FACTA UNIVERSITATIS (NIŠ)
SER.: ELEC. ENERG. vol. 16, Aug. 2003, 185-194

Defining a General Object Model of Distributed Systems Entities in Java

Nenad Jovanović, Ranko Popović, and Zoran Jovanović

Abstract: This paper deals with modeling and simulation of distributed systems in Java programming language. Distributed systems consist of components which communicate and coordinate their actions by passing messages. Components of distributed systems which define some functionality are entities determined with static attributes. The paper presents general model for modeling in Java environment entities distributed systems communicate using message passing protocol. We described a way to present model components and their functional links in a form of an XML document. A model can be executed as an application and (or) an applet. Presented model can be used for modeling heterogeneous systems.

Keywords: Distributed system, modeling, simulation, parallel processing, Java, XML.

1 Introduction

System is a set of entities that interact mutually (together) in purpose of realizing logical goals [3]. Entity is an element of a system which defines some functionality. Modeling is the act of representing a system or subsystem formally. A model might be mathematical, in which case it can be viewed as a set of assertions about properties of the system such as its functionality or physical dimensions. A model can also be constructive, in which case it defines a computational procedure that mimics a set of properties of the system. Constructive models are often used to describe behavior of a system

Manuscript received 16 May, 2003

N. Jovanović is with the Advanced Business School, Kosovo Polje, R. Popović is with the University of Priština, Faculty of Technical Sciences, Kosovska Mitrovica. Z. Jovanović is with the University of Belgrade, School of Electrical Engineering.

in response to stimulus from outside the system. Constructive models are also called executable models.

Design is the act of defining a system or subsystem.

Executable models are sometimes called simulations. Executable models are constructed under a model of computation, which is the set of "laws of physics" that govern the interaction of components in the model. It is a set of rules that are more abstract, and provide a framework within which a designer builds models.

Ptolemy II [10] project describes modeling concurrent heterogeneous systems in Java. Ptolemy II models might be simulations (executable models of system) or implementations (the system itself). They might be classical computer programs (applications), or any of a number of network-integrated programs (applets).

A distributed system is a set of interconnected, independent computers that appears to its users as a single computer. Many distributed systems let their software modules communicate and coordinate their actions by passing messages.

Communication between distributed systems components can be[6]:

- Synchronous and
- Asynchronous.

Entities, which represent components of distributed system, are defined by their attributes (static features) and functions they perform.

The basic goal in this paper is to show how to model static attributes of distributed system entities and their functionality using Java and a manner in which that model can be represent in a form of an XML document. The proposed model based on the object-oriented approach, observes system like a set of objects described by their attributes and behavior, and interaction of objects reflected in mutual changing of their internal states (conditions).

Model can be applied for simulation and visualization processes in real time distributed systems.

2 General Object Model

Distributed system is defined by:

- Entities
- Communication between entities
- Interaction between entities

That system can be described by general hierarchical object model which is represented in the form of directed dependant graph (Figure 1). Both the nodes and the arcs are entities. Entities are objects with their states and behaviors. The nodes represent objects that define some functionality and communicate by passing messages in the model. The arcs present some relations between functional object, usually communication between them.

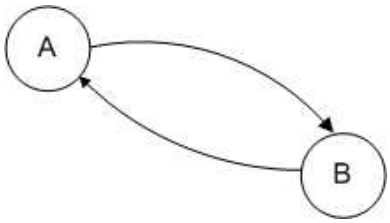

Fig. 1. Object model of distributed system can be described in the form of directed dependant graph.

Entities in observed model are characterized by attributes. State of system is determined with a set of attributes that define static characteristics of entities. Functionality for an entity is defined by functions that represent dynamically review of system.

Entities in model of distributed system can be:

- Computers
- Hubs
- Switches
- Routers
- Transmission paths...

A particular type of entity is a port. Ports are logical constructions that define connecting points between functional objects of models and links. Ports are always unidirectional.

Distributed system entities communicate by message passing. That communication can be synchronous or asynchronous.

The most frequently used type of interaction between distributed system components is client-server interaction.

For instance, if we want to make a model of local network consisting of two computers and one hub connected by UTP cables (Figure 2), that network can be represented with a general object model like on Figure 3.

Computer **PC1** is represented by entity **e1**, computer **PC2** is represented by entity **e2**; **UTP** cable **Link1** is represented by arrows 1 and 2 and

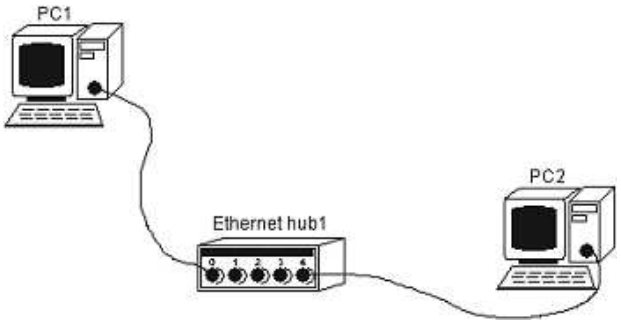

Fig. 2. An example of local computers network.

UTP cable **Link2** is represented by arrows 3 and 4; **Ethernet Hub 1** is represented by entity **e3**. Ports are represented by numbers from 1 to 8.

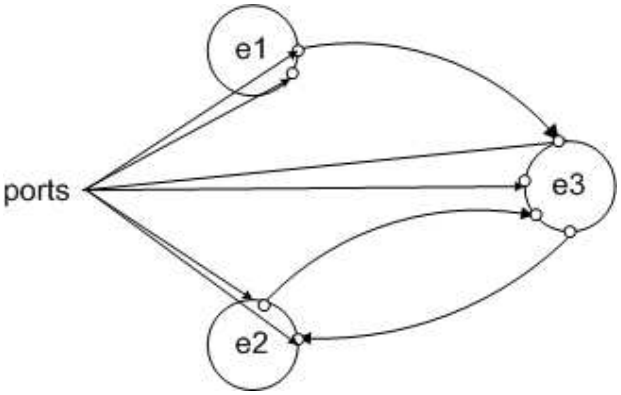

Fig. 3. General model of local computers network described in Figure 2.

2.1 Modeling static attributes of distributed system entities in Java

Entity can be presented in Java programming language by object which is an instance of class that have to define all static attributes of that entity and to enable functional relation of that entity with the other entities (Table 1).

Attribute **type** define to which type of distributed system components does that entity belong. Type can be, for instance, "**computer**", "**hub**", "**link**", "**port**"...

Attribute **nextType** presents reference to a next **entity** of a different type that can be in a connected list. Attribute **nextType** of the last entity

in structure has value **null**.

Attribute **previousType** presents reference to a previous **entity** of a different type that can be in a connected list. Attribute **previousType** of the first entity in structure has value **null**.

Attribute **nextEntity** presents reference to a connected list next **entity** the same type with which is connected certain **entity**.

Table 1. **UML** description (notation) of basic class which models distributed system entities.

Entity
type : String # properties : Property # nextType : Entity # previousType : Entity # nextEntity : Entity # previousEntity : Entity
+ Entity() + Entity(String type) + addProperty(String name, String value) + findProperty(String name) : String + changeProperty(String name, String value) + setNextType(Entity nextType) + setPreviousType(Entity previousType) + setNextEntity(Entity nextEntity) + setPreviousEntity(Entity previousEntity)

Attribute **previousEntity** presents reference to a connected list previous entity the same type with which is connected certain **entity**.

Attribute **properties** presents reference to the connected list head, where elements define all certain entity attributes.

Static attributes of entity are determined by its features (Table 2). For instance, attributes of PC computer model are **name**, **MAC address**, **IP address**, **coordinates** in coordinate system which make possible graphical description of model...

A reference to a head of this list is in entity which attributes list define, and value of attribute **nextProperty** of the last attributes is **null**.

Ports represent logical constructions and they define all access points between link entities and functional entities.

Port attribute **properties** define reference to a connected list which elements determine port attributes. Port attributes are his **name** which is presented by ordinal numeral of port and attribute **I/O** which determines if the port is output (**Out**) or input (**In**).

Table 2. **UML** model of class Property. Objects of class Property serve for preserve static attributes of entities. These attributes are preserved in chained list.

Property
- name : String - value : String - nextProperty : Property
+ Property() + Property(String name,String value) + setName(String name) + getName(String name) : String + setValue(String value) + getValue(String value) : String + setNextProperty(Property next) + getNextProperty(Property next) : Property

If port is output than its attribute **nextType** represents reference to an object **Links**, and **previousType** represents reference to a functional object (for instance computer). If port is output than its attribute nextType represents reference to a functional object and **previousType** represents reference to an object **Links**.

An example of a part LAN model network with all mutual references between entities is presented on Figure 4.

Ptolemy model[10] represents topological collection of entities, ports and relations. Topology of presented model is a collection of entities. Every object (active component, port and link) of distributed systems is specialization of entity. This approach makes possible efficient polymorphism exploitation and modeling a system behavior, where components mutually actively interact.

2.2 Modeling entities functionality

Functionality of entities can be modeled by objects which execute in form of independent threads starting and running some functional activity. **Thread** is a manage flow which can execute concurrently with other threads belonging to the same process. All threads of one process share same address space.

This object is **active object**. Active object is, like every object, instance of some class, in this case active class (Table 3).

In **Java** programming language concept active object is direct support by class **Thread** or by interface **Runnable**. New thread can appear by

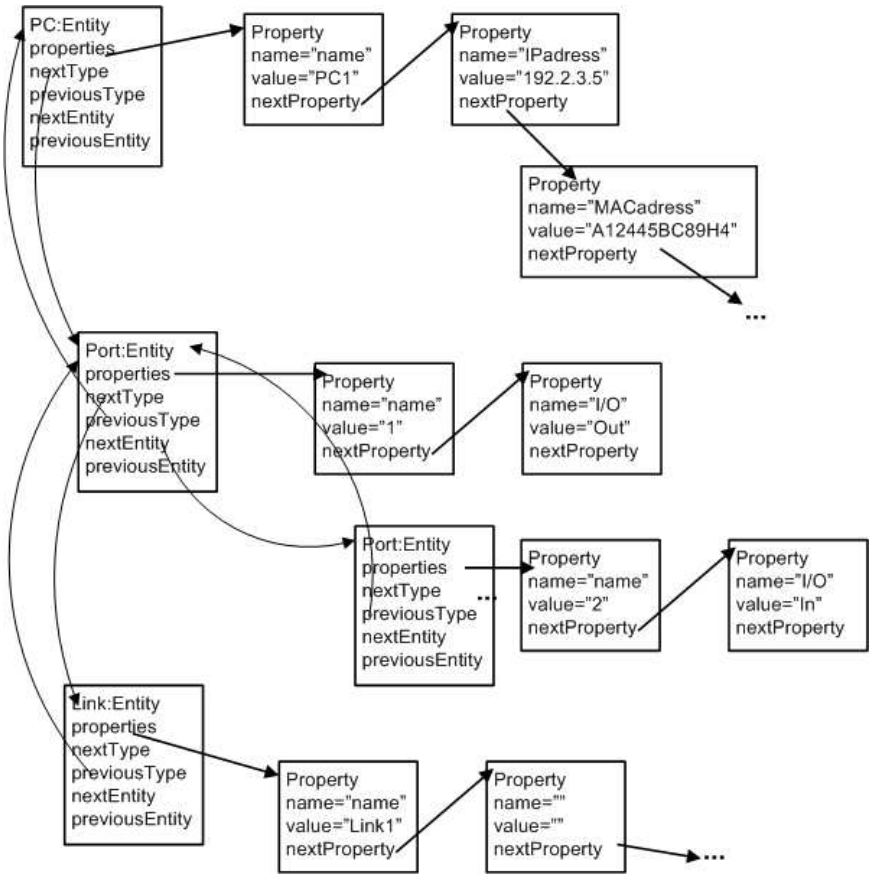

Fig. 4. A part of LAN model presented on figure 2 with appropriate mutual referencing of components.

inheritance of class **Thread**:

Thread thread = new Thread();

Thread starts with calling method **start()** which invokes method run() of new thread when thread starts to execute. Method **run()** presents entry point for new manage stream in program. Activity happens in entity is defined in method **run()**. This manage stream finishing when method **run()** has finished.

Method **run()** can invoke other class methods, declares variables and changes them. Changing fields and calling methods certain objects, method changes internal state of that objects, sends on that way definite messages. Those messages make possible objects mutual interaction and cooperation.

Table 3. In Java functionality modeling is performed by active entities with threads which can define independent manage streams. Threads are defined by inheritance class **Thread** or by interface **Runnable**.

```
class Active extendes implements Runnable{
    //...
    Thread ActiveEntity(){
    public ActiveEntity(){
        // ...
        thread=new Thred(this);
        thread.start();
    }
    public void run(){
        //Define behavior of entity ActiveEntity
        //...
    }
}
```

3 Model Presentation in XML Notation

XML (eXtensible Markup Language) is technology enables creation markup language which can describe data and data structures. Markup language uses label (marker, tag) to define data structures.

In suggested notation markers <**ENTITY**> and </**ENTITY**> are used **where** we define entity structure. Inside these markers can be placed arbitrary number attributes in similar notation.

For example, attribute **name** of computer **PC1** can be presented like:

```
<ENTITY>
    <TYPE>
      PC
    </TIPY>

    <NAME>
      PC1
    </NAME>
       ...
</ENTITY>
```

Suggested model can be presented with document in **XML** notation as illustrated in figure 5.

This presentation make possible to accommodate document-type definitions of distributed system model within the document being Internet transmitted, and exchange of executable models in computer networks.

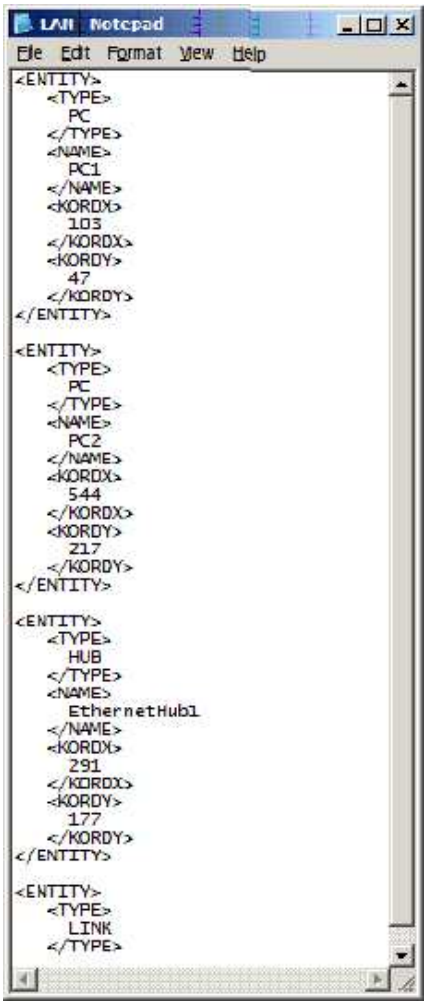

Fig. 5. XML notation document for distributed system object model.

4 Conclusion

In this paper is presented general object model of distributed system entity in Java. Entities are characterized with attributes, which define static state of system, and with their behaviors, which define functionality of system. Suggested model of static system attributes make possible modeling entities and defining reference links between entities which mutual communicate and interact by message passing. Functionality are modeled in Java by active entities with Java threads which can define independent manage stream. In

the paper is proposed a description distributed system model in a form of an XML documents.

References

[1] N. Jovanovic, R. Popovic, *Modeling distributed system in Java programming language*. SYM-OP-IS 2002, pp. VI-25-VI-27, Tara, October, 2002.

[2] G. Coulouris, J. Dollimore, T. Kindberg, *Distributed Systems concepts and design*, Addison Wesley, 2001.

[3] A. M. Law, W. D. Kelton, *Simulation Modeling and Analysis*, Mc Grow Hill, 2000.

[4] D.Ince, *Developing Distributed and E-commerce Applications*, Addison Wesley, 2002.

[5] M. Goel, *Process Networks in Ptolemy II*, MS Report, Berkeley, December, 1998.

[6] N. Smyth, *Communicating Sequential process domain in Ptolemy II*, UCB/ERL M 98/70, Decembe, 1998.

[7] Sh. S. Mukherjee, S .K. Reinhardt, B. Falsafi, *Wisconsin Wind Tunnel II: A Fast, Portable Parallel Architecture Simulator*. In Proc. IEEE Concurrency, October- December, 2000, pp. 12-20.

[8] Philippus C. Homburg, *The Architecture of a Worldwide Distributed System*, PhD thesis, Vrije Unuversiteit, Amsterdam, 2001.

[9] M. Dalpasso, A. Bogliolo, L. Benini, *Virtual Simulation of Distributed IP-Based Designs*, IEEE Design & Test of Computers, September-October 2002 pp. 92-104.

[10] Edward A. Lee et al., *-PTOLEMY II- Heterogeneous Concurrent Modeling and Design in Java*, Originally published as Memorandum UCB/ERL M99/40, University of California at Berkeley, March 15, 2001.