CORBA technologies for railway site reservation and passenger information

Petre Băzăvan
University of Craiova, Romania,
SC Silogic SRL, Romania

Abstract

This paper aims to describe the suitability of CORBA (Common Object Request Broker Architecture, [1]) as a management middleware [2] for a railway communications systems. We present a CORBA solution for passenger information and site reservation.

1. Introduction

Research and simulations presented in this paper was supported by SC Silogic Romania [3] as part of the European Project TrainCom - Integrated Communication System for Intelligent Train Applications (Contract No. IST - 1999-20096) [4].

The intention of TrainCom project was to develop communication system for telematic applications in the railway field integrating on-board networks, GSM radio links and Internet technologies. The TrainCom project proposes a communication infrastructure integrating Internet and GSM radio communication technologies, Train Communication Network (TCN) (on-board network) with client-server type technologies. The infrastructure includes the Railway Open Ground Station (ROGS), several networks and on-board interface (Railway Open GATEway - ROGATE) between TCN and the Internet word. Based on this system the project develops applications related to the fields of dynamic passenger information and locomotive interoperability. The infrastructure is intended to became the standard platform on top of which a number of applications can be built overcoming border line problems and equipment heterogeneity [5].

Middleware is a computer software defined as a layer above the operating system but below the application program that provides a common programming abstraction across a distributed system [2]. It is designed to mask the complexity and heterogeneity inherent in distributed systems.

CORBA is one of the most used middlewares. Other known technologies are Java RMI/Enterprise Java Beans (only for Java users) [6], [7] and Microsoft's DCOM (only on Microsoft's Windows Operating Systems) [8]. All three technologies support the development of object-based distributed applications. The CORBA characteristics (heterogeneity, location transparency, dynamic configuration, etc.) are appealing in many application domain such as embedded real-time data, telecommunications, aerospace, ground vehicle control and automation systems. Services can be written in many different languages and executed on many different platforms. Now, CORBA utilization is in decrease because the price is very expensive but CORBA is ideally to ensure that applications written now will be accessible in the future.

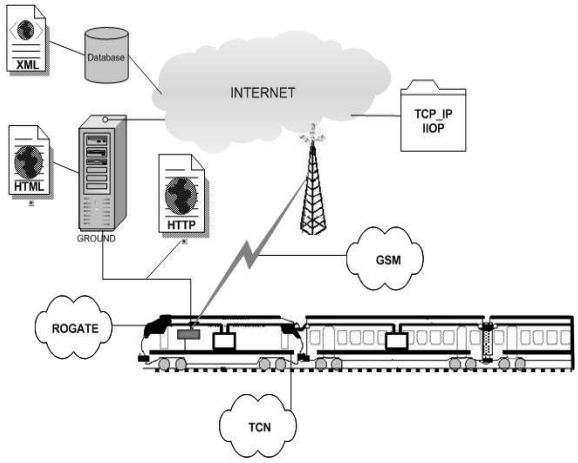

Figure 1. TrainCom Communication Infrastructure

As Part of the TrainCom approach, the SILOGIC solution for railway passengers information and sites reservation is structured around the CORBA technology [9].

In order to make clear our explanation we shortly present the principals CORBA concepts.

CORBA is a standards-based distributed computing

model for object-oriented applications developed by the Object Management Group (OMG) [10]. Finding the CORBA objects, transmitting the method call and returning any results are mediated using an Object Request Broker (ORB) which the application doesn't have to know about. The ORB allows objects to interact in a heterogeneous, distributed environment, independent of the computer platforms on which the various objects reside. CORBA-compliant applications typically comprise a client and a server. The ORB manages the communications between client and server using the Internet Inter-ORB Protocol (IIOP), which is a protocol layer above TCP/IP. The objects in a distributed CORBA application can communicate with the ORB through an interface written in the CORBA Interface Definition Language (IDL). The CORBA IDL is a language included in the CORBA 2.0 specification that is designed to describe the interfaces of objects, including the operations that may be performed on the objects and the parameters of those operations. Clients need only know an object's interface in order to make requests. Servers respond to requests made on those interfaces, and clients do not need to know the actual details of the server's implementation.

The next two sections will show how CORBA can be used to manage resources in railway communication systems.

2. The Architecture of TrainCom CORBA Model

According with the normative framework in railway field, proposed by TrainCom project [4], our TrainCom CORBA Model (TCCM) offers remote access to on-board equipment and integrates technologies such as:

- on-board TCN - is the local train network;

- higher level protocols (e.g. TCP-IP, IIOP) and languages of Internet (XML, JAVA);

- on-board interface ROGATE between TCN and the Internet world (is the radio-link based on GSM);

- the ground infrastructure (e.g. communications and applications servers to support the needed communication and applications services).

Our aim is to be as operating system independent as possible, in order to be flexible enough for possible upcoming changes. A possible solution that could fit within these limits is the 3-tier CORBA server-client architecture where CORBA server would be querying local databases and sends results to CORBA clients. The client in turn would have to combine the results and, if is necessary, can propagates the information at the on-board network level.

The principals TrainCom CORBA-based standard entities, the server and the client, are developed in Java language and, consequently, our model is OS-independent. We

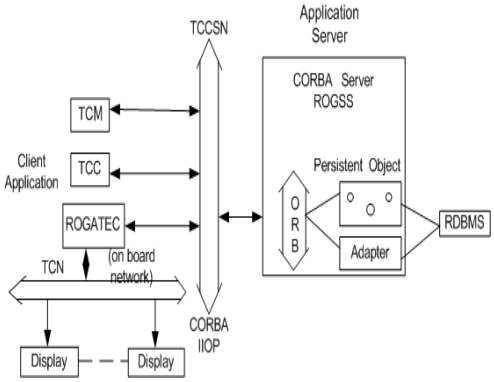

Figure 2. TrainCom CORBA Architecture

use a portable object adapter (POA) to communicate between ORB and the object implementation and CORBA's name service to get the object references.

Our specification addresses:

- a CORBA-based Intelligent Network (e.g. network data changes between CORBA-based entities);

- interworking between CORBA-client entity and the on-board network infrastructure (e.g. TCN).

The server side of TCCM is the Railway Open Ground Station Server (ROGSS) and is hosted in a computer placed on GROUND station.

There are three client applications in our model:

- TrainCom Manager (TCM) runs in a same computer as ROGSS or in the same private network as ROGSS;

- TrainCom Client (TCC) is hosted in a computer placed in a Railway Station or in a Travel Agency;

- The on-board structure includes ROGATE interface and the local train network (e.g. TCN). ROGATE links ROGSS with this network through a GSM radio link using Internet protocols. A mechanisms for IP-addressing, GSM numbering and train identification matching are used. For this on-board structure, the client side of the TrainCom model is the ROGATE Client (ROGATEC) that is hosted by the ROGATE computer.

In communication architecture of TCCM we distinguish a private network TrainCom Client-Server Network (TCCSN) that links ROGSS with TCC, TCM and ROGATEC clients using Internet protocols. In addition, the link ROGSS-ROGATEC uses a set of GSM wireless connection.

The following table describes the protocol architecture of TCCM communication. It uses the generic parts: IIOP, TCP/UDP, IP, GSM, RS232C.

The following table describes the software architecture for TCCM communication network.

Table 1. Protocol architecture of TCCM communication

Layer 5	CORBA Server-Client Application, XML reports
Layer 4	TCP/UDP, IIOP
Layer 3	IP
Layer 2	GSM-RS232C, Internet
Layer 1	TCN

Table 2. Software architecture for TCCM

	CORBA Server-Client Application, XML reports Compiler: JDK Relational Data Base
UDP, IIOP	Communication: VisiBroker [11]
IP	OS: Windows NT/2003/XP or Unix/Linux
TCN	OS: Embedded Windows NT or Unix/Linux

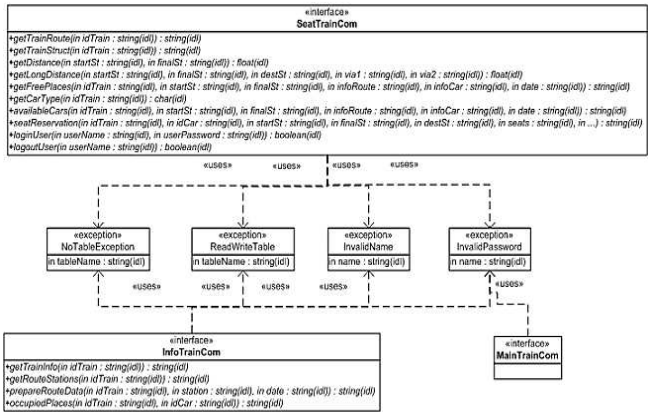

Figure 3. TrainCom CORBA Interfaces

3. Software Design

Two scenarios are implemented in TCCM.

The first scenario is from producer-consumer pattern and regards the sites reservation and the management of base information. It is a typical unidirectional CORBA communication between client and server. This implies that whenever TCM or TCC client is interested it can invoke a method on an object of ROGSS.

The second scenario is a publisher-subscriber communication model [12] and regards the passengers information activity witch implies that ROGATEC client must be notified of recent events occurring at the ROGSS side.

In the both cases TCCM offers interfaces which are CORBA-IDL based API. These interfaces use a CORBA ORB to provide necessary middleware services then, on ROGSS side we have CORBA objects that implement these services.

The first scenario uses the interfaces MainTrainCom, SeatTrainCom, InfoTrainCom and a set of CORBA exceptions. The CORBA interface MainTrainCom provides services for manage the base informations that are trains routes, type and structure of cars, tariffs, etc. These services are used by the client application TCM. The CORBA interface SeatTrainCom provides services for sites reservation. TCC is a dedicated client application which uses these services. Information services are provided by the CORBA interface InfoTrainCom and are used by TCM, TCC and ROGATEC applications. Various report type informations are generated by these applications. In diagram (3) we present the three interfaces and same methods of SeatTrainCom and InfoTrainCom interfaces.

Each method can throws minimum a CORBA exception presented in the diagram. At implementation level, CORBA objects use others non CORBA objects. In case of SeatTrainComImpl CORBA object, same relations with non CORBA objects are represented in the next diagram.

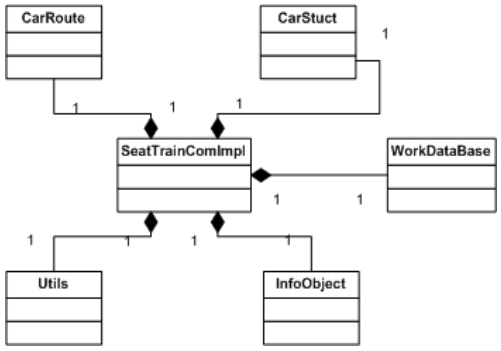

Figure 4. SeatTrainCom Implmentation

Each one of TCM, TCC and ROGATEC client applications uses an object WorkWithServer which has the role to call the methods of CORBA objects on ROGSS side. The call mechanism is presented in Figure (5).

For the second scenario a discussion is necessary. Whenever a remote CORBA object changes its state, an event is generated. The same event should get propagated or notified to all the clients connected to the CORBA server and interested on the remote object. OMG has proposed a standard set of rules and IDLs for realizing event notification [13]. In an other approach, the client periodically invokes a method on remote CORBA object and analyzes the return value to determine whether the remote object has changed

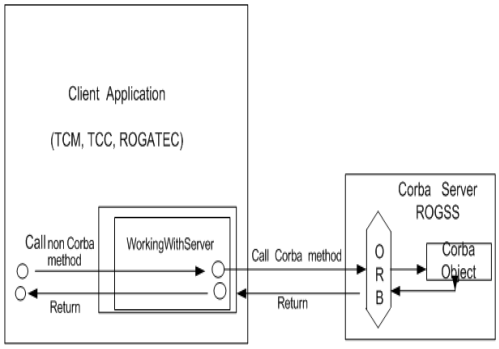

Figure 5. Call TrainCom CORBA method

its state. There are many pros-cons discussions for these solutions.

The first solution is very safe and robust but a separate event server application is necessary that is more expensive to maintain. The communication is little slower and the solution will cause a resource crunch if many clients connect to the same server because for each client there is a thread blocking in server process.

The second solution is easy to implement but increases the network traffic and the real time event notification is not achieved. In our approach, the on-board passengers information about seats reservation and information about linked routes on destination station are not necessary real-time sensitive and then we have adopted the second solution.

We use a periodic call of remote methods "occupiedPlaces" and "prepareRouteData" (diagram of Figure 3). The call mechanism is the same as in Figure (5). The returned information is propagated at TCN level by non CORBA methods.

4. Testing the prototype of CORBA TrainCom Communication Architecture

AS we have presented (Sec. 2, 3), TCCM has a 3-tier architecture. The application has a user interface code layer (an example is in Figure 6), a logic layer, and a Data Base access layer. All interactions between the first two layers occur via the interfaces that all CORBA objects must publish.

The TrainCom Data Base (TCDB) access layer is made up of objects that encapsulate Data Base routines and interact directly with a Data Base server. TCDB is placed on the ground station on the same machine as ROGSS or in a same local network.

The server ROGSS, or logic layer, is server-based code with which the client code interacts via ORB services. The ROGSS is made up of CORBA objects that perform logical functions. These objects invoke methods on Data Base Access layer. The computer, which hosts ROGSS, is placed in the ground station, ROGS.

The user interface layer is the client side and can reside on the user's desktop, on Intranet, or on the World Wide Web (Internet). The GUI implementation is deployed with access to the same server, ROGSS. The GUI usually invokes methods on the logic layer and thus acts as a client of the logic server. The three types clients (see Sec. 2) have different tasks but all access the same Data Base, i.e. TCDB (layer 1) placed on the ground station, ROGS. The access is synchronized.

The test application is composed of three different subsystems:

- SYS1: The local network simulating the ground station ROGS is composed of two computers, the ROGSS and the TCM computer;

-SYS2: The TCC clients is placed in Internet;

- SYS3: The ROGATEC computer and the associated local (TCN) network simulate the on-board network.

The communication test procedure consists in several phases:

1. Test of UDP communication in Intranet (SYS1: ROGSS-TCM link) and Internet (ROGSS-TCC link);
2. Test of UDP communication (GSM link between ROGSS and ROGATEC);
3. CORBA technologies communication in Intranet (ROGSS-TCM link) and Internet (ROGSS-SYS2 link). We use the Dial-Up connection for the ROGSS-SYS2 communication;
4. CORBA communication with GSM link (ROGSS-SYS3 link).

Here is the subsystems description.

4.1 The subsystems description

SYS1

Two computers simulate the ground station ROGS and the associated local network. Each computer is a Windows NT host and has the CORBA software installed. The two computers simulate the server side of the TrainCom application, i.e. ROGSS, and respectively the client TCM. Optional, the ROGSS computer can host the TCM software.

The ROGSS computer has four IP address: 127.0.0.1 - localhost // local feedback address (himself); 192.168.0.1 - device1 // on local station network; 10.0.0.1 - ROGSS // example of IP address with GSM connection and 213.154.145.1 // example of IP address with Dial-Up connection.

The TCM computer has two IP address: 127.0.0.1 - localhost // local feedback address (himself); 192.168.0.2 - device2 // on local station network;

SYS2

SYS2 simulate a local network of TCC clients, that are placed in a remote station. The two computer of SYS2 are Windows NT hosts and have the CORBA software installed. Each computer simulates the client TCC side of the TrainCom application.

The first TCC is the local server and has three IP address: 127.0.0.1 - localhost // local feedback address (himself); 192.168.0.1 - device1// on local station network; 213.233.97.104 - TCC // example of IP address with Dial-Up connection.

The second TCC computer has two IP address: 127.0.0.1 - localhost // local feedback address (himself); 192.168.0.2 - device2 // on local station network.

SYS3

SYS3 simulate the on-board network (TCN). The ROGATEC computer has IP address: 10.0.0.2 - ROGATEC // example of IP address with GSM connection; 192.168.0.1 - device1 // the IP address for on-board network;

For the phases 1 and 2 of the communication test we use our specific Java UDP client-server application based on DatagramSockets. This application transfers a text file from the server to the client. For the phases 3 and 4, which imply a CORBA communication, the files containing IP address must be completed on each machine.

For our GSM communication links we have used a GSM Terminal, type TC 35, made by SIEMENS.

A GUI of TCC application where a seats reservation is realized is presented in Figure (6) and in Figure (7) is presented an example of passengers information on TCN display.

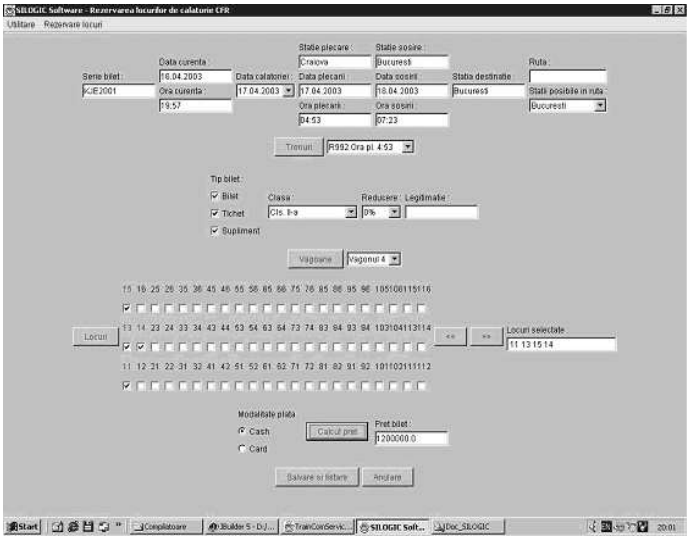

Figure 6. TrainCom Seats Reservation

Figure 7. TrainCom Passenger Information

5. Conclusions and comparisons with related works

This paper reports that CORBA technologies could be used in railway communication systems. We have demonstrated that CORBA is a management middleware easy to use in this activity field. Until this moment we did not find studies and experiments of CORBA technologies int the railway field. But many papers present aspects of CORBA use in other spheres of activity.

In [14] is described the suitability of CORBA as a management middleware in mobile communications systems. Similar with our study an interworking between intelligent networks is reported.

The paper [15] present a "Composite Objects" approach for integrating of CORBA with real-time requirements.

Alike in our paper, in [16] is proposed a web-based network simulation framework to provide a flexible, extensible and platform-independent environment that is suitable for large-scale deployment.

One of our subjects, i.e event notifications from CORBA server to a CORBA client, is presented in [17] but another mode of treatment is used.

References

[1] OMG; THE COMMON OBJECT REQUEST BROKER ARCHITECTURE AND SPECIFICATION, Object Management Group, version 3.0, July 2002, http://www.omg.org

[2] Encyclopedia of Distributed Computing, Kluwer Academic Press,2002

[3] SC SILOGIC SRL ROMANIA, http://www.silogic.ro

[4] TRAINCOM - INTEGRATED COMMUNICATION SYSTEM FOR INTELLIGENT TRAIN APPLICATIONS, http://www.traincom.org

[5] TRAINCOM - PROJECT PRESENTATION, http://www.traincom.org

[6] http://java.sun.com/products/jdk/rmi/

[7] THOMAS A., Enterprise Java Beans Technology - Server Component Model for the Java Platform. White Paper, Sun Microsystems

[8] http://www.microsoft.com/com/tech/DCOM.asp

[9] SIEGEL M., CORBA 3 Fundamentals and Programming (second ed.), Wiley, 2000

[10] OBJECT MANAGEMENT GROUP, http://www.omg.org

[11] BORLAND VISIBROKER, http://www.borland.com/us/products/visibroker

[12] RAJKUMAR R., GAGLIARDI M., SHA L., The Real-Time Publisher/Subscriber Inter-Process Communication Model for Distributed Real-Time Systems: Design and Implementation, Proceedings of the 1995 IEEE Real-Time Technology and Applications Symposium

[13] EVENT SERVICE SPECIFICATION VERSION 1.1, OMG, March, 2001

[14] MUTLU U. AND EDWARDS R., CORBA in Mobile Communications, Electronic Paper

[15] POLZE A. AND RICHLING J., Data Replication and Weak Memory Consistency: Predictable CORBA Interactions with Composite Objects

[16] CHOLKAR A., A Web-based Distributed Network Simulation Framework using CORBA IDL-based APIs, Electronic Paper

[17] JAGASIA M., NARAYANAN A. K., Realizing CORBA Client-Server Event notification using Threads and Synchronisation, Techniche' 2003