Creating and Running Mobile Agents with XJ DOME

Kirill Bolshakov, Andrei Borshchev, Alex Filippoff,
Yuri Karpov, and Victor Roudakov

Distributed Computing & Networking Dept.
St.Petersburg Technical University
Experimental Object Technologies (XJ)
195251 St.Petersburg Russia
dome@xjtek.com
http://www.xjtek.com

Abstract. XJ DOME is a set of tools and techniques for those who wish to speed up development of Distributed COM applications and improve their quality. DOME supports graphical modeling, code generation, simulation, deployment, monitoring and management. The simulation mode enables the developer to simulate the entire distributed application in virtual time on a single machine. After simulation step the application can be deployed onto the target network and managed via DOME Application Viewer. During run-time DOME platform enables the developer to collect and watch statistics, inspect threads and synchronization objects, view logs. DOME platform supports building of mobile agent systems on top of DCOM services. It provides for agent migration and employs DCOM security.

1 Introduction

Mobile agents are attracting interest from fields of distributed systems, information retrieval, electronic commerce and artificial intelligence as a rapidly evolving technology. This area suffers a lack of industrial-strength development tools support.

In this paper we present XJ DOME – run-time and development environment for building mobile agent system on top of Microsoft DCOM services [2,3]. It also provides graphical specification of object behavior, as well as simulation and visualization services. XJ DOME may tightly integrate with MS Visual C++ Developer Studio. Thus, DOME provides the developer with friendly environment from specification through debugging to real execution stage.

2 The Current State of Mobile Agent Systems Market

The vast majority of the agent systems available are research prototypes and only a few of them have users outside their own university or research institute [1]. The most known are Aglets (IBM, Japan), Mole (University of Stuttgart, Germany), Telescript

V. Malyshkin (Ed.): PaCT-99, LNCS 1662, pp. 410-416, 1999.

(General Magic, USA) and AgentTcl (Dartmouth College, USA). Aglets and Mole sup-port Java, AgentTcl supports Tcl, Telescript has its own language.

However, there are no high-level rapid application development (RAD) environments for agents development. Also, at the moment there is no mobile agent system based on DCOM platform. The implementation of mobile agent system for DCOM enables to utilize its performance advantages (execution of native code and full access to OS services) and integration with MS Windows NT security.

3 XJ DOME Run-Time Environment

XJ DOME run-time environment supports the following models of mobile agents:

- Lifecycle
- Computational
- Communication
- Navigation
- Security

The lifecycle model provides services to create, destroy, save and restore mobile agents. The computational model heavily relies on Win32 services at the moment. The navigation model handles all issues referring to transporting an agent between two places. The communication model defines communication between agents. The security model defines rules of mutual access for agents and network.

We build our communication, navigation and security models on top of Microsoft DCOM. This allows the developer to fully exploit the advantages of Microsoft industry standard for Windows environments. This also greatly simplifies dealing with numerous security issues intrinsic to mobile agent systems.

4 DOME Application Editor

The basic services provided by XJ DOME Developer Studio are:

- Creation of COM components in visual environment with complete code generation (for both static and dynamic components)
- Debugging of timings and synchronization and performance estimation of a distributed application by simulating it in virtual time on the developer's machine
- Deployment of the application components over the target network
- Monitoring of the distributed application: collect and display statistics, inspect threads and synchronization objects, view logs, etc.
- Management of the distributed application via COM interfaces using standard controls

For rapid prototyping of distributed COM applications XJ DOME offers Application Editor, see Figure 1. It includes graphical COM Object Diagram and Statecharts

editors. DOME Application Editor generates the complete application code including IDL, C++, resources, and MS Visual C++ project. It builds the application components using MS Visual C++ command-line compiler. DOME Application Editor drastically saves developer's time on the early design stages. Later on, when "COM skeleton" of the application becomes more stable, the developer can continue with MS Visual C++ environment using built-in DOME Wizards.

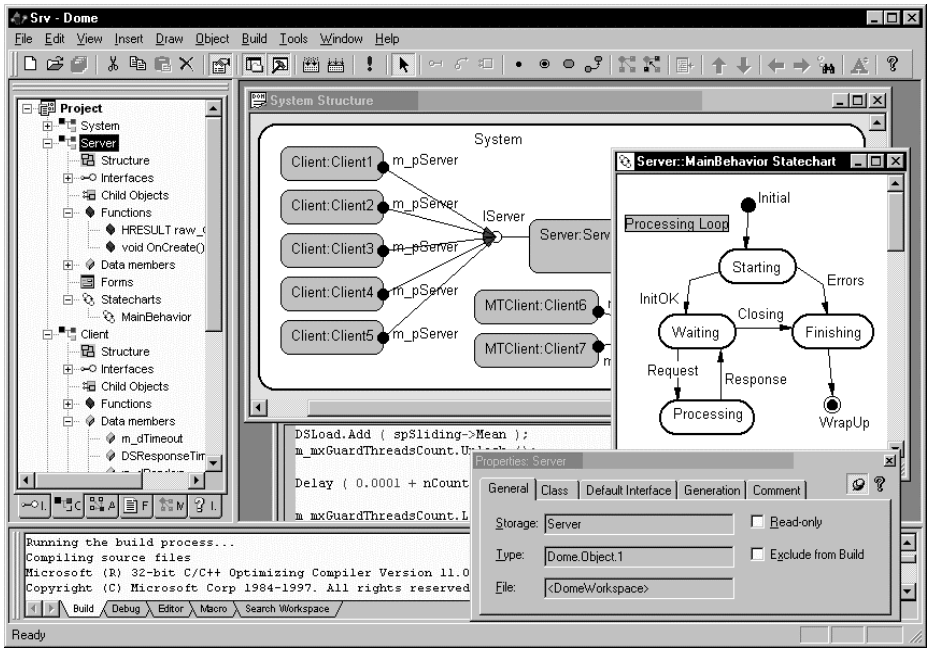

Fig. 1. XJ DOME Application Editor

5 Wizards for MS Visual C++

DOME can be used with new projects as well as with the legacy software. In the latter case DOME functionality can be added to the application gradually. By following a set of simple rules one can easily instrument COM objects under development to use DOME simulation, deployment, visualization and management facilities. To automate this work DOME adds to MS Visual C++ development environment a set of wizards covering all related tasks:

- *DOME App Wizard* — creates a skeleton application project.
- *Add DOME Object* — adds an object with support of standard DOME interfaces and skeleton implementation to the application project.

- *Add Child Object* — adds aggregated or contained child object to DOME object.
- *Add COM pointer* — adds a COM pointer to DOME object. DOME Objects communicate by calling methods of each other via COM pointers that are in turn set up by DOME run-time environment.
- *Add DOME Thread* — adds a control thread to DOME object. DOME objects can have several control threads and spawn them dynamically.
- *Add DOME Statistics* — adds an ability to collect statistics to DOME object. On execution stage the statistics is available for monitoring in DOME Viewer.
- *Add DOME Log* — adds a log access point to DOME object. DOME a feature that allows the user to watch logs of several objects deployed to different hosts.

6 Simulation

Simulation is used for preliminary analysis of the application correctness and estimation of its performance. In the simulation mode the most detailed information on threads and synchronization objects is available online in DOME Application Viewer, see Figure 2. The user can run the application step-by-step, stop upon a certain condition, e.g. when enough statistics is collected, etc. Using automation the developer can program DOME to run multiple simulation sessions to find optimal parameter values or to test the application scalability. All the simulation is performed on a single developer's machine.

DOME simulates the application in virtual time, thus making arbitrary complex experiments possible on a single workstation.

Simulation is supported by DOME Engine that implements IDomeEngineSite interface. The application objects developed according to DOME technology invoke the functions creating new objects, threads, synchronization objects, delaying thread execution, waiting for events, etc. through IDomeEngineSite. In the normal execution mode such call is transparently passed to the local operating system (in fact, the call does not even leave the local machine, as the engine is represented there by a lightweight DOME Engine Proxy object). In the simulation mode, the engine takes care of thread scheduling, synchronization and time.

7 Visualization

The user can monitor the running application with DOME Application Viewer. The viewer retrieves the information via IDomeObject interfaces implemented by DOME-compatible application components and displays the global picture of the application, including:

- Application objects and their hierarchy
- Threads
- Synchronization objects

- Statistics
- Logs
- Inspection views
- `IDispatch` interfaces of objects

The details of the displayed information depend on the execution mode. Namely, in the simulation mode the user can watch the current states of the synchronization objects and threads, and wait queues, whereas in the normal mode these details are not available.

The user can request information about running agents on the specific nodes and view their statistics, as well as access their properties and status information.

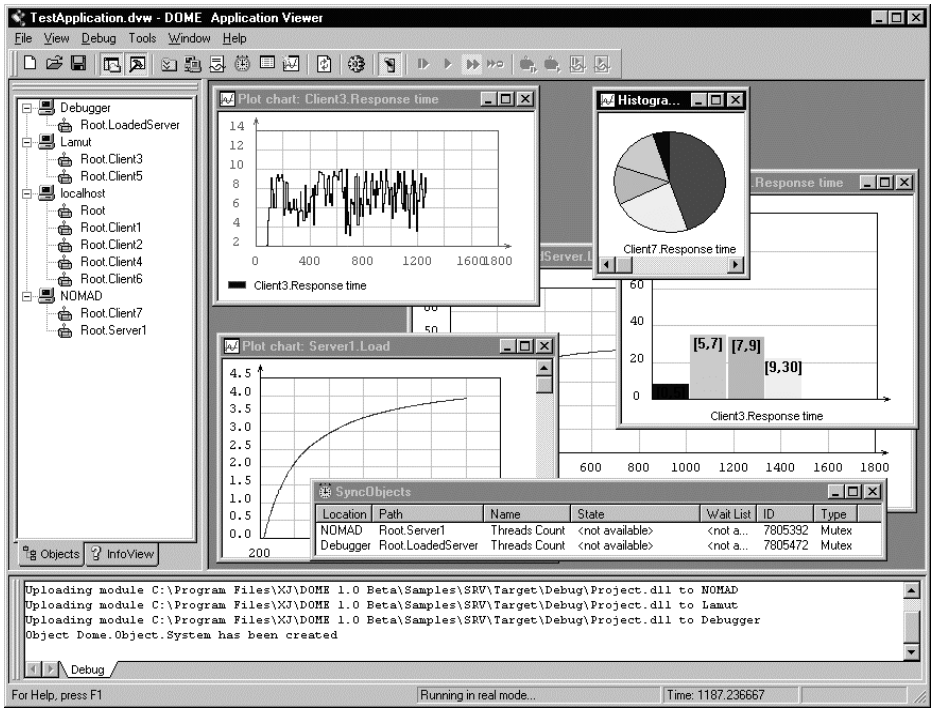

Fig. 2. XJ DOME Application Viewer

8 Management

The user can manage DCOM application with DOME Application Viewer. Before the application starts the user chooses the execution mode (simulation or normal), root objects and gives deployment instructions for static objects.

When the application is running, commands available in the viewer depend on the execution mode. In simulation mode the user has full control over the application execution. Since DOME Engine manages time and synchronization, the user can run the application step-by-step, stop, watch the activity of the selected object, etc. In the normal execution mode time and synchronization are managed by the operating system. In any mode the user can change the COM properties of any application object with DOME IDispatch Browser.

9 Example

As an example (see Figure 3), consider a file searching system. Its purpose is to find a file with a name corresponding to the given pattern and containing given text in it. The search system consists of two parts. The first one is an application that interacts with the user, requests corresponding patterns and reports the result on search completion. Another part of the search system is a mobile agent that travels through the domain and performs search procedure on every host in the domain. Due to the use of DCOM security model the agent has the same privileges as the user who launched it. As the agent finishes visiting hosts in the domain, it returns to the launching system and transfers the results to the front-end application, which reports them to the user.

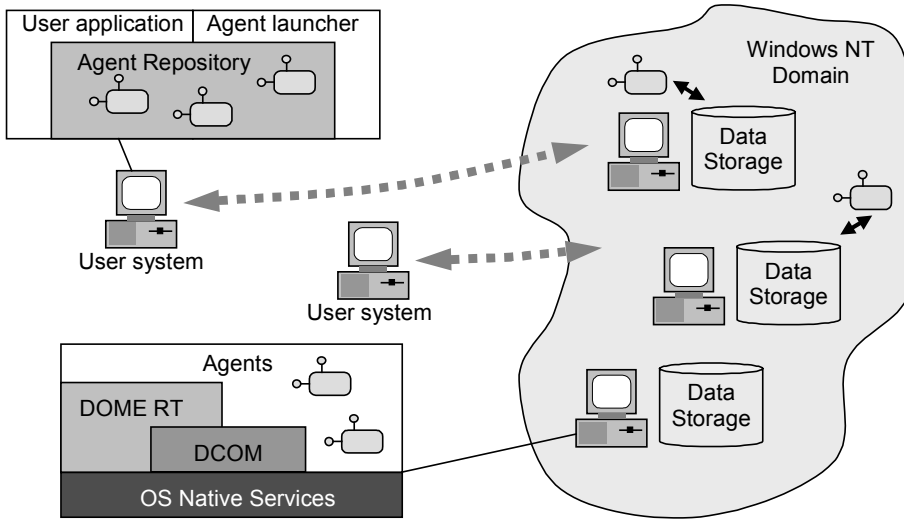

Fig. 3. An example of information retrieval system for distributed data storage

10 Future Work

As DOME is considered as a framework technology that can be used as a basis for building distributed applications with predictable quality of service, we are working in three directions:

- Implementing general-purpose distributed algorithms in DOME objects, such as distributed termination, distributed snapshot and distributed deadlock detection.
- Developing DOME object-compatible simulation models of communication and navigation models for mobile agent systems and communication media (networks and protocols) for better prediction of application performance.
- Incorporating UML Statecharts engine into DOME objects for enhancing clarity and expressive power of object behavior specification.

References

1. J. Baumann, F. Hohl, K. Rothermel and M. Strasser. Mole - Concepts of a mobile agent system. World Wide Web, 1 (1998), pp.123-127.
2. D. Krieger and R.M. Adler. The Emergence of Distributed Component Platforms. IEEE Computer, March 1998, pp. 43-53.
3. Andrei V. Borshchev, Alex E. Filippoff and Yuri G. Karpov. Developing, Simulating and Managing Distributed COM Applications with XJ DOME. In Proceedings of the 1st International Workshop on Computer Science and Information Technologies, Moscow, January 18-22, 1999, Volume 2.