Implementation Method for Time-Critical Embeded Systems Using SES Approach

Hiroki Morimoto

School of Information Science,
Japan Advanced Institute of Science and Technology
Rebruary 15, 2000

Keyword : SES Approach, Aspect-Oriented programming, Embeded System, Real-Time Operating System.

Systems which are embedded in devices and controls them are called embedded systems.Today the embedded system are used in many areas and their scale is becoming larger and larger with the advances of hardware technologies and increase of services of they have to offer. This makes their developer to switch their development methods to object-oriented methods which have been successfully used for large scale information systems. However, development of embedded systems requires their specific non-functional properties to be taken into consideration such as time and hardware constraints.

In this paper, I proposed a development method to facilitate these constraints in the lower phase of their development process using SES approach. The SES approach is an object-oriented development method for time-critical embedded systems proposed by T.Aoki, et.al. In this approach, we construct a design model called SES model and implement software on real-time operating system (RTOS) using it. The SES model consists of process sequences which are called synchronous execution sequences, and we refer to each of them as SES. However, this method does not have been applied to practical system developments. We need to accumrate the experiences of developments where it is used and modify it. In my research, first, I implemented a telephone system on RTOS which is conform to μ ITRON using the SES approach. Software implemented on RTOS consists of tasks and they are scheduled based on priorities assigned to them and on other synchronization mechanisms such as semaphores and event flags. Though using the reference architecture, we can systematically implement software from the SES model, the following issues have to be still considered.

- How to bag SESs
  appearing in the SES model into tasks.Bagging SESs into tasks decides timing characteristics of the tasks such as dead lines and activation cycles. We have to consider how SESs should be bagged.

- How to schedule tasks.
  The tasks are scheduled based on their assigned priorities and on other synchronization mechanisms. We have to implement the scheduling strategy using them as the timing constraints are met.

- How tasks communicate.
  To deal with these issues, I incorporated the concept of AOP(Aspect-Oriented Programming) into the SES approach. In AOP, aspects of the system are independently programmed and then they are weaved with component programs. In this paper, I proposed an aspect programming language for each of these three issues and I described the telephone system using them.

In the aspect programming language for the bagging issue, we specify which task we put a SES in. Therefore, the abstract syntax of a programming language for the bagging aspect is defined by a function which maps a SES to a task.

In the scheduling aspect, we specify how to control tasks using priorities and synchronization primitives such as wup_tsk, slp_tsk, etc, wup_tsk wakes up a slept task, and slp_tsk puts a task to sleep.

Though thess primitives are usually scattered in source code, in the SES approach, they are put only before or after executing a SES because processes which the SES consists of are synchronously executed. In the aspect programming language for the scheduling issue, for each of SESs appearing in the SES model, we specify what sychroniztion primitives are put after or before it.

In the aspect programming language for the communication issue, we specify how to communicate between tasks if there is a communication between them exclusion. Such communication is implemented using system calls on RTOS or other primitives existing in the target environment.We describe only how to use variables which are passed to other SESs or shared by them in the implementation of processes and, in this

aspect programming language, we specify how to pass these variables.