Agent-based Scientific Simulation Using Java/Swarm, J2EE, RDBMS and Autonomic Management Technologies

Yingping Huang, Xiaorong Xiang and Gregory Madey
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556

Steve Cabaniss
Department of Chemistry
University of New Mexico
Albuquerque, NM 87131

Abstract

The authors present an agent-based stochastic simulation model of NOM transformations and its implementation using Java/Swarm. An autonomic self-manageable infrastructure is designed to facilitate remote invocation of simulations, analysis and reports of the resulting large simulation datasets. The NOM simulation system may be useful in many areas including chemistry, geology, microbial ecology and environmental science and the infrastructure may be used to support scientific simulations in other areas.

1. Introduction

Natural Organic Matter (NOM) is a complex mixture of compounds formed as a result of the breakdown of animal and plant material in the environment. NOM is ubiquitous in terrestrial, aquatic, and marine ecosystems, playing a crucial role in such important processes as the evolution and fertility of soils; the mobility and transport of pollutants such as trace metals, radionuclides and hydrophobic organic compounds; the availability of nutrients to microorganisms and plant communities; the growth and dissolution of minerals; and the global biogeochemical cycling of the elements [1].

NOM has a significant impact on all aspects of drinking water treatment. NOM is responsible for the majority of the coagulant demand. Therefore water with high dissolved organic carbon levels usually has a high coagulant requirement and consequent high treatment costs. NOM can cause major problems in the treatment of water as it reacted with chlorine to form disinfection by-products. Many of the disinfection by-products (DBPs) formed by the reaction of NOM with disinfectants, are reported to be toxic and carcinogenic to humans if ingested over an extended period of time. The removal of NOM and hence reduction in DBPs is a major goal in the treatment of any water source.

The importance of NOM attracts numerous researchers, including chemists, geologists and even computer scientists. Despite decades of research, we still know relatively little about the structure, chemical composition, and chemical properties such as molecular weight, functional group concentrations, structure, composition, and reactivity [2, 3, 4].

During the last 50 years, some research has been done on the functional behavior of NOM molecules, and different models are proposed to handle NOM research, including ODE (Ordinary Differential Equation), PDE (Partial Differential Equation), etc (see Section 2). But the diversity of the compounds present in NOM leads to the difficulty of describing the mixture adequately and makes models computationally expensive.

In this paper, we describe a new agent-based stochastic modeling approach to simulate the behavior of NOM. The simulation was implemented using Java and the Swarm library [5]. To make the simulation accessible by scientists around the world, we employ the J2EE (Java 2 Enterprise Edition) and RDBMS (Relational DataBase Management System) technologies. We design and implement an autonomic self-manageable multi-tiered infrastructure using IBM's autonomic computing technology which permits scientists to run simulations anytime and anywhere reliably.

The rest of the paper is organized as follows. Section 2 identifies some previous simulation models and points out their limitations. Section 3 presents our new agent-

based stochastic model of NOM, performance discussions and model verification and validation. Section 4 describes the supporting self-manageable infrastructure. Finally, section 5 draws conclusions.

2 Background

Despite decades of research, we still know relatively little about the structure, chemical composition and reactivity of NOM primarily because it is a complex mixture of molecules with different physico-chemical properties such as molecular weight, functional group concentration, structure, composition and reactivity. The evolution of NOM from its biological precursor compounds is both an interesting biogeochemical problem and an important aspect of predictive environmental modeling. On the one hand, carbon cycling models based on average properties of various organic carbon pools are too simplistic to represent the heterogeneous structure of NOM and its complex behavior in the environment. On the other hand, molecular models employing connectivity map or electron density are too computational expensive to be useful for large-scale environmental simulations.

With awareness of the drawbacks of the previous models, we propose a middle path: development of an innovative stochastic model that will allow, for the first time, forward modeling of the evolution of NOM structure and properties. This stochastic model of NOM evolution represents individual molecules as discrete objects of specified elemental and functional group composition, size and reactivity. Temporal evolution of NOM from biological precursor compounds such as lignin, polysaccharides and proteins is simulated using Monte Carlo algorithms in which specific probabilities are assigned to particular transformations. These algorithms employ newly-developed pseudo-random number generators with long periods and robust statistical properties provided by the Swarm library. Both physico-chemical and biochemical effects are incorporated probabilistically. The reactivity of the resulting NOM assemblage over time is predicted based on the distributions of molecular properties.

This stochastic approach has several advantages: it is much less computationally intensive than molecular modeling or explicit kinetic simulation of hundreds of compounds, it can readily be adapted to a variety of time scales and processes, and it intrinsically handles NOM structural and functional heterogeneity.

Furthermore, this approach employs a self-manageable infrastructure which permits users to run simulations anytime from anywhere with reliable results. In the next two sections, we first present the details of the agent-based stochastic simulation model, then we present the design and implementation of the supporting self-manageable infrastructure.

3 Agent-based Stochastic Model

In this section, we present the new stochastic simulation model using agent-based technology. This stochastic model of NOM represents individual molecules as discrete objects of specified elemental and functional group composition, size and reactivity. Temporal evolution of NOM from biological precursor compounds such as lignin, polysaccharides and proteins is simulated in which specific probabilities are assigned to particular transformations. The reactivity of the resulting NOM assemblage over time can be predicted based on the distributions of molecular properties.

3.1 Modeling of NOM Evolution

Since representing each molecule as a detailed chemical structure is prohibitively computationally expensive, we design a data structure which is much simpler to work with but which encapsulates much of what we know and when we wish to know about NOM. The data structure includes the following:

- The elemental formula, i.e., the number of C, H, O, N, S and P atoms in the molecule.
- A record of the molecular ”origin”, i.e., the type and size of precursor molecule and the time at which it was entered into the simulation. This allows the calculation of separate ”turnover time” and apparent ages for individual molecules and fractions.
- Functional group counts: carboxylic acid, alcohol, ester, keton, aldehyde, thiol, sulfato, amine and peptide groups are counted in each molecule.

The NOM simulation is implemented in a discrete 2D space with discrete time. The simulated space is a rectangular lattice. Each molecule can occupy at most one cell, and each cell can host one or more molecules. The molecules in the simulation are intended to represent a sample from a large population in the system under study. During execution of the simulation, each molecule may move to another cell or stay in a fixed cell according to predefined simple rules that describe physical processes. In chemical reactions, one molecule could split; two or more molecules could combine and occupy just one cell.

There are 10 types of chemical reactions represented in the simulation system: ester condensation, ester hydrolysis, amine hydrolysis, microbe uptake, dehydration, strong C=C oxidation, mild C=C oxidation, alcohol (C-O-H) oxidation, aldehyde C=O oxidation, and decarboxylation. Each

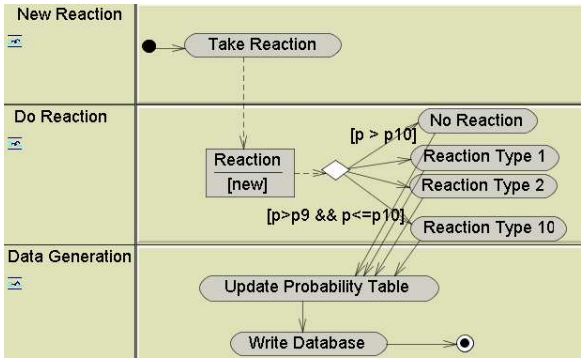

Figure 1. Simulation process for each molecule and each time step

molecule has a probability for each type of chemical reaction. The calculation of the probability is based on the structure of the molecule, and the environment in which the molecule resides. These environmental variables include the length of the time step, microbe density, fungal density, pH value, temperature, pKw (the equilibrium constant for the autolysis of water, which is very close to 14.0), oxygen density, light density, etc. After each chemical reaction, the probabilities of these reaction types are re-calculated. The reaction probabilities are stored in an array associated with each molecule.

In each time step, for each molecule, a random number is generated which is used to determine whether a chemical reaction will occur, and if one occurs, which reaction type. In the simulation, the sum of all the reaction probabilities is controlled to be less than 1 percent. The interval $[0, 1]$, is partitioned into 11 subintervals. The length of the first interval is equal to the probability of the first reaction type; the length of the second interval is equal to the probability of the second reaction type, etc. The length of the last interval is the probability in which no reaction will occur. The generated random number from the interval [0,1) will reside in one of these intervals, and it will decide which chemical reaction will occur, if any at all. Figure 1 shows the above processes.

If the chosen chemical reaction type is a second order reaction (the probability of higher (> 2) order reactions is so small that we can safely ignore them), i.e., there will be two molecules involved in the reaction, the second molecule will be chosen from one of its nearest neighbors who are not yet involved in a chemical reaction. After the reaction takes place, the probability tables for involved molecules are updated at the end of the time step and the new probability tables are assigned to newly produced molecules.

The NOM simulation is implemented using Java and the Swarm library, the Oracle RDBMS, JDBC (Java Database Connectivity) and SQL*Loader.

3.2 Performance Discussion

Agent-based modeling (ABM), also known as individual-based modeling (IBM) [6], and equation-based modeling (EBM) are two approaches for modeling complex systems. ABMs and EBMs are significantly different with respect to which characteristics they focus on. ABMs focus on the characteristics of each individual and track them through time. EBMs, on the other hand, focus on the characteristics of the population, which are averaged and simulation changes in the averaged population characteristics [7]. It involves a process for solving a set of differential equations. EBMs can describe already known global properties of a system, but often can neither explain the origin of those properties nor track the behavior of individual components. The objectives of ABMs are describing the heterogeneous aspects of individual agents and system, providing a mechanism for interactions between individual agents, and predicting phenomena at higher levels based on the actions of individual agents, in a bottom-up approach. The NOM complex system consists of a large number of heterogeneous molecules and microbes. Individual molecules can be transported through the soil medium via water flow, adsorb on the soil particle surfaces, and react with other molecules or microbes. The properties of each individual molecule can change over time. Also, new molecules are emerging (one molecule is split into two) and molecules disappear (reacting with microbes) in the system. It is hard to capture the global properties of the NOM system accurately with EBMs modeling approach. Cabaniss [9] presents several examples showing that the use of "average" values to represent the complexity of NOM is problematic. It can result in discrepancies when the model results are compared with the results from the laboratory studies. Therefore, ABMs is more suitable for modeling the NOM complex system that is composed of interacting individuals and exhibit a wide range of dynamic behavior. Agent-based modeling of molecules as objects is also being investigated in Cell Biology. For example, at Cambridge University, in the UK Shimizu et al. [10][11][12] create a stochastic simulator for chemical reactions among molecules. In their model, each molecule is represented as an independent software object called agent in our model.

The agent based approach is more flexible and accurate modeling the evolution process of NOM, however, compare to the equation based modeling approach, it is much slower. The performance become an important issue for the agent based simulation. Determining and understanding the factors that affect the performance of simulation from a software engineering perspective and identifying and

eliminating the bottlenecks that limit scalability at the software development stage are necessary for achieving high performance. Several approaches for exploring the performance and scalability improvement of the simulation model is taken. These approaches include runtime optimization, databased access, objects usage, and parallel and distributed computing. We discovered that by employing these technologies and implementation the distributed simulation model with MPJ [8], the performance can be improved by 25 times. The performance is expected to increase more while the number of agents in the system increases. The advantages of using ABMs in the NOM system may overcome performance concern.

3.3 Verification and Validation (V & V)

Verification and validation (V & V) are processes to increase confidence in simulations. Verification is about getting the "simulation right" while validation is about getting the "right simulation". Although neither process guarantees absolute confidence, we used numerous V & V techniques on the NOM simulation. We describe these techniques using an adapted version of Sargent's V & V process shown in Figure 2 [13]. The simulation process starts with an identification of research questions of interest. Through analysis and modeling, a conceptual NOM model is developed that includes the important features relevant to the research question. The conceptual model is based on theory and domain knowledge from environmental chemistry. This theoretical foundation includes 1) the heterogeniety of NOM molecules, 2) the important NOM iterations with mineral surfaces such as adsorption, hemi-micelle formations, acid or complexing dissolution, and reductive dissolution, 3) NOM interaction with pollutants, 4) relationships between NOM adsorption to mineral surfaces and the molecular weight of the NOM molecules, and 5) probabilistic reaction kinetics based on elemental composition and the nature of functional groups in the molecules. The incorporation of such theory and domain knowledge provides us initial face validity, ie., the logic of the conceptual model appears to domain experts to include appropriate mechanisms and properties of the research problem. Six scientists - two biologists, a chemist, a geomicrobiologist, and two soil scientists - evaluated the conceptual model for face validity. Once the conceptual model achieves its initial validation, coding of the agent-based simulation took place. At this step, verification methods such as code walk throughs, trace analysis, input-output testing, and boundary testing were used to verify the correctness of the simulation. To date, the validation of the simulation has included comparisons of simulation behavior with mathematical models and experimental results. For example, the simulation has been used to study the relationship between the adsorption

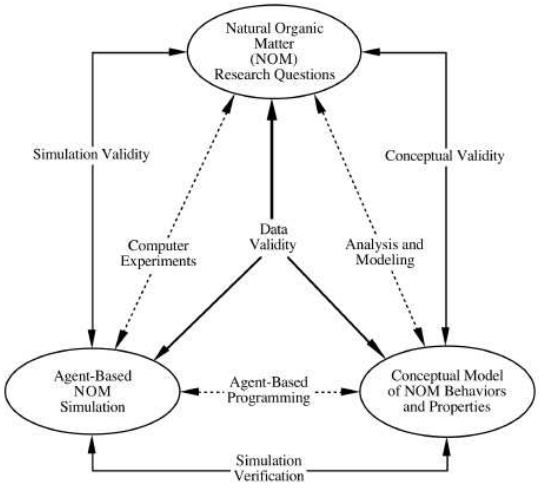

Figure 2. The verification processes of the agent-based stochastic model

of NOM molecules on mineral surfaces and the molecular weight of the molecules. This relationship has been empirically observed to have a lognormal distribution. The simulation has yielded a similar lognormal distribution as reported by Arthurs et al [14]. Additional such comparisons between theoretical, empirical and simulation predictions have been completed.

3.4 Simulation Technologies

In the simulation described in this paper, multiple information technologies are integrated to investigate a new paradigm of scientific inquiry. Java is reaching performance parity with other languages (e.g., C/C++) and has begun to be used in scientific simulations [15] [16] [17]. Fox examined the paradigms of scientific study and introduced a fourth paradigm [18]. We used to speak of three approaches to science: experiment, theory and the computational approach. The fourth paradigm of scientific study uses IT technologies such as web, databases and data mining. We support the molecular simulation model using these IT technologies.

The NOM simulation system includes a web interface which allows users to provide inputs for the simulations and to obtain data analysis reports for their simulations through their browsers, as shown in Figure 3.

The web interface is implemented using J2EE (Servlets, Java Server Pages and Enterprise Java Beans) with the popular MVC (Model, View and Controller) architecture, running on an Oracle9i Application Server. Simulation data is stored into Oracle databases through Java Database Con-

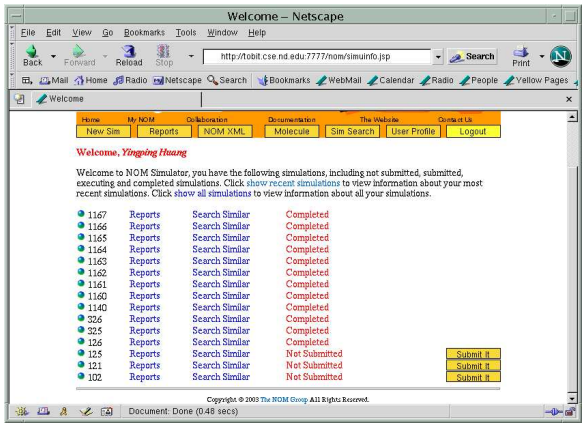

Figure 3. A snapshot of the NOM web interface

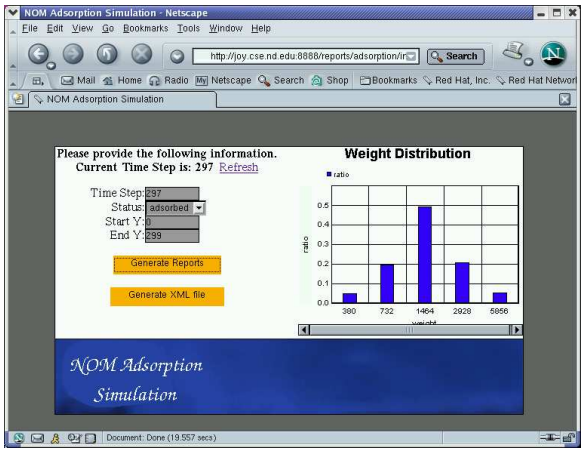

Figure 4. A sample report produced by the reports server

nection (JDBC) and SQL*Loader. Reports of the simulation results are generated using SQL, PL/SQL, XSQL (XML SQL) and Oracle Reports Server. Figure 4 shows a sample report page for the NOM project.

In the following section, we briefly introduce these technologies.

- Java/Swarm: Swarm is a Java library which implements agent-based technology. Swarm can be used to model multi-agent simulation of complex systems.
- Oracle RDBMS: The purpose of databases is to store and retrieve related information. The information includes database schema objects to manage simulations, and input and output of simulations.
- JDBC: Java Database Connectivity is a standard Java interface for connecting from Java to relational databases such as Oracle.
- J2EE: The Java 2Enterprise Edition has historically been an architecture for building server-side deployments in the Java programming language. It can be used to build traditional web sites, software components, or packaged applications.
- XSQL: XML-SQL delivers XML documents of simulation results to users by querying the database.

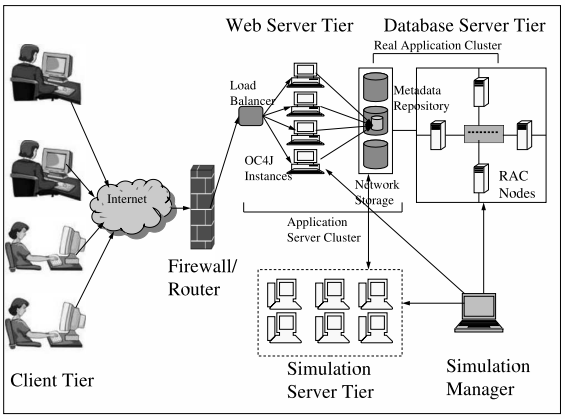

Figure 5. The Multi-tier Infrastructure

4 Self-manageable Infrastructure

The NOM simulation uses an infrastructure built on a multi-tier architecture to support high scalability through load balancing and cross simulation server migration, and high availability through redundancy and simulation resuming, as shown in Figure 5. The client tier communicates with the web server tier behind a firewall by specifying simulation input data, which is stored in the database server tier. The simulation manager dispatches simulation jobs to the simulation server tier. An intelligent agents on the simulation server tier invokes simulations as directed by the simulation manager.

An intelligent agent is an autonomous process (implemented using SQL scripts and Bourne shell scripts) running on a simulation server. It performs the following operations:

- Checks for simulations submitted by users and dispatched by the simulation manager.
- Runs simulations and queues simulation results in the case that the simulation manager is down.
- Handles data transportation from local disk drives to the database server tier.

- Creates database schema objects (such as tables and indexes) for reporting.

- Cancels simulations jobs as directed by the simulation manager.

The simulation manager employs a greedy algorithm and dispatches simulations to simulation servers with least load average.

The infrastructure is designed to be highly scalable at all tiers. We use Oracle9i Application Server in the web server tier and Oracle9i in the database server tier [19].

- Web Server Tier: Application servers are platforms to run J2EE (Java 2 Enterprise Edition) applications. An Oracle9i application server cluster is a collection of application servers which can be configured to be a single group. The application server cluster uses a metadata repository to configure each OC4J (Oracle9iAS Container for J2EE) instance. The metadata repository is configured to be stored in the real application cluster in the database tier. New application servers can be added during the operation of the cluster without downtime. Clients interact with the cluster as if they are interacting with one single application server. Client requests are routed to instances of the cluster through a load balancer.

- Simulation Server Tier: The simulation server tier can be scaled linearly by installing new simulation servers running identical simulations. When the intelligent agent starts to run on the new simulation server, the simulation server is automatically recognized by the simulation manager and the simulation manager can dispatch simulations to the new simulation server. The simulation servers are included into our load balancing and simulation resuming algorithms which are discussed in section 4.4 below.

- Database Server Tier: Oracle Real Application Cluster (RAC) is used in this tier. In a RAC environment, all active nodes concurrently execute transactions against a shared database.

Availability of the web server tier and database server tier is achieved through some level of redundancy.

- Web Server Tier: An Oracle9iAS cluster eliminates the single point of failure by redundancy and failover in the system. A failure of any single instance does not bring down the whole system. Client session state is replicated throughout the cluster, thereby protecting against the loss of session state in case of process failure.

- Simulation Server Tier: Availability of the simulation server tier is handled by a simulation checkpoint and resuming feature, which is discussed in later sections.

- Database Server Tier: Each instance node is isolated from each other so that a failure on one node does not affect the whole database system.

The designed infrastructure is self-managing, i.e., self-configuring, self-healing, self-optimizing and self-protecting [20]. For a detailed implementation of these features, please visit the project web site http://www.nd.edu/ nom.

4.1 Self-Configuring

Self-configuring means that the infrastructure can configure itself to meet goals specified by the system administrators. This infrastructure has the following self-configuring properties:

- Automatic deployment of new web server, simulation server, database server, and removal of existing ones. For example, the intelligent agent automatically recognizes a new simulation server, and automatically checkpoints running simulations on an existing simulation server in case the simulation server is to be removed from the simulation server tier.

- Autonomatic deployment of a new simulation model based on simulation metadata (simulation description data). For example, web interface and database objects are automatically generated from simulation metadata.

- Automatic reconfiguration of the infrastructure for newly submitted simulations. For example, a new simulation with high priority is executed first by checkpointing necessary simulations with lower priority.

4.2 Self-Healing

Self-healing means that the infrastructure can detect and identify failed components (such as web server, database server, simulation server and running simulations), and recover then from failure. This infrastructure has the following self-healing properties:

- Automatic restart of failed web server and database server.

- Automatic resume of crashed simulations based on simulation checkpointing.

Checkpoint is the solution for long-running simulations. The simulation status is checkpointed to the database server tier. In case of a crash, the simulation can be restarted from

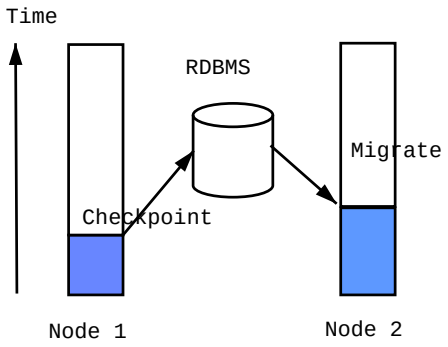

Figure 6. Checkpoint-initiated load balancing

a previous checkpoint. In the case a simulation server is failed, the simulation manager temporary removes the simulation server from the simulation server tier and dispatches simulations to other simulation servers by restarting them from the most recent checkpoints. Once the failed simulation server recovers, its intelligent agent can register the recovered simulation server to the simulation manager so that new simulations can be dispatched to the recovered simulation server.

The interval (i.e., the number of time steps) between checkpoints is simulation varient. It is determined by the mean time to crash, the number of requested time steps, the required time to restart and the required time to checkpoint.

4.3 Self-Optimizing

Self-optimizing means the infrastructure can optimize itself to meet user requirements. This infrastructure has the following self-optimizing properties:

- Load balancing among web servers.
- Load balancing among database servers.
- Greedy load balancing algorithm among simulation servers. The simulation manager picks the simulation server with lowest load average to run a new simulation.
- Checkpoint-initiated load balancing algorithm cross simulation servers.

When a running simulation completes a checkpiont, the intelligent agent contacts the simulation manager whether migrating the simulation to another simulation server is beneficial. The simulation manager decides which simulation server the simulation should migrate, taking into account the overhead of simulation migration. Figure 6 shows the checkpoint-initiated load balancing algorithm.

4.4 Self-Protecting

Self-protecting means the infrastructure can prevent unauthorized access. This infrastructure has the following self-protecting features:

- Role-based access control: users are assigned different roles, such as owner, public. The benefit of role-based access control is that users can share simulation results with restrictions.
- Firewall-enabled using IPTABLES. A port scan is performed on all servers so that only necessary ports are opened to the public. The infrastructure is an intranet behend a firewall.

5 Conclusion

In this paper, we presented a new agent-based stochastic simulation approach to model the behavior of natural organic matter (NOM). The simulation was implemented using Java and the Swarm library. To make the simulation accessible to scientists around the world, we designed and implemented a self-manageable multi-tiered infrastructure to support the reliability and availability of the web-based simulations.

6 Aknowledgements

This research was partially supported by the NFS ITR Grant 0112820 and the Center for Environmental Science and Technology at University of Notre Dame.

References

[1] G. Aiken, D. McKnight, R. Wershaw, and P. MacCarthy. *Humic substrances in soil, sediment, and water - geochemistry, isolation and characterization*. John Wiley and Sons, New York, 1985.

[2] D. McKnight and G. Aiken. *Sources and age of humus*. Springer-Verlag, Berlin, 1985.

[3] S. Cabaniss, Q. Zhou, P. Maurice, Y. Chi, and G. Aiken. A log-normal distribution model for the molecular weight of aquatic fulvic acids. In *Environ. Sci. Technol. 34*, pages 1103–1109, 2000.

[4] K. Kalbitz, S. Solinger, J. Park, B. Mchalzik, and E. Matzner. Controls on the dynamics of dissolved organic matter in soils:a review. In *Soil Sci.165*, pages 277–304, 2000.

[5] Swarm. http://www.swarm.org, 2002.

[6] J. Gross. Agent-based modeling in ethnobiology: a brief introduction to outside. http://www.tiem.utk.edu/ gross, 2002.

[7] H.V.D. Parunak, R. Savit, R.L. Riolo. Agent-based modeling vs equation-based modeling: a case study and user's guide. Proceedings of *Multi-agent systems and Agent-based Simulation* , 1998.

[8] B. Carpenter, V. Getov, G. Judd, A. Skjellum, G. Fox. MPJ: MPI-like message passing for Java. *Concurrency: Practice and Experience*, 2000

[9] S. Cabaniss. Modeling and stochastic simulation of NOM reactions, working paper. http://www.nd.edu/ nom, 2002.

[10] L.N. Norere, T.S. Shimizu. StochSim: modeling of stochastic biomolecular processes. *Bioinformatics*, 2001.

[11] T.S. Shimizu, D. Bray. Foundations of Systems Biology. *Compuational cell biology - the stochastic approach*, 2001.

[12] T.S. Shimizu, S.V. Aksenov, D. Bray. A spatially extended stochastic model of the bacterial chemotaxis signalling pathway. *Journal of Molecular Biology*, 2003.

[13] R.G. Sargent. Validation and verifi cation of simulation models. *Proceedings of the 31st Winter Simulation Conference*, Phoenix, 1999.

[14] L. Arthurs, P.A. Maurice, X. Xiang, R. Kennedy, G.R. Madey. Agent-based stochastic simulation of natural organic matter adsorption and mobility in soils. *Eleventh International Symposium on Water-Rock Interaction*, 2004.

[15] V. Getov, G. von Laszewski, M. Philippsen, and I. Foster. Multiparadigm communications in java for grid computing. *Communications of ACM, Volume 14, Issue 10*, 2001.

[16] G. Thiruvathukal. Java at middle age: Enabling java for computational science. *IEEE: Computing in Science and Engineering*, pages 74–84, 2002.

[17] O. Vormoor. Quick and easy interactive molecular dynamics using java3d. *IEEE: Computing in Science and Engineering*, pages 98–104, 2001.

[18] G. Fox. E-science meets computational science and information technology. *IEEE: Computing in Science and Engineering*, pages 84–85, 2002.

[19] Oracle. http://www.oracle.com, 2002.

[20] Autonomic Computing. http://www.ibm.com/autonomic, 2003.