IMAGE: a middleware for heterogeneous simulations interoperability

Nancy Rodriguez
Jean-Pierre Jessel
Patrice Torguet
IRIT
rodri@irit.fr, jessel@irit.fr, torguet@irit.fr

Ana Carrillo
LABEIN
ana@labein.es

Jacques Delteil
OKTAL
Jacques.delteil@oktal.fr

Jan Depauw
RHEA System
j.depauw@rheagroup.com

Keywords:
Interoperability, distributed simulation, middleware, CORBA, HLA

ABSTRACT: *Middlewares are used to simplify the integration of applications with different hardware and software requirements. Currently, a wide variety of middleware technologies are made available off-the-shelf to alleviate some of the complexity of developing distributed simulations. In our project, IMAGE, we have designed a middleware-based architecture to support coupling of existent heterogeneous simulators. This paper discusses the IMAGE middleware design based on a combined CCM/HLA technology.*

1. Introduction

When working in a distributed system that is made up of heterogeneous simulators, an intermediary layer of software could be included in order to support the correct interchange of data. This software layer - known as a middleware - must provide services, enabling the applications of the distributed simulation, to communicate in a transparent way. Middleware main concern is to ensure interoperability between different applications (i.e. with separate hardware and software requirements) by defining communication mechanisms. Middleware may also provide security, synchronization, administration, naming services and others common functions used by the diverse applications. Therefore, distributed simulation developers can use the solutions provided by the middleware instead of implementing their own protocols and services, increasing in this way software reuse. COTS[1] middleware technology, now commonly available, can reduce development time and cost of distributed simulations. Several middlewares have been developed for distributed systems, the most significant being OMG CORBA and DMSO HLA.

IMAGE is an European project started in January 2003, which aims at specifying and developing value-added simulations by coupling heterogeneous autonomous simulators together. One of the main goals of the IMAGE project is to create a generic environment facilitating the cooperation between different platforms federated in a same simulation. This paper presents IMAGE and discuss how it benefits from CORBA and HLA features to enable interoperability of existing aeronautical simulations. The paper is structured as follows: First an overview of CORBA and HLA middleware technologies is given. Next IMAGE requirements are briefly described. The technological choice for IMAGE implementation is then presented. The paper finishes with a description of applications deployed with the IMAGE middleware.

[1] commercial off-the-shelf

2. Middleware technology

Applications interoperability can be considered from two perspectives [ref]:

- Technical interoperability will ensure that the systems can be connected together and there is a link to perform the data interchange. Technical interoperability is the capability of simulation participants to physically connect and exchange data. Technical interoperability just can insure that data can be transported from one system to the other.
- Data interoperability will support understanding and correct data interpretation. For systems to meaningfully collaborate, data interoperability (concerning format and semantics of data) is essential.

Interoperability is more easily achieved when a middleware is used to provide common functionality. Middleware mediates between application programs and the underlying operation systems, network protocol stacks and hardware, in order to coordinate how parts of the applications are connected and how they interoperate [ref]. It enables and simplifies the integration of components developed independently.

HLA (High Level Architecture) is a standard for distributed simulation focusing on interoperability of geographically dispersed simulation applications. CORBA (Common Object Request Broker Architecture) is a widely accepted middleware supporting distributed software systems. CORBA and the CORBA Component Model (CCM) allow objects to interoperate across networks regardless of the language in which they were written or the platform on which they are deployed. It offers complementary services to those of HLA.

2.1 HLA

The High Level Architecture (HLA) is a general architecture developed to allow reuse of US Department of Defense (DoD) simulations. In HLA, a federation is a collection of federates (simulations) designed to operate together to form a distributed virtual world. HLA is defined by three components: a set of rules for building and executing distributed simulations (HLA Rules), an interface specification for the Run-Time Infrastructure (RTI) to provide certain services for data sharing and coordination between these simulations, and a standard Object Model Template (OMT) format for describing the data and interactions to be shared between these simulations.

The HLA Rules is a set of rules that must be followed to achieve proper interaction of simulations in a federation. They describe the responsibilities of simulations and of the runtime infrastructure in HLA federations. These rules are divided into two groups: federation and federate rules. Federations, or sets of interacting simulations or federates, are required to have a FOM in the OMT format. The Federation Object Model (FOM) offers a precise description of the objects and interactions that the federates will share. During runtime, all object representation takes place in the federates, with only one federate owning any given attribute of an instance of an object at any given time. Information exchange among federates takes place via the RTI using the HLA interface specification.

Additional rules apply to individual federates. Under the HLA, each federate must document their public information in their Simulation Object Model (SOM) using the OMT. The Simulation Object Model (SOM) describes the data that a particular federate shares with the federation. The SOM is specific to the given simulator application and every federate must have a defined SOM. A federation's FOM is composed of parts of the SOMs of all of its participating federates Based on the information included in their SOM, federates must import and export information, transfer object attribute ownership, updates attributes and utilize the time management services of the RTI when managing local time [ref].

The RTI is a distributed operating system for the federation. It provides a set of general-purpose services that support federate-to-federate interactions and federation management and support functions. The HLA runtime interface specification provides a standard way for federates to interact with the RTI, to invoke RTI services to support runtime interactions among federates and to respond to requests from the RTI. There are six basic RTI service groups: Federation Management, Declaration Management, Object Management, Ownership Management, Time Management and Data Distribution Management. The HLA interface specification defines the way these services are accessed, both functionally and in an application programmer's interface (API). The HLA API is defined as a series of procedural functions that the application developer uses to communicate over the network. At present, APIs in CORBA IDL, C++, Ada and Java are incorporated in the interface specification.

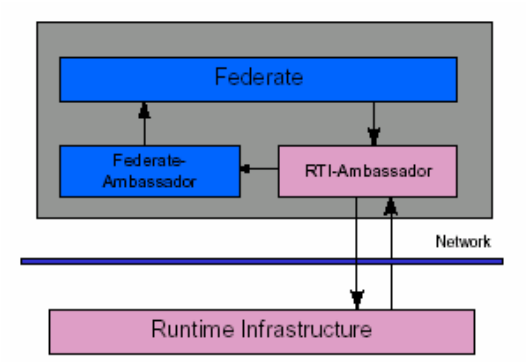

Figure 1. The RTI-Federate communication

Federates communicate with the RTI through the RTI-Ambassador and the FederateAmbassador (Figure 1). The RTI Ambassador is a component provided with the RTI to enable a federate to request services from the RTI and to publish information to other federates in the federation execution. The Federate Ambassador is the mechanism the RTI uses to communicate back to the federate e.g. to reflect information to the federate during federation execution. The Federate Ambassador is a component of the RTI with a well-defined interface. The developer is responsible for implementing the desired features while complying with this interface. The federate specific code performs all simulation activities and communicates information to and from the RTI via the RTI Ambassador and the Federate Ambassador.

2.2 CORBA

The OMG's Common Object Request Broker Architecture (CORBA) is a standard specification providing an architecture for the construction of distributed object-oriented applications [Otte96]. It is part of the Object Management Architecture (OMA) defined by the Object Management Group (OMG) and it specifies the Object Request Broker (ORB) component of OMA. It has been developed to answer the need for interoperability among the rapidly proliferating number of hardware and software products [ref]. CORBA allows objects to communicate with each other independently from their physical location, the language in which they were written or the platform on which they are deployed [ref].

In CORBA, an object is an encapsulated entity whose services can be accessed only through well-defined interfaces. Object interfaces (the object available services) are specified first in the OMG Interface Definition Language (OMG IDL), which is then compiled into code in one of the supported languages (including C, C++, Ada and Java). Client and server “stubs”, each of which is an interface to the actual object on the server, are generated from IDL. The server stub is often referred to as a “skeleton”. Only the skeleton code is used on the server, while only the stub code is used on the client. The object services are implemented in the chosen language using the generated code as the basis. With IDL, object interfaces definitions are separated from object implementations. This allows communication between objects constructed using different programming languages and development of several implementations based on the same interface.

In a CORBA application, clients issue request to objects to perform services. All messages between client and server objects must go through the Object Request Broker (ORB). The ORB provides 'transparent' communication services: objects are not concerned about the implementation details of the underlying network services or the server objects.

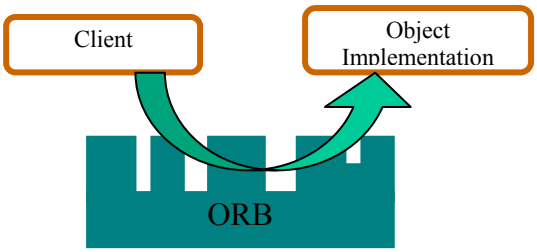

Figure 1. A CORBA request

The ORB consists of software residing on both client and server machines. When a client makes a request to a CORBA object, the message is processed first by the ORB on its machine. The client ORB establishes communication with the ORB on the server, which answers with a reference to the requested object. This reference is then passed to the client ORB who returns it to the client making the request. The key feature of the ORB is the transparency of the client/object communication. It allows application developers to work in their own application logic rather than in low-level distributed system programming issues like object location and implementation, communication protocols, etc.

Server and client ORBs communicate via the Internet Inter-ORB Protocol (IIOP). The IIOP ensures that CORBA-based applications can interoperate across computer architectures, operating systems, programming languages and ORB vendors. The IIOP uses the TCP/IP reliable network protocol to pass data point-to-point between the clients and server for each information transfer [ref]. .

The services one might expect to find in a middleware product such as CORBA (e.g., naming, transaction, and asynchronous event management services) are actually specified as services in the Object Management Architecture (OMA). OMA is itself a specification (actually, a collection of related specifications) that defines a broad range of services for building distributed applications.

OMA services are partitioned into three categories: CORBAServices, CORBAFacilities, and ApplicationObjects. CORBAServices are considered fundamental to building non-trivial distributed applications. These services currently include asynchronous event management, transactions, persistence, externalization, concurrency, naming, relationships and lifecycle. CORBAFacilities may be useful for distributed applications in some contexts, but are not considered as universally applicable as CORBAServices. These "facilities" include: user interface, information management, system management, task management and a variety of "vertical market" facilities in domains such as manufacturing, distributed simulation, and accounting. Application Objects provide services that are specific to an application or class of applications. These are not (currently) a topic for standardization within the OMA, but are usually included in the OMA reference model for completeness, i.e., objects are either application-specific, support common facilities, or are basic services.

The CORBA Component Model architecture is based on the CORBA Middleware. In short, the idea of component-based development is to divide an application into small reusable components that can be connected to other components via ports, or, speaking the other way around, to compose applications by reusing and interconnecting existing components [ref].

Below is a comparison table to summarize the differences among CORBA and HLA.

CORBA	HLA
Object Interface Language	
IDL (Interface Definition Language) Stubs and skeletons automatically generated by the IDL compilers.	OMT (Object Model Template) While the FOM describes the simulation's data elements providing the necessary contract between Federates in a Federation, it does not specify uniform programming language mappings for these elements.

CORBA	HLA
CORBA guarantees interoperability Defines mechanisms to find objects dynamically (Naming Service, Trader Service).	This often leads to diverging implementation practices that adversely affect the reusability and extensibility of the implementation In HLA the responsibility of interoperability depends on the languages specified in RTI FOM is static
Time Management	
CORBA is missing services for managing time between simulation applications.	The Time management services provided by the HLA allow the transparent running of federates under different time regimes including real-time, time-stepped and event-driven simulations.
Interoperability between service providers	
ORB to ORB Internet Inter-Orb Protocol (IIOP) General Inter-Orb Protocol (GIOP)	RTI to RTI: HLA does not specify a protocol, but leaves the choice up to the RTI implementers.
Data interchange	
CORBA supports direct communications between objects. The CORBA Event Service supports communications with multiple suppliers and consumers based on the publish/subscribe paradigm.	HLA allows publishing and subscribing to attributes and interactions.
Ownership	
In CORBA, a server always owns an object instantiated by it.	HLA supports transfer of object ownership.
Platform Independence	
Applications code is portable because stubs/skeletons are used in the same way, independently of the ORB	Code must be rewritten

3. IMAGE Requirements

IMAGE main goal is to specify and develop a standard environment to insure the deployment of complex simulations for the aeronautical industry. Our objective is not to develop new simulation software but to interconnect existing ones and to convert them into standard components. To do so, existing software (with no access to software source code) will be encapsulated to give them compatibility, in certain specific cases encapsulation will be done through an interface module (when access to software source code do exists). Therefore, IMAGE focuses on such themes as interoperability, modularity, flexibility, cost reduction, etc.

The project scientific and technical objectives are to:

- Undertake research on a set of methods and component libraries to insure the deployment of complex simulations.
- Achieve a technological development that enables simulation designers to make the best of their available software and hardware resources.
- Implement research results validation into three prototypes (real time, non-real time and mixed real/non-real time simulators and simulations applications); Industries will later be able to integrate this in their own already existing systems.
- Accomplish a first important step towards distributed simulations and simulators that are no longer restricted to development experts.

We made an initial consultation of our partners and clients in order to specify clearly the needs they encounter. All the expressed needs cannot be developed in the project's framework but will prove useful to open the conception of our environment in order to integrate them thereafter.

Through the applications they develop or use, items associated to systems interoperability appear as crucial to our users:

- Allowing the cooperation between heterogeneous simulations, in terms of hardware, software, data, network protocols, etc.
- Handling the constraints of synchronous/asynchronous modules through a simulation supervision module.
- Remaining coherent in the sense that all the distributed simulation participants must be aware of each other's action on objects concerning them.
- Exchanging data among simulation modules with few manual interactions.
- Keeping an acceptable level of performance.

We also know that standardization and normalization are an obligation if we want the IMAGE system to support extensions, new modules and to remain modular. The IMAGE system must then provide development rules and tools so that the user can comply with this requirement as easily as possible.

Nowadays, many simulation applications already exist which could be used to develop simulations of higher complexity. Therefore, reusing the existing simulators or simulation modules is a requirement very strongly expressed by users. The examples of existing simulations that would be interesting to make cooperate are numerous. For example, in order to improve the realism of a simulation, it would be interesting to couple a flight simulator (real time) and all the modules simulating the meteorological data (non real time). To support an application of this kind we must study the interoperability and distributed aspects of the application as a whole: way to communicate in a generic manner between heterogeneous and distributed processes, location of simulation modules, analysis of computers resources in terms of CPU, memory and disk, allocation of resources and time scheduling of the different simulation modules… The IMAGE project aims at providing a standard environment that helps the aeronautical industry to find a solution to these difficulties arising when building more and more complex simulations required by the aeronautical industry.

Users consultation and the scenarios like the one described above helped us to refine our analysis and to define the services of the IMAGE API. A further selection of services to be considered for IMAGE prototyping was derived based on the priority scoring provided with the initial user requirements gathering. This resulted in a set of services to be considered for prototyping:

- Security:
  Information Hiding, Authentication, Access Control, Integrity Checking, Non-Repudiation, Security Administration;
- Administration:
  Federation Management, Declaration Management, Object Management, Ownership Management, Time and Time Advancement Management, Data Distribution, Transportation, Configuration Management, Configuration Maintenance Management;
- Supervision:
  Simulation Entity, Simulation Time, Simulation State, Simulation Management, Simulation Recording and Replaying, Simulation Controlling Tools, Simulation Testing Tools, Assembling and Deploying Simulation Components, Clustering, Loads Balancing, Error Handling, Fault Tolerance;
- Middleware
  Event, Publish and Subscribe, Scheduler, Network Protocols, Communication, Data Management, Analysis and Statistics, Repository, Interface, Wrapper, Gateway, Naming & Trader, Marshalling/Unmarshalling, Remote Procedure Calling.

In order to be able to derive a detailed design of the IMAGE backbone being comprised of these services, a set of tools have been surveyed and analyzed. The following major technology families were identified for this: CORBA Component Model, HLA, Java Beans, and Grids. Next section details technologies evaluation and presents the technical choice we have adopted for IMAGE.

4. IMAGE Technological Choice

For implementing IMAGE services, we have evaluated the most significant middleware technologies (CCM, HLA, J2EE, Globus) according to the criteria defined by user and technical IMAGE requirements. From literature, we know that CORBA CCM is the most generic middleware, because it is platform and language independent (vs. J2EE or COM/DCOM). It isn't dedicated to a specific application domain, which is presently not the case with HLA. It allows any kind of application to use it. CORBA offers the followings advantages: automating software generation (using description language and compilers), standardizing model description using description language (IDL, UML, etc.), providing generic services (naming, event, scheduling, trading).

The technology choice task in IMAGE consists in identifying possible technologies applicable for the IMAGE project, defining important criteria, and evaluating each technology according to all criteria. It was observed that on many occasions the major characteristics of representatives of separate technology classes complemented each other. From this observation, it was decided to not only evaluate the individual technologies within each technology class (i.e. single technology evaluation), but to perform evaluation of technology tuples formed by members of separate technology classes (i.e. dual technology evaluation). This dual technology evaluation would then need to also consider adaptations to the scoring against the evaluation criteria as each individual technology representative in the tuple carried its own evaluation score against the various criteria. The values of both the single and the dual technology evaluations have then been compared to allow for the selection of the best technological solution for the IMAGE implementation.

Considering the available evaluated set, an interesting combination would be CORBA CCM and HLA. It is clear that the choice of only one of these two types of technologies leads to the absence of adequate technical solutions for the other part of the problem. For instance, taking CCM only would remove the advantages of abstract simulation time concepts, the benefits of the Object/Interaction way of doing simulation, and so on. On the other hand, taking HLA only would remove all practical solutions for doing load balancing, clustering, remote procedure calling, etc. Gathering the benefits of two different technologies is also good if the technologies rely on widely accepted or normalized ways of working. It is the case for CORBA and CCM in the distributed computing community, as well as for HLA in the Distributed Simulation community. HLA has been adopted as the specification for simulation by the Defense Modeling and Simulation Office. IMAGE users will certainly feel more comfortable if they see that IMAGE relies on well-normalized and widely accepted concepts.

From this discussion, it has been decided that within the context of the IMAGE project all further design and development activities would be performed based on the combined CCM + HLA technology selection. We have chosen to combine CCM and HLA and gather the benefits of these two different technologies. By adopting technologies that complement each other, IMAGE will minimize the amount of efforts needed to implement the system. Prototyping can be done easily to prove concepts, since the main underlying implementations are already at hand. Therefore, IMAGE can really focus on what is truly missing, i.e. tools that will ease the use of, or the adaptation to these technologies in a consistent way.

5. The IMAGE Middleware

The IMAGE project aims at specifying and implementing a generic environment in order to make complex, distributed and heterogeneous integrated simulators cooperate. To achieve this goal, IMAGE provides various simulation services and is based as discussed above on two different technologies: CCM for the functional coupling and HLA for the simulation coupling.

Functional coupling consists of all interactions between two systems that are to function in collaboration on the application level. Simulation coupling guarantees that all simulators participating in describing the common simulated world do it so that all the other participants can understand it. It supposes the definition of a common language. The simulation coupling is a little different than the functional coupling. There are different rules that apply to events that are linked to the description of the simulated world. It does not consist in function calls, threads, return values, but in elements that are referring to a certain conception of reality. The IMAGE system insists that a language should be defined in order to express what is happening in the simulated reality. This language is defined in terms of two constitutional elements: Simulation Entities and Simulation Events. A Simulation Entity is something in the simulated reality that has a beginning and an end in time. It is also composed of attributes, which value can change between its appearance and disappearance (during its lifetime). Simulation Entities are to be opposed to Simulation Events, which

are events that take place at a particular point in time (they have no lifetime). A Simulation Entity is an element of the common language simulators will use to communicate. Simulation Entities have a lifetime, as opposed to Simulation Interaction, which are atomic events related to the simulated world.

IMAGE first prototype has been built using TAO/CIAO and DMSO's HLA 1.3NG implementation. TAO is a C++ ORB that is compliant with most of the features and services defined in the CORBA 3.0 specification, which includes the Real-time CORBA specification. TAO is based on the standard OMG CORBA reference model, with the enhancements designed to support high-performance and real-time applications. A subset of services defined in the design task are deemed to be already available without further development through the TAO/CIAO and HLA 1.3NG libraries. For the other IMAGE services it will either need to be adapted or developed based on the technology libraries and additional external developments. This concerns mainly the security issues and some administration and supervision services such as Simulation Recording and Replaying and Configuration Maintenance.

IMAGE services and generic components were implemented in a real size concrete industrial production context. We insist on the fact that the applications developments' approach is totally new: the generic components once prototyped allow us to launch full real size cooperative executions of real-time simulation, numerical simulation and hybrid simulation. The three applications were selected with the purpose of validating the integration of the IMAGE components. They are:

- A real time simulation (full flight simulator coupled with ATC training simulator and computer based training). This validation case uses IMAGE components for communication and management of ATC simulation processes.
- Numerical simulation application. This validation case demonstrates that the IMAGE results are applicable to the entire simulation domain. Presently, computing methods such as CAE, CAM, simulation, virtual product management are numerous, efficient but heterogeneous and non-compatible. The generic IMAGE result is an environment for interactive simulations, allowing an easier simultaneous operation of these different tools. The genericity of IMAGE's project applications is tested on a non-real time numerical simulation.
- A mixed real-time and numerical simulation application. The task explores how to use IMAGE supervision, administration and security components for the set up and execution of an heterogeneous real-use simulation application based on a mixed set of numerical and real-time simulations. The application involves the use of wind tunnel numerical and real-time planification and scheduling simulations. The application is tested against a set of predefined application functional and performance requirements.

The validation applications were modeled, assembled and deployed using IMAGE as their development environment. The deployed applications allowed heterogeneous and multi platform applications to interoperate. In the performed validation cases, the various applications that have to interoperate run on heterogeneous and distributed platforms. Moreover, these applications made their simulation components communicate with others simulation components, which are located in a distant site, making possible the communication capability between applications currently non compatible.

The validation applications were able to demonstrate that the IMAGE services are fulfilled in the three cases (real-time application, numerical application and mixed real-time and numerical application), which corresponds to real size concrete industrial production context. Therefore it demonstrated IMAGE's ability to develop the three kinds of applications.

6. Conclusion

The IMAGE project aims at implementing a generic and universal environment in order to make complex, distributed and heterogeneous integrated simulators cooperate. We have evaluated several technologies for distributed computing development which fall in two categories of technologies that target distributed computing and technologies that focus on high level distributed simulation concepts.

For the IMAGE middleware implementation, we have chosen to gather the benefits of two different technologies : HLA and CORBA CCM. This allows us to make the most of distributed simulation concepts and mechanisms provided by HLA and to benefit of all practical solutions for doing load balancing, clustering and remote procedure calling supplied by CORBA. Our experience indicates that a selection of complementary technology tools provided a better coverage of the design and development needs of our project.

The validation of the IMAGE prototype through the development of several complex simulations demonstrate that the initial objectives of the project have been achieved. The implemented middleware for real time and numerical simulations applications is a step forward in the integration of different softwares, the environment makes possible the communication between heterogeneous software/hardware and enables the optimization of the sharing of computers and network resources.

7. References

[1] J. Dahmann: "HLA and Beyond: Interoperability Challenges", Fall Simulation Interoperability Workshop, 1999

[2] R. E. Schantz, D. C. Schmidt: "Middleware for Distributed Systems, Evolving the Common Structure for Network Centric Applications", Encyclopedia of Software Engineering., Wiley & Sons, New York, 2001

[3] Defense Modeling and Simulation Office : "RTI 1.3-Next Generation Programmer's Guide Version 5", February 2002

[4] Object Management Group: "The Common Object Request Broker: Architecture and Specification" Version 3.0, December 2002

[5] S. Vinoski : "CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments", IEEE Communications Magazine, Vol. 35, No. 2, 1997

[6] A. Buss, L. Jackson: "Distributed Simulation Modeling: a Comparison of HLA, CORBA and RMI", Winter Simulation Conference, 1998

[7] Object Management Group : "CORBA Components Specification Version 3.0", June 2002