

# Verification and Validation of an Agent-based Simulation Model

Nadine Shillingford      Gregory Madey      Ryan C. Kennedy  
Department of Computer Science and Engineering  
University of Notre Dame  
Notre Dame, IN 46556  
[nshillin@nd.edu](mailto:nshillin@nd.edu), [gmadey@nd.edu](mailto:gmadey@nd.edu), [rkenned1@nd.edu](mailto:rkenned1@nd.edu)

## Abstract

*Verification and validation (V&V) are two important tasks that need to be done during software development. There are many suggestions in the literature for techniques that can be used during V&V. Some of these are varied and need to be adapted to the particular software product being developed. There are no defined V&V techniques that can be applied to agent-based simulations specifically. It is, therefore, left to the software team to develop a subset of V&V techniques that can be used to increase confidence in the simulation. This paper introduces a subset of techniques gathered from several authors and applies them to a computer simulation of natural organic matter.*

**Keywords:** Verification, Validation, Computer Simulation, Natural organic matter, Agent-based model

## 1 Introduction

The terms verification and validation are often used together. However, these two tasks are quite different. In simple terms, verification asks the question “did I produce the simulation right?” Validation, on the other hand, asks the question “did I produce the right simulation?” Sargent [1] states that a simulation is valid “if its accuracy is within its acceptable range of accuracy which is the amount of accuracy required for the model’s intended purpose.” The goal of validation is two-fold. First, it serves to produce a model that represents true system behavior closely enough for the model to be used as a substitute for the actual system for the purpose of experimenting with the system. Secondly, it increases to an acceptable level the credibility of the model [2]. Verification is concerned with the representation of the model structure. Proper software engineering practices indicate that the model should be analyzed and designed sufficiently before coding can begin. This design referred to as the model in simulation studies needs to be accurately followed. Verification is complete when it

can be asserted that the design/model has been implemented accurately.

The organization of this paper is as follows: First, we introduce the principles of V&V in Section 2. Section 3 discusses some of the V&V techniques found in literature. Next, we introduce the natural organic matter case study that was used for this research in Section 4. In Section 5, we give some technical details on the project. Section 6 discusses the V&V techniques used in the case study, followed by, Section 7 which gives a summary of the results. A conclusion and future work is discussed in Section 8.

## 2 Principles of V&V

Balci [3] gives fifteen (15) principles of VV&T (he includes testing along with verification and validation). Five (5) of these principles include:

1) VV&T must be conducted throughout the life cycle of the simulation study. In his paper entitled *Verification, Validation and Testing*, he displays a diagram of the different phases of the life cycle of a simulation [3].

2) The outcome of the simulation model VV&T should be considered as a binary variable where the model is absolutely correct or absolutely incorrect. He notes, however, that perfect representation is never expected. It is up to the individual or team performing the VV&T to determine at which point the model can be considered to be either absolutely correct or absolutely incorrect.

3) A simulation model is built with respect to the study objectives and its credibility and is judged with respect to those objectives.

4) Complete simulation model testing is not possible.

5) Simulation VV&T must be planned and documented.

These principles were a result of research by Balci and knowledge gained through his experience with VV&T. A complete discussion of the principles can be found in his paper *Verification, Validation and Testing* [3].

### 3 V&V Techniques

There are several authorities on V&V techniques. Some of these include Sargent, Banks, Balci and Adrion.

The following are some of the techniques suggested by Sargent [1]. He states that a simulation model can be compared to other models built for a similar purpose. This process is commonly called docking. He also details the ideas of face validity and Turing tests. Face validity involves the process of allowing an expert to look at the results of the simulation and tell whether the results appear to be correct. In a Turing test, an expert is allowed to distinguish between unmarked simulation results and unmarked system results. The aim is to determine whether the results are so similar that the expert is not able to tell the difference. This would be a very important test in V&V. Unfortunately, since simulations are usually built due to impossibility of attempting actual experiments (due to lack of time and resources), a Turing test is often hard to produce.

In their book, Discrete-event System Simulation, Banks et al give several V&V techniques. Some of these include:

- 1) Have the computerized version checked by someone other than the developer.
- 2) Closely examine the model output for reasonableness under a variety of settings of the input parameters.
- 3) Have the computerized representation print out the input parameters at the end of the simulation.
- 4) Verify that what is seen in the animation imitates the actual system.
- 5) Analyze the output data using goodness of fit tests such as Kolmogorov-Smirnov tests and other graphical methods.

Along with his fifteen (15) VV&T principles, Balci specified seventy-five (75) different techniques. He categorized them into informal, static, dynamic and formal techniques. The informal techniques rely heavily on “human reasoning and subjectivity without stringent mathematical formalism” [3]. Informal techniques, as well as static techniques do not require machine execution of the model. Static techniques deal with assessing the accuracy of the model based on the characteristics of the static model design and source code. Dynamic techniques require model execution and are intended for evaluating the model based on its execution behavior. Formal techniques are based on mathematical proof of correctness.

Adrion et al [4] discusses several interesting techniques for V&V. Manual testing includes tasks such as desk checking, peer reviews, walk-throughs, inspections and reviews. He also mentions proof-of-

correctness techniques which he states have been around since “von Neumann time.”

Other V&V techniques that can be used include structural testing such as coverage-based testing. Coverage-based testing is similar to what Pressman [5] refers to as Basis Path Testing using flow graph notation. During this process, the model is analyzed to ensure that all possible paths have been thoroughly tested. Functional testing includes boundary value analysis and cause-effect graphing. Both structural and functional testing can be rather difficult depending on the complexity of the software product being tested. However, there are several software testing tools such as JStyle and TestWorks TCAT that simplify this process.

### 4 Case Study – NOM Simulation

Natural organic matter (NOM) is a “poly-disperse mixture of molecules with different structures, compositions, functional group distributions, molecular weights and reactivities that forms primarily as the breakdown product of animal and plant debris” [6]. It can be found almost anywhere and is rather important in biogeochemistry of aquatic and terrestrial systems. Some of the processes that involve NOM include the evolution of soils, the transport of pollutants and the global biogeochemical cycling of elements [7].

Throughout the years, it has been rather difficult to study the true structure and behavior of NOM. Most research has involved the investigation NOM on a wide scope. Little work has been done in focusing on individual NOM molecules. Perhaps the main reason for this is the amount of time and resources required to research on the molecular level.

A computer simulation is a program written to perform research that would be considered costly and resource-intensive in an actual lab situation. An agent-based computer simulation is a special type of simulation that can “track the actions of multiple agents that can be defined as objects with some type of autonomous behavior” [6]. In the case of NOM, each agent is a molecule. The molecules include protein, cellulose, lignin and so on.

The NOM simulation developed by a group of graduate students of the University of Notre Dame is a stochastic agent-based simulation of NOM behavior in soils. There are presently four different model implementations. These include:

- 1) SorptionFlowModel
- 2) SorptionBatchModel
- 3) ReactionFlowModel
- 4) ReactionBatchModel

This case study focuses on the ReactionBatchModel which is a modeling of laboratory batch adsorption experiments.

In order to successfully represent molecules as individual agents, several properties need to be included. In the ReactionBatchModel, the data used to represent individual molecules include elemental formula, functional group count and record of molecular origin. The elemental formula consists of the number of C, H, O, N, S and P atoms in each molecule. The functional groups may include carboxylic acid, alcohol and ester groups. The record of molecular origin is the initial molecule, its starting position in the system and its time of entry into the system.

In an actual system, NOM molecules move around interacting and reacting with each other. Twelve (12) types of reactions are built into the ReactionBatchModel. These reactions are categorized into:

- 1) First order reactions with split
- 2) First order reactions without split
- 3) First order reactions with the disappearance of a molecule
- 4) Second order reactions

Table 1 gives examples of reactions that fall into each of these categories.

Reaction Name	Reaction Type
Ester condensation	Second order
Ester hydrolysis	First order with split
Amide hydrolysis	First order with split
Microbial uptake	First order with molecule disappear
Dehydration	First order with split
Strong C=C oxidation	First order with split (50% of the time)
Mild C=C oxidation	First order without split
Alcohol (C-O-H) oxidation	First order without split
Aldehyde C=O oxidation	First order without split
Decarboxylation	First order without split
Hydration	First order without split
Aldol condensation	Second order

**Table 1 Chemical Reactions in the Conceptual Model [6]**

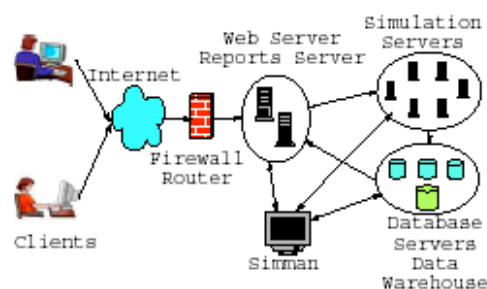
The ReactionBatchModel is designed to include probabilities to determine which reaction will occur with a particular molecule within a 2D space. These are expressed in terms of intrinsic and extrinsic factors. The intrinsic factors are derived from the molecular structure including the number of functional groups and many other structural factors.

The extrinsic factors arise from the environment including concentrations of inorganic chemical species, light intensity, availability of surfaces, presence of microorganisms etc [6].

## 5 Technical Details

The ReactionBatchModel was programmed using J2EE, Repast and an Oracle database. The front-end includes a web interface where the user can log on and run simulations. To begin a simulation, the user enters the extrinsic parameters such as pH, light intensity, enzyme concentrations as well as parameters such as the number of time steps and a seed. Next, the user selects which types of molecules he would like to include in the sample along with the percentage of each molecule. The user is also given the option of creating a new molecule by entering a new name as well as the structural properties. The user then submits the simulation which is given a unique simulation id. Each user who logs on to the web interface can view the status of the simulations that he has submitted.

On the back end there are eight (8) simulation servers and two (2) database servers. A special file server contains a Java program which apportions simulations to each simulation server using a load balancing algorithm. The simulation servers then communicate to the database server to retrieve input parameters and store results. Figure 1 is a diagram of the structure of the system.



**Figure 1 ReactionBatchModel structure [8]**

The data stored in the database produced by the simulation is listed in Table 2.

Name	Comments
Number of Molecules	Number of molecules in the system. It changes as molecules condense, split or are consumed.

MWn	The number-average molecular weight.
MWw	The weight-average molecular weight
Z average	The average charge on each molecule at pH 7.
Element mass	Weight of C, O, H, etc in the system.
Percent of element	The weight percentages of each element
Reactions (1...n)	Number of reactions that occur for each type of reaction

**Table 2 Molecular properties**

### 6 V&V Techniques Used in this Case Study

The V&V techniques used for this case study are a subset of the techniques mentioned in Section 3. Some of these techniques include manual testing, static and structural testing, docking and internal validation.

The first test we conducted on the ReactionBatchModel was manual testing starting with desk checking. This served as a means of familiarizing ourselves with the model as well as checking the code for errors that can be picked out without actually running the system. The other type of manual testing conducted was a face validation of the system. Sargent states that “‘Face validity’ is asking people knowledgeable about the system whether the model and/or its behavior are reasonable. This technique can be used in determining if the logic in the conceptual model is correct and if a model’s input-output relationships are reasonable.” [1] For this test, we addressed two post-doctoral scholars who looked at the data and gave an opinion on the results.

The second set of tests we conducted was static and structural testing. The focus was on code review and complexity analysis using an evaluation copy of software by Man Systems called JStyle. We performed this testing on the Java source files. The results of these tests were on three (3) levels – project, file and class. The project level tested features such as reuse ratio, specialization ratio, average inheritance depth and average number of methods per class. On the file level, two features were noted – the number of source lines per file and the comment density. The features noted on the class level included cyclomatic complexity (CC), weighted methods per class (WMC), response for a class (RFC) and so on. Each of these results was

compared against recommended values found in literature. See Table 3 for further information on each of these metrics [9].

Metric	Measurement Method	Interpretation
Cyclomatic Complexity	# algorithmic test paths	Low => decisions deferred through message passing Low not necessarily less complex
Weighted methods per class	# methods implemented within a class	Larger => greater impact on children through inheritance; application specific
Response for a class	# methods invoked in response to a message	Larger => greater complexity and decreased understandability
Lack of cohesion of methods	Similarity of methods within a class by attributes	High => good class subdivision Low => increased complexity
Depth of inheritance tree	Maximum length from class node to root	Higher => more complex; more reuse

**Table 3 Structural metrics**

For the docking process, we compared the AlphaStep implementation to the ReactionBatchModel. The AlphaStep simulation was developed by Steve Cabaniss. Its conceptual model is the same as the ReactionBatchModel although its implementation is slightly different. Table 4 lists the differences between AlphaStep and the ReactionBatchModel implementations [6].

Features	AlphaStep	ReactionBatchModel
Programming Language	Delphi 6, Pascal	Java
Platforms	Windows	Red Hat Linux cluster
Running mode	Standalone	Web based, standalone
Simulation packages	None	Swarm, Repast libraries
Initial Population	Actual number of different molecules	Distribution of different molecules
Animation	None	Yes
Spatial representation	None	2D grid

Second order reaction	Random pick one from list	Choose the nearest neighbor
First order with split	Add to list	Find empty cell nearby

**Table 4 Differences of features in AlphaStep and ReactionBatchModel implementation**

Table 5 lists the input parameters we used during the docking process. We only used three types different molecules for all the experiments. They are cellulose, lignin and proteins. Since the AlphaStep and ReactionBatchModel implementations have different ways of expressing the initial population of molecules (see Table 4), it was important to estimate so that the number of molecules used in each implementation were comparable. In previous experiments we realized that the average number of molecules of the initial population of the ReactionBatchModel was about 754. Therefore this value was used in the proportions of 34%, 33% and 33% for cellulose, lignin and protein respectively. The equivalent in the AlphaStep implementation was 256, 249 and 249 molecules for cellulose, lignin and protein respectively.

Parameter	AlphaStep	ReactionBatch Model
pH (affects oxidation and decarboxylation rates)	7	7
Light Intensity (affects oxidation and decarboxylation rates)	0.0001 $\mu\text{mol cm}^{-2} \text{hr}^{-1}$	0.0001 $\mu\text{mol cm}^{-2} \text{hr}^{-1}$
Dissolved O <sub>2</sub> (affects oxidation)	0.1 mM	0.0001 M
Temperature (affects thermal (dark) reaction rates)	24.8 C	298 K
Water (scaled 0-1, affects hydrolysis and hydration reactions)	1	1
Bacterial Density (scaled 0-1, affects rate of microbial utilization)	0.1	0.1
<b>Enzyme Activities</b>		
Protease (scaled 0-1, affects hydrolysis rate)	0.1	0.1
Oxidase (scaled 0-1, affects oxidation rate)	0.1	0.1
Decarboxylase (scaled 0-1, affects decarboxylation rate)	0.1	0.1
<b>Batch Control</b>		
Reaction Time (hrs)	1000	1000.25

Delta T	0.1	0.25
Sample Interval	500	1

**Table 5 Parameters Used.**

Table 6 gives a list of elemental composition and functional groups of each of the molecules used.

Elemental Composition	Protein	Cellulose	Lignin
Carbon	240	360	400
Hydrogen	382	602	402
Nitrogen	60	0	0
Oxygen	76	301	81
<b>Function Groups</b>			
C=C bonds	15	0	160
Rings	5	60	40
Phenyl rings	5	0	40
Alcohols	10	182	2
Phenols	0	0	1
Ethers	0	119	89
Acids	6	0	0
Amines	6	0	0
Amides	54	0	0

**Table 6 Elemental and functional group composition of molecules used.**

We ran 25 simulations of each implementation and graphed the average number of molecules, number-average molecular weight, weight-average molecular weight, weight percentage of Carbon and total mass of Carbon after a total of 1000 simulation hours. In addition to graphing, which allowed for visual analysis of the results, we ran statistical tests on each of the results. Two different types of tests were ran – Student’s t test and Kolmogorov-Smirnov tests.

Xiang et al state that “a simulation model that uses random seeds must have statistical integrity in that independent simulations with the same input data should have similar results. If the simulation model produced large variabilities because of random seeds, there would be a considerable problem with it” [6]. In order to determine internal validation of the ReactionBatchModel simulation we ran 141 simulations. The actual number of simulations attempted was much higher however a problem with the load balancing algorithm resulted in a significant number of failed simulations. Each of the 141 simulations was run for 1000 simulation hours. A histogram of the total number of molecules was generated.

## 7 V&V Results

Unfortunately, the results of the face validation were inconclusive. The post-doctoral scholars informed us that it was difficult to determine whether the results were reasonable based on the lack of actual system data involving similar processes. They were concerned that the study did not indicate the number of the types of reactions that occur in the system. Also, they admitted that the area of microbiology was not their specialty and therefore suggested repeating the experiments with a bacterial uptake of 0.

The results of the static and structural tests conducted using JStyle were generally good. The results on the file level, however, were questionable. The average number of source lines per file was too high. Longer files are usually harder to debug and is generally not recommended. The comment density was also too low.

Although the overall results for the structural tests were good, this does not indicate that structurally the entire simulation was acceptable since only a subset of the files was tested. Table 7 displays the results.

Parameters	Value	Interpretation
<b>Project Level</b>		
Number of classes	20	Good
Number of abstract classes	0	Good
Reuse ratio	0.05	Low (but Repast modules were not considered)
Specialization ratio	19	High – not good
Average inheritance depth	1.2	Good
Class hierarchy depth	3	Good
Number of methods per class	10.650	Good
Method size – mean	13.620	Good
<b>File Level</b>		
Average number of source lines per file	185.8095	Very high
Comment density	16.59952	Very low
<b>Class Level</b>		
Cyclomatic Complexity	2.6	Good
Weighted methods per class	30.77	Not good
Response per class	24.5387	Good
Lack of cohesion of methods	50.79365	Fair

Depth of inheritance tree	1.33	Good
---------------------------	------	------

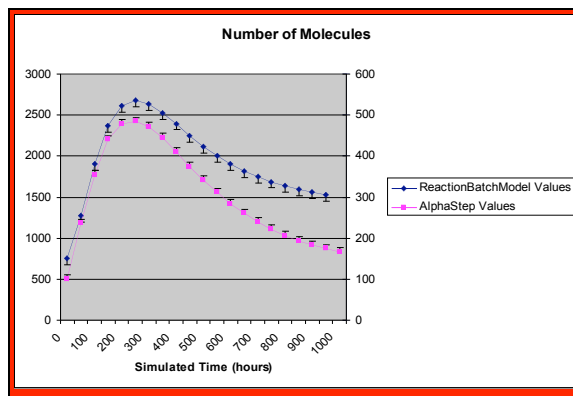
**Table 7 Structural results.**

The docking tests had to be conducted twice. The results of the initial test did not appear to be correct based on previous tests performed by Xiang [6]. After further investigation, the following factors were altered to give much more accurate results:

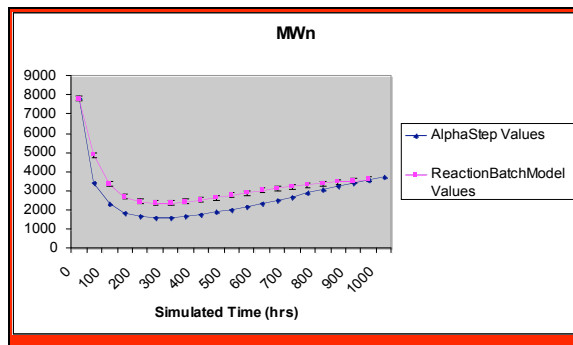
1) **Units of measurement.** The units for the input parameters were different for the AlphaStep and ReactionBatchModel implementations. For example, the AlphaStep implementation accepts temperature values in degrees Celsius while the ReactionBatchModel implementation accepts temperature values in Kelvin.

2) **The initial population of molecules** had to be adjusted. As noted in Table 4, the initial population of molecules in the AlphaStep implementation is the actual number of molecules while the ReactionBatchModel implementation uses the distribution or percentage of different molecules.

Figure 2 a-d shows the results of the first trial while Figure 3 a-e shows the results of the second trial.



**Figure 2 a) Number of Molecules (first trial)**



**Figure 2 b) MWn (first trial)**

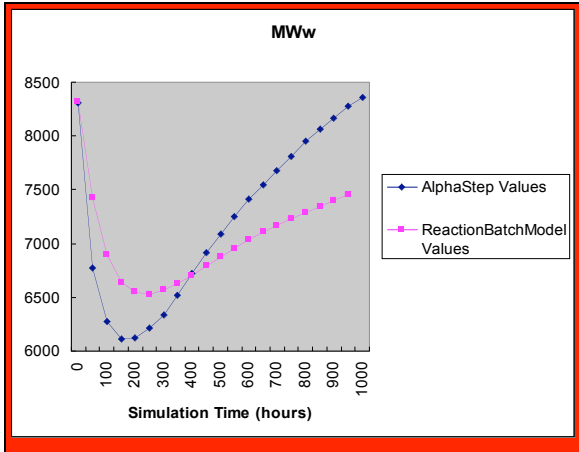


Figure 2 c) MWw (first trial)

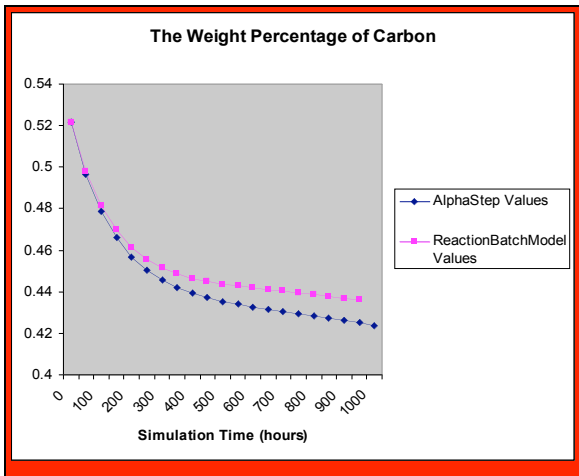


Figure 2 d) Weight percentage of Carbon (first trial)

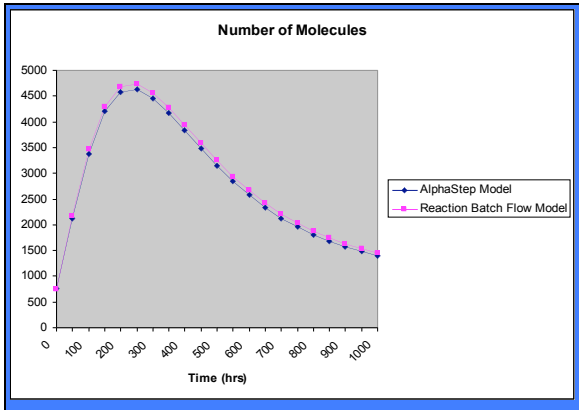


Figure 3 a) Number of Molecules (second trial)

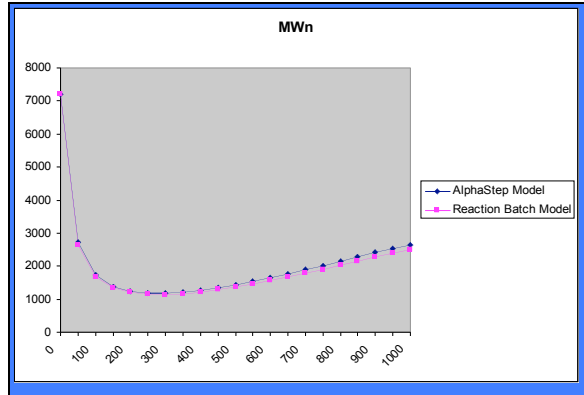


Figure 3 b) MWn (second trial)

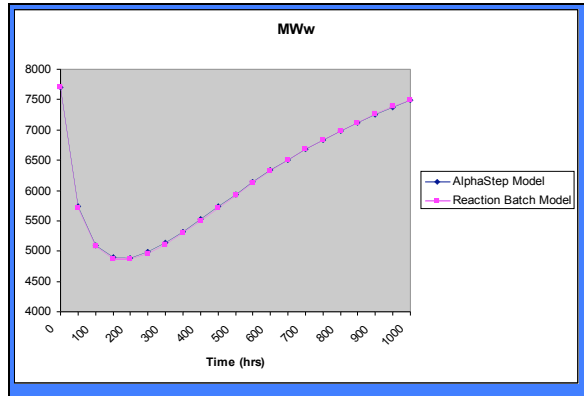


Figure 3 c) MWw (second trial)

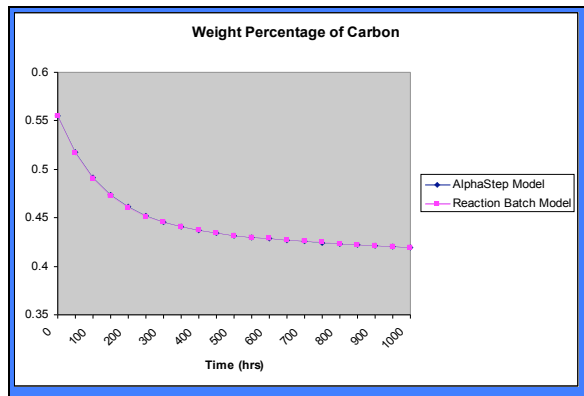
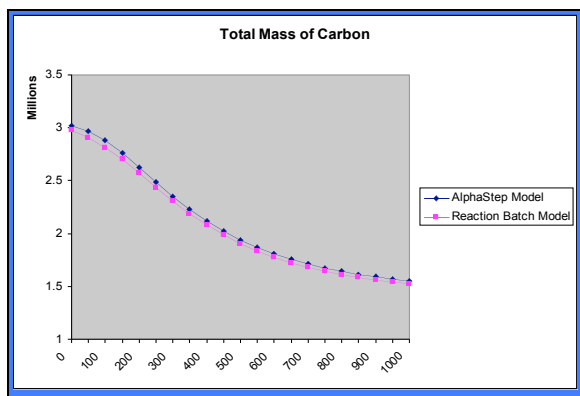


Figure 3 d) Weight percentage of Carbon (second trial)



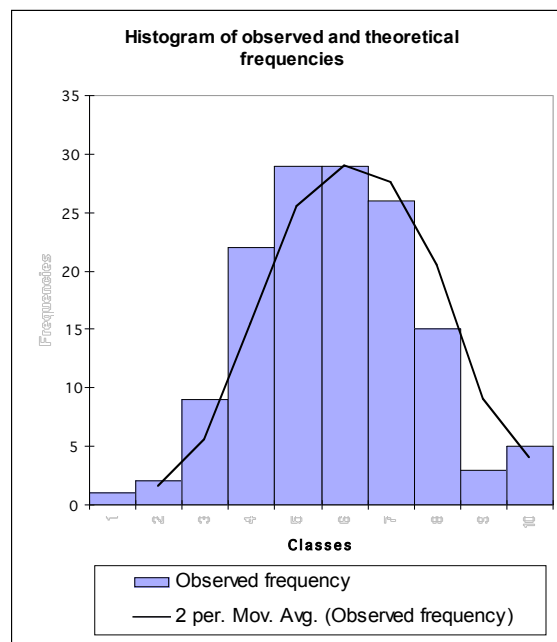
**Figure 3 e) Total mass of Carbon (second trial)**

Although the results of the second trial appeared to indicate that both simulations produced the same molecular properties, the statistical tests proved otherwise. Table 8 gives the result of the statistical tests. For the Student's t tests results, significant indicates a rejection of the null hypothesis of equality of means at a level of significance  $\text{Alpha}=0.05$  and a confidence interval at 95%. Not significant indicates that the difference between the means is not significant. For the Kolmogorov-Smirnov test significant indicates a rejection of the null hypothesis that the samples are not different. Not significant indicates that the difference between the samples is not significant.

	Student's t test	Kolmogorov-Smirnov Test
<i>Number of Molecules</i>	Significant	Significant
<i>MWn</i>	Significant	Significant
<i>MWw</i>	Not Significant	Not Significant
<i>Weight Percentage of Carbon</i>	Not Significant	Not Significant
<i>Total Mass of Carbon</i>	Not Significant	Significant

**Table 8 Results of statistical tests.**

The result of the internal validation is shown in Figure 4. We also applied Kolmogorov-Smirnov and Chi-square tests to these results. The results of the Kolmogorov-Smirnov test indicate whether the difference between empirical and theoretical cumulative distributions is significant. The results of Chi-square test indicate whether the difference between the observed frequencies and the theoretical frequencies (where  $\mu = 1469.816$ ,  $\text{Sigma}^2 = 9192.294$ ) are significant. Table 9 gives the results.



**Figure 4 Probability distribution fitted to the data: Normal N ( $\mu = 1469.816$ ,  $\text{Sigma}^2 = 9192.294$ ).**

Test	Significance
Kolmogorov-Smirnov	Not significant
Chi-Square	Not significant

**Table 9 Results of statistical tests on 141 simulation runs of the ReactionBatchModel.**

## 8 Conclusions and Future Work

The verification and validation of simulation models is quite important. Unfortunately, even after conducting a battery of tests it is still often difficult to come to a conclusion. In this case although tests such as the internal validation appear to indicate that the model is valid, other tests such as the statistical tests and face validation are inconclusive. Several activities can help improve the confidence in the system in the future. Some of these would be:

1) Face Validation – include an analysis of the different types of reactions that occur during the simulation hours. The tests included in this study focused on the molecular properties retrieved from the system. Further work would have to be done in graphing and statistically testing the number of reactions. This would help in determining the validity of the system.

2) Static and Structural Tests – include all the code including the GUI in the tests.

3) Internal Validation – redoing the test to include more than 141 simulation runs.

## References

- [1] Robert G. Sargent. Verification and Validation of Simulation Models. D.J. Medeiros, E.F. Watson, J.S. Carson and M.S. Manivannan, eds. *Proceedings of the 1998 Winter Simulation Conference*, 1998.
- [2] J. Banks, J.S. Carson, B.L. Nelson, and D.M. Nicol. *Discrete-event System Simulation*. Industrial and Systems Engineering, Prentice Hall, 3rd ed., 2001.
- [3] O. Balci. *Handbook of Simulation: Principles, Methodology, Advances, Applications and Practice*, Chapter 10 Verification, Validation and Testing. John Wiley & Sons, New York, 1998.
- [4] W. Richards Adrion, Martha A. Branstad and John C. Cherniavsky. Validation, Verification and Testing of Computer Software, *Computer Surveys*, Vol 14, No. 2, June 1982.
- [5] Roger S. Pressman. *Software Engineering: A Practitioner's Approach*, McGraw-Hill Publishing, 6<sup>th</sup> ed, 2005.
- [6] Xiaorong Xiang, Ryan Kennedy and Gregory Madey. Verification and Validation of Agent-based Scientific Simulation Models. *Agent-Directed Simulation Conference*, San Diego, CA, April 2005.
- [7] S. E. Cabaniss. Modeling and Stochastic Simulation of NOM reactions, working paper [http://www.nd.edu/~nom/Papers/WorkingPaper\\_s.pdf](http://www.nd.edu/~nom/Papers/WorkingPaper_s.pdf), July 2002.
- [8] Yingping Huang and Greg Madey, "Towards Autonomic Computing For Web-Based Simulations", *International Conference on Cybernetics and Information Technologies, Systems and Applications (CITSA 2004)*, Orlando, July, 2004
- [9] Natural Aeronautics and Space Administration, Software Quality Metrics for Object Oriented System Environments, June 1995.