High Level Object Oriented Real-Time Simulation Programming and Time-triggered Message-triggered Object(TMO) Scheme

Chan-Joo Jeong* · Sang-Dong Na*

ABSTRACT

The object-oriented(OO) distributed real-time(RT) programming movement started in 1990's and is growing rapidly at this turn of the century. Distributed real-time simulation is a field in its infancy but it is bounded to receive steadily growing recognition for its importance and wide applicability. The scheme is called the distributed time-triggered simulation scheme which is conceptually simple and easy to use but widely applicable. A new generation object oriented(OO) RT programming scheme is called the time-triggered message triggered object(TMO) programming scheme and it is used to make specific illustrations of the issues. The TMO structuring scheme is a general-style components structuring scheme and supports design of all types of component including hard real time objects and non real time objects within one general structure.

Keyword

Object Oriented(OO), Real-Time(RT), Time-triggered Message triggered Object(TMO)

Ⅰ. Introduction

Object oriented real time distributed computing is a rapidly growing branch of computer science and engineering. Its growth is fueled by the strong needs present in industry for the RT programing methods and tools which will bring about orders of magnitude improvement over the traditional RT programing practiced with low-level programing languages and styles.

RT simulator developments are under increasing demands[1,2,3,4]. For example, continuing advances in virtual reality application accompany increasing demands for more powerful RT simulation capabilities. Numerous other examplescan also be found in the RT computing control field. Not only description but also simulation of application environments is often performed as integral steps of validating control computer system designs.

RT simulators of application environments can often enable highly cost-effective testing of the control computer systems implemented. Such testing can be a lot cheaper than the testing performed in actual application environments while being much more effective than the testing based on non-RT simulators of environments.

The new generation OO RT programing scheme called the time-triggered message triggered object programing scheme[3,5,6]. In the course of developing a RT system engineering methodology based on this TMO programing scheme, a new approach to RT simulation which is conceptually simple and easy to use but widely applicable, has also been established.

*조선대학교 컴퓨터공학과

*순천청암대 컴퓨터정보과

접수일자 : 2002. 10. 21

In the next section, the motivations for pursuing the OO RT programing approach are reviewed and then in section 3, a brief overview is taken of the particular programing scheme. This programing scheme called the time-triggered message triggered object(TMO) programing scheme [3,6] is used on several occasions in the rest of this paper to make specific illustrations of the issues and potentials of OO RT programing.

II. High-Level Approach to Programming Real-Time Distributed Systems

As a concrete example of a high-level OO RT distributed programming approach that has been based on the philosophy discussed in the preceding section, the time-triggered message-triggered object (TMO) programming scheme is briefly summarized in this section[2,3,4,5,6].

The TMO scheme was established in early 1990's with a concrete syntactic structure and execution semantics for economical reliable design and implementation of RT systems. The TMO scheme is a general-style component structuring scheme and supports design of all types of components including distributable objects and distributable non-RT objects within one general structure.

Calling the TMO scheme a high-level distributed programming scheme is justified by the following characteristics of the scheme :

(1) No manipulation of processes and threads :Concurrency is specified in an abstractform at the level of object methods. Since processes and threads are transparent to TMO programmers, the priorities assigned to them, if any, are not visible, either.

(2) No manipulation of hardware-dependent features in programming interactions among objects : TMO programmers are not burdened with any direct use of low-level network protocols and any direct manipulation of physical channels and physical node addresses/names.

(3) No specification of timing requirements in (indirect) terms other than start-windows and completion deadlines for program units (e.g., object methods) and time-windows for output actions : TMOs are devised to contain only high-level intuitive and yet precise expressions of timing requirements. Priorities are attributes often attached by the OS to low-level program abstractions such as threads and they are not natural expressions of timing requirements. Therefore, no such indirect and inaccurate styles of expressing timing requirements are associated with objects and methods.

At the same time the TMO scheme is aimed for enabling a great reduction of the designer's efforts in guaranteeing timely service capabilities of distributed computing application systems. It has been formulated from the beginning with the objective of enabling design-time guaranteeing of timely actions. The TMO incorporates several rules for execution of its components that make the analysis of the worst-case time behavior of TMOs to be systematic and relatively easy while not reducing the programming power in any way.

TMO is a natural and syntactically minor but semantically powerful extension of the conventional object(s)[6]. As depicted in Fig. 1 the basic TMO structure consists of four parts :

ODS-sec = object-data-store section : list of object-data-store segments(ODSS's);

EAC-sec= environment access-capability section : list of gate objects (to be discussed later) providing efficient call-paths to remote

object methods, logical communication channels, and I/O device interfaces;

SpM-sec = spontaneous-method section : list of spontaneous methods;

SvM-sec = service-method section.

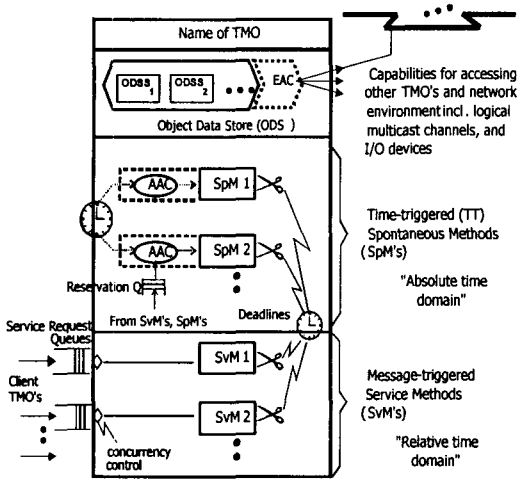

Figure 1. Structure of the TMO

Major features are summarized below.

(a) Distributed computing component :

The TMO is a distributed computing component and thus TMOs distributed over multiple nodes may interact via remote method calls. To maximize the concurrency in execution of client methods in one node and server methods in the same node of different nodes, client methods are allowed to make non-blocking types of service requests to server methods.

(b) Clear separation between two types of methods :

The TMO may contain two types of methods, time-triggered (TT-) methods (also called the spontaneous methods of SpMs), which are clearly separated from the conventional service methods (SvMs). The SpM executions are triggered upon reaching of the RT clock at specific values determined at the design time whereas the SvM executions are triggered by service request messages from clients. Moreover, actions to be taken at real times which can be determined at the design time can appear only in SpMs.

(c) Basic concurrency constraint (BCC) :

This rule prevents potential conflicts between SpMs and SvMs and reduces the designer's efforts in guaranteeing timely service capabilities of TMOs. Basically, activation of an SvM triggered by a message from an external client is allowed only when potentially conflicting SpM executions are not in place. An SvM is allowed to execute only when and execution time-window big enough for the SvM that does not overlap with the execution time-window of any SpM that accesses the same ODSSs to be accessed by the SvM, opens up. However, the BCC does not stand in the way of either concurrent SpM executions of concurrent SvM executions.

(d) Guaranteed completion time and deadline :

The TMO incorporates deadlines in the most general form. Basically, for output actions and method completions of a TMO, the designer guarantees and advertises execution time-windows bounded by start time and completion times.

Triggering times for SpMs must be fully specified as constants during the design time. Those real-time constants appear in the first clause of an SpM specification called the autonomous activation condition (AAC) section. An example of an AAC is

"for t = from 10am to 10:50am every 30min
start-during (t, t+5min) finish-by t+10min"
which has the same effect as
"start-during (10am, 10:05am)
finish-by 10:10am",

"start-during (10:30am, 10:35am)
finish-by 10:40am"

A provision is also made for making the AAC section of an SpM contain only candidate triggering times, not actual triggering times, so that a subset of the candidate triggering times indicated in the AAC section may be dynamically chosen for actual triggering. Such a dynamic selection occurs when an SvM within the same TMO object requests future executions of a specific SpM. Each AAC specifying candidate triggering times rather than actual triggering times has a name.

An underlying design philosophy of the TMO scheme is that an RT computer system will always take the form of a network of TMOs. The designer of each TMO provides a guarantee of timely service capabilities of the object. The designer does so by indicating the guaranteed execution time-window for exery output produced by each SvM as well as by each SpM executed on requests from the SvM and the guaranteed completion time (GCT) for the SvM in the specification of the SvM. Such specification of each SvM is advertised to the designers of potential client objects. Before determining the time-window specification, the server object designer must convince himself/herself that with the object execution engine (a composition of hardware, node OS, and middleware) available, the server object can be implemented to always execute the SvM such that the output action is performed within the time-window. The BCC contributes to major reduction of these burdens imposed on the designer.

Middleware which together with node OSs andhardware make up TMO execution engines, have been developed.

III. Multi-Level Multi-Step Desing with The TMO Structuring

First, the system engineering team describes the application environment as the TMO Mini-Theater in Figure 2, without the components enclosed by square brackets.

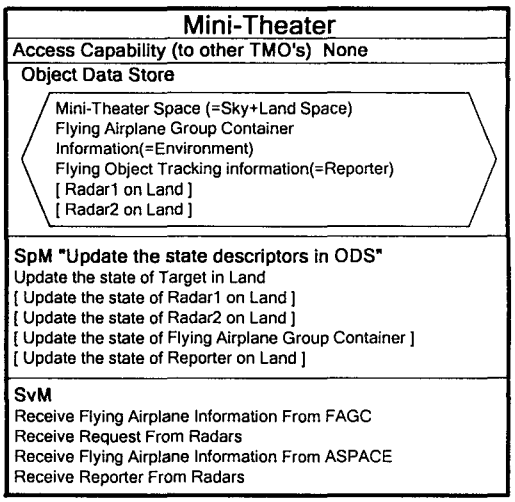

Figure 2. High-level specification of the Mini Theater TMO.

The components in brackets describes sensors (such as radar) which do not yet exist because the system engineering team has not decided which types to use.

The information kept in Mini-Theater is a composition of the information kept in all the state descriptors within its object data store. Here the object data store basically consists of the state descriptors for the following three environment components:

- Flying Airplane Group Container information (Environment)
- Flying Object Tracking Information(Reporter)
- Mini-Theater Space(Sky and Land)

Corresponding to each of these state descriptors of environment components is a spontaneous method that periodically updates the state descriptor. Conceptually, spontaneous methods in Mini-Theater TMO are activated continuously and each of their executions is completed instantly. Spontaneous methods thus represent continuous state changes that occur naturally in the environment components. Multiple spontaneous methods activated simultaneously can be used to precisely represent the natural parallelism that exists among environment components.

The state descriptor for the theater space not only provides geographical information about the theater but also maintains the position of every moving component in the Mini-Theater. This information is used to determine the occurrences of collisions among components and to recognize the departure of any component from the Mini-Theater.

The Mini-Theater object is more than a mere description of the application environment; it is also a simulation model. To support simulation, the designers choose an activation frequency for each spontaneous method such that it can be supported by anobject execution engine. The behavior of the environment can be simulated. This practical simulation is of course less accurate than the unexecutable description based on continuous activation of spontaneous methods. In general, the accuracy of a TMO-structured simulation is a function of the chosen activation frequencies of spontaneous methods.

Next the system engineering team decides which sensors to deploy. Sensors include two radars locatedon land. Once this is done,

```
Control Computer System In Reporter

Access Capability (to other TMOs)
        Radar (Accept_spot_check_request)

Object Data Store
        Radar data received, Flying airplane tracking information

SpM
SpM1 Radar Data Processing Step
    - "Process all the radar data received since the last processing cycle,
       update the flying object tracks"
    AAC : for T = from TMO_START + WARMUP_DELAY_SECS
        to TMO_START + SYSTEM_LIFE_HOURS every PERIOD
        start-during (T, T + START_WINDOW) finish-by T + DEADLINE
    InputSpec : Radar data received in the object data store
    OutputSpec : <deadline : xxx msec> Reflect changes onto the object data store,
          i.e., Radar data received, Flying airplane tracking info.
        <deadline : xxy msec> Send spot-check radar requests to Radar if ···;

SvM
SvM1 Receive_from_Radar_on_Land (pos_list)
       < Accept-with-Delay_Bound-of ACCEPTANCE_DEADLINE
          under MAX_REQUEST_RATE finish-within EXECUTION_TIME_LIMIT>
    - "Receive from Radar_on_Land the information on all recent detections."
    InitiationCond : Other SvM1 invocations are not in place.
    InputSec : pos_list = array of (return_type (=scan_search/spot_check),
       position, time, predicted_time)
    OutputSpec : <deadline : yyy msec> Deposit the radar data received
       in the object data store
SvM2 Accept Advice from ··· < Accept-via···· .>
```

Figure 3. Intermediate Specification of the control computer system for Command Post.

Mini-Theater can be expanded to incorporate all the components enclosed by square brackets in Fiqure 2. The object data store now contains the selected sensors. The two radars loaded on Reporter are described in the state descriptor for the Reporter.

Now the system engineering team should also decide how to deploy the computer-based control system in the Mini-Theater. The functions of the control system will be determined by the control theory logic adopted.

In this experimental development, we deployed one control system such as Reporter.

The Reporter contains a control system. Initially, the system engineers proceed each control computer system out of Reporter and generate single TMO specification, as shown in Figure 3. The specification in Figure 3 shows a more complete specificationstructure than shown in Figure 2. It has the autonomous activation condition for the spontaneous method, the input and output specifications for both the spontaneous and the service methods, and the initiation condition for the service method.

▸ The input specification for a method describes the actions of picking data during the execution of the method such as receiving the data coming from the external client in the form of call parameters, picking data from the object data store, or picking data from the input devices.

▸ The output specification for a method describes the action of sending data to other TMOs, sending data to the output devices, and depositing data into the object data store.

▸ The initiation condition for the service method describes when the service method execution can be initiated after being called by a client. It is in a sense a concurrency specification.

Now Mini-Theater is a network of three objects. The system engineering team is now ready to give the computer engineering team the specification structuredin the form of three TMOs, plus an overall specification of the type. Embed one control computer system in the Reporter such that the computer system follows the chosen control theory logic to control the chosen sensorssuch as radars.

In the real time simulation techniquesbased on TMO object modeling, we have observed several advantages to the TMO structuring scheme. TMO object modeling has a strong traceability between requirement specification and design, cost-effective high-coverage validation, autonomous subsystems, easy maintenance and flexible framework for requirement specification.

Ⅳ. Conclusion

We believe that using this scheme for the uniform, integrated design of complex real time systems and their application environment simulators offers great potential in significantly reducing the development costs and increasing the dependability of the real time systems. Also,the goal of the TMO structuring scheme, is to realize RT computing in a general manner not alienating the main-stream computing industry and yet enabling the system engineer to confidently produce certifiable real time simulator for safety-critical applications.

Although the potential of the TMO scheme has been amply demonstrated, much further research efforts are needed to make the TMO structuring technology easily accessible to common practitioners. Further development of TMO support middleware, especially those running on new-generation RT kernels and multiprocessor hardware, is a sensible topic for

future research. Tools assisting the TMO designer in the process of determining the response time to be guaranteed are among the most important research topics.

References

[1] A. Attoui and M. Schneider, "An object-oriented model for parallel and reactive systems", Proc. IEEE CS 12th Real-Time Systems Symp., pp. 84-93, 1991.

[2] K. H. Kim et al.," A timeliness-guaranteed Kernel model DREAM kernel and implementation techniques", Proc. 1995 Intl Workshop on Real-Time Computing Systems and Applications (RTCSA 95), Tokyo, Japan, pp. 80-87. Oct. 1995.

[3] K. H. Kim, C. Nguyen, and C. Park, "Real-time simulation techniques based on the RTO.k object modeling", Proc. COMPSAC 96 (IEEE CS Software & Applications Conf.), Seoul, Korea, pp. 176-183, August 1996.

[4] K. H. Kim and C. Subbaraman, "Fault-tolerant real-time objects", Commun. ACM 75-82. 1997.

[5] K. H. Kim, C. Subbaraman, and L. Bacellar, "Support for RTO.k Object Structured Programming in C++", Control Engineering Practice 5 pp. 983-991, 1997.

[6] K. H. Kim, "Object Structures for Real-Time Systems and Simulators", IEEE Computer 30 pp.62-70, 1997.

[7] H. Kopetz and K. H. Kim, "Temporal uncertainties in interactions among real-time objects", Proc. IEEE CS 9th Symp. On Reliable Distributed Systems, pp. 165-174, Oct. 1990.

저자소개

나상동(Sang-dong Ra)

1968년 조선대학교 전기공학과 졸업(공학사)

1980년 건국대학교 대학원 졸업(공학석사)

1995년 원광대학교 대학원 졸업(공학박사)

1995년~1996년 Dept. of Electrical & Computer Eng. Univ. of California Irvine 연구교수

1998년 조선대학교 전자계산소 소장 역임

1973년~현재 조선대학교 컴퓨터공학부 교수

2001년~2002년 Dept. of Electrical & Computer Eng. Univ. of California Irvine 연구교수

※관심분야: 실시간 통신, 디지털 통신망, 데이터 및 이동통신, TOM, 적응 신호처리 등임.

정찬주(Chan-Joo Jeong)

1993년 조선대학교 컴퓨터공학과 졸업(공학사)

1996년 조선대학교 컴퓨터공학과 대학원 졸업(공학석사)

1997년~현재 조선대학교 컴퓨터공학과 박사수료

1997년~2002년 순천청암대학 컴퓨터정보과 조교수

※관심분야: 실시간 통신, 인터넷 통신, 데이터 및 이동통신, TOM 등