Dynamic Management Software Design in Embedded System using Middle

Suk Hwan Moon¹, Cheol sick Lee²

¹ Department of Computer Information, Cheju halla University,
38 Halladaehak-ro, Jeju-si, Jeju Special Self-Governing Province, Republic of Korea
msh@chu.ac.kr

² Department of Computer Information, Cheju halla University,
38 Halladaehak-ro, Jeju-si, Jeju Special Self-Governing Province, Republic of Korea
sigma@chu.ac.kr

Abstract. With high performance of hardware and development of diverse devices such as sensors, the embedded system must satisfy instant and diversified demands. Embedded middle ware should perform a mediating role of continuous management for productivity of complex embedded system, recombination of device, and provision of convenience in management by analyzing functions of existing system and more effectively handling new functions. To do so in this paper, a middle ware framework and dynamic module reuse method appropriate for small embedded system are designed, proposing a method for embodiment of software for reuse of resources.

Keywords: Embedded Middleware, Embedded System Management, Embedded System reusability

1 Introduction

Embedded system is currently applied in almost all fields including home, automobile, office, building, and plant. Especially, while there is no functional difference, development of embedded system for special purpose is being carried out for each purpose without consideration on the use of reusable device, due to convenience in maintenance.

Currently, many embedded system developing companies are supplementing hardware process for overlapping or similar functions with software technology. Embedded software must be able to perform complex functions in diverse fields with basic functions such as real-time processing, high reliability, optimization technology, application of specific system, and network and multimedia handling functions. Embedded operation system technology is quickly developing with the focus on light weight, low power, fast booting, and high reliability. With recent appearance of diverse devices based on sensor technology, embedded software requires a flexible and expandable platform structure. [1][2] For this reason, design of embedded systems has become extremely complex with very low productivity. Studies on recycling of existing system resources are being actively conducted nowadays, but
there are limits to the adapter method with simple addition of device and modification of software on the existing system. [3]

Accordingly in this paper, a middleware framework is configured to easily examine the interface between existing embedded hardware and software, and to reassign or reuse resources. This framework provides an additional processing module other than the reusable processing module connected with the functional module added by the middleware framework according to input and output method or information of input, processing and output modules, which are configured in unit of reusable functional module. In addition, an embedded system that constitutes an additional processing module by including it in the basic module provided by the above middleware is suggested. Also, UI and processing algorithm between modules are proposed to acquire input, processing and output module information of the embedded system, to display this information on the work window, and to reassign or add a functional module.

2 System Design for Reuse of Dynamic Embedded System Resources

General module reuse is operated according to generalization rule by generalizing the interface of unit module, and flexibility is extremely reduced because of limitation in the logic caused by generalization. Therefore in this paper, only the minimal part needed for system operation is generalized. For the remaining part, reflection technique is used to dynamically analyze the module, verify connection of each module, and set dynamic connection. Embedded middleware provides the base for configuring modules and allowing easy management of modules. It provides a framework for processing of base functions (queue management, thread management, scheduling, device management, etc.) and information (memory, storage device information, etc.) when a third developer wishes to develop and mount a unit module on the embedded system.

![Configuration of system for reuse of dynamic module (left), middleware framework (right)](image)

**Fig. 1.** Configuration of system for reuse of dynamic module (left), middleware framework (right)
Configuration shown in Fig. 1 can use communication devices (bluetooth, USB, Wi-Fi, CDMA, etc.) of the embedded system to remotely reconfigure one or more embedded system middlewares on a remote terminal (mobile, laptop, computer, etc.). Since module unit control and embedded system status can be checked here, it offers the base for easier management of complex system.

In other words, expansion of the embedded system can be made easier by configuration of unit modules, and time and cost of development can be reduced and unit modules can be developed easily by reusing unit modules. As doing so provides basic base functions and information to the embedded system through the framework of middleware, time and cost of unit module development can also be reduced. In addition, it can be utilized for development of an embedded system management tool for easy configuration and convenient management of embedded middleware, allowing for easy understanding and configuration of unit module with GUI method.

Fig. 2. Embedded system management for remote control software UI

Fig. 2 is a UI used for management of the embedded system. System configuration can be examined at a glance when the corresponding system devices are connected remotely or locally. An embedded system with new functions can be configured by reassigning input, processing and output modules based on existing modules or modifying existing modules for reuse. That is, it can include a manager element that manages one or more among contents management for one or more modules composed of functional module unit, thread management, event management, time management, file management, queue management, resource management, and sound management, or log data, status change and error generated by the above middleware.
Fig. 3 shows the processing logic for creation of a dynamic module, which includes the procedure for analyzing the embedded system resources, dynamically loading them, and registering them on middleware, as well as reuse or reassignment of each module by middleware and management software.

Component method mainly used for embedded system resource reuse technique has an effect of simplifying the structure and reducing complexity of development through generalization of the overall module, but application of this method in an actual embedded system device shows many difficulties in embodiment as diverse variables and exceptions occur. The aim of this paper is to effectively use system resources using reflection technique, which somewhat increases complexity of programming but alleviates the generalization process, to resolve such problems.

3 Experiments

To reuse embedded system resources, add embedded hardware device or add function, middleware must first analyze current device information. For this, middleware is allowed to load dynamic module based on plug-in, register the module with module manager, and share or reassign it on embedded software.

To mount new functional module on the embedded system for easier configuration of an improved embedded system, first, information of multiple modules composed of reusable functional module unit is obtained from the embedded system through the middleware framework. These multiple modules are classified into one or more of input, processing and output modules and displayed on the work window of the user device. User equipment include but does not need to be limited to mobile, laptop and PC attached with a communication device for management of the embedded system. Functional modules can be displayed on user equipment using GUI method. When a signal is entered on user equipment to select one of reusable multiple modules and
additional modules, input method and output event information of the selected module are displayed. As such, with consideration on input method and output event information of each module, the user connects two or more modules among the above reusable multiple modules and additional modules through the middle ware framework and appropriately assigns modules to configure an improved embedded system.

![Fig. 4. Screen assignment after analysis on the existing system (left), addition and setting of new module (processing function) (center), configuration of embedded system with new functions and reused resources (right)](image)

As shown in Fig. 4, this paper analyzed modules of the current system using reflection technique, added processing modules specifying processing data of the added devices, configured new processing event creation, and processing logic through reuse of existing modules to perform an experiment for change to a system with new functions.

### 4 Conclusion

System recycling and expansion based on the reuse of embedded system resources proposed in this paper were applied to a system with embedded operating system and profile storage memory. Accordingly, configuration of operating system base and framework design were proposed. As a result, increase in embedded systems subject to analysis allows diverse modules to be recorded and managed, and recorded module information can be continuously reused in the future. Also, the analyzed embedded system configuration can be visualized by management software to conveniently reassign or add resources according to dynamic processing procedure. Moreover, middle ware was designed based on service-oriented architecture for attachment on end terminal and remote management.

Embedded system resource reuse and management software based on reflection technique suggested in this paper reduces the generalization process of modules compared to the component method, resulting in slight increase of programming complexity. However in the end, system can have quick response and time and cost of system design and manufacture can be saved by configuring a system appropriate for diverse fields using a high performance embedded hardware and allowing efficiency management with development tool that takes distributed environment into account.

Future plan is to study a technique for reducing the complexity of programming, which appeared as a disadvantage of the system proposed in this study, by adding a
middle ware function for conversion of common protocol for various devices based on further segmentation of embedded system resources.

References