The JDEVS modelling and simulation environment

Filippi, J.B., Delhom,M. and Bernardi, F. [a]

[a]UMR CNRS 6134, University of Corsica, Quartier Grossetti 20250 Corte, France

Abstract: This paper describes the JDEVS modelling and simulation environment. JDEVS has been developed for over a year to serve as an experimental framework for natural systems modelling techniques. It enables discrete-event, general purpose, object-oriented, component based, GIS connected, collaborative, visual simulation model development and execution. The sample models implementations shows that this experimental environment might be used to solve any complex problems solvable by discrete-event simulation and is especially suited for natural system modelling and simulation. Already, hierarchical block (static) and cellular models can be modelled and simulated within the environment. We are now extending those capabilities with the development of a multi-layered modelling paradigm for spatially distributed systems (with vector and cellular models) that will be implemented in the toolkit.

Keywords: Discrete Events Simulation; Modelling Environment; Artificial Neural Networks; Distributed modelling; 3d visualization.

1 INTRODUCTION

The research in defining and creating a general purpose modelling and simulation environment began nine years ago at the UMR CNRS 6134 lab of the University of Corsica. This research has resulted in the definition of the Object Oriented Modelling and Simulation formalism, Delhom [1997]. Delhom has also proven that this formalism, based on DEVS, Zeigler [1976], can be used to solve any problems that can be solved by discrete event simulation. OOMS can serve as a framework for any kind of modelling environments. Nevertheless, it needs to be refined to suit domain specific needs, such as environmental modelling.

1.1 Environmental modelling specific problem

Our problematic is now to adapt the OOMS environment to the field of environmental modelling. A top-down design approach has been used, first trying to identify the needs from the literature to propose an up to date software solution to these problems. The JAVA language has been chosen in that purpose. Several tools have been implemented around a java simulation engine: A graphical modelling interface, an easy to use models library, a cellular simulation panel and a connection to a GIS (Geographic Information System). The Neuro-DEVS methodology is also proposed to easily implement Artificial Neural Networks (ANN) into OO models. Neural networks are widely used in the field of natural system modelling since they can simulate systems that are not well defined and when a large amount of empirical data is available.

The different modules that composes the toolkit will be presented after a brief identification of the fundamental requirements for an environmental modelling software. Sample implementations of cellular and component diagram models are also described to illustrate the modelling process under the environment. The toolkit is still under development, our current research is the development of a topological vector based modelling and simulation formalism that will enable propagation of models in a different manner than with cellular models and runtime models cooperation.

1.2 OOMS environment

The OOMS (Object Oriented Modelling and Simulation) is based on the DEVS (Discrete EVents system Specification) formalism defined by B.P Ziegler. To be able to give the most general description of any system, the DEVS's description hierarchy has been extended with two other concepts in OOMS, the abstraction hierarchy, Oussalah [1989] and time hierarchy, Euzenat [1994]. Currently time and description hierarchy are implemented in JDEVS. Integration of those concepts allows the automatic generation of a simulator from the model, as defined in the DEVS formalism. Within the formalism are specified basic models (atomics models) from which larger ones are built (coupled models) and connection between them.

2 ENVIRONMENTAL MODELLING SOFTWARE REQUIREMENTS

The massive amount of literature available on environmental modelling is reflecting strong needs in this domain, and many software approaches like the VSE from Balci et al. [1998], OOPM/RT from Lee and Fishwick [1999], SME from Maxwell [1999] or CD++ from Wainer and Giambiasi [2001] have been developed to satisfy them. Fundamental requirements have been raised from those different approaches to serve as building blocks for this software toolkit.

Environmental data storage and retrieval is a basic requirement, like in other approaches, a GIS (Geographic Information System) is used in the toolkit. But if the choice of using GIS seems obvious, the main concern was how to perform the coupling to keep the architecture as modular as possible. The parsi-model Approach to Modular Simulation introduced by Maxwell [1999] and implemented in the SME demonstrate that environmental modelling is a collaborative process that can be simplified by using a modular and hierarchical methodology. The OOMS approach is quite similar in terms of models abstraction. Distributed modelling is enabled by the integration of a generic models database defined by Bernardi et al. [2002].

As identified in Gimblett et al. [1995] environmental simulations, especially for geographically distributed models, are generating a heavy computer load in terms of calculation and memory transfer, consequently another requirements is the use of high performance simulation. OOMS is satisfying this need because it is based on discrete event simulation, the memory load is reduced since the simulator is only processing significant changes of the model. Calculations can also be accelerated by the use of parallel computer with no needs to redefine the model thanks to the DEVS formalism. Gimblett et al. [1995] also identified a need for interactive visualization, the use of java is simplifying the integration of 2d/3d runtime visualization. Another environmental modelling domain specific problem is the lack of understanding of some natural systems. This leads to the situation where self learning algorithms, such as artificial neural networks (ANNs) can offer better results than physical models. Due of their ability to generalize from empirical data, ANN are often used in the environmental modelling field. One of the alternative software requirements is the easy integration of ANN models for specific applications.

Our current research is now based on agent modelling and topological displacement vector modelling. This can be seen as a methodology that will avoid splitting space into cellular models by offering to the modeler the ability to describe the behavior of a polygonal shape model (represented by agents). Such dynamic models will also facilitates runtime cooperation and will serve as a basis for multi-layered modelling.

3 THE JAVA DEVS TOOLKIT

JDEVS toolkit is composed of five independent modules. A simulation kernel, a graphical block modelling interface, a models library, a connection to a GIS and a cellular simulation panel. They can interact with other modules that are already developed and some elements, including the java simulation kernel, might be changed for better performance. Figure 1 describe the architecture of the toolkit and the interaction with the modelers.

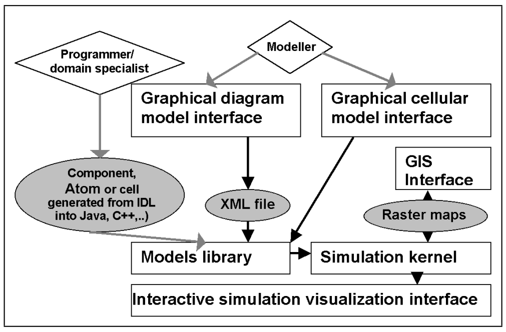

Programmer/ domain specialist
Modeller
Graphical diagram model interface
Graphical cellular model interface
GIS Interface
Component, Atom or cell generated from IDL into Java, C++,..)
XML file
Raster maps
Models library
Simulation kernel
Interactive simulation visualization interface

Figure 1: JDEVS toolkit

The only programming task that the domain specialist has to do is the redefinition of the 4 methods of an atomic model from the IDL interface. Once the atom is created, it is stored in the library to be used later in a federation (coupled model). In case of a spatially distributed system, the federation is automatic in the cellular panel, atom cells are instantiated according to the raster property map exported from the GIS. For a component diagram model, the coupling between models is made graphically by the modeller in the block diagram GUI, then stored in XML into the library. The toolkit is developed by the first author and available from the project website at ”http://spe.univ-corse.fr/filippiweb/”.

3.1 Modelling and simulation kernel

The modelling and simulation kernel is a java implementation of the DEVS formalism. Atomics and coupled models are described as follow.

Atomic DEVS models definition. The DEVS formalism is offering well defined interfaces for the description of systems. The concept of model abstraction permits to use models that are coded in various object oriented languages. Those models are then accessed though a software interface specified in DEVS. JDEVS is a java toolkit. Modelling atomic models directly in the toolkit can be done directly in this language. To help the modeler in this task, the GUI generates a java skeleton, stores it in the models library and compiles it. A formal DEVS atomic model is described as:
M = $< X, S, Y, \delta_{int}, \delta_{ext}, \lambda, t_a >$ With **X** is the input events set, **S** is the state set, and **Y** is the output events set. There are also several functions: δ_{int} manages internal transitions, δ_{ext} external transitions, λ the outputs, and ta the elapsed time. The resulted code is a generated java skeleton for :

Atom=$< X\{in1\}, Y\{out1\}, S\{first\}, \delta_{int}, \delta_{ext}, \lambda, t_a >$

```
    public class Atom extends AtomicModel {
        Port in1 = new Port(this,"in1","IN");
        Port out1 = new Port(this,"out1","OUT");
    public Atom () {
        super("Atom");
        states.setProperty("first",""); }
        EventVector outFunction(Message m) {
            return new EventVector();}
        void intTransition() {}
        EventVector extTransition(Message m) {
            return new EventVector();}
        int advanceTime()return 1;}
```

External transition function is returning events that are appended to the event queue.

Coupled models description in JDEVS. If the user wants to interact directly with the simulation engine, the coupling between models can be made directly in a java file. However it is possible to graphically construct the model structure with the GUI and save it in XML. A DEVS coupled model is defined as:
CM = $< X, Y, D, \{M_i\}, \{I_i\}, \{Z_{ij}\} >$ Here, **X** is the set of input events, and **Y** is the set of output events. **D** is an index of components, and for each i $\in$ **D**, $\mathbf{M}_i$ is a basic DEVS model. $\mathbf{I}_i$ is the set of influences of model i. For each j $\in \mathbf{I}_i$, $\mathbf{Z}_{ij}$ is the i to j translation function. Part of the resulted XML document type definition for a coupled model is:

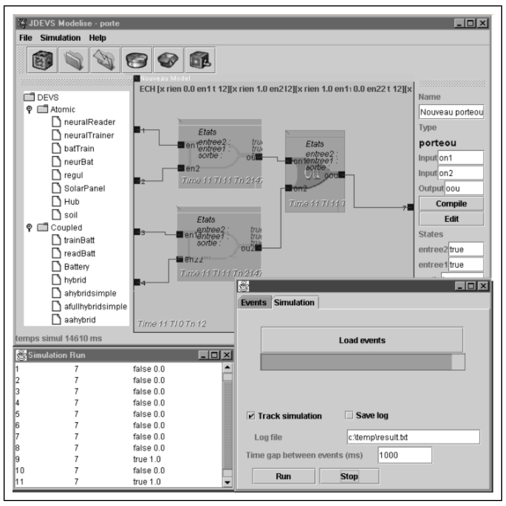
$<!ELEMENTMODEL(TYPE, NAME, BOUNDS?, INPUT*, OUTPUT*, CHILD*, EIC?, EOC?, IC?) >$

With TYPE defining the kind of coupled model (Cellular, kernel, coupled...), NAME the name of the model, BOUNDS the position of the model on the screen (used only by the GUI), INPUT the set of input ports, OUTPUT the set of output ports, CHILD the index for the components of the coupled model (in a priority order), EIC is the external input coupling, EOC the external output coupling and IC the internal coupling. Each coupled model is stored in a different XML file, the parser automatically instantiates the models and creates the links during the loading.

Neuro-DEVS, ANN models definition. Neuro-Devs, introduced in Filippi et al. [2002] is a proposition for an implementation of Neural Networks in the DEVS formalism to satisfy the requirement of self learning models integration. An interface is created to separate the ANN from the modelling process to manipulate it as a stand-alone object. Three main applications have been identified for neuro-DEVS models. The concurrent simulation can be used to avoid an unexpected behavior of a neural network by comparing the neural network output with the output of a simple model to validate the result. Adaptive models can be used to modify the neural network runtime according to an error feedback (Difference between the model's forecast and the real world data collected afterwards). ANN as a sub-component can be used if Neural Networks provides better results for only a piece of the whole system (like battery in an energetic system Jungst et al. [2000] (see section 5 for sample application). The ANN is separated of the model during the modelling process, and is only accessed during the run (through an object broker, or a link to a file). It is up to the modeler to define how and where the neural network should be used. If the ANN object is not used as an adaptive model, it should have already "learned" the patterns, as initial weights must be defined before the simulation starts.

3.2 Hierarchical block diagram modelling and simulation interface

The graphical user interface is the modelling front-end of the toolkit, using this front end, the user can graphically create, compile, link and store atomic and coupled models, debug the resulting model and perform the simulation. Distributed modelling is made using the GUI, if different modelers works on sub-coupled models and store them in the same library, it is possible to federate those models in another graphical modelling client. Figure 2 shows the modelling and simulation interface.



ECH [x rien 0.0 en1 t 12][x rien 1.0 en2 l 2][x rien 1.0 en1: 0.0 en22 t 12][x

Time 11 Tl 11 Tn 2147

Time 11 Tl 0 Tn 12

Name: Nouveau porteou
Type: porteou
Input: on1
Input: on2
Output: oou
entree2: true
entree1: true

temps simul 14610 ms

Simulation Run

```
1         7          false 0.0
2         7          false 0.0
3         7          false 0.0
4         7          false 0.0
5         7          false 0.0
6         7          false 0.0
7         7          false 0.0
8         7          false 0.0
9         7          true 1.0
10        7          false 0.0
11        7          true 1.0
```

[✓ Track simulation] Save log

Log file: c:\temp\result.txt

Time gap between events (ms): 1000

Figure 2: JDEVS diagram M&S GUI.

At the left stands the models library (with atomic and coupled models), with a mouse click the selected model is added to the selected coupled model. In the center stands the hierarchical block diagram representation of the model, all components can be moved with the mouse, the linking between models is performed by a click from the origin port to the destination port. On the right stand the property panel of the selected component. If it is an atomic model, the user can edit and compile it from this properties panel. At the bottom of the figure stands the simulation panel, the user load the input events from this panel and run the simulation to the screen or to a file. To debug the model, it is possible to track the simulation. In this mode, a chosen delay is set between the processing of each event. The diagram representation of the models is then displaying the event queue and the states of the selected models during the run.

3.3 Generic models library

A complete description of the library can be found in Bernardi et al. [2002]. The implementation of the library description in JDEVS is resulting in a module in the GUI. This module presents models according to its domain and sub-domain, all classified in a tree like architecture.

3.4 GIS interconnection

Brandmeyer [2000] have detailed various GIS coupling methodologies. To keep the modular architecture of the toolkit, the connection to the GIS has been be made through a loose coupling. In this kind of coupling, the data is exported from the GIS to the simulator, and results are imported back after the simulation. Since no open formats like OpenGIS or SEDRIS have yet became standards, simple ASCII files are used to transfer data from the database to the simulation engine and back. To perform the coupling, the user has to select a zone in a GIS, then rasterize the zone and export the resulting map in an ASCII file. During the initialization of the simulation the cellular simulation panels automatically instantiates each cell using the attribute from the file. To enable 3d visualization of the model, the same cut has to be made in an elevation grid map (at any resolution) then exported aside. As the simulation will eventually be displayed and imported back in a GIS, the ASCII map can be refined with a link to a background image, the cell size and the absolute coordinates of the map; those data are simply stored in the header of the file and are not used by the simulation engine. The simulation output is a set of discrete events (containing the cell coordinates, the cell new state, and when the change has occurred). Those events are also flattened during the run to recompose discrete time maps that can be imported back to the GIS.

3.5 Cellular simulation panels

This module allows the user to perform (and debug) simulation of a cellular model. The user can directly interact with the mouse during the simulation run. The general architecture shown in Figure 3 has been adopted to model cellular systems.

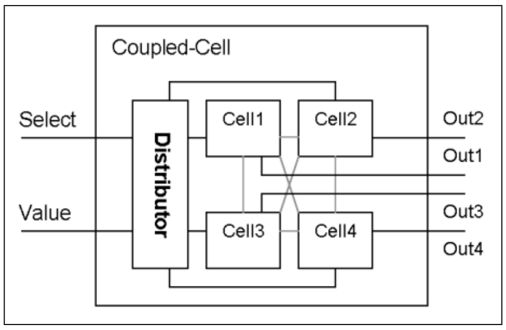


Figure 3: Cellular models architecture.

It is composed of a distributor and cells in a cellular coupled model. The general inputs are connected to the inputs of the distributor, then the distributor will send them to either all the cell or to the cell that would have been selected by an event in its ”Select” port. All cells are connected to the general output. During the initialization, the cellular coupled model is loading the GIS generated file, calculates the number of cells needed (width*height) and then instantiates and performs the coupling for every cell. The modelling is a straight forward process, to describe a model the user only need to define the java class that implements the characteristic functions of a cell-model according to the DEVS formalism. Using this tool, the user can interact during the run with the simulation, with a click on the map it is possible to select a cell and send a specific event. The main panel offers a 2d representation but a 3D simulation panel (figure 4) has also been developed to enable a better visualization of phenomena.

4 APPLICATION: CELLULAR POLLUTION MODEL WITH 3D VISUALIZATION

To manage natural resources such as water, it is necessary to model the phenomena that alter those natural resources in order to quantify and qualify them. Some models, like pollution dispersion models or water flow models require 3d visualization since they represent 3d data. Figure 4 shows a model that is currently tested to quantify phosphates pollution in catchments basins. Like any other atomic model, this cellular pollution model is described in one file, the atom cell description file.

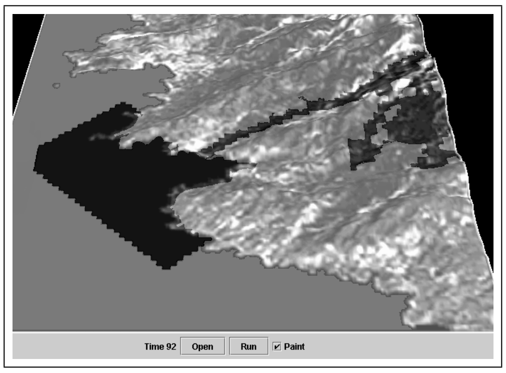


Figure 4: Pollution model in the 3D panel

The behavior is described in programming code, either in Java or in C++. The skeleton for the file is generated from an IDL interface, it contains the four functions of the atomic model as well as the following state set :

$<X\{N, S, E, W, in\}, Y\{N, S, E, W, out\}, S\{poros, elev, pollut\}>$ (N,S,E,W corresponds to the North, South, East and West ports of their neighborhood).

In this model, the λ function (output) is sending to the neighboring cells its elevation and quantity of pollutant. This function is called by the simulator in case of an activation.

The δ_{ext} function (input) is receiving the quantity of pollutant and altitude from the neighboring cells, and send and activation message if it's elevation is significantly lower than the elevation of the neighboring cell.

The δ_{int} function (internal) is called when the cell receive an activation. It updates the states (here the quantity of pollutant) according to the quantity received and the porosity of the ground.

Finally the t_a function (time advance) defines the time to the next self-activation of the cell according to the quantity of pollutant (thus defining the flow speed). Here is the Java transcription of the output function as a code example:

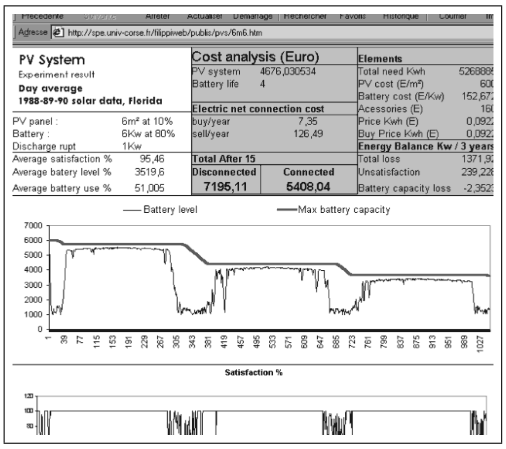
```
EventVector outFunction(Message m){
    e = new EventVector();
    e.add(new Event(N,"Elev","Pollut"));
    e.add(new Event(S,"Elev","Pollut"));
    e.add(new Event(E,"Elev","Pollut"));
    e.add(new Event(W,"Elev","Pollut"));
    return e;}
```

Those four functions are the only code that the specialist has to implement in order to have his model working. The program is in Java or C++ code, thus it is possible to define subroutines in the description file (eg. to send the same event to all ports). Once the behavior of the atomic cell model is described, the only work that has to be done is in the data preprocessing into the GIS to generate raster ascii maps of the initial states (here three files for porosity, elevation and quantity of pollutant). The GIS connector is then loading the maps, automatically instantiates the right number of cells and perform the coupling. Once the model is created, it is sent to the simulator that performs the simulation. The 3d simulation panel serves for the visualization of the simulation of these phenomena, it uses Java3D in order to paint the outputs of 2d or 3d cellular models. The elevation map exported from the GIS permits to reconstruct a 3d world and the 3d panel enables here the visualizations of the polluted zones. To interact with the model, it is possible to click on the map during the simulation run and set a pollution point.

5 APPLICATION: A DIAGRAM MODEL WITH ANN: PHOTOVOLTAIC POWER PLANT

The solar power plant (known as the Photovoltaic(PV) system) is composed of solar power cells, a battery and a switch. The resulted coupled model is composed of four atomic models. The solar panel is modelled using the physical description provided). The battery is modelled like an electric power tank, with charge and throughput efficiency. The damage model use a neural network to decrease the battery maximum capacity runtime according to the deepness and duration of the discharges. Finally the switch is responding to the power demand.

The benefits of using JDEVS for the modelling of this model is clear, the structure of the model is quickly created with the GUI. The neural network is directly included in the atomic damage model and data (for the solar radiation) is imported from the GIS. Also, the neural-net can be trained with different experimental data to model different brand of batteries. Figure 5 shows the experiment results by JDEVS. For every experiment with any initial states, an HTML page is generated showing graphs and analysis of the simulation results.



http://spe.univ-corse.fr/filippiweb/publis/pvs/6m6.htm

PV System
Experiment result
Day average
1988-89-90 solar data, Florida

PV panel :	6m² at 10%
Battery :	6Kw at 80%
Discharge rupt	1Kw
Average satisfaction %	95,46
Average batery level %	3519,6
Average battery use %	51,005

Cost analysis (Euro)	
PV system	4676,030534
Battery life	4
Electric net connection cost	
buy/year	7,35
sell/year	126,49
Total After 15	
Disconnected	**Connected**
7195,11	**5408,04**

Elements	
Total need Kwh	526888
PV cost (E/m²)	60
Battery cost (E/Kw)	152,67
Acessories (E)	16
Price Kwh (E)	0,092
Buy Price Kwh (E)	0,092
Energy Balance Kw / 3 years	
Total loss	1371,9
Unsatisfaction	239,22
Battery capacity loss	-2,352

Figure 5: Experiment result page

This model can be used as it is to perform cost viability studies for PV system. Simulation time for 3 years of hourly solar data and consumption is about 20 seconds on a Pentium 4 1.6 Ghz.

6 CONCLUSION

This paper has presented all the features already implemented in the JDEVS toolkit. The tool saves modelling and simulation time in the field of environmental modelling. It has been successfully used in different applications showing its versatility, and the flexibility of the toolkit. It also satisfies the fundamental requirements that has been identified from other approaches in environmental modelling software. We are now extending those features by the development of polygonal agents models methodology to allow the description of models using dynamic polygons and displacement vectors rather than with cells. This method will also serve as a basis for multi-layered models runtime cooperation during simulation.

REFERENCES

Balci, O., I. Anders, C. Bertelrud, M. Esterbrook, and R. E. Nance. Visual simulation environment. *Proceedings of the 1998 Winter Simulation Conference*, 1:279–287, 1998.

Bernardi, F., E. De-Gentili, and J. Santucci. Reusable models integration in a devs-based modeling and simulation environment. *Proceedings of the SCS ESS 2002 conference on simulation in industry*, 1:644, 2002.

Brandmeyer, J.E. Karimi, H. Coupling methodologies for environmental models. *Environmental Modelling and Software*, 15(5):479–488, July 2000.

Delhom, M. *Modélisation et Simulation Orientées Objet, Contribution à l'Etude du Comportement Hydrologique d'un Bassin Versant*. PhD thesis, University of Corsica, 1997.

Euzenat, J. Granularité dans les repréentations spatio-temporelles. Technical Report 2242, INRIA Rhône-Alpes, 1994.

Filippi, J., P. Bisgambiglia, and M. Delhom. Neurodevs, an hybrid methodology to describe complex systems. *Proceedings of the SCS ESS 2002 conference on simulation in industry*, 1:647, 2002.

Gimblett, R., G. Ball, V. Lopes, B. Zeigler, B. Sanders, and M. Marefat. Massively parallel simulations of complex, large scale, high resolution ecosystem models. *Complexity International*, 2, april 1995.

Jungst, R., A. Urbina, and T. Paez. Stochastic modeling of rechargeable battery life in a photovoltaic power system. Technical Report 1541C, Sandia national laboratories, 2000.

Lee, K. and P. Fishwick. Oopm/rt: A multimodeling methodology for real-time simulation. *ACM Transactions on Modeling and Computer Simulation*, 9:141–170, 1999.

Maxwell, T. Parsi-model approach to modular simulation. *Environmental Modeling and Software*, 14:511–517, 1999.

Oussalah, C. A framework for modeling and linking the structure and the behavior of a system. *Artificial Intelligence in Scientific Computation*, 1989.

Wainer, G. and N. Giambiasi. Application of the cell-devs paradigm for cell spaces modelling and simulation. *Simulation*, January 2001.

Zeigler, B. *Theory of Modeling and Simulation*. Academic Press, 1976.