High Level API for CORBA-Based High-Precision Real-Time Programming

Seok-Joong Kang, Hiroshi Miyazaki, and Kane Kim
DREAM Lab, UC Irvine
{seokjook, hmiyazak, khkim}@uci.edu, http://dream.eng.uci.edu/

June 2001

Outline

• Key features for real-time computing
• TMO (time-triggered message-triggered objects) structuring scheme adapted to CORBA environment
• Example application
• Tool for visual development of real-time applications
• Conclusion
Key Features for Real-Time Distributed Computing

- Global time base
  - Synchronized clocks of computing nodes (e.g. GPS)
- Location transparency
  - Uniform method invocation of both local and remote objects
- Lockable data structure
  - Increase concurrency of object method execution
- Message-triggered methods
  - Conventional object methods in remote objects
- Time-triggered methods
  - Triggered when real-time clock reaches specific values
- Guarantee of timeliness
  - Server’s guarantee of method completion times & client’s imposition of a deadline for arrival of the results returned from the invoked server object’s method

Location Transparency

Real-Time Distributed Computing Applications

RT middleware may rely on ORB to provide location transparency
Lockable Data Structure

- Lockable data storage unit
  - Concurrency control should be handled by RT middleware
  - Easy-to-use facility to achieve exclusive access of shared data
- Specifications of potential access of a data storage unit by methods can increase the efficiency of the scheduling of method execution
- Each method should register with RT middleware data storage units that it will access (read-only or read-write) during execution
- Application should request RT middleware to lock and unlock a data storage unit

Time-Triggered Methods

- Triggered by RT middleware when real-time clock reaches at specific values determined at the design time
  - Timing specification of each time-triggered (TT) method should be registered with RT middleware at initialization of an RT application
  - RT middleware schedules TT methods based on the specifications

EX. for <time-var> = from <activation-time> to <deactivation-time>
    [every <period>]
    start-during (<earliest-start-time>, <latest-start-time>)
    finish-by <guaranteed completion time>

for t = from 10am to 10:50am every 30min
    start-during (t, t+5min)
    finish-by t+10min

= (*start-during (10am, 10:05am)
    finish-by 10:10am*,
    *start-during (10:30am, 10:35am)
    finish-by 10:40am*)
**Message-Triggered Methods**

- Conventional service methods triggered by messages from clients
  - Can be implemented as operations of CORBA objects
  - Server’s guarantee of method completion times
  - Client’s imposition of a deadline for arrival of the results returned from the invoked server object’s method

![Diagram](image)

**TMO (Time-triggered Message-triggered Objects) Execution Support (TMOES)**

- A middleware architecture supporting execution of CORBA-based TMOs
- No modification of ORB required
- Supports distributed, real-time programming on COTS platforms with various ORBs (TAO, OmniORB, Orbix, ...)
- Performance of the prototype implementation
  - Supports the time-window for activating a method as small as 20ms
  - Supports the execution deadline as short as 30ms
- TMOESL (TMOES library), a user-friendly C++ API, is provided
- Tool for Visual development of CORBA-based TMO applications is under development
Time-triggered Message-triggered Objects (TMO) Structuring Scheme

- Object Data Store (ODS)
  - List of object data store segments (ODSS)
  - Environment access capabilities (EAC)

- Time-triggered (TT-) or spontaneous methods (SpM's)
  - Triggered when real-time clock reaches at specific values determined at the design time (Autonomous Activation Condition)

- Message-triggered or service methods (SvM's)
  - Conventional service methods triggered by messages from clients

Logical Multicast Channels
- RMMC (Real-time Multicast and memory-replication channel)

TMOES Library (TMOESL)

- User-friendly API library for CORBA-based TMO programmers
  - A collection of C++ classes
  - Functions wrapping TMOES services

Translated by IDL compiler

Skeleton

Use an object

Interface TMO1 {...

Class POA_TMO1 {

IDL file

Group of functions of IO Management

Group of functions of Real-time Clock Management
TMOES Library (TMOESL) (cont.)

class TMOBaseClass {
public:
    TMOBaseClass();  ~TMOBaseClass();
    int activate( const char* TMO_Name,
                 const tms& TMO_start_time);
    int get_TMO_ID() const;
    ...
};

class ODSSBaseClass {
protected:
    int EnterODSS_RO();  int ExitODSS_RO();
    int EnterODSS_RW();  int ExitODSS_RW();
public:
    ODSSBaseClass();  ~ODSSBaseClass();
};

class SpMBaseClass {
public:
    SpMBaseClass();  ~SpMBaseClass();
    void build_regist_info_SpM_name( const char* name);
    void build_regist_info_ODSS( int odss_id, access_mode_type mode);
    void build_regist_info_AAC( const AACclass& AAC);
    int RegisterSpM();
    int ActivateAACcandidate( const char* AAC_label);
    int DeactivateAACcandidate( const char* AAC_label);
    virtual void SpMBody() = 0;
    ...
};

class SvMBaseClass {
public:
    SvMBaseClass();  ~SvMBaseClass();
    void build_regist_info_SvM_name( const char* name);
    void build_regist_info_ODSS( int odss_id, access_mode_type mode);
    void build_regist_info_max_invoke_rate( const max_invoke_rate_type& mir);
    void build_regist_info_guranteed_completion_time( const MicroSec& gct);
    int RegisterSvM();
    ...
};

typedef void (*TMOESL_WRAPPER_FUNC)();

int BlockingCallForAgent( TMOESL_WRAPPER_FUNCWrapperFunc,
                          const MicroSec& ResponsePeriod);
int BlockingCallForAgent( TMOESL_WRAPPER_FUNCWrapperFunc,
                          const tms& RTDeadline);
int NonBlockingCallForAgent( TMOESL_WRAPPER_FUNCWrapperFunc,
                            tmsp& Timestamp);
int BlockingCheckOfAgentResultWithDeadline( const tmsp& Timestamp,
                                              const MicroSec& ResponsePeriod);
int BlockingCheckOfAgentResultWithDeadline( const tmsp& Timestamp,
                                              const tms& RTDeadline);
int NonBlockingCheckOfAgentResultWithDeadline( const tmsp& Timestamp);
int ReportSvMStart( SvMBaseClass& svm);
int ReportSvMCompl( SvMBaseClass& svm);

Message-Triggered Methods in TMOES

- Support multiple types of service requests
- Blocking call
  - No client's deadline imposition
    - CORBA synchronous method invocation
  - Client's deadline imposition
    - Client can specify a deadline for the result arrival (DRA) when it invokes service request
- Nonblocking call
  - No client's deadline imposition
    - CORBA asynchronous method invocation (AMI)
  - Client's deadline imposition
    - Client can specify a DRA when it checks the result return
- Uniform APIs for blocking and nonblocking calls are provided
**Execution of Message-Triggered Methods - Server Side**

- The first action is to report the starting of the execution to TMOES and the last action is to report the completion of the execution
  - TMOES can control the execution of the methods and detect deadline (guaranteed completion time) violations can be detected

**Blocking Call of Service Methods with No DRAs Imposed - Client Side**

- Service requests can be issued by direct calls to service methods (synchronous method invocation)
  - TMOES in the client side isn't involved in handling service requests
  - Application programmers can't exploit the capability of TMOES for checking timeliness of service completions and result returns
Blocking Call of Service Methods with DRAs Imposed - Client Side

- Service requests can be issued by indirect calls to service methods through TMOES API
  - Take advantage of timeliness-checking capability of TMOES
  - Programmers need to supply a wrapper function which calls the service method
  - Using of a blocking service request API results in passing a pointer of the wrapper function to TMOES along with a deadline for result return
  - TMOES checks if the result returns within the deadline and notifies the client of any deadline violation

```
BlockingCallForAgent(pWrapperFunc, deadline);
```

Nonblocking Call of Service Methods with No DRAs Imposed - Client Side

- Service requests can be issued by direct calls to service methods (asynchronous method invocation)
  - Client is not blocked during the processing of the service request and might continue doing other jobs
  - TMOES in the client side isn't involved in handling service requests
  - No DRA is imposed on checking of the result return

```
server_obj->method7();
```

```
AMI callback method
Do_something();
```
Nonblocking Call of Service Methods with DRAs imposed - Client Side

- Service requests can be issued by indirect calls to service methods through TMOES API
  - Programmers need to supply a wrapper function which actually calls the service method
  - Control of execution and a timestamp returns to the client immediately after it calls nonblocking service request function
  - Client can continue doing other jobs

Nonblocking Call of Service Methods with DRAs imposed - Client Side (cont.)

- Client may check later the availability of the service method result
  - Blocking result check APIs with deadline imposition and nonblocking result check APIs are provided
  - Timestamp is used as the identifier of the earlier method call in the result check
Performance of TMOES

- TMOES has been set to run on the following ORBs:
  - TAO
  - OmniORB
  - Orbix

- Performance studies are underway

- Optimizations of TMOES toward use of special capabilities of RT ORBs are yet to be achieved
Tool for visual Development of CORBA-based TMO application

- TMO network diagram
- Property of each TMO
- Interface definition

ViSTMO

- C++ code for CORBA-TMO class definitions
- TMO config.ini files, Visual Studio work space, and projects files

Requirement analysis and Design

Code Generator & IDL Compiler

Code generation

Compiler / Debugger (MS Visual Studio)

Coding and debugging

Timing Analyzer

Timing analysis

Other tools

Objective of Visual Development Tool

- Automatic generation of C++ source code (class definitions only)
  - It generates C++ source code (class definitions for TMO, ODSS, SpM, SvM)
  - Increase programming efficiency
- Allowing smooth transition from design to coding
  - It can be integrated with C++ compiler (MS Visual studio) and will create a workspace and projects for the application
  - Minimize the gap between design and coding
- Efficient management of design documents and source code
  - It will help application TMO designers and programmers to manage design documents and source code
  - Increase productivity
Screen shot of Visual Development Tool (I)

Screen shot of Visual Development Tool (II)
Conclusion

- TMOES enables fast and efficient development of CORBA-based TMO-structured RT distributed applications supported by ORBs

- TMOES doesn’t require any change in the CORBA standards

- Tool for visual development of CORBA-based TMO-structured application will be available later this year

- A prototype implementations of DCOM-based TMO programming facility has recently been realized

- Until ideal language tools arrive, a pragmatic approach to enable high-level real-time distributed object programming today is to provide abstract APIs