DUNIP TECHNOLOGIES

WHITEPAPER

Automated Attention Agent Assessment of Current States and the Prediction of Future States using DEVS Cognitive Agent

Phil Hammonds, Saurabh Mittal, Bernard P. Zeigler

(2007)

Abstract

This paper describes a methodology for modeling attention to a set of conditions that are defined by an expert or an interested party who knows or intuits that some set of indices deserves regular attention and that attention will result in useful knowledge for near term prediction of future states of the indices. The ability to monitor many indices efficiently and capture information that may be important is the primary goal of the approach. The logical model is based on a hierarchy of "needs conditions" that are satisfied if the criteria are met and then decay with time. The methodology is based on the dynamic generation of Discrete Event System Specification (DEVS) models and their simulation and subsequent comparison to snapshots of real states. A webservice prototype is presented and described.

Introduction

The ability to predict future states from presently available information has been a common pursuit in most human endeavors and has met with varying degrees of success. The ability to pay attention to the relevant information at the right interval is a fundamental aspect of prediction. In the physical world, Newtonian mechanics has been one of the most successful models for prediction. An observer with a good understanding of the principles and an efficient model of attention to the current state of variables and rate of change, may reliably predict the future position of a body in motion with known direction and forces acting upon it. The finer the resolution of observation or attention, the better the model may be used to predict the next position of interest. Of course, all of the important variables must be known in order for the current state to be used to predict a subsequent future state. Large gaps in attention to those states, may result in poor prediction, but constant attention to the states is inefficient.

Predicting future states of human associated or influenced behavior has been much less successful. The number of variables, the number of participants and the number of potential states, as well as the inability to pay attention to many variables at the

appropriate resolutions simultaneously contribute to the inability to reliably predict. However, in many cases restricted local or probabilistic models with high resolution and high observational frequency do have some degree of success. Combinations of valid models that can be executed by powerful computers combined with subject matter interpretive expertise often lead to very useful predictions. Just as a Newtonian model of a body in motion requires laws or contextual restrictions for its future predictive ability, so do models of human associated behavior. Predicting human influenced action (HIB) is not so different in the sense that the variables that combine must be well understood and observable at the appropriate resolution. It is often the case that we suspend "low contribution" variables or reduce the attention in terms of frequency of state intervals observed when predicting the future states of physical bodies in motion, but exclaim that human behavior has too many variables and must be constantly monitored in order to understand or predict. However, there is every day evidence that suggests that human behavior and performance are predictable to a degree that causes the populations being modeled to be concerned. A case in point is the careful observation by a football or basketball coach of films or video of players on an opposing team. If this were not a useful and predictive practice, there would be no restrictions for observation and especially the frequency of observation of team practices. In most professional leagues, observation for the purpose of modeling for analysis and subsequent prediction of behavior are severely restricted by sanctioning organizations. It is a fact that most human organizations such as a states, governing bodies, military organizations and corporations limit the access of outsiders and consequently outsiders' ability to model and simulate behavior in order to make useful predictions that could be profitable in some way to the observer's own organization. It is also a fact that organizations who are under constant observation display behavior that is meant to cause the outside observer to generate invalid models and therefore restrict the predictive capability of those models. Poker players regularly employ bluff in wagering or deception of facial or physical movement to give opponents false "tells." Restricted access and deception are fundamental human behaviors designed to limit a competitor's ability to defend themselves or gain advantage or from observation and behavior modeling. Law enforcement, courts and other organizations that have broad powers to detain and interrogate are generally able to remove or substantially reduce access barriers and they can generally test whether human subjects are engaged in deception, thus they are able to pay attention and generate models that aid in the reconstruction of means, motives and opportunities for criminal behavior and in many cases the same models may help predict aspects of future behavior.

There are three areas of interest in which the proposed attention agents could be particularly useful. The first is the ability to examine a domain with an unknown future. In this case we attempt to predict a future where likely states are unknown based only upon a combination of historical, current or developing information, e.g., an unfolding disaster such as hurricane Katrina, or the Indonesian Tsunami. The second scenario focuses on a known or "hoped for" optimal future state. An example of the last scenario would be a case in which there is a known standard that is the goal, but the current state

of affairs/practices/process is not aligned with the steps needed to attain that standard, e.g., a small software company seeking CMMI, or ISO certification of processes. The third scenario focuses on an unknown but likely future, e.g., a stock may go up, go down or stay the same value from day to day, so this is a case where there are limited likely outcomes with degrees of value change. The attention model provides the capability for an expert's attention to some matters to be emulated. In particular, this approach could be used to change the attention to broadcasts or monitoring software to recreate the amount of attention that an expert might pay. Although, it would not be able to emulate the capability once the attention was focused, that capability would need to be developed separately.

High Level Model Description

This model focuses on the amount of attention and duration one should devote to a given index or behavior. The goal of this design is to maximize attention to important information or changes in information while minimizing the overall attention and therefore processing load. It is possible to pay attention to *n* details of *n* indices as long as *n* is very small. This model seeks to provide and approach by which we can examine larger numbers of details and indices in an expert like way. Experts can keep track of several details of many indices, but there are limits to human attention. This effort seeks to maximize the efficiency of an expert to examine very large numbers of details and indices. Once the model is validated to the satisfaction of the user, it can then be used to deploy other agents that can act in predefined circumstances or combinations of circumstances or it can simply alert the user that it is time to look at an index. The ability to dynamically update the model is anticipated as necessary in future versions.

The model does not rely on probability in any way. It is simply a matter of availability, thresholds and decays of satisfaction states and multiple interactions. The model is loosely based on a biological model were living creatures devote a certain amount of attention to certain activities based on internal needs that are satisfied for a short period of time, but require attendance and maintenance. For example, humans can only go without water for a relatively few hours before the need to drink water becomes overwhelming at which point all activity is focused on finding and consuming water. Once the need for water is satisfied then other needs can be addressed that have progressively lower priorities. The model in this paper seeks to satisfy high priority needs first and then moves to fulfill lower priority needs. Once a need is fulfilled, a clock starts and the “satisfaction state” begins to decay at a predetermined linear rate. Multiple needs may be addressed in this model, but no two needs may have the same priority, .e.g., a human model might require that water always is the highest priority, then food, then shelter, then companionship, then art, etc. It is important to remember, that this model does not take into account external events explicitly except availability. If something is not available at all it may be due to some catastrophic event that is external to the agent, but arguably it is just a matter of perspective. For example, the destruction of an

environment or some aspect of the environment is only relevant in the sense that an agent's ability to satisfy needs is adversely affected.

Modeling and Simulation Architecture

This model and simulation relies on Discrete Event System Specification (DEVS) for its design and execution. The DEVS formalism was introduced by Bernard Zeigler [1], [2] to provide a means of modeling discrete event systems in a hierarchical and modular way. DEVS exhibits the concepts of system theory and modeling, and supports capturing the system behavior in the physical and behavioral perspectives. A DEVS model can be either an atomic or coupled model. In the DEVS formalism, a large system can be modeled by both atomic and coupled models. The atomic model is the basic model that describes the behavior of a component. A Discrete Event System specification (DEVS) atomic model is defined by the structure in Figure 4.

$M = \langle X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$

where

X is the set of input values
S is the set of state
Y is the set of output values
δ_{int}: S S is the internal transition function
δ_{ext}: Q x X S is the external transition function, where
Q = {(s,e)|s εS, 0 □ e □ ta(s)} is the total state set, and e is the time elapsed since last transition
λ: SY is the output function
ta: $SR_0^+{}_{,inf}$ is the time advance function

Figure 1: Classic DEVS Specification

Atomic and coupled models can be simulated using sequential computation or various forms of parallelism. The basic parallel DEVS formalism extends the classic DEVS by allowing bags of inputs to the external transition function, and it introduces the confluent transition function to control the collision behavior when receiving external events at the time of the internal transition. The parallel DEVS atomic model is defined by the structure in Figure 5.

$M = \langle X, S, Y, \delta_{int}, \delta_{ext}, \delta_{con}, \lambda, ta \rangle$
where

X is the set of input values
S is the set of state
Y is the set of output values
δ_{int}: S S is the internal transition function
δ_{ext}: Q x X^b S is the external transition function,
where X^b is a set of bags over elements in X, Q is the total state set.
δ_{con}: S x X^b S is the confluent transition function,
subject to $\delta_{con}(s, \Phi) = \delta_{int}(s)$
λ: SY^b is the output function
ta: S $R_{0}^{+}{}_{,inf}$ is the time advance function

Figure 2: Parallel DEVS Specification

A DEVS-coupled model designates how atomic models can be coupled together and how they interact with each other to form a complex model. The coupled model can be employed as a component in a larger coupled model and can construct complex models in a hierarchical way. The specification provides component and coupling information. The coupled DEVS model is defined as the structure in Figure 3.

$M = \langle X, Y, D, \{M_{ij}\}, \{I_j\}, \{Z_{ij}\} \rangle$
Where

X is a set of inputs
Y is a set of outputs
D is a set of DEVS component names
For each i D,
$\mathbf{M_i}$ is a DEVS component model
$\mathbf{I_i}$ is the set of influences for I
For each j I_i,
$\mathbf{Z_{ij}}$ is the i-to-j output translation function.

Figure 3: Coupled DEVS Specification

Three different DEVS formalisms have been introduced. The classic DEVS formalism treats components sequentially, and the parallel DEVS formalism treats components concurrently. These formalisms also include the means to build coupled model from atomic models.

The Hierarchy of System Specifications

System theory deals with a hierarchy of system specification which defines levels at which a system may be known or specified. Table 1 shows this Hierarchy of System Specifications.

- At level 0, we deal with the input and output interface of a system over a time base.
- At level 1, we observe the behavior of the system by gathering a collection of all I/O pairs.
- At level 2, we add the initial states to the specification. When the initial states are known, there is a functional relationship between the inputs and outputs.
- At level 3, the system is described by the state space and the state transition functions. The transition function describes the state changes as the system responds to inputs and generates outputs.
- At level 4, we specify how the system is composed of interacting components in a coupling structure. Each component is a system on its own with state set and state transition functions. One property of a coupled system, called "closure under coupling," guarantees that a coupled system at level 3 itself specifies a system. This property allows hierarchical construction of systems, i.e., that coupled systems can be used as components in larger coupled systems.

Level	Name	What we specify at this level
4	Coupled Systems	System built up by several component systems which are coupled together
3	I/O System	System with state and state transitions to generate the behavior
2	I/O Function	Collection of input/output pairs constituting the allowed behavior partitioned according to the initial state the system is in when the input is applied
1	I/O Behavior	Collection of input/output pairs constituting the allowed behavior of the system from an external Black Box view
0	I/O Frame	Input and output variables and ports together with allowed values

Table 1: Hierarchy of System Specification

As we shall see in a moment, the system specification hierarchy provides a mathematical underpinning to define a framework for modeling and simulation. Each of the entities (e.g., real world, model, simulation, and experimental frame) will be described as a

system known or specified at some level of specification. The essence of modeling and simulation lies in establishing relations between pairs of system descriptions. These relations pertain to the validity of a system description at one level of specification relative to another system description at a different (higher, lower, or equal) level of specification.

Framework for Modeling and Simulation

The modeling and simulation framework defines entities and their relationships that are central to the M&S enterprise [1]. The basic entities of the framework are source system, model, simulator, and experimental frame as illustrated in Figure 4, and they are linked to the modeling and simulation relationships.

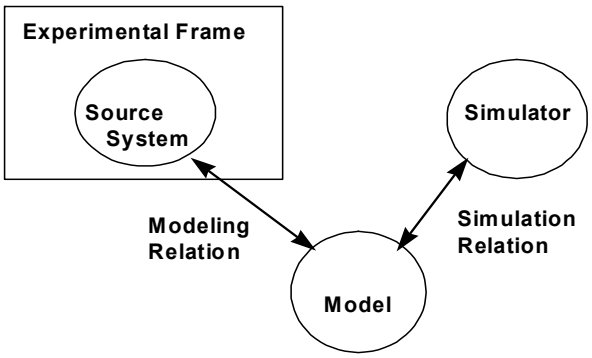

Figure 4: Basic M&S Entities and their relationships

The source system is the real or virtual environment that we are interested in modeling. It is viewed as a source of observable data in the form of time-indexed trajectories of variables. The data that has been gathered from observing or otherwise experimenting with a system is called the system behavior database. The data is viewed or acquired through experimental frames of interest to the modeler.

The Experimental Frame is a specification of the conditions under which the system is observed or experimented. It reflects the objectives of the experiment performed on a real system or simulation. An experimental frame specification consists of four major subsections: input stimuli, control, metrics, and analysis [1].

Multiple experimental frames can be used for a single system, and the single experimental frame can be applied to many systems. Conversely, there are multiple objectives to test a system, and a single objective is applied to many systems. There are two valid views of an experimental frame. A frame can be viewed as data element type that is entered into a database. Another view is that a frame can interact with the system to obtain data under specified conditions. In the second view, the frame can be treated as

an observer, and it has three components: generator, acceptor, and transducer, as illustrated in Figure 8. The Generator describes the inputs applied to the system or model. The Transducer observes and analyzes the system output. The Acceptor monitors the experiment to see the experimental condition, and compares the generator inputs with the transducer outputs.

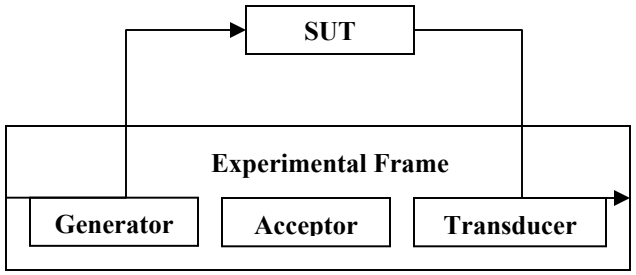

Figure 5: Experimental Frame

Dynamics of the Learning Agent[1]

The agent has following attributes [3]):

1. ***DirectiveThresholdValue***: This is the value above which the agent will strive for a higher priority directive. This is currently statically defined for all the directives that the agent is programmed to search and learn. It is adjustable for each agent. For example, if the threshold is set at and arbitrary value of 4.5 and the agent's satisfaction level stays above 4.5, it will continue to try to satisfy the next highest priority directive.

2. ***DecayFactorPerSecond***: This value decays the memory strength of the agent with respect to time in seconds. If the agent is searching for a directive, i.e., it is in phase 'Directive1", if not replenished, the strength of this directive will diminish over time, guided by this value. This is programmable and is passed during agent configuration

3. ***List of Directive and corresponding Strengths***: This is an internal attribute of the agent and constitutes its memory. If the environment fails to provide constant information that satisfies the imperatives or directives, the memory decays over time. If agent is "fed" from environment, its memory strengthens and is cumulatively increased

[1] Dr. Saurabh Mittal created the DEVS implementation from author's design

4. ***List of Directive and Phases***: This is also an internal attribute of the agent and it maintains a list of directives to satisfy and their priority. The directives are number ordered starting from priority 1. The agent starts in this highest priority search mode and as long as it remains satisfied, it moves on to higher order directives.

Environment Model

The environment is specified as a list of tuple

< Directive, Priority, availabilityInterval >

It is specified in file DataInputFiles/DirectiveGenerators/ModelX.txt

Each tuple configures a DEVS generator with period 'availabilityInterval'. During the agent initialization, the agent builds up the list of 'DirectiveAndPhases' based on data specified in the environment file. The list constitutes a digraph Environment model with atomic DEVS directive availability generators.

Testing Model

The environment is specified in a list of tuples

< Directive, Priority, testingInterval >

It is specified in file DataInputFiles/DirectiveAnalyzers/AnalyzerX.txt

Each tuple configures a DEVS generator with period 'testingInterval'. The list constitutes a digraph Tester model with atomic DEVS generators that generate the 'test' traffic.

Modeling Attention

When input from the environment, i.e., a condition that satisfies a directive, the agent consumes it and strengthens itself. The agent will only attend to the condition or information if it is looking for the same Directive. For example, if the agent is looking for Directive2 information and the environment presents it with Directive 1 or 3 information, the agent will ignore the 1 or 3 level information if the directive conditions are either satisfied or not addressed yet. However, if Directive2 information appears in the environment, the agent reacts positively.

Primitive Learning

The agent maintains a database of tuple <Directive, strength> that constitutes its memory. This decays and get replenished over time based on the agent's observation behavior and the environment. The agent is consistently is aware of high priority directives and if any of them falls below the individual directive threshold value, the agent abandons the current directive satisfaction search and moves to satisfy the higher priority. Once satisfied above the threshold value, the agent then moves to lower priority directives in successive manner.

Testing

The agent is continuously monitored by the test environment and periodically tests each Directive's satisfaction level based on data provided in AnalyzerX.txt. If the agent is looking to satisfy DirectiveX and a higher priority Test request comes i.e. 'tell me the status of DirectiveY where priorityY>priorityX, the agent suspends looking DirectiveX and starts looking for DirectiveY. Once the period of lookup (as specified in environment specifications) elapses, it resumes looking for DirectiveX.

Agent Behavior

1. The agent continuously moves to satisfy lower priority directive if the higher priority directive is satisfied, i.e., the value exceeds the threshold value.

2. The agent continuously checks each directive's satisfaction level

3. The agent memory continuously decays over time according to variable *DecayFactorPerSecond*

Simple Application of the Model

Simple Creature Example
Water: 8
Food: 6
Shelter: 10
Education: 20
Art: 30
Decay Factor: 0.09 units per second

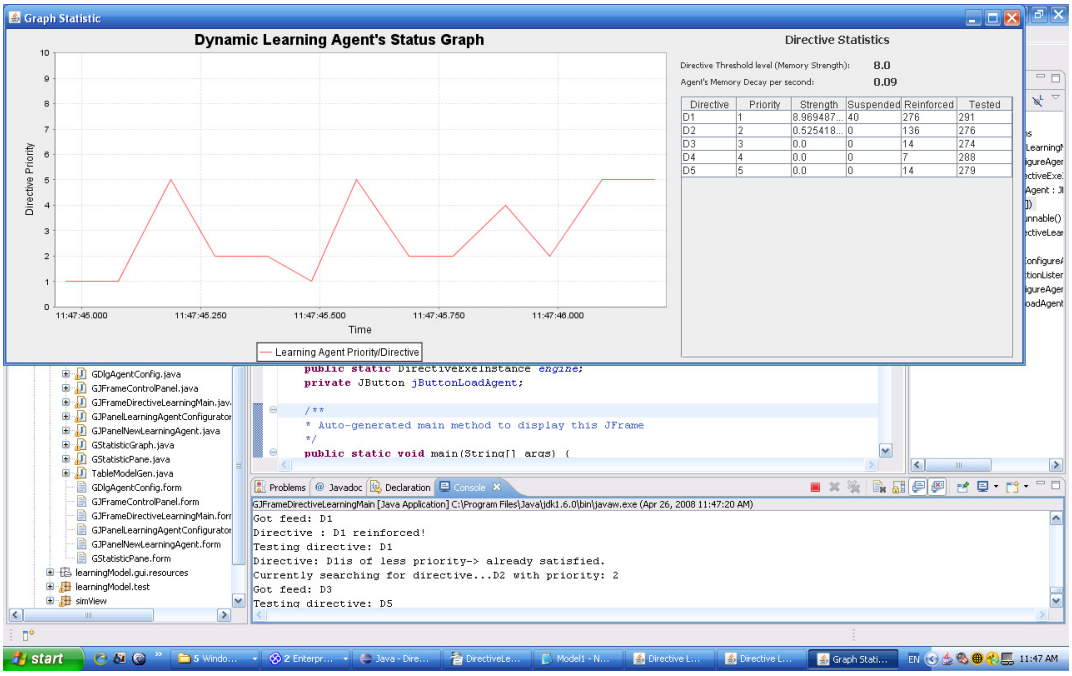

Figure 6.

Interpretation of Statistic Graph

Graph on the left hand side is self Explanatory. It's a plot of agent's status, i.e., in state 'looking for Directive X', with respect to time. The right hand side table gives a count of Strength, Suspended, Reinforced, and Tested on a per Directive basis.

The agent starts by seeking to satisfy the highest priority directive, which is water. It continues to look for water until it is satisfied with 8 units. Once satisfied, it moves to find food and starts looking for Food. While searching for food, the water level diminishes and if it goes below 8, the agent abandons food and searches for water. This

process continues and the agent keeps on vacillating between Water and Food. When Food touches 8, the agent starts looking for shelter and the process continues, with decay happening in the background for all water, food and shelter. All the directive strengths are initially zero when the agent initializes. In this example even though the time scale is small, it is evident that the small amounts of water and food available to the creature only allow short forays for shelter and the lower priority items, even though those items above shelter are in plentiful supply. This short example illustrates the utility of the model. We can examine a domain for those items of interest or that satisfy some need and then depending on the availability of those items and their priority we can conduct automated searches of many domains at once and receive alarms or notification or generate automated responses when thresholds are exceeded.

Interpretation of the Chart

Graph on the left hand side is a plot of the agent's status (i.e. in state 'looking for Directive X', with respect to time. The right hand side table gives a count of Strength, Suspended, Reinforced, and Tested on a per Directive basis.

Below is the information generated as the model is executed. This model executes very quickly so it is difficult to make a screen capture and stop the model before the queue fill up, but the output allows one to see how the creature's search through its environment results in satisfaction of the directives assigned in the configuration file.

```
Directive Strengths: [(D5,0.0)(D4,0.0)(D3,0.0)(D2,2.447)(D1,8.16)]
=================

Testing directive: D3
Currently searching for directive...D1 with priority: 1
Directive: D3is of higher priority-> searching...
Suspending searching of D1....
Initiating search for directive: D3
Got feed: D1
Testing directive: D1
Directive: D1is of less priority-> already satisfied.
Currently searching for directive...D3 with priority: 3
Got feed: D2
Testing directive: D4
Current directive strength is very low.
Can't test higher priority directive: D4
Got feed: D3
Directive : D3 reinforced!
Testing directive: D5
Current directive strength is very low.
Can't test higher priority directive: D5
Got feed: D1
Got feed: D2
Got feed: D4
Testing directive: D2
```

```
Directive: D2is of less priority-> already satisfied.
Currently searching for directive...D3 with priority: 3
Directive: D1     NEEDS reinforcement!
Searching for directive: D1
----------------
Agent status:
Directive: <D1    1     7.0>
Directive Strengths:
[(D5,0.0)(D4,0.0)(D3,0.53)(D2,1.524)(D1,7.236)]
=================

Testing directive: D3
Current directive strength is very low.
Can't test higher priority directive: D3
Testing directive: D1
Directive: D1is of less priority-> already satisfied.
Currently searching for directive...D1 with priority: 1
Testing directive: D4
Current directive strength is very low.
Can't test higher priority directive: D4
Got feed: D3
Testing directive: D5
Current directive strength is very low.
Can't test higher priority directive: D5
Got feed: D2
Got feed: D1
Directive : D1 reinforced!
----------------
Agent status:
Directive: <D1    1     7.0>
Directive Strengths: [(D5,0.0)(D4,0.0)(D3,0.0)(D2,0.894)(D1,7.606)]
```

Table 2. System level output of simulation

Parser

In order to capture data of interest a web based parser is needed. The service allows you to scan text from a web page or take in an Really Simple Syndication (RSS) feed and define directives, their priorities, frequencies, thresholds and then test criteria. The parser searches the XML space for tags and text defined by the user. The inconsistency of RSS formatting represents a significant challenge to the automation of this effort. The parser used in this project searched HTML tags and text for patterns of interest.

Model Parameter Loader

Once the directives of interest, thresholds and rates of decay have been identified, they must be input into into the model for simulation. A simple formating program in the

form of a jar file is used to capture the data for input into the model in the form of two text files. One for the model and one for the tests -The model file: Model1.txt and the test file: Analyzer1.txt

```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Phil Hammonds.KRECHET>java -jar "C:\Documents and
Settings\Phil
Hammonds.KRECHET\MyDocuments\NetBeansProjects\DirectiveModelGenerator
\dist\DirectiveModelGenerator.jar"
To terminate input, type the end-of-file indicator
when you are prompted to enter input.
On UNIX/Linux/Mac OS X type <ctrl> d then press Enter
On Windows type <ctrl> z then press Enter

Enter DirectiveName (> 0), DirectivePriority, DirectivePeriod, DirectiveThreshol
d and DirectiveDecay.
? 1 1 17 .005 1.5
Enter DirectiveNumber (>0), DirectivePriority, DirectivePeriod, DirectiveThresho
ld and DirectiveDecay.
? 2 2 6 .003 20.5
Enter DirectiveNumber (>0), DirectivePriority, DirectivePeriod, DirectiveThresho
ld and DirectiveDecay.
? 3 3 3 .09 2.5
Enter DirectiveNumber (>0), DirectivePriority, DirectivePeriod, DirectiveThresho
ld and DirectiveDecay.
?4 4 7 .6 5.5
```

Table 3. Input to Java model parameter formatting program.

1	1	17	0.005	1.5
2	2	6	0.003	20.5
3	3	3	0.09	2.5
4	4	7	0.6	5.5

Table 4. Formatted Model input

Financial Example

In order to test the usefulness of the analytical and predictive engines, a domain of interest is needed with constraints in which the learning engine can function. For the purposes of this paper, I have chosen to examine four different indices of financial market health, the Federal Funds Discount Rate (DR) (The interest rate which an eligible depository institution is charged to borrow short-term funds directly from a Federal Reserve Bank), The Dow Jones Industrial Average, The US Dollar Index and the Commodity Research Bureau (CRB) index.

The primary index of interest is the Discount Rate. The agent will monitor the Discount for changes that take it below a certain yearly adjusted rate. The agent can be programmed to look at very fine levels of resolution, so that one could take either a strategic or a minute by minute tactical look at a market and be notified. If the rate remains above this rate the Agent will next monitor the DJIA, as long the average remains above a rate which is adjusted for the year. The next index of interest will be the US Dollar Index, the next will be Gold and then CRB. For each year a threshold was defined and the model adjusted to threshold for each index. It is important to mention that there is no preconceived cause and effect relationship implied. It may be the case that the DR is lowered for several reasons - a few of which may be indicators based on these indices, it also may be the case that the other indices then respond to change in the DR

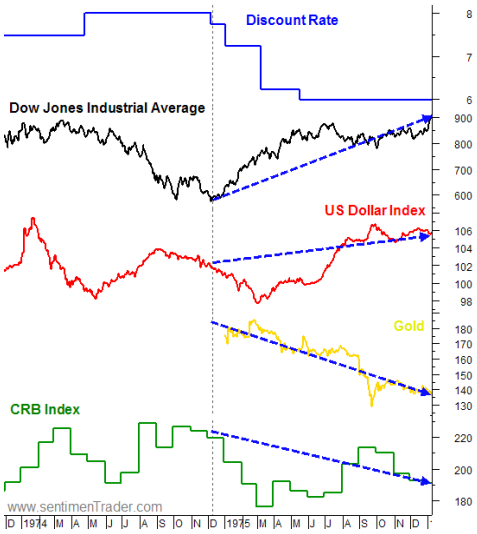

figure 7. **1974 showed a 2 point reduction in 3 months in the DR**

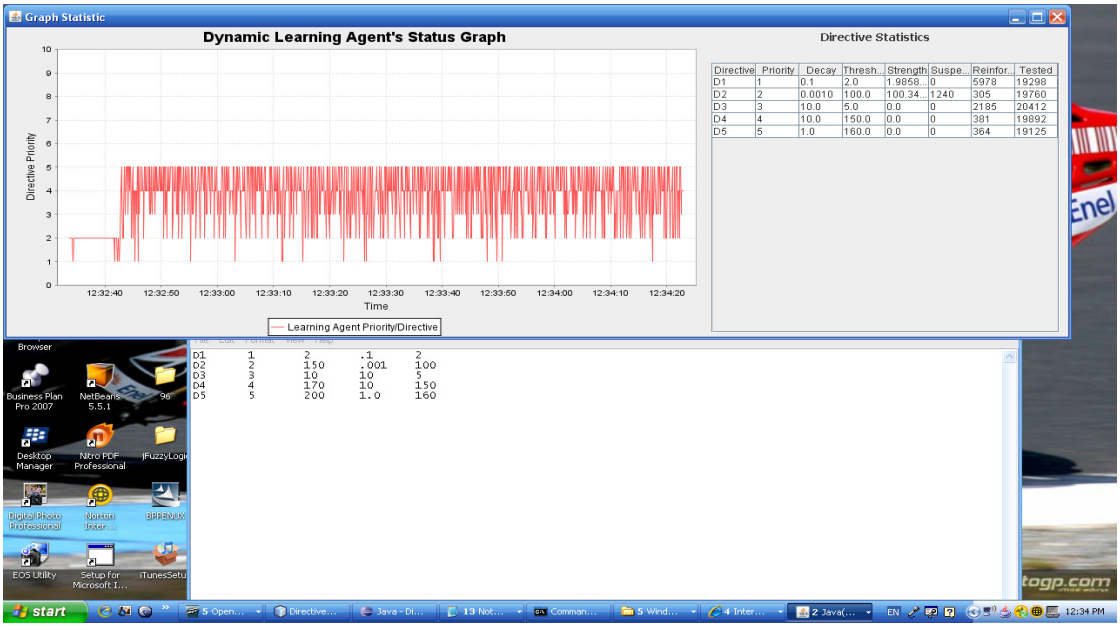

Figure 8. Simulation output for 1974 model.

Interpretation: The scale adjusted parameters indicate in this set of market conditions that the lower priority items deserve attention because the higher priority needs are being satisfied, namely the DR, DJIA and Dollar Index are above threshold, so the other indices should be watched more carefully.

This graph (figure 8.) represents the attention paid to the indices. For the DR, I suggested that the availability of information would be 2 points per unit time with a decay rate of .1 unit per unit time and a threshold of 2. In other words, the agent looks to see if there are at least 2 units of information per unit time and that a threshold of 2 is maintained. The decay rate for DR is set at .1 unit per unit time so that if not satisfied, the agent will need to look for information when the threshold goes below 2. Once satisfied, the agent's attention may be focused on the next priority until the information decays below the threshold. Since the information is available at the rate of 2 points per unit time, that is unless the discount rate is below 2, the agent will continue to pay attention to the DJIA, the Dollar index, the Gold Index and the CRB index depending on their information availability, decay rates and thresholds. The important question is does this design maximize attention to important details while minimizing the overall attention to any index. In this case, if we examine 30 time units, in any month. If we assume that 30 seconds is equivalent to 30 days, then the monitoring behavior would adequately monitor the trends and changes while minimizing the attention to details that could be ignored. In this case the attention to the CRB index seems excessively high.

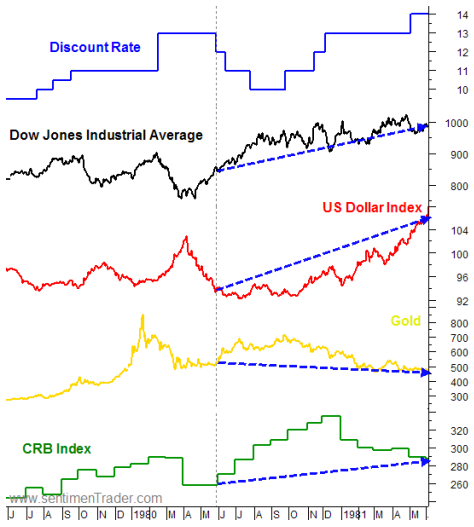

Figure 9. 1980 showed a 4 point decrease in 3 months

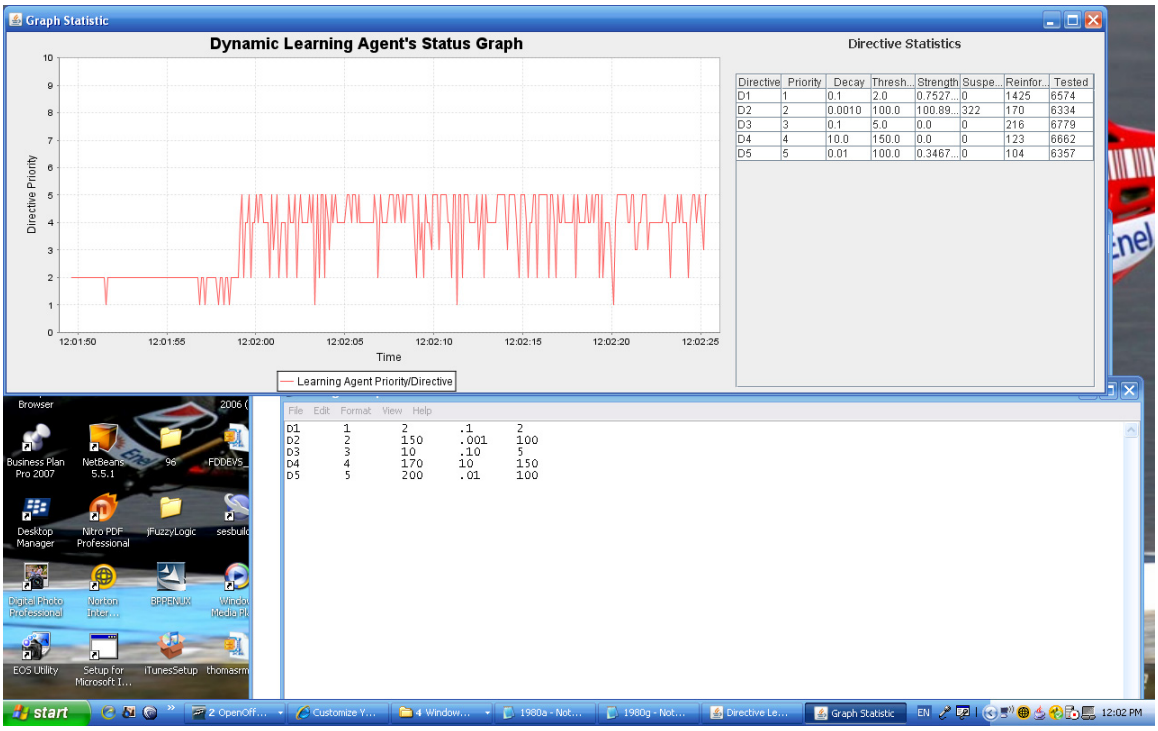

Figure 10. Model Revision for 1980

In this adjustment D3 for for the dollar index the decay rate for the dollar index was reduced from 10 units per unit time to .1 units per unit time and the threshold for D5 for the CRB index was reduced from 160 100. The model is still paying too much attention to the the CRB per unit time.

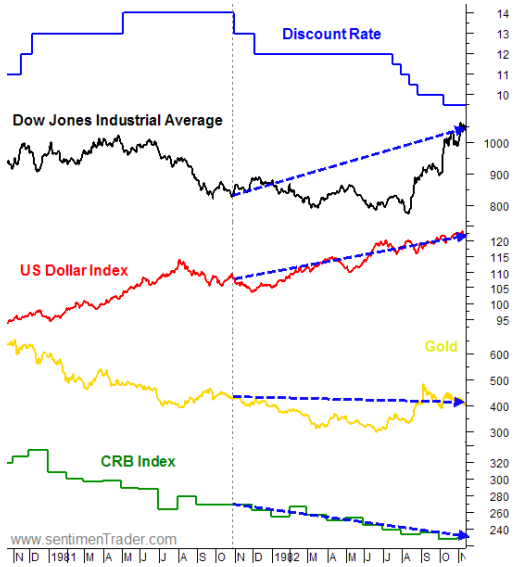

Figure 11. 1981 4.5 % reduction in 6 months

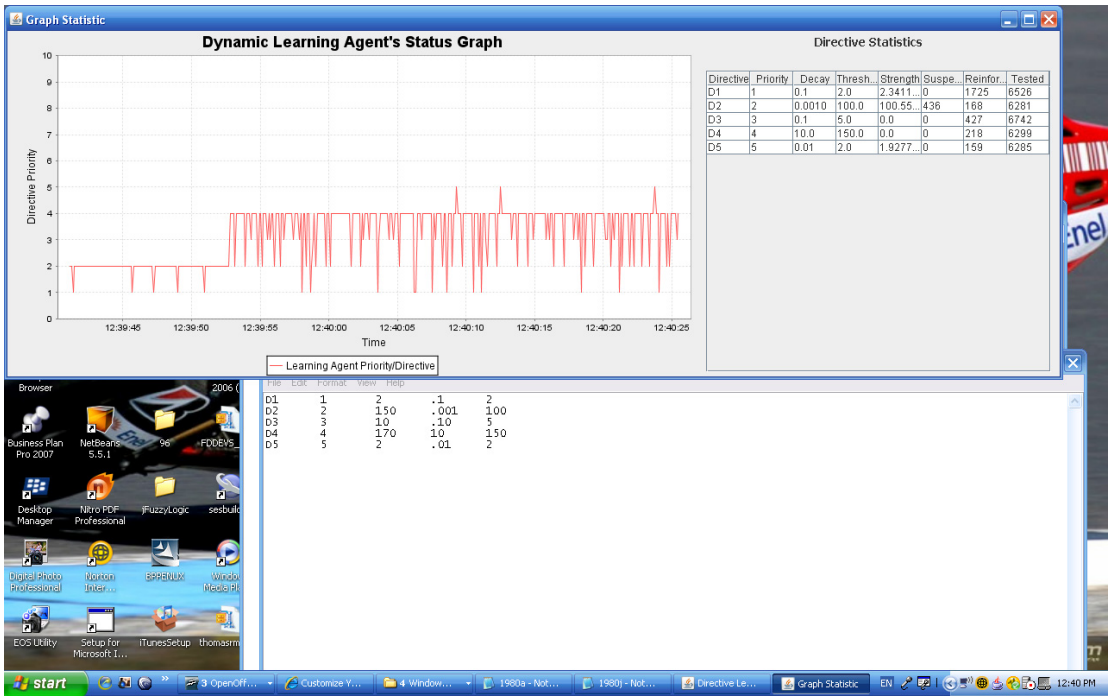

Figure 12.

In figure 12 the decay rate for D5 CRB was reduced from to .01 units per unit time. The model primarily focuses on the DJIA, Dollar Index and Gold Index.

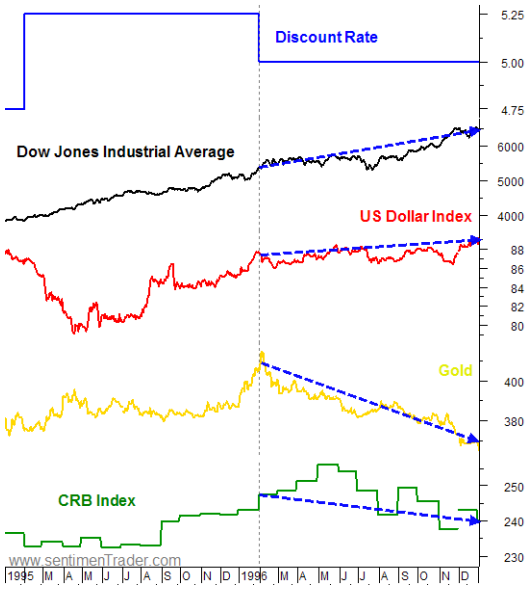

Figure 13. 1996 .25% in 11 months

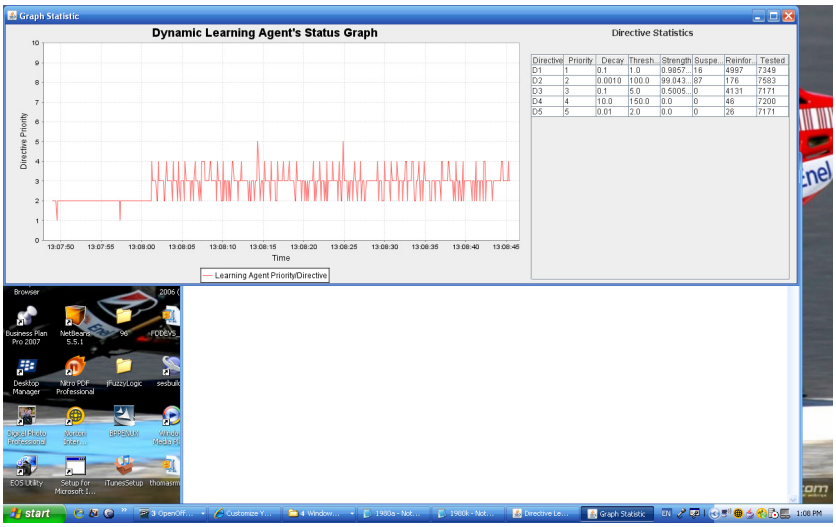

Figure 14.

This version of the model reduced the threshold of D1, the DR to 1 from 2. The model now focuses appropriately on the DJIA, dollar Index, Gold Index. This nearly has the appropriate levels of resolution at the DR through the CRB index, however it may be necessary to return the threshold to 2 for DR or increase the decay rate slightly.

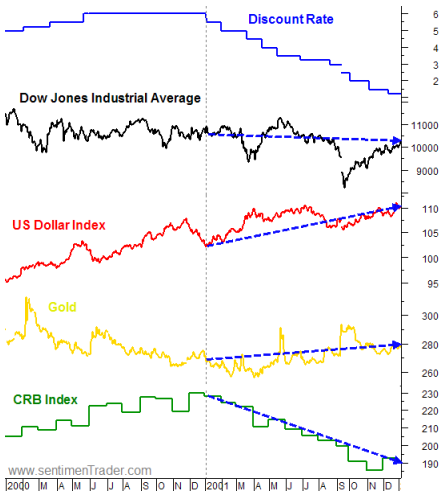

Figure 15. 2001 5% reduction 12 months

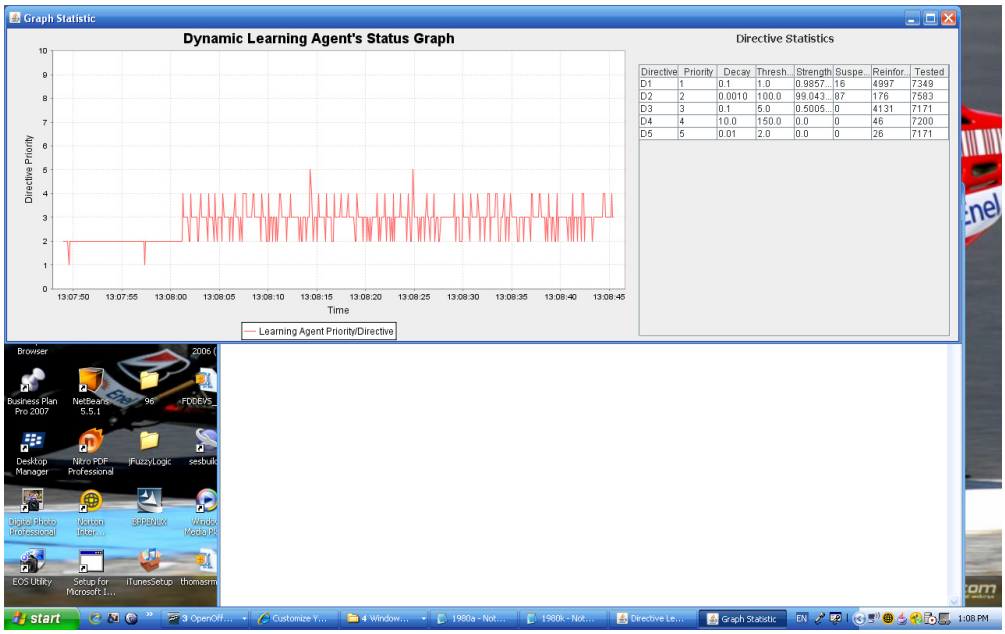

Figure 16. Analysis of the 2001 data.

J2EE Architecture based design and implementation for an Agent based web service.

In order for this capability to be useful to a wider and perhaps commercial audience, a web service approach is advisable. The prototype web service consists of two applications. The first client based service allows the user to choose web sites and then parse web sites for relevant information. An application where a seriesof RSS feeds was originally planned, but the reformatting required proved to be an obstacle. The second service is server based and isolated from the user to prevent misconfiguration and loss of proprietary information regarding the methodology for the attention management/learning agents.
The J2EE architecture is arguably the most popular for the development of high-quality web applications. It popularity and ease of use were the determining factors in choosing J2EE for the development of the Attention Agent service.

Architecture

The J2EE platform uses a distributed multitiered application model for enterprise level applications [4]. The logical architecture is divided into several components according to function, and the various application components that make up a J2EE application are installed on different machines depending on the tier in the multitiered J2EE environment to which the application component belongs. This DirectiveLearning Agent is intended to operate in a three tiered environment. Where the user has an application client that allows the agent to be easily configured, and will interface with the server machine through JSP pages and Enterprise Beans with the database. The server side application which carries the actual agent functionality uses Enterprise Beans to interface with the database where information regarding the historical searches and directives data are held [5].

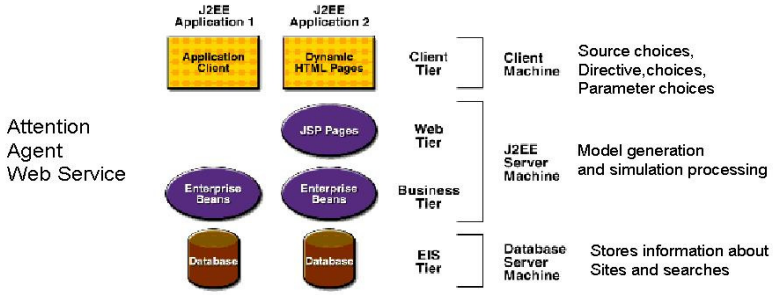

Figure 17. Attention Agent Web Service Architecture

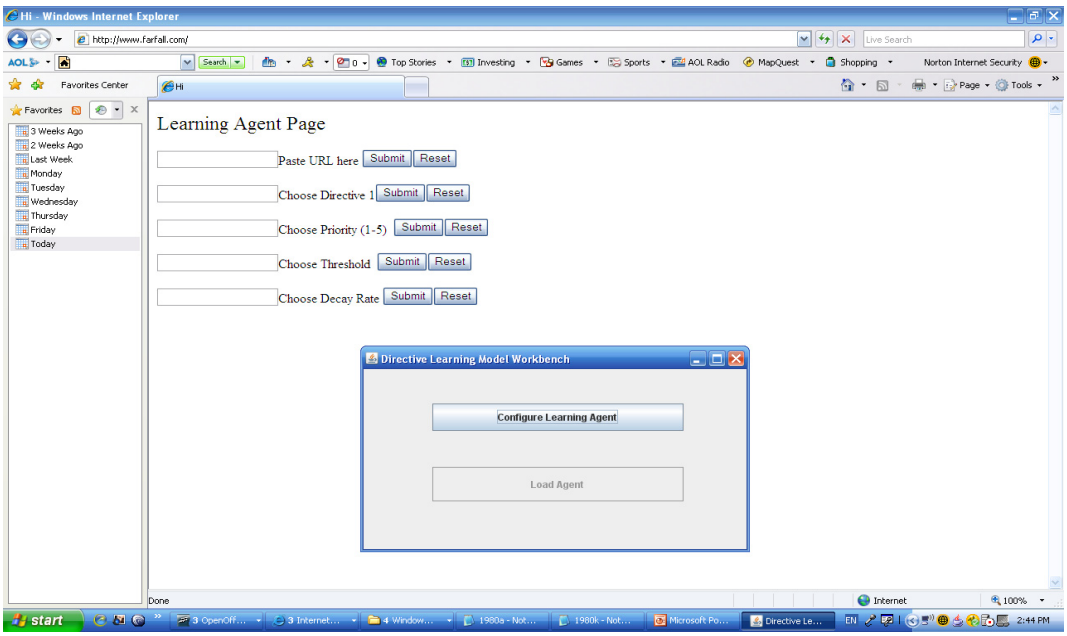

Figure 18. Prototype Attention Agent Web Service Client

Conclusion

The prototype model has shown usefulness in the automated monitoring of user defined hierarchical systems. Although, I have no expertise in the financial model chosen, I was able to examine historical empirical evidence provided by a syndicated financial service to create a model that simulated the attention an expert might pay to a series of financial indices. The simulation experience produced an attention model that would have been difficult to generate without the advice of a subject matter expert. The effort produced a model that attends to 5 indices and is able to provide the user times when he or she should check the index for themselves.

The J2EE architecture allows for commercial deployment of these attention agents, but the ability to combine a series of RSS streams proved to be problematic. The formatting of XML does not take into account the detailed level of tagging that would be required for this application to work as intended so the Parsing tool never worked correctly. The use of an advance schema modeler that employed System Entity Structure data engineering would have been most beneficial [6].

Future Work

The next steps include expanding the input capabilities such that a dynamically changing data can be input in real time. The current model only allows for historical data to be modeled forward. The ability to update thresholds and decay rates in real time should enhance the ability of the model to generate better predictions based on more recent input. The most likely way to develop this capability is a client side initiated stop and re-start method where the period, decay and threshold rates are updated and then the model is allowed to continue with the simulation in the same graph space. The graph should be able to display the new parameters and the point in time in which they were injected.

In order to resolve the RSS problem encountered in this study, an effort is underway to generate RSS feeds that conform to SES guidelines.

Several applications of this methodology should be considered. Surveillance and situation awareness seem to be good candidates. For example a watch center or operations center could use this methodology to capture an expert's attention to system or network monitors so that the screens would display information from various sources at the rate at which an expert might pay attention to them. Although this does not guarantee that a non-expert observer would respond appropriately, it may act as a training activity or as an automated expert capability if many systems are involved.

References

[1] B.P. Zeigler, H. Praehofer, T.G. Kim, “Theory of Modeling and Simulation,” Academic Press, 2000.

[2] E. Mak, S. Mittal, M. Hwang, J. Nutaro. Automated Link 16 Testing using DEVS and XML. Paper submitted to ITEA 2007.

[3] S. Mittal, “Cognitive Learning Agent using DEVS”, Correspondence with the Author. 2007.

[4] P. J. Deitel , H. M. Deitel, “Java: How to Program”, 7th Ed. , Prentice Hall, 2007

[5] http://java.sun.com/j2ee/1.4/docs/tutorial/doc/

[6] B.P.Zeigler, P. E. Hammonds, "Modeling & Simulation-Based Data Engineering: Introducing Pragmatics into Ontologies for Net-Centric Information Exchange”, Elsevier, 2007.

[7] Financial Graphs from www.sentimentrader.com