DISTRIBUTED SIMULATION AND MODELING ENVIRONMENT

Valentine PENEV, Tatiana ATANASOVA and Ivanka VALOVA

1. Introduction

Distributed simulation has become a widely popular and useful tool for various applications, including military simulation and training, for example, ModSAF.

The goal of Distributed Simulation and Modeling (DSM) is to create a virtual world incorporating various simulators distributed over the network. It needs to manage different kinds of time infrastructures and communications strategies, based on real-time information processing.[1,2] A rich variety of simulators for different purposes have already been developed, but the integration of these simulators into a common framework needs further investigation. The work consists of simulation, seamless integration, and distributed virtual environments. The virtual environment includes its two-dimensional representation, three-dimensional visualization, physical simulation, behavior, and networking.

In this paper a distributed environment which shares the computational resources is proposed. Distributed simulation systems use models, visual representation and calculation of object movements. The network distribution allows remote access to heterogeneous data sources and functionally divided operation of the system. The status of the environment and its parts are updated and sent over the network so that all participants are fully aware of the environmental situation. Different optimization algorithms are used for tasks of simulation, computation, and visualization.

2. Distributed System for Modeling and Simulation

Simulations are very complex systems with many interactions between the involved variables.[2] A lot of computations is needed to perform the various simulation tasks. Figure 1 presents the proposed scheme for modeling and simulation system. Different

parts of the synthetic environment with their corresponding algorithms are distributed among several computational resources connected by a network.

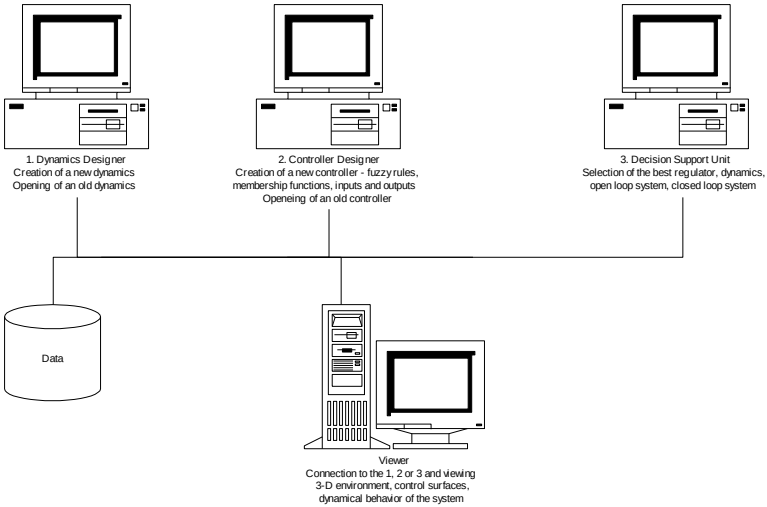

Figure 1: DIS architecture

The blocks in this scheme have the following functions:

Dynamics Designer (DD) of the simulated objects:

- Creates a new dynamics set of modeling objects
- Reads and shows old dynamics set
- Defines the dynamics set inputs for test
- Saves a dynamics set
- Displays dynamics set and dynamical response to the inputs
- Saves dynamical response

Controller Designer (DC):

- Creates a new controller
- Reads and shows old controller
- Defines controller input for test
- Saves controller
- Displays controller and controller output behavior to the inputs

- Saves controller output behavior

The primary function of this block is to create appropriate controller for the defined simulated object.

Decision Support Unit (DSU):

Performs selection of the best regulator, dynamics, open loop system, closed loop system on the base of defined criteria. DSU consists of several integrated parts with approximate reasoning capabilities. The system provides fuzzy reasoning combining fuzzy sets, hedges and fuzzy if-then rules into non-procedural collections – policies.

Viewer defines connections between Distributed Knowledge Data Base, DD, DC and DSU:

- Creates new loop (dynamics, controller). The loop consists of the following items – Dynamics, Controller and connections between them.
- Reads and shows old loop
- Defines input to the loop
- Saves loop
- Displays loop dynamical response to the inputs
- 3D – viewer of behavior of object and control surfaces
- Saves dynamical response of loop

Viewer utilizes the information about the simulated object, its dynamics, behavior and environmental conditions. Solutions for the visualization include interactive real-time 3D displays of virtual worlds in a range of graphics tools and interfaces including Direct3D and others.

Distributed Knowledge Data Base (DKDB) is represented by series of data elements, rules, documents and software modules. DKDB keeps the following kinds of data:

- Dynamical set - equations and numerical coefficients.
- Inputs to Dynamical Set - step, linear, sinusoidal, joystick.
- Controller - The controller has the following characteristics: inputs and outputs, fuzzy input and output variables, membership functions, fuzzy rules and defuzzification procedure.
- Dynamical responses - time sequences of dynamics outputs and state variables.
- Environmental data. An essential step in environmental modeling is the collection or creation of data to represent the specific type of environment required. This includes information about the terrain, atmosphere, ocean surface, floor, vegetation, imagery, etc. The list is extensive and the storage requirements are large. The terrain database is distributed over the network so that any

simulator entering the virtual world could get background information from the network.

Data units are sent over the network by addressing them to a particular computer and preprocessing the data to satisfy some specific types recognized by the receiving program. The calculation process is implemented using modular applets (ActiveX) which automatically gather the corresponding data over the network. The simulation also receives or sends data produced from user inputs.

3. Description of the database

Entity in the DKDB describes any physical or virtual object defined by a set of properties such as its geometrical dimensions, geographical position, dynamical response, behavior, modeling parameters, etc. It has attributes and functions (variables and methods). The attributes are as following:

- Entity ID: a unique reference identifying each object.
- data field: it varies for different categories of entities. The field contains two sorts of information: parametric data and simulation data. Parametric data could be downloaded dynamically from a remote data base at the initialization of this kind of entity. The simulation data would be initialized at the creation of the entity through its parent. It consists of the following information:
    - entity type: a universal type identifier
    - status
    - time step and time synchronization
    - representation (from parametric data): properties used for entity modeling and simulating
- link to the related entity
- universal locator: a unique address

The entities in the DKDB are dynamical models of simulated objects, controllers, a set of views, predicted trajectory, documents, graphics, methods, software modules.

3.1 Functions of the entities

The methods attached to an entity differ depending on the entity. Some common functions that each entity implements are:

- Initialization of the parametric and simulation data through the remote data bases
- Update data field

- Update entities
- Get data
- Create entity
- Remove entity
- Change computational resource
- Specific - this category of function depends directly on the type of entity

This structure enables an entity to have function. It is useful for the surface modeling, because in this case the entity uses its own modeling algorithm, without depending on whether it would be grass, sand, or soil. For military applications, this could enable a terrain to have mines, hydrology models, vegetation, ground characteristics, instead of just a color or a texture map.

The description of data is based on HLA - the High Level Architecture. HLA is the standard technical framework for U.S. Department of Defense simulations. The HLA specifies the interface that simulations must use to communicate. This interface is implemented in the Run-Time Infrastructure (RTI).[3]

The developed services take advantage of network processing to improve distributed simulation accuracy, scalability, and performance. Beside this the computational and data caching, transport protocols, and optimized algorithms for locating and activating active code within a network are considered.

4. Integration of information applications in the DSM System

The interchange of information applications needs to be considered at four levels: data level; application program interface level; method level; and user interface level.

Data-level approach is concerned to the information extracting from one database, processing that information as needed, and updating it in another database. *Method-level approach* is the sharing of the logic that exists within the system. By sharing the logic or methods contained inside or outside any given application it is possible to share both data and processes among many applications, and therefore integrate the applications. *Application program interface-level* supports the interfaces to access both simulation processes and simple information (data). Using these interfaces, many applications work together, sharing logic and information to access processes and data, extract the information, place it in a format understandable by the target application, and transmit the information. Message brokers are one of the solutions, they are able to move messages among different parts of the system to reformat the messages, so they are understood by the target application. At *User interface-level* information is integrated on the basis of user interfaces as a common point of integration.

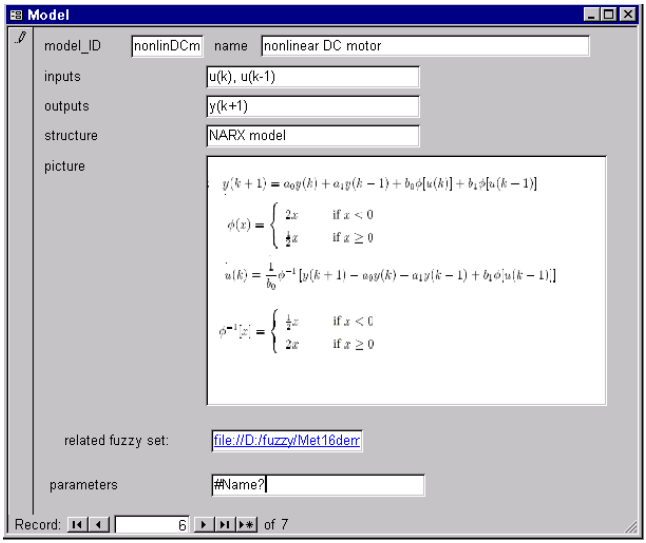

Figure 2.: Preview of Dynamic Designer

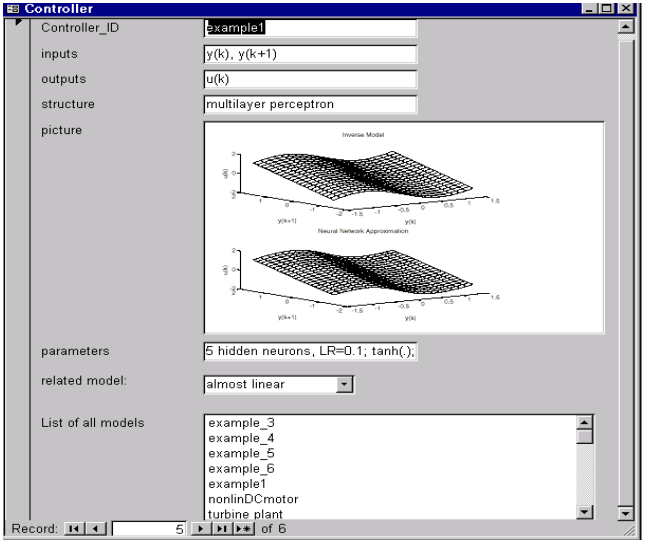

Figure 3: Preview of Controller Designer

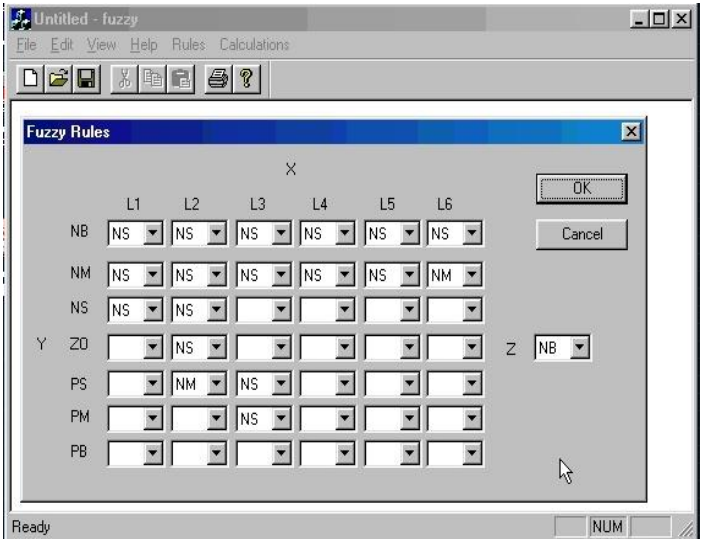

Figure 4: Preview of Fuzzy Rule for Controller Designer

Figure 5: Preview of Viewer

4.1 Enabling Technologies

The integration of information is necessary for creating environments that realize collaboration with distributed computing and modern modeling and simulation technologies. It relies on the 3-tier architecture with the distributed object middleware, visual front-ends and suitable simulation, computation or information/database objects in the back-ends. In the front-end, the most promising collaboratory technologies include: Java3D, VRML, DirectX, XML.

The DirectX is a common infrastructure from Microsoft Inc. in the implementation of high performance real-time applications of modeling and simulations. The component of DirectX that supports multiplayer networked applications is called DirectPlay. The functions provided by DirectPlay are similar to those provided by the HLA RTI with a few significant differences. DirectPlay provides some features specific to gaming. However, it does not provide any time synchronization features; it is built to support a DIS-like, loose causality model. DirectPlay also differs from the RTI in that it does not provide any support for determining data routing. In DirectPlay, there is no concept of a common object or data definition.

DirectPlay is possibly the more suitable technology for multiuser PC games. However, combining DirectPlay with concepts of DIS and HLA might create a hybrid technology that could be useful for both military simulations and the entertainment industry.

Conclusion

We have presented a distributed Modeling and Simulation environment. The scalability limitations dictated by existing networking technologies continue to obstruct the achievement of high-performance, large-scale, distributed simulation. The objective of our future work is to exploit the novel capabilities of active networks to overcome these limitations through the development of advanced protocols and services specifically for distributed modeling and simulation. The future key research areas include using active network tools to provide dynamic management, computational and data caching in distributed data sources, and to interchange data from heterogeneous sources.

References

1. Roger Smith, *Military Simulation Techniques & Technology*, Detailed Course Outline. Available at http://www.simulation.com. An abridged version is published in the current issue of I&S.

2. Katherine L. Morse, *Interest Management in Large-Scale Distributed Simulations*, UC Irvine, Information and Computer Science Technical Report, ICS-TR-96-27.
3. IEEE Standard for Distributed Interactive Simulation - Application Protocols, IEEE 1278.1-1995.

VALENTINE PENEV see page 90.

TATIANA ATANASOVA received her Diploma in Automatics from the Power Institute in Moscow. Currently she is research associate at the Institute of Control and System Research. E-mail: tatiana@iusi.bas.bg.

IVANKA VALOVA is a researcher at the Institute of Control and System Research. E-mail: vania@iusi.bas.bg.