

The TMO Structuring Approach and its Potential for Telecommunication Applications

K. H. (Kane) Kim * and Chittur Subbaraman

Dept. of Electrical & Computer Engineering
University of California
Irvine, CA, 92697, U.S.A.
kane@ece.uci.edu (*: contact author)

Abstract: The TMO (Time-triggered Message-triggered Object) structuring scheme has been formulated in recent years with the goal of improving the productivity by multiple times in the design of complex real-time computer systems (RTCS's). The TMO scheme is intended to facilitate the pursuit of a new paradigm in designing RTCS's which is called the "General-form timeliness-Guaranteed (GG)" design paradigm. The essence of the GG design paradigm is to realize real-time computing with a common and general design style not alienating the main-stream computing industry and yet allowing system engineers to confidently produce certifiable RTCS's for safety-critical applications. The TMO structuring scheme is a syntactically simple but semantically major extension of the conventional object structuring approaches and as such, its support tools can be based on various well-established object-oriented programming languages such as C++ and JAVA and on ubiquitous commercial real-time operating system kernels. The scheme enables a great reduction of the designer's efforts in guaranteeing timely service capabilities of information systems. In this paper we discuss the major features of the TMO structuring approach for real-time computer systems and its potential for use in the telecommunication application domain. Some of the major computing requirements that are imposed by a large class of telecommunication applications are identified and then the discussion on how well the TMO approach can satisfy these requirements, follows. As a concrete illustration, we also present the top-down design of a simple yet concrete multi-party video conferencing application using the TMO structuring scheme. Finally, the major benefits of the TMO structuring scheme in the design of complex RTCS's are discussed.

Keywords: real-time, object-oriented, telecommunication, video conferencing.

1. Introduction

In the last several years, there has been a growing trend of research activities aimed for extending the conventional object-oriented structuring approaches to support real-time computer system design [Att91, Ish92, Kim94b, Kim98]. Among the object extensions developed, the *time-triggered message-triggered object (TMO) model*, formerly called the *RTO.k object model*, is unique in its aim for enabling the design-time guarantee of

timely service capabilities of objects [Kim94a, Kim94b]. Significant extensions of the conventional object model that are incorporated into the TMO model are the following:

(a) Spontaneous methods (SpM's) clearly separated from service methods (SvM's):

For some methods of a TMO, a real-time clock serves as the mechanism for triggering the method executions as the clock reaches some values specified at design time. Such methods are called time-triggered (TT-) methods, also called the spontaneous methods (SpM's), and clearly separated from the conventional service methods (SvM's) triggered by messages from clients. The two types of methods in a TMO are different not only in the way their executions are triggered but also in that

"actions to be taken at real times which can be determined at the design time can appear only in SpM's".

(b) For each execution and completion of a method of a TMO, a deadline is imposed;

The designer of each TMO provides a guarantee of timely service capabilities of the object by indicating the *deadline for every output* produced by each SvM (and each SpM which may be executed on requests from SvM's) in the specification of the SvM (and some relevant

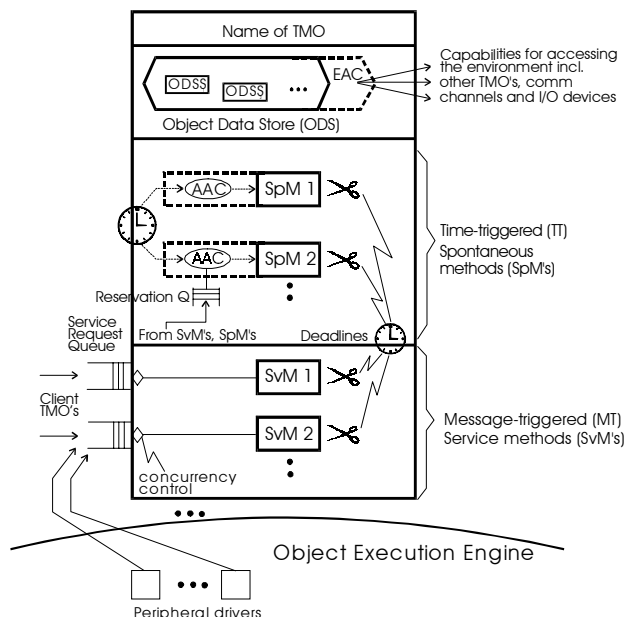


Figure 1. Structure of the TMO

SpM's) advertised to the designers of potential client objects. An execution rule has been incorporated into the TMO model in order to reduce the designer's efforts in ensuring that these deadlines can be met on a given execution engine. It is called the basic concurrency constraint (BCC) and prevents conflicts between SpM's and SvM's. 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.* Therefore, SpM's are given higher priorities for execution over the potentially conflicting SvM's. At least this BCC and the fact that triggering times of SpM's are fixed at the design time, make it very easy to analyze the execution time behavior of SpM's.

In order for the TMO's to provide guaranteed timely services to external clients, the engine that supports the execution of TMO's must obviously provide guaranteed timely services to the requesting TMO's. In addition to containing a hardware platform, such an execution engine should be formed by a *timeliness-guaranteed operating system*. Existing commercial operating systems are not yet capable of providing guaranteed timely services needed by a great deal of real-time application software.

The DREAM kernel has been formulated as a model of an operating system kernel that can support guaranteed timely services not only for TMO structured real-time applications but also for conventional process-structured real-time applications. The DREAM kernel guarantees the timely response to service requests by adopting a unique approach for the layering of its components [Kim95, Kim98]. This approach is based on the organizational principle called the time-leasing machine layering, which is of fundamental nature. Several prototype implementations of the DREAM kernel have been produced by the authors and their research collaborators to run on networks of PC's connected by Ethernet. Among the real-time distributed applications supported by the DREAM kernel include a traffic management system, a steel factory control system, and a military command-control system.

Supports for TMO execution have been built on top of both Solaris and Windows NT as middleware components [Sho97, Sho98]. Both these COTS operating systems provide some real-time application support with their real-time threads mechanisms. The middlewares that were built ensure correct and timely execution of the TMO's according to their timing specifications.

A user-friendly interface to the DREAM kernel can reduce the burden imposed on the TMO implementers. To achieve this goal, the authors have designed a library, called the DREAM Library [Kim96a], which is a collection of several C++ classes. Each class functions as an interface between a component of a TMO and the kernel support for that component. The DREAM library hides various details of parameter passing between application and the DREAM kernel from the application programmer and thus functions as a user-friendly interface. Some work on incorporating the TMO

structuring capabilities into other languages such as Java is also a schedules research action in the authors' laboratory.

One of the major objectives of this paper is to investigate the potential of the TMO structuring scheme for telecommunication applications where timely performance could be very important in many situations. In Section 2, we present the major computing requirements of a large class of telecommunication applications and discuss how the TMO structuring scheme caters to these requirements. A secondary objective of this paper is to illustrate the top-down design of a simple yet concrete telecommunication application using the TMO structuring scheme. In Section 3 we present the design of such a simple multi-party video conferencing application. In order to highlight the potential drawbacks of using conventional approaches in designing telecommunication applications we present, in Section 5 the design of the video conferencing application using a client-server approach. The major benefits of using the TMO structuring scheme are then summarized. This paper concludes in Section 6.

2. Potential of the TMO approach in the telecommunications field

In this section, we investigate the potential of the TMO structuring approach in the telecommunication application domain. Telecommunication applications are large and complex and typically impose a wide variety of requirements on the computing infrastructures that support them. The following are some of the major requirements imposed by a large class of telecommunication applications identified by communities such as the Telecommunications Information Networking Architecture Consortium (TINA-C) [Gie95, Bos97].

(1) *Support for precisely timed actions:*

(1.a) Applications such as multimedia telecommunication applications require

- QoS (quality of service) in terms of delays, jitter bounds, and throughput,
- flow control based on QoS contracts, and
- presentation synchronization between voice and video data.

(1.b) Transport network operation and management applications require

- network surveillance for the fast detection of failures in the system, and
- fast response to events such as alarms.

(1.c) Delay-bounded response to device-generated events.

(1.d) Guaranteed real-time behavior.

(2) *Generic connection management package:* Setup, maintenance and release of connections.

(3) *Modular and configurable structuring to manage the complexity of software.*

(4) *Platform-transparent and location-transparent distributed software modules and ease of scaling up and*

down.

(5) *Enable reuse of existing telecommunication software (complete interoperability between new and existing systems).*

(6) *Guaranteed dependability (availability and security).*

In order to address (3), (4), and (5), TINA-C and other telecommunication R&D communities are opting for a CORBA-based distributed software architecture. In order to address (1), these R&D communities are proposing real-time extensions to the CORBA ORB supported by a real-time operating system such as Chorus as well adding some service packages on top of the CORBA ORB such as the QoS management service. Requirement (2) is proposed to be handled by a Connection Management service package while requirement (6) is proposed to be handled by service packages such as the transaction service and security service built on top of a CORBA ORB.

The TMO structuring scheme, being an extension of the conventional object structuring scheme, naturally addresses (3) and (5). A network of TMO's supported by the DREAM kernel offers efficient migration of TMO's among various nodes in the system since a logical multicast communication scheme called the Hitachi-UCI Data Field scheme [Kim95b] supports location-transparent inter-TMO communication. In addition, efficient support for the development of TMO programs in C++ would facilitate the execution of such programs in a wide variety of hardware platforms. Thus, the TMO scheme can address the requirement (4) effectively.

The TMO scheme can also address the requirement (2) well. The TMO scheme together with the DREAM kernel offers support for various types of communication, i.e., one-to-one, one-to-many, many-to-one and many-to-many. Such a high degree of flexibility is not offered by a CORBA-type distributed software architecture which is inherently client-server based. Thus, the TMO scheme would offer a wide range of choices in managing connections between different communication parties.

The major strength of the TMO scheme, however, comes out in addressing requirements (1) and (6). As we mentioned in Section 1, the TMO scheme aims at achieving the design-time guarantee of timely service capabilities of objects. The inclusion of time-triggered methods which carry out critical tasks at real-times chosen at the design-time and the message-triggered methods which respond to client requests (which may include device-generated events as well) within a bounded delay contribute to performing precisely timed actions. Thus, as we will discuss in Section 3, it is not a difficult task to achieve synchronization (such as multimedia presentation synchronization) between different actions using the TMO scheme while such a task would be difficult using a conventional client-server approach as we will see in Section 5.

Even though current approaches to building

telecommunication software are aimed at addressing fault tolerance issues to some extent, they do not particularly address the issue of providing fault tolerance in a time-bounded manner. In other words, they do not particularly address the issue of producing output actions within a certain bounded delay after the time of performing the corresponding input actions despite the occurrence of hardware and/or software faults. This *time-bounded fault tolerance* issue could be of great importance in some telecommunication applications such as telephone switching applications. The primary-shadow TMO replication (PSTR) scheme [Kim97] has been formulated to specifically address this issue. The potential of this scheme has been convincingly demonstrated with many realistic application scenarios with hard real-time requirements. An analysis of this scheme to identify some tight bounds on recovery times has also been performed.

We thus feel that the TMO approach can be an improvement over current approaches in many ways in addressing the requirements imposed by complex telecommunication applications. One should also note that a network of TMO's may utilize an inter-object communication framework such as CORBA if necessary. In fact, some TMO support middlewares we built on top of Solaris and Windows NT [Sho97, Sho98] utilize the communication facilities offered a CORBA-compliant ORB.

3. TMO based design of a simple multi-party video conferencing (MVC) system

In order to investigate and demonstrate the potential power of the TMO structuring scheme, we have developed several real-time distributed computing application systems. These include prototypes of a military defense command-control system, an advanced traffic management system, and a steel factory control system. In this section, we discuss the top-down design of a simple multi-party video conferencing (MVC) system using the TMO scheme.

Figure 2 shows the application environment under consideration. n participant computer systems are interconnected by a communication network which may typically span over a wide area. Each computer system is equipped with a microphone, a video camera, a speaker, and a display unit. A person participating in a conference session typically sits in the line of sight of the camera and also in close proximity to the microphone. The camera and the microphone are sensors which interface with the computer system. The display unit typically shows the images of all the participants of a conference session including the participant sitting in front of it. Speakers attached to the computer system convey the audio signals generated during a conference session to the corresponding participants.

Initially the high-level requirements are given by the customer who places an order for the MVC system:

1. Each participant computer system must be delivered

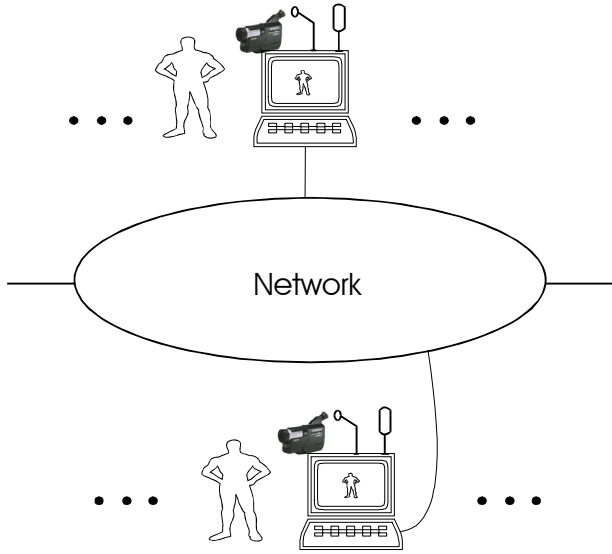


Figure 2. The multi-party video conferencing application environment

video frames (each of size x kilobytes) at the rate of $v \pm \Delta$ frames/sec, where $v \gg \Delta$, such that the time interval between the arrival of two successive frames must be within $v_d \pm \gamma$ seconds, where $v_d \gg \gamma$.

2. Each participant computer system must be delivered audio frames (each of size y kilobytes) at the rate of $a \pm \epsilon$ frames/sec such that the time interval between the arrival of two successive frames must be within $a_d \pm \eta$ seconds.
3. The time interval between the arrival of an audio frame and the corresponding video frame at a participant site must be within $av_d \pm \phi$ seconds.

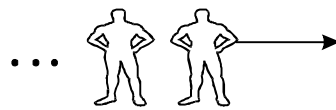
3.1 Step 0: High-level specification of the application environment of the MVC as a TMO

Initially, sensors such as microphones, cameras etc., and output units such as display units and speakers do not exist because the system engineer has not decided which types to use. As the first step, the system engineer (team) may describe the application environment of the MVC system as a TMO as depicted in Figure 3.

This TMO is called the *Video Conferencing System TMO*.

As we mentioned before, the MVC system consists of up to n participant site computer systems (PSCS's). Hence, the object data store (ODS) of the Video Conferencing System TMO consists of the state descriptors for $(0-n)$ PSCS's. The information kept in all these state descriptors constitutes the information kept in the Video Conferencing System TMO.

The PSCS state descriptors are periodically updated by a spontaneous method (SpM). Conceptually, this SpM in the Video Conferencing System TMO is *activated continuously* and each of its executions is *completed*



instantly. The SpM thus represents the continuous state changes that occur naturally in the real PSCS's. The SvM in the Video Conferencing System TMO functions as an interface to "external clients". The only conceivable clients here are the people each of who will enter a seat located in front of a participant computer system.

So far, the Video Conferencing System TMO in Figure 3 was interpreted as a mere description of the application environment. However, if the activation frequency of each SpM is chosen such that it can be supported by an object execution engine, then the resulting TMO becomes a simulation model. The behavior of the application environment is represented by this discrete-time simulation model somewhat less accurately than by the aforementioned description model based on continuous activation of SpM's. In general, the accuracy of a TMO structured simulation is a function of the chosen activation frequencies of SpM's. Note that this style of simulation is real-time simulation in which the simulation objects are designed to show the same timing behavior that the simulation targets do [Kim96b].

3.2 Step 1: High-level design of a PSCS using the TMO model

After creating the high level specification of the application environment, the system engineer (team) now decides to produce a high-level design of each PSCS. For this, the engineer first decomposes the Video Conferencing TMO into multiple *PSCS TMO's*. Such a decomposition would also involve the introduction of one or more SvM's in each PSCS TMO to establish some connections among the TMO's and between the TMO's and the human clients.

Next, the engineer decides on the types of sensors and output units to be used. Once those devices are

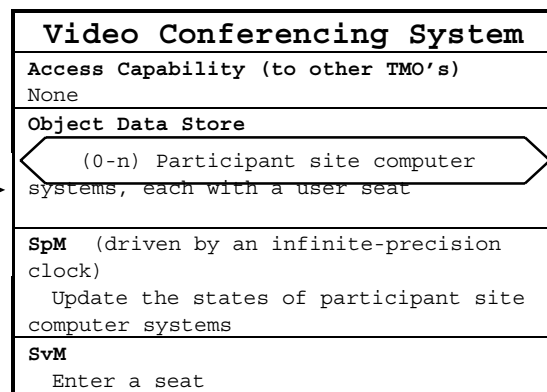


Figure 3. Video conferencing system

chosen, then the algorithms for operating the devices will be determined. Figure 2 already showed all the sensors and output units chosen. The PSCS augmented with chosen sensors and output units can be described as a TMO shown in Figure 4. The ODS of this TMO contains

state descriptors for the audio server, the video server, the audio receiver-player, and the video receiver-player, respectively. The ODS also shows the sensors and output units chosen by the computer engineer (team). The SpM "Update the state of audio server" is responsible for updating the state representation for the audio server which periodically sends out the audio frames generated by the microphone connected to the host computer system. The SpM "Update the state of video server" is responsible for updating the state representation for the video server which periodically sends out the video frames generated by the camera connected to the host computer system. The SpM "Update the state of audio receiver-player & speaker" is responsible for updating the state representation for the audio receiver-player and speaker which periodically send out the audio frames received at the host computer system to the speakers. Similarly, the SpM "Update the state of video receiver-player & display unit" is responsible for updating the state representation for the video receiver-player and display unit which periodically send out the video frames received at the host computer system to the display units. These SpM's can be viewed as core parts of the requirement specifications for the computer-based video conferencing system at this stage of the design cycle.

Remote method calls coming through message channels (a part of object execution engine)

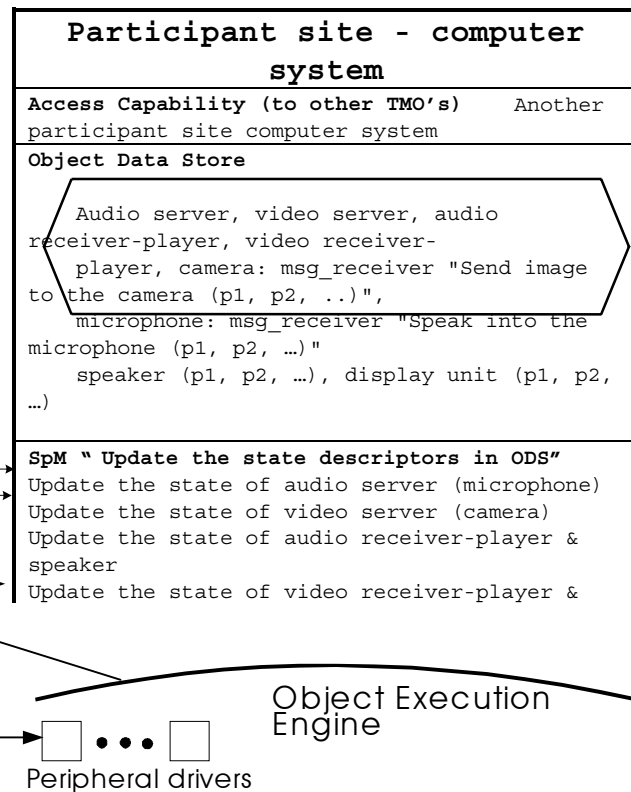


Figure 4. A participant site - computer system

In addition, the PSCS TMO contains 4 SvM's as shown in Figure 4. The first two SvM's are invoked by the client PSCS's running at remote sites and the functions of these SvM's are self-explanatory. The last two SvM's are invoked by the user sitting in front of the host computer system as shown in Figure 4.

3.3 Step 2: Decomposition of the PSCS TMO into 4 different TMO's

As the system engineer decomposes the single TMO representation of the PSCS TMO in Figure 4 as a part of more detailed design, a component of the ODS becomes a new TMO. When these new TMO's are created, the SvM's that serve as the front-end interfaces of these new TMO's should also be created. After the decomposition, the MVC system may be composed of a network of four TMO's: the *Audio server TMO*, the *Video Server TMO*, the *Audio Receiver-player TMO*, and the *Video Receiver-player TMO* (Figure 5).

In this process, the requirement specifications associated with the MVC system may be refined. The

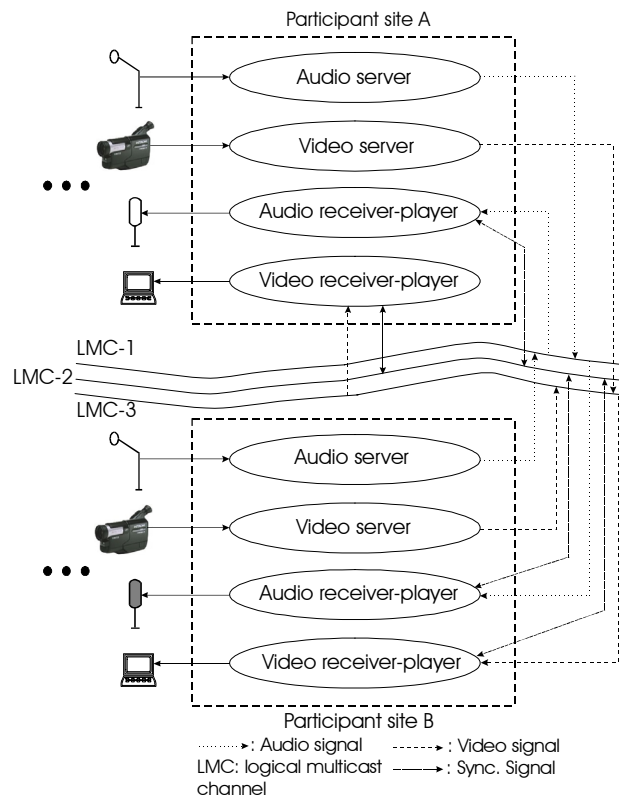


Figure 5. Decomposition of the PSCS TMO

Audio Server TMO may now describe or simulate the desired audio frame server function more accurately than the PSCS TMO did. Similar is the case with the other 3 TMO's shown in Figure 5. Figure 5 also shows the interaction among the TMO's in different PSCS's as well as the TMO's hosted in one PSCS.

3.4 Step 3: Detailed design of the MVC system

Next, the computer engineer designs in detail the TMO's created in Step 2. Figure 6 shows the detailed design of the Audio Server TMO. Due to space limit, we are not showing the detailed design of 3 other TMO's. By appropriate choice of timing parameters for each TMO method, the designer ensures that the MVC system designed as a network of TMO's meets the requirement specifications of the customer.

5. MVC design using conventional client-server approach

In order to highlight the major potential benefits of the TMO scheme in comparison with conventional approaches, we now consider the design of the MVC application using the conventional client-server approach. Figure 7 shows the MVC application designed using the client-server approach. The figure shows two different participant computer nodes A and B in the system. Both nodes A and B contain the audio server object, the video server object, the audio receiver-player object, the video receiver-player object, and the sync manager object. However, only the server objects are shown inside node A and only the receiver-player objects and the sync manager object are shown inside node B. Here it is assumed that the execution engine in each node provides multithreading/multitasking capability. Thus, the multiple

methods within an object all execute in a concurrent fashion.

The audio server object consists of two methods. The second method receives audio frames from the microphone and updates the ODSS while the first method responds to the request from a client for audio frames. The video server has a similar functionality. The audio receiver-player object has two different methods. The two methods may be invoked by two different threads and each method may run for X seconds, where X is designated by the customer of the MVC application. The first method is a continuously "looping" method that requests an audio server for audio frames. This method thus polls all the audio server objects in the system. Once this method obtains the audio data from the audio server, it updates its ODSS. In addition, it also informs the sync manager object of the ID of the audio data received. The second method of the audio receiver-player object is also a continuously "looping" method. This method is mainly responsible for checking with the sync manager whether proper synchronization with the video data has been achieved. Only if the audio and video data are properly synchronized, this method reads the audio data from the ODSS and sends it to the speaker. The functionality of the video receiver-player object is similar to that of the audio receiver-player object. The sync manager object manages the synchronization between the audio-player method and the video-player method. This object maintains the ID's of the most recent audio and video data frames in its ODSS. The ODSS is updated by the first method. The second method of the sync manager object responds to synchronization requests from the audio and video receiver-player objects.

This client-server type of design of the MVC application has the following major weaknesses:

- (1) Concurrency conflicts among multiple object methods (such as methods 1 and 2 of the audio receiver-player object) combined with their ability to have unrestricted competition for ODSS access makes the timely service guarantees a difficult task. This in turn would make it hard to meet the customer specifications.
- (2) Instead of a video (and audio) server multicasting

video (and audio) signals, multiple receiver-players poll the same video (and audio) server in a competing manner. This aggravates the message traffic problem severely.

- (3) Service methods such as the first method of the audio or the video server object can become a bottleneck if requests from multiple receiver-players occur at high rates. Employing multiple threads to serve client requests can reduce the problem somewhat but can worsen the concurrency conflict problem.

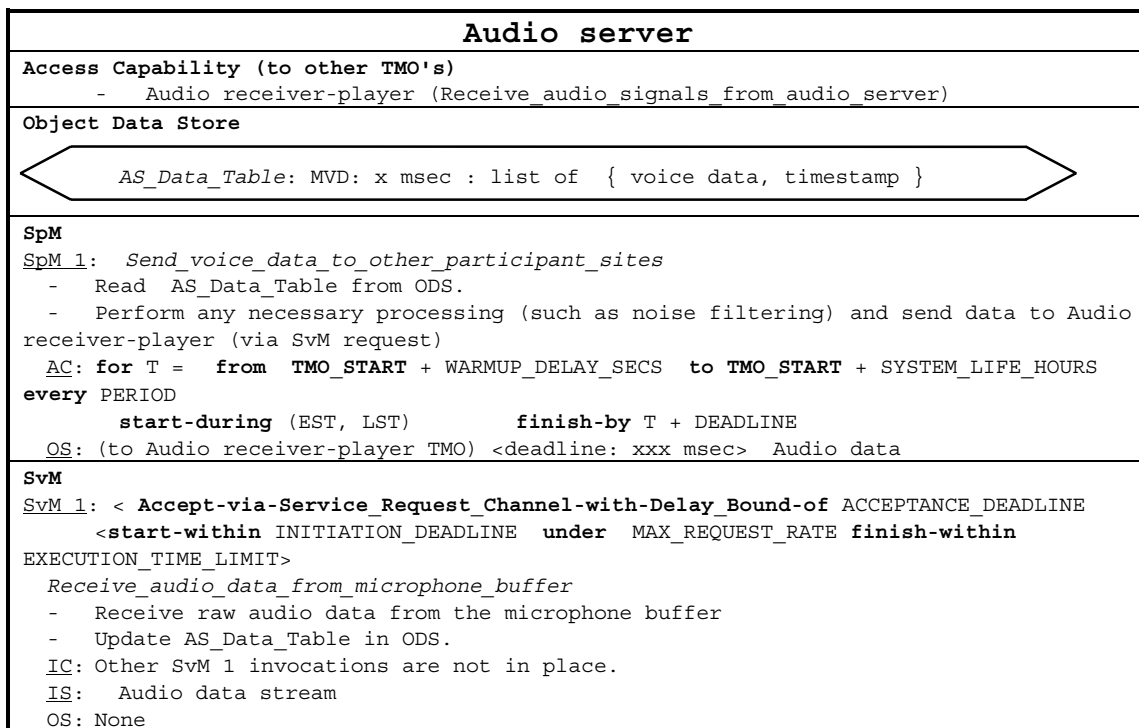


Figure 6. The audio server TMO specification

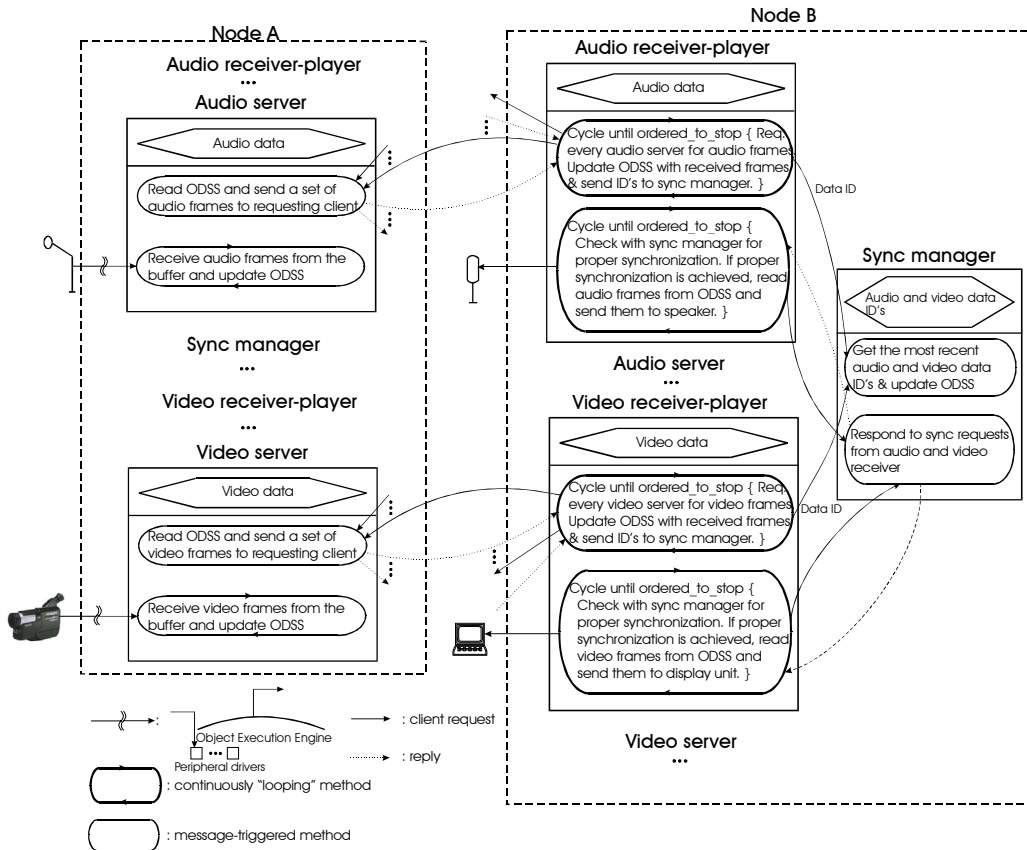


Figure 7. MVC design using a client-server approach

(4) Ensuring presentation synchronization executing jitter control among the audio and video signals at the client site are also major issues, even with the sync manager object. This is because even the synchronization of audio and video frames from one source may involve frequent competing accesses to the sync manager object and there can be quite a few sources. Conventional client and server objects are designed not to rely on the availability of a global time base and their typical execution engines make it difficult for the OO application designer to ensure timely interaction among the objects. High-quality synchronization involving audio and video frames from multiple participants can become a very difficult task.

4. Achieving a high degree of reliability by using the TMO structuring approach

The TMO based approach for engineering real-time computer systems allows the systems to achieve a high degree of reliability due to the following major reasons:

(1) *Timeliness-guaranteed design*: The TMO structuring and the use of a timeliness-guaranteed operating system kernel such as the DREAM kernel make it practical to realize a timeliness-guaranteed design of a control computer system. Much of the timing analysis and verification can be automated in principle although constr-

ucting effective tools possessing such capability will require multiple years of large-scale research efforts. Timeliness-guaranteed design leads to a significant reduction in the testing efforts required since the most difficult errors to detect, locate and correct are the timing errors. (2) *Cost-effective high-coverage validation*: Connecting a TMO-structured control computer system with a TMO-structured real-time simulator of the application environment produces a uniform TMO network representing the "application universe" and enables high-coverage testing. Such testing could be much cheaper than the testing involving the actual environment. The latter testing is inevitable but its duration can be

shortened by conducting the real-time simulator based testing first. The basic principle of a time-bounded fault-tolerant task execution scheme called the Distributed Recovery Block (DRB) scheme has been integrated with the TMO structuring scheme to produce the *primary-shadow TMO replication* (PSTR) scheme [Kim97].

(3) *Variable-degree abstraction of control computer systems*: The TMO structuring approach is an effective mechanism for the variable-degree abstraction of control computer systems. This gives the computer engineer (team) who develops the control computer system the freedom of incorporating fault tolerance mechanisms at the level of abstraction he chooses. For instance, the computer engineer (team) might initially describe the entire control computer system with one TMO. This object which contains the information about the entire control system in the state descriptors within its ODS could be chosen to be made fault-tolerant. Alternatively, the engineer might decide to separate one or more components from the ODS of the high-level TMO and create one or more new TMO's. These new TMO's could then be chosen to be made fault-tolerant.

(4) *Ease of incorporating time-bounded fault tolerance mechanisms*: The methods of a TMO are guaranteed to produce their outputs by relevant deadlines. The occurrence of hardware and/or software faults is the only possible reasons for methods of objects to miss their

deadlines. To reduce the probability of such an undesirable situation from occurring, time-bounded fault tolerance mechanisms need to be incorporated into the objects. The TMO structuring scheme allows easy incorporation of time-bounded fault tolerance mechanisms.

(5) *Strong traceability between requirement specification and design*: In both the requirement specifications and multi-step designs of control computer systems, the same notation, the same structuring styles, and the same refinement procedures are used. It is thus quite apparent that the relationship between requirement specifications and designs can be easily recognized.

(6) *Autonomous subsystems*: In a TMO network structured computing system, each object tends to be highly autonomous since most of the computations tend to be done by SpM's which do not have execution dependency or message communication among themselves. It is easy to maximize the autonomy of subsystems in a TMO network structured computing system as discussed in [Kim95a]. High degree of subsystem autonomy naturally leads to the ease of maintenance.

6. Conclusion

We believe that the TMO structuring scheme and its use for the uniform design of real-time computer systems and their application environment simulators offer great potential in significantly reducing the development costs of complex RTCS's such as those needed in telecommunication applications and increasing the dependability of the RTCS products from what is achieved under the current practice. Although limited in scope, the RTCS development experiments conducted so far are strongly supportive of this positive assessment. However, to fully realize the potential, many new specification, design, and execution tools need to be developed.

Acknowledgments: The research work reported here was supported in part by the US Defense Advanced Research Project Agency under Contract N66001-97-C-8516 monitored by NRad, in part by the University of California's MICRO Program under Grant 96-169, and in part by LG Electronics.

References

- [Att91] Attoui, A. "An Object Oriented Model for Parallel and Reactive Systems", *Proc. IEEE CS 12th Real-Time Systems Symp.*, 1991, pp. 84-93.
- [Bos97] Bosco, P.G., et al., "The ReTINA Project: An Overview", <http://www.chorus.com>, 1997.
- [Gie95] Gien, M., and Stefani, J-B., "Open Microkernel Technology, Key to Evolving Telecommunication Systems and Networks", *TELECOM 95*, Oct. 1995, Geneva, Switzerland.
- [Ish92] Ishikawa, Y., Tokuda, H., and Mercer, C. W., "An Object-Oriented Real-Time Programming Language", *IEEE Computer*, October 1992, pp. 66-73.
- [Kim94a] Kim, K.H. and Kopetz, H., "A Real-Time Object Model RTO.k and an Experimental Investigation of Its Potentials", *Proc. 1994 IEEE CS Computer Software and Applications Conf. (COMPSAC)*, Nov. 1994, Taipei, pp.392-402.
- [Kim94b] Kim, K.H. et al., "Distinguishing Features and Potential Roles of the RTO.k Object Model", *Proc. 1994 IEEE CS Workshop on Object-oriented Real-time Dependable Systems (WORDS)*, Oct. 94, Dana Point, pp.36-45.
- [Kim95a] Kim, K.H., et al., "Realization of Autonomous Decentralized Computing with the RTO.k Object Structuring Scheme and the HU-DF Inter-Process-Group Communication Scheme", *Proc. 1995 IEEE CS's 2nd Int'l Symp. on Autonomous Decentralized Systems*, Phoenix, AZ, April 1995, pp. 305-312.
- [Kim95b] Kim, K.H. et al., "A Timeliness-Guaranteed Kernel Model - DREAM Kernel and Implementation Techniques", *Proc. 1995 Int'l Workshop on Real-Time Computing Systems and Applications (RTCSA 95)*, Tokyo, Japan, Oct. 1995, pp. 80-87.
- [Kim96a] Kim, K.H. et al., "The DREAM Library Support for PCD and RTO.k programming in C++", *Proc. IEEE CS Workshop on Object-oriented Real-time Dependable Systems*, Feb. 96, Laguna Beach, pp. 59-68.
- [Kim96b] Kim, K.H., Nguyen, C., Park, C., "Real-Time Simulation Techniques Based on the RTO.k Object Modeling", *Proc. COMPSAC '96 (IEEE CS Software & Applications Conf.)*, Seoul, August 1996, pp. 176-183.
- [Kim97] K.H. Kim, and C. Subbaraman, "Fault-Tolerant Real-Time Objects", *Communications of the ACM*, January 1997, pp. 75-82.
- [Sho97] Shokri, E., et al., "ROAFTS: A Real-Time Object-Oriented Adaptive Fault Tolerance Support Middleware", *Proc. IEEE Workshop on Middleware Services and Systems*, San Francisco, CA, Dec. 1997.
- [Sho98] Shokri, E., et al., "An Implementation Model for Time-triggered Message-triggered Object Support Mechanisms in CORBA-compliant COTS Platforms", *Proc. 1st IEEE International Symp. on Real-time distributed Computing Systems*, Kyoto, Japan, April 1998.

Proceedings

**1998 IEEE WORKSHOP
ON
APPLICATION-SPECIFIC SOFTWARE
ENGINEERING AND TECHNOLOGY**

ASSET-98

March 26-28, 1998
Clarion Hotel and University of Texas at Dallas
Richardson, Texas

Co-sponsored by
IEEE Computer Society
and
Center for Application-Specific Systems and Software Engineering (CASSE)
at UT-Dallas

Edited by
Simeon Ntafos



Los Alamitos, California

Washington • Brussels • Tokyo
