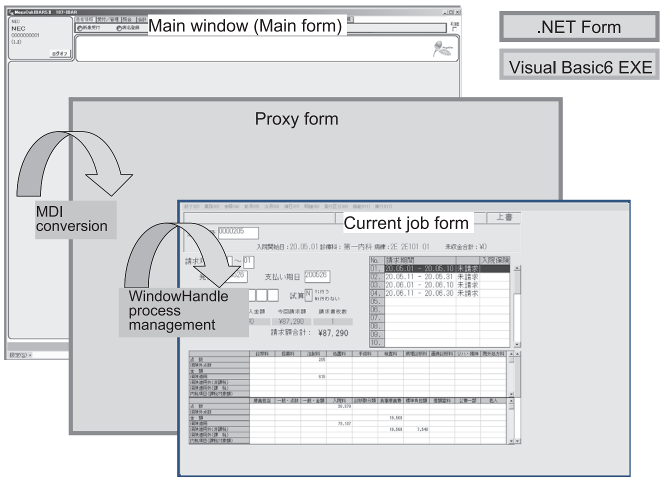

Fig. 3 MDI display of MegaOakIBARSII.

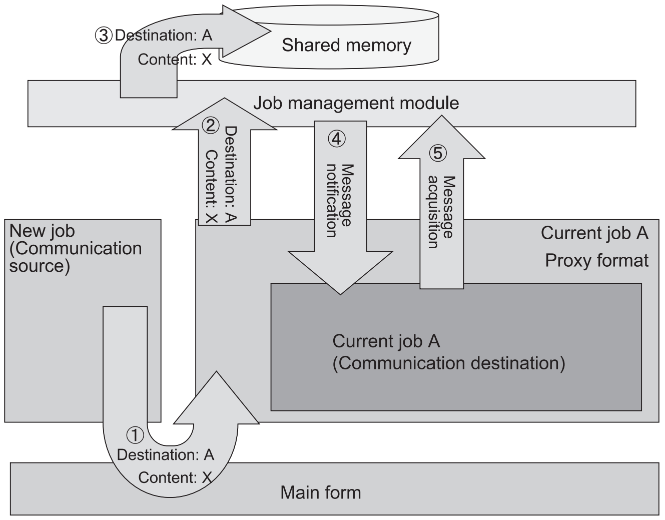

Fig. 4 Procedure of communication from new job to current job.

(2) Inter-Job Communications

When new and current jobs coexist, it is necessary to provide a means for inter-job communications from new jobs to current jobs and from current jobs to new jobs.

To make this possible we used both the main window (main form) managing the new jobs and the job management module managing the current jobs and packaged the inter-job communications using the following procedures.

● Procedures for communication from new job to current job (see **Fig. 4**)

1) A new job sends a notification (Destination: Current job A, Content: X) to the proxy form of the current job A via the main window (main form).

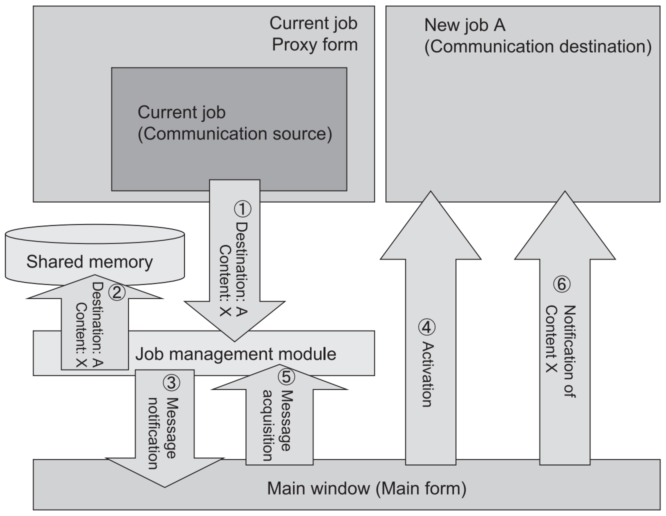

Fig. 5 Procedure for communication from the current job to the new job.

2) The proxy form sends the notification destination to the current job process (Destination: Current job A, Content: X) to the job management module.

3) The job management module saves the notification information (Destination: Current job A, Content: X) in the shared memory.

4) The job management module sends a message notification to current job A.

5) Current job A receives the notification and acquires the content from the job management module.

● Procedure for communication from current job to new job (see **Fig.5**)

1) The current job sends a notification (Destination: New job A, Content: X) to the job management module.

2) The job management module saves the notification information (Destination: New job A, Content: X) in the shared memory.

3) The job management module sends notification to the main window (main form).

4) If the destination new job is not activated, the main window (main form) reactivates it.

5) The main window (main form) acquires the content of the notification to New job A from the shared memory.

6) The main window (main form) sends the notification (Content: X) to the new job.

The above procedures enable intercommunications between the new and current jobs by simply modifying the main window (main form) and job management module.

4. Conclusion

As described above, we have succeeded in developing MegaOakIBARSII as a new system that inherits the functions of the previous system by hybridizing the new and previous systems.

We believe that the method described above makes possible the smooth transition of about 800 nationwide users who have introduced the previous system to the new system.

*Visual Basic is a trademark or registered trademark of the Microsoft Corporation in the United States and other countries.

Authors' Profiles

MANO Makoto
Senior Manager,
Medical Systems Division,
Community and Medical Solutions Operations Unit,
NEC Corporation

KIKUCHI Hirohito
Group Manager,
Medical Systems Division,
Community and Medical Solutions Operations Unit,
NEC Corporation

YANAGIHARA Keisuke
Assistant Manager,
Medical Systems Division,
Community and Medical Solutions Operations Unit,
NEC Corporation

KURAMOCHI Atsushi
Assistant Manager,
Medical Systems Division,
Community and Medical Solutions Operations Unit,
NEC Corporation