

Loma Linda University

TheScholarsRepository@LLU: Digital Archive of Research, Scholarship & Creative Works

Loma Linda University Electronic Theses, Dissertations & Projects

6-2004

Development of Web Tools for NLX Simulation Software

Daniel Chai Siriphongs

Follow this and additional works at: <https://scholarsrepository.llu.edu/etd>



Part of the [Biology Commons](#), and the [Software Engineering Commons](#)

Recommended Citation

Siriphongs, Daniel Chai, "Development of Web Tools for NLX Simulation Software" (2004). *Loma Linda University Electronic Theses, Dissertations & Projects*. 779.

<https://scholarsrepository.llu.edu/etd/779>

This Thesis is brought to you for free and open access by TheScholarsRepository@LLU: Digital Archive of Research, Scholarship & Creative Works. It has been accepted for inclusion in Loma Linda University Electronic Theses, Dissertations & Projects by an authorized administrator of TheScholarsRepository@LLU: Digital Archive of Research, Scholarship & Creative Works. For more information, please contact scholarsrepository@llu.edu.

LOMA LINDA UNIVERSITY
Graduate School

Development of Web Tools for NLX Simulation Software

by

Daniel Chai Siriphongs

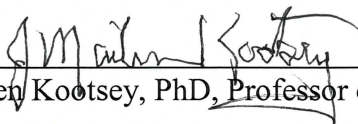
A Thesis submitted in partial satisfaction of
the requirements for the degree of
Master of Science in Biology

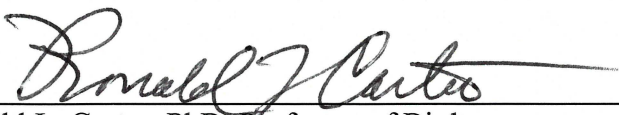
June 2004

© 2004

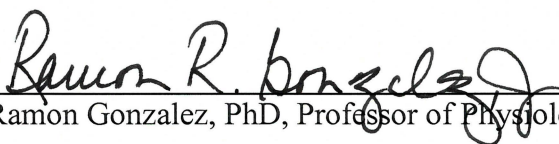
Daniel Chai Siriphongs
All Rights Reserved

Each person whose signature appears below certifies that this thesis in his opinion is adequate, in scope and quality, as a thesis for the degree of Master of Science.


_____, Chairperson
J. Mailen Kootsey, PhD, Professor of Physiology and Pharmacology



Ronald L. Carter, PhD, Professor of Biology



Ramon Gonzalez, PhD, Professor of Physiology and Pharmacology

ACKNOWLEDGEMENTS

I would like to express my appreciation to the individuals who helped me complete this project. I am grateful to the Department of Natural Sciences for have the vision to support this project. I am thankful for the advice of my guidance committee members – J. Mailen Kootsey, Ronald L. Carter, and Ramon Gonzalez. I would also like to thank J. Mailen Kootsey and Grant McAuley for their guidance and assistance.

CONTENTS

Approval Page.....	iii
Acknowledgements.....	iv
Table of Contents.....	v
List of Figures.....	vii
List of Tables.....	ix
List of Abbreviations.....	x
Abstract.....	xi
Chapter	
1. Introduction.....	1
Biological and Mathematical Simulation.....	1
The Benefit of Mathematical Modeling and Simulation.....	1
The Educational Benefit of Mathematical Simulations.....	2
The Need for Simulation Software to be Modular and Usable.....	2
Web Technologies – HTML and Java.....	3
Number Linx (NLX).....	4
Project Goals.....	4
2. NLX Graphical User Interface.....	6
Development Software and Hardware.....	6
Building the NLX Graphical User Interface.....	6
3. Models.....	15
Building a Model Template.....	15
Choosing Models.....	16
Hardy-Weinberg Selection Model.....	16
Sweating Model.....	18
Arms Race Model.....	20
4. Discussion.....	24
References.....	26
Appendices	
A. Object UI – JavaScript Sample File Code.....	27
B. Property Inspector Sample File Code.....	29
C. Graph Floating Panel Sample File Code.....	31

D. NLX Code for Insertbar.xml.....	46
E. NLX Code for Menus.xml.....	47
F. Instructions for User Study.....	48

FIGURES

Figure	Page
1. Overall NLX Project Plan.....	5
2. NLX Graphical User Interface Development Workflow.....	7
3. Screenshot of an Object Form.....	7
4. Sample HTML Code for an Object User Interface.....	8
5. Screenshot of Object User Interface Form.....	9
6. Screenshot of Form Validation Check for Required Fields.....	9
7. Screenshot of Error Message When Not All the Required Fields Have Values.....	9
8. Screenshot of Applet Code for Insertion into the User HTML Page.....	10
9. Screenshot of Comment for Dreamweaver to Identify the Property Inspector.....	11
10. Screenshot of Hard-coded Identifier for Matching to the APPLET tag.....	11
11. Screenshot of Graph Floating Panel Load Button.....	12
12. Screenshot of Graph Floating Panel Save Button.....	12
13. Screenshot of Code Added to Insertbar.xml File.....	12
14. Screenshot of Code for the Button Image and the Location of the Associated Object UI File.....	12
15. Screenshot of Object Buttons within Dreamweaver.....	13
16. Screenshot of Menus.xml File.....	14
17. Screenshot of Graph Editor Option on the Command Bar.....	14
18. Screenshot of Hidden Form for Variable and Parameter Names in the Model Template.....	15
19. Parameter/Variable Selection List.....	16
20. Reload Defaults Button.....	16

21. Screenshot of Model and Equation Solver Parameters in the Model Template	16
22. Hardy-Weinberg Selection Equations.....	17
23. Screenshot of Hardy-Weinberg Selection Simulation Page	18
24. Sweating Model Equations	19
25. Screenshot of Sweating Model	20
26. Arms Race Model Equations	21
27. Screenshot of Arms Race Model	23

TABLES

Table	Page
1. Hardy-Weinberg Selection Model Variable Definitions	17
2. Sweating Model Variable Definitions	19
3. Arms Race Model Variable Definitions	22

ABBREVIATIONS

GUI	Graphical User Interface
HTML	Hypertext Markup Language
NLX	NumberLinx
UI	User Interface

ABSTRACT OF THE THESIS

Development of Web Tools for NLX Simulation Software

by

Daniel Chai Siriphongs

Master of Science, Graduate Program in Biology

Loma Linda University, June 2004

Dr. J. Mailen Kootsey, Chairperson

Valuable mathematical equations have developed within biology and other sciences (physics, chemistry, etc.) that model various processes, but the complexity of the equations are not easily understood by students, scientists, and those with limited knowledge of the subject area. Simulation software used to visually explain a model is normally proprietary to the specific model equations and cannot be easily adapted to new models by non-programmers. Simulation software needs to be modular and based upon web technologies so that the software can be run on multiple platforms. The NLX simulation software is a group of Java-based objects that can be combined with any model equations to create interactive HTML simulation pages rapidly. No simple interface exists to build simulation pages with the software. The goal of the project is to create a tool by which biologists or any scientist, instructor, or general user, could convert an abstract mathematical model into a visually comprehensible and interactive simulation using NLX. The project succeeds in providing a graphical user interface for the NLX software that opens the technology to users that have a limited coding background, as well as providing users with assistance in understanding the behavior of complex mathematical models.

CHAPTER ONE

INTRODUCTION

Biology and Mathematical Simulation

Over the last fifty years, biology has developed from a primarily observational science of classification and description into a full branch of science, involving complex theories and mathematical models (Brown, 1993). Biology has the potential to be the most mathematical of the sciences, due to the fact that living systems involve a complex interaction of chemical and physical properties that can be mathematically described (Spain, 1982b). Unfortunately, most biologists do not have the computer programming background to create their own simulations to test these models (Hannon, 1997). Some biological simulation software exists, but often computer programmers, who have teamed up with biologists to simulate specific models, create this software. There is great value in providing a tool that biologists with limited coding background could pick up and create their simulations from any model they choose.

The Benefit of Mathematical Modeling and Simulation

Mathematical modeling has developed in various sciences in an attempt to quantify and predict behaviors and processes of various systems – from the law of gravity to the cellular sodium-potassium pump. Due to the more concise and precise language of the models, they can be used to calculate and predict system behavior (Maxim, 2002). This precision creates a valuable niche for mathematical modeling within the sciences. Yet often, these mathematical models are too complex for other scientists to understand, since they may not have the necessary mathematical background. Therefore, there need to be ways to create interactive model simulations that would allow a user (whether they

are scientist or student) to visually manipulate the parameters of the models. Through this activity, the behavior of the equations would become real to the user.

The Educational Benefit of Mathematical Simulations

As budget cuts permeate academia, from the elementary school level up to the universities, software simulation is beginning to supplement student classroom teaching and laboratory experience. Only a few years ago, professors did not have the simulation tools to offer hard-science laboratories that could mimic the student laboratory experience. But slowly, some academic institutions have begun projects to create virtual laboratories based upon software simulation (Carnevale, 2002). Certainly virtual laboratories provide a cost-efficient way to introduce a large number of students to complex concepts, but another benefit arises – the virtual laboratories allow students to experiment on their own rather than following stringent recipes. Hence, students and other software users become involved in the discovery process (Hurwitz, 2002) and learn to manipulate real systems to understand cause and effect (Hannon, 1993). Classroom benefits arise as well from the use of simulations in lectures. Software simulation and virtual laboratories and classrooms are not practical for some areas of science and could never replace actual laboratory experience, but science can certainly benefit from the simulations as a valuable educational tool.

The Need for Simulation Software to be Modular and Usable

A glaring problem has arisen as various institutions begin to meet their own software simulation needs. The custom software for one project does not function well beyond the scope of the original project, nor does it facilitate a simple way of changing the software to meet the needs of a new project. Many of these software projects are not

commercially viable, and consequently, the underlying code is unstructured and difficult to modify or scale (Hurwitz, 2002).

To remedy this situation, simulation software needs to be modular and developed using web technologies. The software should be in small components that would provide a specific function and be self-contained. The modularity and customization would come from the ability to combine these components in any configuration. By basing the components on web technology and languages (e.g., Java, HTML, etc.), distribution and use of the components will be easier, since web technology is available at all academic institutions and in most homes.

Finally, by creating a tool that is easy to use – even for non-programmers – users will be able to easily create simulations. Instructors will be able to quickly build numerous model pages for teaching. Students will be able to easily build their own model simulations and vary the parameters to their desires.

Web Technologies – HTML and Java

Hypertext markup language (HTML) is a language used to create standard web pages that can be viewed in any web browser (such as Internet Explorer, Netscape, etc.). The language allows a developer to write text and insert images and animations. The HTML files are placed on a web server that will host the website. The website can be made available to the entire Internet, thereby providing a global distribution method for the spread of ideas.

Java is a programming language designed with the mentality of “write once, run anywhere”. This phrase means that a program written in Java can be run on any type of

platform that supports Java. This widespread interoperability of Java makes it a desirable choice for a project requiring a broad audience across multiple platforms.

Number Linx (NLX)

Prior to this project, NLX, created by Dr. J. Mailen Kootsey, provided users with the ability to create model simulations by defining their models and then creating a HTML simulation page using various Java-based objects (Kootsey, 2003). Currently, seventeen objects are available and are broken down into five main categories—input (changing parameters and values), control (controlling the model), output (displaying the results), model (equations for running the model), and equation solver (numerical processing method). Users can reuse the objects in different simulations—except for the model object, which is specific to each model. Unfortunately, the software’s user base is limited to those with knowledge of Java programming and HTML page design and coding. Users have to either type new code themselves, or they must “cut-and-paste” – essentially copy – existing code and modify the code to their specific models. The NLX software has the recommended modularity and customization, but it does not have the ease of use and assembly necessary for a wide audience.

Project Goals

The primary goal of the project is to create a tool by which biologists or any scientist, instructor, or general user, could convert an abstract mathematical model into a visually comprehensible and interactive simulation. The project is Phase II of an overall plan to make NLX available to non-programmers (Figure 1).

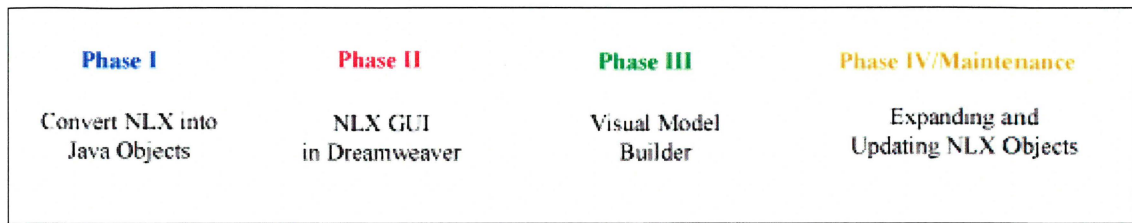


Figure 1. Overall NLX project plan

Users of all skill levels would have a method of building the simulation HTML pages and importing the various NLX components (objects). The users will build a NLX simulation page just by clicking on various icons and filling out short forms. Each object within the page will have an associated block of pre-formed HTML code that will be combined with the specific parameter values typed into the form by the user. The tool will be able to output an HTML page with all the various objects along with their respective parameters and values.

The tool will be created as an extension to Macromedia Dreamweaver 2004. Dreamweaver is a visual web page editor that allows a user to insert and edit HTML pages without having to know HTML code. Extending Dreamweaver rather than creating a builder tool from scratch is the most parsimonious solution, since it allows the user to edit the resulting HTML page within Dreamweaver, thus increasing usability while reducing complexity and the number of steps to a finished page. The extension tool also allows the editing of previously created model simulation pages and allows the user to edit the objects' parameters and export the HTML page without damaging the layout of the page.

CHAPTER TWO

NLX GRAPHICAL USER INTERFACE

Development Software and Hardware

- Dreamweaver MX 2004 (Macromedia Inc., 600 Townsend Street, San Francisco, CA, 94103 – <http://www.macromedia.com>)
- Xinox JCreator IDE (Integrated Development Environment) (Xinox Software, Oostplantsoen 115, 2611 WL Delft, Netherlands – <http://www.jcreator.com>)
- NLX Model Builder (J.M Kootsey and G. McAuley, LLU Biomedical Simulation Lab, Loma Linda University, Loma Linda, CA, 92350)
- Adobe Photoshop 7.0 (Adobe Systems Inc., 345 Park Avenue, San Jose, CA 95110 – <http://www.adobe.com>)
- Java Software Development Kit (Version 1.4.1_02) (Sun Microsystems Inc., 4150 Network Circle, Santa Clara, CA 95054 – <http://java.sun.com>)
- Personal Computer

Building the NLX Graphical User Interface

The software design phase commenced with learning how to create the extension to Macromedia Dreamweaver MX 2004. In order to create this extension, three necessary components needed to be created. First, the initial object user interface (UI) must be created to allow a user to input the desired parameter values into the NLX object. Once the object has been inserted into the simulation web page, a property inspector must be created that will allow the user to easily edit the attributes of the NLX object without having to revert back to the underlying code. Finally, a visual representation of the objects (e.g., icons) must be created and inserted into Dreamweaver's configuration files

so that the software program will recognize the new extension and provide the user with access to the NLX objects. Figure 2 is a diagram of the workflow for the development process.

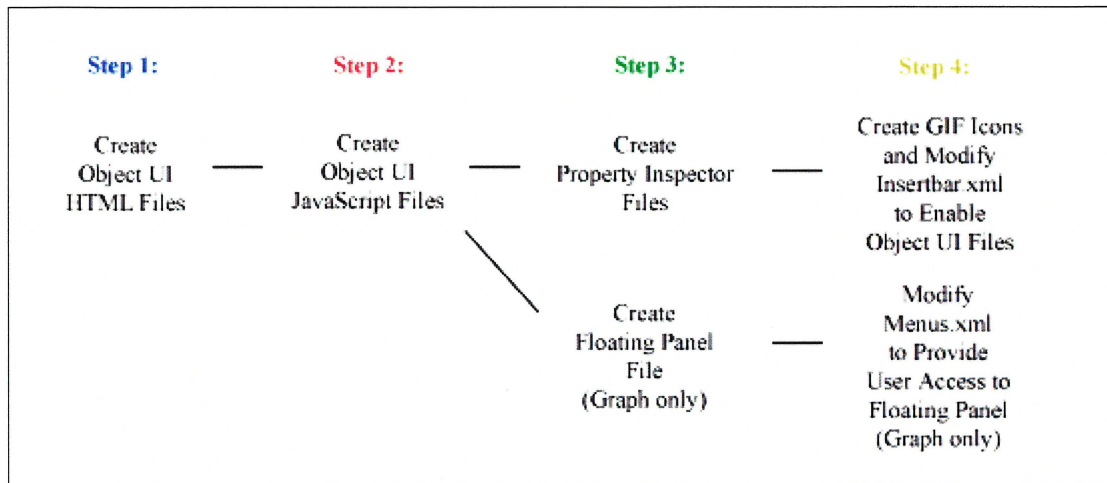


Figure 2. NLX Graphical User Interface Development Workflow

The object user interface is broken down into two files. The first file is the actual HTML form file that is displayed when a user clicks on the icons. Figure 3 is a screenshot of the visual representation of the form during development. Figure 4 is the sample code for this particular file.

Log Points: *

** Required Fields*

Figure 3. Screenshot of an object form

```

1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
2 <html>
3 <head>
4 <!-- Copyright 2003 Loma Linda University. All rights reserved. -->
5 <title>One-Shot Control</title>
6 <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
7 <script language="javascript" src="OneShotControl.js"></script>
8 </head>
9 <body>
10 <form name="theform">
11   <table width="300">
12     <tr>
13       <td align="right" valign="bottom" nowrap>Log Points:</td>
14       <td valign="middle">
15         <input name="logpoints" type="text" value="1000">
16         *</td>
17     </tr>
18     <tr>
19       <td colspan="2" align="center">
20         <b><i>* Required Fields</i></b></td>
21     </tr>
22   </table>
23 </form>
24 </body>
25 </html>

```

Figure 4. Sample HTML code for an object user interface

The HTML file contains the fields that will be populated by the user. In Figure 5, this particular example only has one form field for log points, but the number of fields displayed for an object is only limited by the size of the window and the number of fields one could fit in that finite window. The <title> tag (line 5, Figure 4) will display whatever name is assigned to object in the titlebar of the window, as shown in Figure 5.

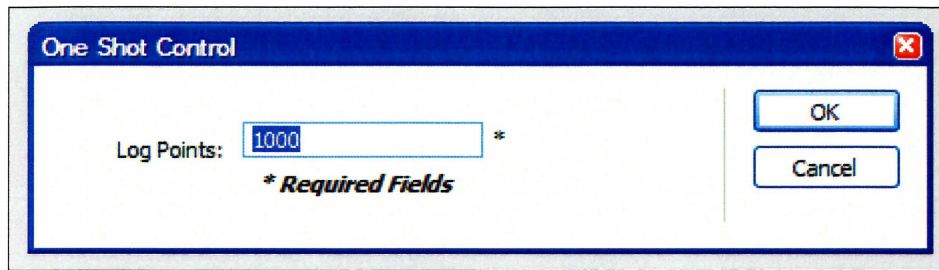


Figure 5. Screenshot of object user interface form

The final important addition to the HTML user interface file is a reference importing the associated JavaScript file that will process the data inserted into the form (line 7, Figure 4).

The JavaScript file (full sample code in Appendix A) is the other half of the object user interface. Initially, the scripting checks the form to make sure that all the required fields (denoted by an asterisk *) have some type of value (line 9, Figure 6).

```
8 function insertObject() {  
9     var valid = checkForm(document.theform); // Check the form  
10    var theDOM = dw.getDocumentDOM(); // Used to get the inserted
```

Figure 6. Screenshot of form validation check for required fields



Figure 7. Screenshot of error message when not all the required fields have values

If any required fields are missing values, an error message is displayed (Figure 7) and the form processing stops, thereby returning the focus back to the form so that the user can rectify the problem. If all the required fields have values, the processing of the form continues. Each form field value is identified by its name and its value is stored in a

- the JavaScript for reading user inputted values and changing the code is embedded in the HTML file versus being a separate script file
- a special HTML comment is at the top of the file (Figure 9) that identifies the file as being applicable to APPLETS tags
- the NLX object is identified by matching the code value of the APPLETS tag to the value hard-coded into the property inspector file (Figure 10)

```
1 <!-- tag:APPLET,priority:1,selection:within -->
```

Figure 9. Screenshot of comment for Dreamweaver to identify the property inspector

```
theObj.getAttribute("code") == "nlx.control.runcontrol.CommandRunControl.class");
```

Figure 10. Screenshot of hard-coded identifier for matching to the APPLETS tag

The property inspector files are stored in the Inspectors subdirectory within the Configurations directory of the Dreamweaver file structure. A version of this file was created for sixteen out of the seventeen NLX objects.

One limitation of the property inspector is the size of the window within Dreamweaver. Due to the size limitation, the window can only hold approximately fourteen field objects and labels. Therefore, the Graph object required a special inspector. Instead of using the standard property inspector, a floating panel was created.

Floating panels in Dreamweaver are a feature that allows a user to access a large window that can be resized or locked into locations around the workspace. The floating panel for the Graph object (full sample code in Appendix C) is similar to a property inspector for the other NLX objects. Two exceptions with the floating panel are that once the object is selected, it has to be loaded into the floating panel (Figure 11), and after

making changes, the new values have to be saved manually (Figure 12) – unlike the property inspector that automatically updates the HTML code with any changes.

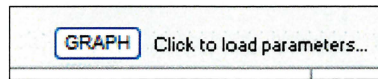


Figure 11. Screenshot of Graph floating panel load button

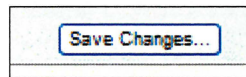


Figure 12. Screenshot of Graph floating panel save button

Once all the various components for each NLX object were created, the configuration files for Dreamweaver were appended so that Dreamweaver could recognize the new NLX extension. In order for the object user interface files to be recognized, a section of code (full sample code in Appendix D) was added to the insertbar.xml file located in the Configurations/Objects directory. First, an NLX category is created (line 411, Figure 13), and then buttons for each NLX object are created (line 412-414, Figure 13). Each button is assigned a specific GIF image representing that NLX object, as well as the location of the object's associated HTML file (Figure 14).

```
411     <category id="DW_Insertbar_NLX" name="NLX" folder="NLX">
412         <button id="DW_DefaultRunControl" name="Default Run C
413         <button id="DW_ContinuousRunControl" name="Continuous
414         <button id="DW_OneShotControl" name="One Shot Control
```

Figure 13. Screenshot of code added to insertbar.xml file

```
image="NLX\OneShotControl.gif" file="NLX\OneShotControl.htm"/>
```

Figure 14. Screenshot of code for the button image and the location of the associated object UI file

Within Dreamweaver, the objects are divided into four color schemes grouping like objects together (Figure 15):

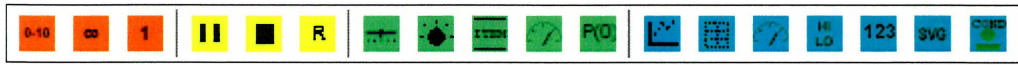


Figure 15. Screenshot of object buttons within Dreamweaver

- Run Control objects are orange
 - Default Run Control
 - Continuous Run Control
 - One Shot Run Control
- Control Panel objects are yellow
 - Default Panel
 - Simple Panel
 - Reset Panel
- Input objects are blue
 - Slider
 - Knob
 - Item Selector
 - Dial Input
 - Parameter Initializer
- Output objects are green
 - Graph
 - Legend
 - Dial Output
 - Conditional Text

- Numeric Output
- SVG Animation
- Conditional Graphic

For the Graph floating panel, a separate command had to be included in Dreamweaver in order for a user to access it. Two lines of code (Figure 16, full code in Appendix E) were appended to the menus.xml file located in the Configuration/Menu directory. This code added an NLX and a Graph Editor option to the command bar at the top of the Dreamweaver workspace (Figure 17), which allows the user to access the panel.

```
<menu name="_NLX" id="DWMenu_NLX">  
  <menuitem name="Graph Editor" enabled="true">
```

Figure 16. Screenshot of menus.xml file

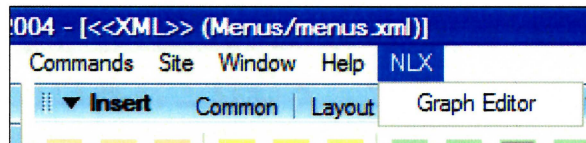


Figure 17. Screenshot of Graph Editor option on the command bar

CHAPTER THREE

MODELS

Building a Model Template

A model template page is required when creating a model simulation page using the NLX graphical user interface. The model template contains the standard code that is common to all NLX simulation pages. There are only two areas within the code that must be edited by the user. The first area contains two lines of code that are in a hidden form in the header of the file (Figure 18). These two lines provide the names of the variables and parameters unique to the specific model in use. The inclusion of the hidden form is part of a key usability enhancement where the developer of the simulation page can have easy access to the model variable and parameter names without having to remember their exact spellings. The names are displayed in a list in the initial object user interfaces (Figure 19), the property inspectors, and in the Graph floating panel. If a user is developing multiple simulation pages for different models, a reload default button (Figure 20) was added to the object user interface forms, since Dreamweaver cannot automatically reload the list values. The tool will also recognize if the chosen option is a parameter or a variable so that the name of the PARAM tag entered into the simulation page is correct.

```
<form name="defaultnames">
  <input type="hidden" name="vars" value="VARIABLE NAMES (INI
  <input type="hidden" name="pars" value="PARAMETER NAMES, SE
</form>
```

Figure 18. Screenshot of hidden form for variable and parameter names in the model template

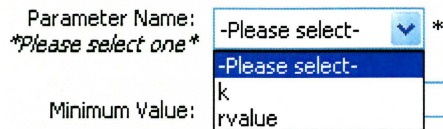


Figure 19. Parameter/Variable selection list

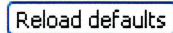


Figure 20. Reload defaults button

The other section of required coding is within the standard NLX Starter applet (Figure 21). The user must denote the specific model and equation solver classes being used. When a user creates a new model simulation page, it must be based upon the model template file with the changes necessary to specialize the file for the specific model being developed.

```
<PARAM NAME = "modelclass" VALUE = "nlx.model.SUBPACKAGE(S).MODEL_CLASS GOES
<PARAM NAME = "solverclasses" VALUE = "nlx.solver.SUBPACKAGE(S).SOLVER_CLASS
```

Figure 21. Screenshot of model and equation solver parameters in the model template

Choosing Models

Three biological models – Hardy-Weinberg selection, sweating, and arms race – were chosen to provide a test of the ease of creating simulation pages. These particular models cover areas of biology where mathematical models are prevalent. The models were also chosen based upon their likely use for educational purposes. The models are being presented in their order of mathematical complexity, with the Hardy-Weinberg model being the most simple to Arms Race being the most complex.

Hardy-Weinberg Selection Model

The classic Hardy-Weinberg Selection genetics model deals with the selection of the dominant or recessive allele within a population over multiple generations (Spain,

1982a). The model is an essential foundation block for an understanding of Mendelian genetics and is commonly taught in biology courses in high school and universities; hence, it is a model likely to be simulated for educational purposes. The linear algebraic model equations are:

<i>Equation 1:</i>	$r = (.5T + k*S)/(k*S + T + U)$
<i>Equation 2:</i>	$S = r^2$
<i>Equation 3:</i>	$T = 2r(1 - r)$
<i>Equation 4:</i>	$U = (1 - r)^2$

Figure 22. Hardy-Weinberg Selection equations

Table 1. Hardy-Weinberg Selection Model Variable Definitions

Variable	Definition
<i>r</i>	Probability of one parent donating a gene to an offspring
<i>k</i>	Fitness of the homozygous recessive in relation to the other genotypes
<i>S</i>	Homozygous recessive genotype frequency
<i>T</i>	Heterozygous genotype frequency
<i>U</i>	Homozygous dominant genotype frequency

Prior to building the page, the model template was changed to recognize four variables (*time*, *S*, *U*, and *T*) and two parameters (*k*, *r-value*). The Starter applet was modified to point to the “HardyWeinbergSelectionModel” Java class. The equation solver used was “RK4Solver” for linear algebraic equations. The simulation page contained a simple dial input to adjust the *k* value that selects for an allele, three numeric outputs to show the genotype frequency, and a graph to show the results of the selection (Figure 23).

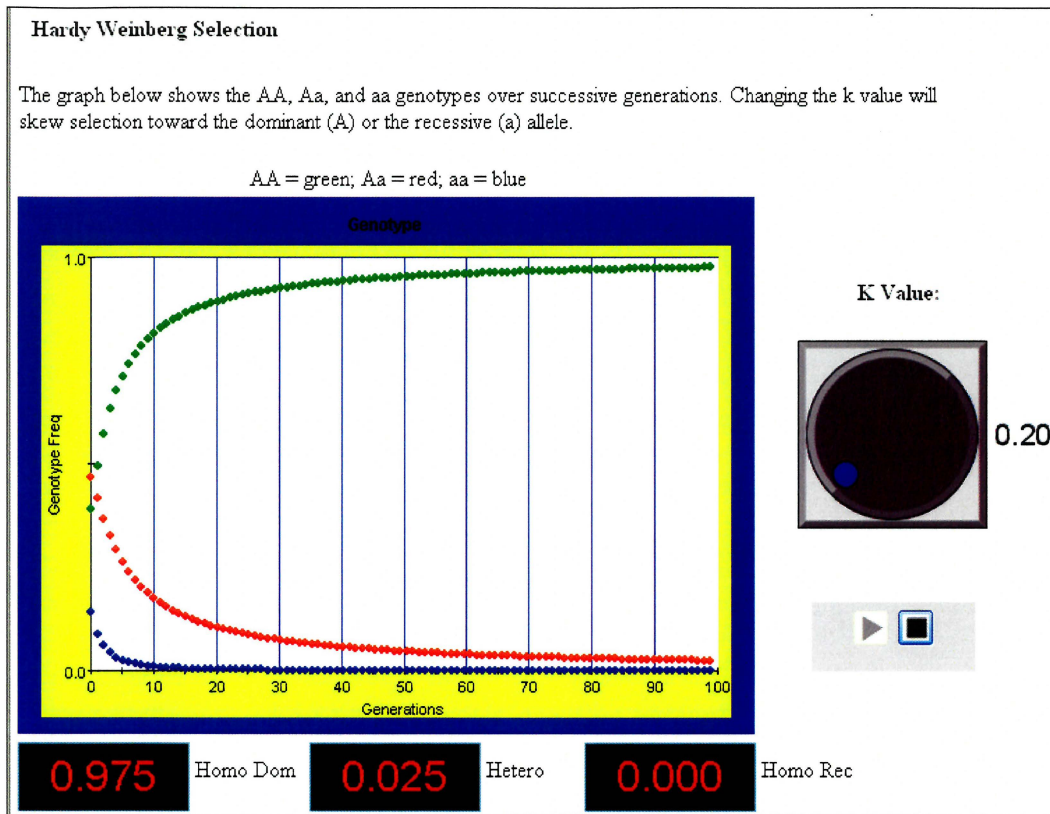


Figure 23. Screenshot of Hardy-Weinberg Selection simulation page

Sweating Model

The simple sweating model deals with thermoregulation of the human body via perspiration in warm to fatally hot temperatures (Spain, 1982c). The sweating model combines complex thermodynamic concepts of chemistry and physiology, yet it is a process common to many organisms and intimately familiar to all humans. The linear algebraic model equations were:

Equation 1:	$\Delta Q_i = 75(Q_{10}^{(T_B - 37)/10}) \Delta t$
Equation 2:	$\Delta Q_r = k(T_B - T_A) \Delta t$
Equation 3:	$\Delta Q_e = (E_{max} * T_d(1 - R_H) \Delta t) / K + T_d$
Equation 4:	$\Delta Q_B = \Delta Q_i - \Delta Q_r - \Delta Q_e$
Equation 5:	$Q_B \leftarrow Q_B + \Delta Q_i - \Delta Q_r - \Delta Q_e$
Equation 6:	$T_B = Q_B / 70$

Figure 24. Sweating model equations

Table 2. Sweating Model Variable Definitions

Variable	Definition
Q_i	Metabolic heat production
Q_{10}	Factor by which heat production increases for a 10°C rise in temperature
T_B	Body temperature
t	Time
Q_r	Convective and radiative heat loss
T_A	Environmental temperature
k	Heat loss coefficient
Q_e	Evaporative heat loss
E_{max}	Maximum evaporative heat loss
T_d	Number of degrees of body temperature above 37°C
R_H	Relative humidity
K	Half maximum parameter in the hyperbolic function employed
Q_B	Heat content

The model template was changed to recognize seven variables (*time*, Q_i , Q_r , Q_e , Q_B , ΔQ_B , *bodytemp*) and eight parameters (T_B , T_A , k , T_d , E_{max} , R_H , K , Q_{10}). The Starter applet was pointed to the “SweatingModel” Java class. Once again, the equation solver used was “RK4Solver” for linear algebraic equations. The simulation page for the sweating model contained a slider to change the external temperature and a graph to show the results of the equations (Figure 25).

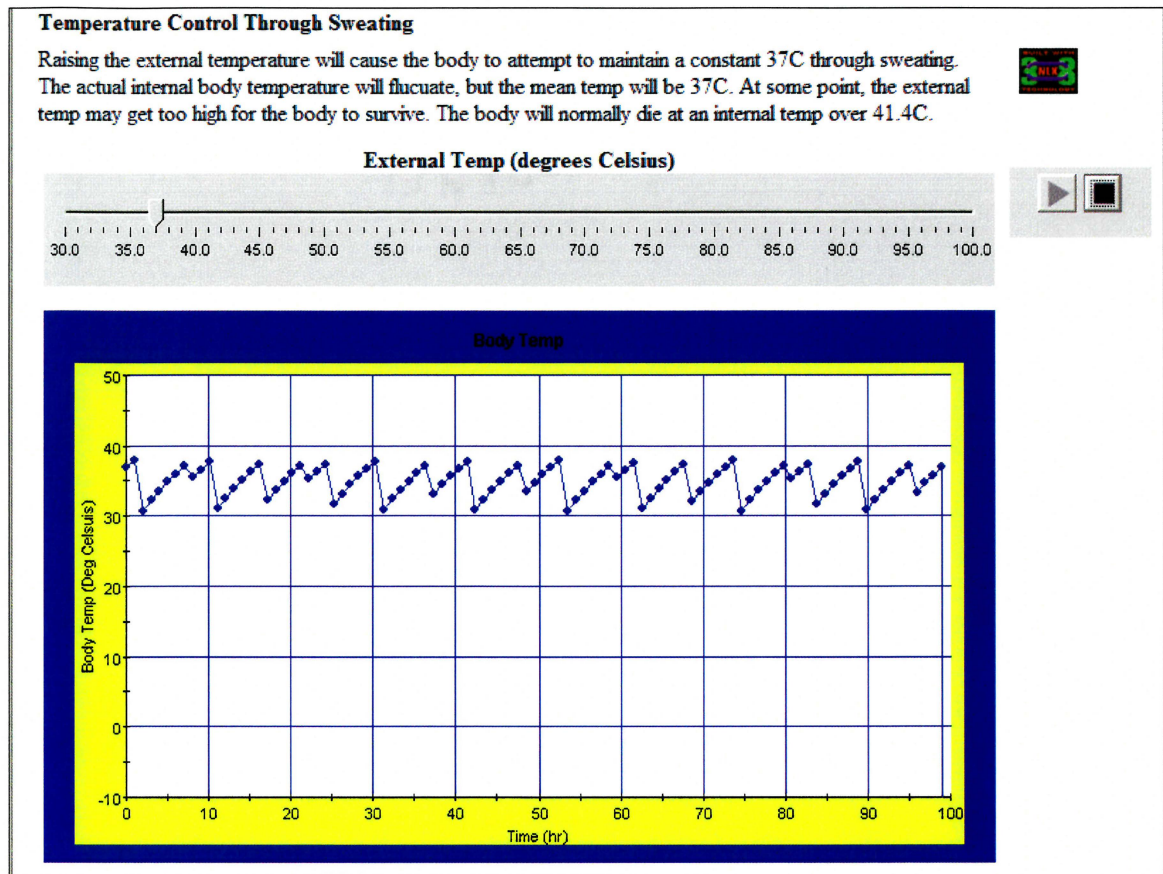


Figure 25. Screenshot of Sweating model

Arms Race Model

The Arms Race model involves three genotypes of a prey species and three genotypes of a predator species (Waltman, 2002). The arms race is created when only the homozygous recessive prey develops a poison that kills the homozygous dominant and heterozygous predator genotypes, while the homozygous recessive predator develops immunity to the prey poison. The complexity of the model appeals to the researcher due to its differential equations and application to previous behavioral and genetics research and personal investigation.

Equation 1:

$$x'_1 = \frac{\alpha}{x} \left(x_1 + \frac{x_2}{2} \right)^2 - \frac{\alpha x_1 x}{K} - \frac{m_1 x_1 y}{a+x},$$

Equation 2:

$$x'_2 = \frac{2\alpha}{x} \left(x_1 + \frac{x_2}{2} \right) \left(x_3 + \frac{x_2}{2} \right) - \frac{\alpha x_2 x}{K} - \frac{m_1 x_2 y}{a+x},$$

Equation 3:

$$x'_3 = \frac{\alpha}{x} \left(x_3 + \frac{x_2}{2} \right)^2 - \frac{\alpha x_3 x}{K} - \frac{m_3 x_3 y}{a+x},$$

Equation 4:

$$y'_1 = \frac{T(x_1, x_2, 0)^2}{(a+x)T(x_1, x_2, x_3)y} \left(y_1 + \frac{y_2}{2} \right)^2 - \frac{m_3 x_3 y_1}{a+x} - s y_1,$$

Equation 5:

$$y'_2 = 2 \frac{T(x_1, x_2, 0)y_1 + \frac{1}{2}T(x_1, x_2, 0)y_2}{T(x_1, x_2, x_3)y} \times \frac{T(x_1, x_2, x_3)y_3 + \frac{1}{2}T(x_1, x_2, 0)y_2}{a+x} - \frac{m_3 x_3 y_2}{a+x} - s y_2,$$

Equation 6:

$$y'_3 = \frac{(T(x_1, x_2, x_3)y_3 + \frac{1}{2}T(x_1, x_2, 0)y_2)^2}{T(x_1, x_2, x_3)(a+x)y} - s y_3.$$

Equation 7:

$$T = (m_1 * x_1 + m_2 * x_2 + m_3 * x_3)$$

Figure 26. Arms Race model equations

Table 3. Arms Race Model Variable Definitions

Variable	Definition
x_1	Prey – Homozygous dominant genotype frequency
x_2	Prey – Heterozygous genotype frequency
x_3	Prey – Homozygous recessive genotype frequency (poisonous)
y_1	Predator – Homozygous dominant genotype frequency
y_2	Predator – Heterozygous genotype frequency
y_3	Predator – Homozygous recessive genotype frequency (poison-resistant)
$\acute{\alpha}$	Rate prey approaches the carrying capacity
m_1	Difficulty of prey capture
m_3	Difficulty of prey capture
T	Calculated function
K	Carrying capacity
a	Biological constant
s	Death rate of the predator in the absence of prey

The model template was changed to recognize nine variables (*time*, *xx*, *yy*, x_1 , x_2 , x_3 , y_1 , y_2 , y_3) and thirteen parameters (*alpha*, *m*, m_3 , *a*, *K*, *T*, *s*, *init_x1*, *init_x2*, *init_x3*, *init_y1*, *init_y2*, *init_y3*). The Starter applet was pointed to the “ArmsRaceModel” Java class. Again, the equation solver used was “RK4Solver” but this time for differential equations. The simulation page contained knob inputs for each parameter and two graphs for showing the prey and predator genotype frequencies (Figure 27). The simulation was setup to run over the course of five thousand generations. If the model is run for less than five thousand generations, an observer would not see the change in genotype frequencies as the poisonous prey and adapted predator do not become prominent if the run is too short.

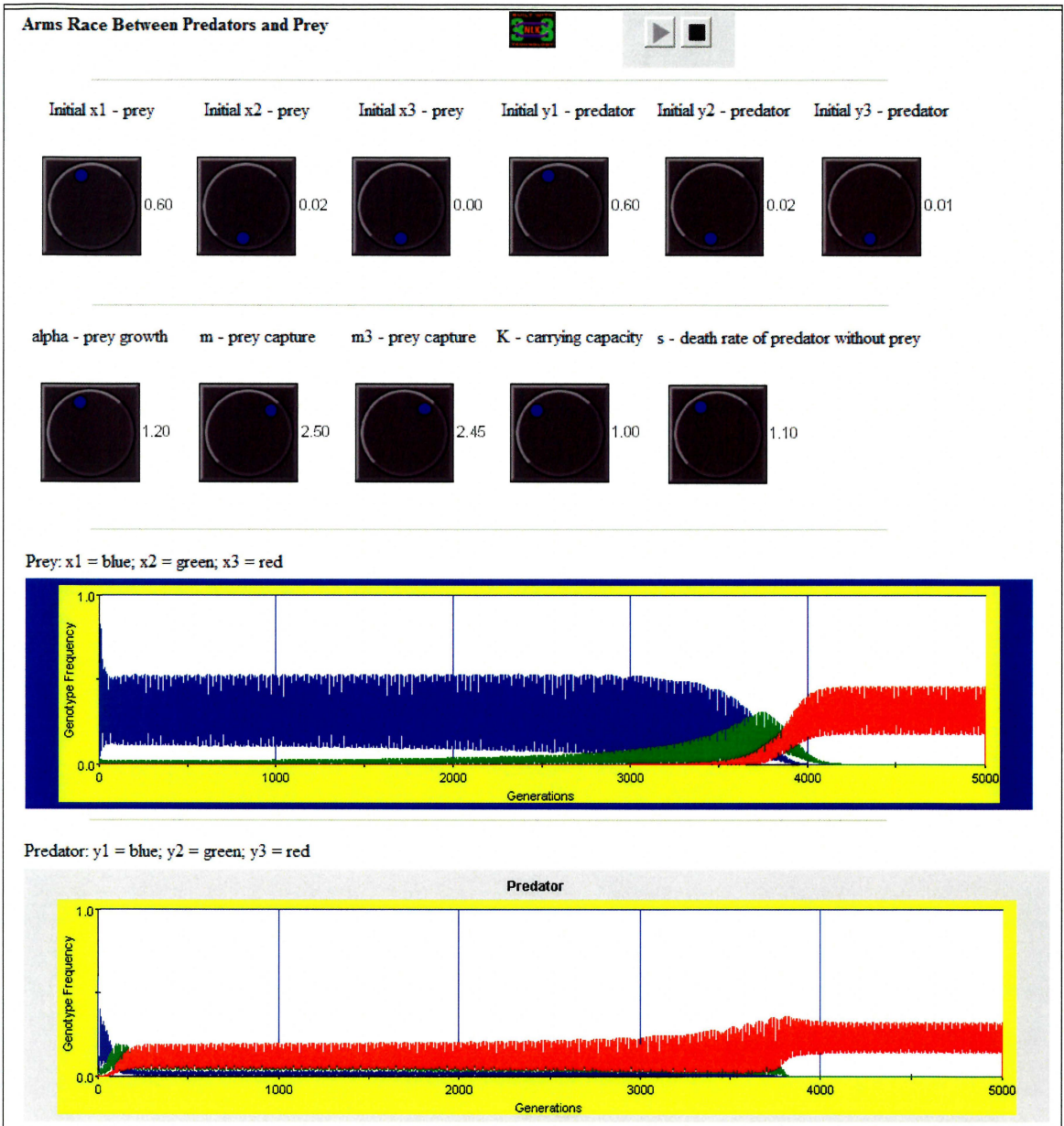


Figure 27. Screenshot of Arms Race model

CHAPTER FOUR

DISCUSSION

The project achieves the goal of creating a tool by which biologists or any scientist, instructor, or general user, could convert an abstract mathematical model into a visually comprehensible and interactive simulation. Through the NLX graphical user interface, any designer can create a model simulation page by simply clicking on buttons to insert objects, instead of the previous method of cutting and pasting code or manually typing the code by hand. These users can now easily bring the complex mathematical equations of models like the Arms Race to a more understandable level.

A small usability study was performed on novice and advanced users of the graphical user interface. The users were given instructions (Appendix F) for inserting specific objects and parameter values into a Hardy-Weinberg template page. The users went through the instructions twice – once by cutting and pasting code and once by using the graphical user interface. The order of the runs was randomized to help reduce the effect of user familiarity with the test. Novice users experienced a three-fold decrease in development time while advanced users experienced a four-fold decrease in development time. The study confirmed the expected improvement in efficiency that most tools provide over having no tool whatsoever.

Developing pages with the NLX graphical user interface leads to a discussion of the design of the simulation page. In general, the user has entire flexibility as to the layout of the simulation page as long as the basic core components are included in the page. To create a functional page, each page must have at least the Starter object (included with the template and modified for the specific model), the hidden form for

variable and parameter names, a run control object, a panel object, and one output object. Beyond these basics, the designer could increase the usability and interactivity of the page by including input objects that allow the users to change parameter values and by including further output objects to help make the page more visually appealing. Text boxes explaining the model and what each parameter is about will also help with the educational value of the page.

The power and significance of the graphical user interface tool and NLX are limited only by the ingenuity of the designer. NLX can have applications well beyond biological sciences and can be used in the physical, financial, behavioral, or any scientific discipline where mathematical models are used. Such versatility is necessary to provide wide acceptance of this tool and simulations in general.

Further research and development will continue through the subsequent phases of the overall NLX project. Phase III will bring the creation of a tool to visually build the Java model file rather than coding the file by hand. Phase IV will achieve further refinement and enhancement of the tool by adding more objects, creating cleaner, easier-to-understand icon graphics, and integrating a help file into the user interface.

REFERENCES

- Brown, D., Rothery, P. "Preface." *Models in Biology: Mathematics, Statistics, and Computing*, pages xvii – xix. Wiley Publishing: New York. 1993.
- Carnevale, Dan. "The Virtual Lab Experiment." *The Chronicle of Higher Education*. January 31, 2002.
- Hannon, Bruce, Ruth Matthias. "Preface." *Modeling Dynamic Biological Systems*, pages ix – xi. Springer Publishing: New York. 1993.
- Hurwitz, Judith S. "Software for Life Science: A Few Requests." *Bio-IT World*. August 13, 2002.
- Kootsey, J.M., McAuley, G. "NLX: Building Web pages with interactive calculations in minutes." *Proceedings 25th Annual Conference IEEE/EMBS*, pages 3505-3508. Cancun, Mexico. 2003.
- Maxim, Bruce R. "Mathematical Modeling and Formal Specification Languages Lecture." University of Massachusetts – Dearborn. 2002.
- Spain, James D. "The Effect of Selection on Random Populations". *BASIC Microcomputer Models in Biology*, pages 124 – 129. Addison Wesley Publishing : London. 1982a.
- Spain, James D. "Introduction: The Role of Computer Modeling and Simulation in Biology." *BASIC Microcomputer Models in Biology*, pages 1 – 2. Addison Wesley Publishing, London. 1982b.
- Spain, James D. "Temperature Control Through Sweating". *BASIC Microcomputer Models in Biology*, pages 209 – 211. Addison Wesley Publishing : London. 1982c.
- Spanar, Megan. "Science Simulation and Virtual Laboratories on the World Wide Web." *Science, Computing, and Instrumentation*. May 1999.
- Waltman, Paul, Braselton, James, and Braselton, Lorraine. "A Mathematical Model of Biological Arms Race with a Dangerous Prey." *Journal of Theoretical Biology*, Volume 218, pages 55 – 70. 2002.

APPENDIX A

Object UI – JavaScript Sample File Code

```
//----- NLX -----
//----- One Shot Control -----

function isDOMRequired() {
    // Return false, indicating that this object is available in
    //code view.
    Return false;
}

function insertObject() {
    // Check the form for required field values
    var valid = checkForm(document.theform);
    // Used to get the insertion point
    var theDOM = dw.getDocumentDOM();
    var rtnStr="";

    if (valid) {
        // Parameter values for the object
        var logpoints = document.forms[0].logpoints.value;

        rtnStr = "<div id='NLXObjectLayer' style='position:absolute;
            width:42px; height:33px; z-index:1'>";
        rtnStr = rtnStr + "<APPLET \n";
        rtnStr = rtnStr + "    ARCHIVE    = 'nlx_c.jar,nlx_s.jar,
            nlx_lx.jar,nlx_jc.jar,nlx_model.jar' \n";
        rtnStr = rtnStr + "    CODE      = 'nlx.control.runcontrol.
            CommandRunControl.class' \n";
        rtnStr = rtnStr + "    NAME      = 'CommandRunControl' \n
            WIDTH      = 0 \n HEIGHT      = 0 \n";
        rtnStr = rtnStr + "    HSPACE    = 0 \n VSPACE    = 0 \n
            ALIGN      = middle \n >";
        rtnStr = rtnStr + "    <PARAM NAME = 'cache_archive' VALUE =
            'nlx_c.jar,nlx_s.jar,nlx_lx.jar,nlx_jc.jar,nlx_model.jar' />
            \n";
        rtnStr = rtnStr + "    <PARAM NAME = 'cache_version' VALUE =
            '1.0.0.0,1.0.0.0,1.0.0.0,1.0.0.0,1.0.0.0' /> \n";

        if (logpoints.length > 0){
            if (logpoints.charAt(logpoints.length - 1) != '\n'){
                logpoints = logpoints;
            }
            rtnStr = rtnStr + "<PARAM NAME = 'logpoints' VALUE = '" +
                logpoints + "' /> \n";
        }

        // Grab the insertion point and insert the code
        rtnStr = rtnStr + "</APPLET></div> \n";
        var selection = theDOM.source.getSelection().toString();
        for (var i = 0; i < selection.length; i++) {
            if (selection.charAt(i) == ',') {
                selection = selection.substring(0,i);
            }
        }
    }
}
```

```

    }
    theDOM.source.insert(parseInt(selection), rtnStr);
} else {
    rtnStr = "Please input values into the required fields
denoted by an asterisk (*).";
    return rtnStr;
}
}

function checkForm(input) {
    var spacecounter = 0;
    var elementlist = new Array("logpoints");
    empty = false;
    var i = 0;
    var j = 0;

    while (i < input.elements.length && !empty) {
        while (j < elementlist.length && !empty) {
            if (input.elements[i].name.toLowerCase() ==
                elementlist[j].toLowerCase()) {
                for (var k = 0;
                    k < input.elements[i].value.length;
k++) {
                    // Checks for spaces only in a field
                    if
(input.elements[i].value.charCodeAt(k) ==
                        32) {
                            spacecounter++;
                        } // end of if
                    } // end of for loop
                // Counts the number of fields with just spaces
                if ((spacecounter
                    >= input.elements[i].value.length)
                    || (input.elements[i].value == "")) {
                        empty = true;
                    } // end of if/else
                }
                spacecounter = 0;
                j++;
            }
            j = 0;
            i++;
        }

        if (empty) { // If any required field is empty
            return false;
        } else {
            return true;
        } // end of if/else
    }
}

```


APPENDIX B

Property Inspector Sample File Code

```
<!--tag:APPLET,priority:1,selection:within →
<!DOCTYPE HTML SYSTEM "-//Macromedia//DWExtension
  layout-engine5.0//pi">
<HTML>
<HEAD>
<TITLE>NLX Object Inspector</TITLE>
<SCRIPT LANGUAGE="JavaScript">
//Global variables to access form and layers
var top = document.topLayer.document.topLayerForm;
var bottom = document.bottomLayer.document.bottomLayerForm;

function canInspectSelection(){
    var theDOM = dw.getDocumentDOM();
    var theObj = theDOM.getSelectedNode();
    return (theObj.nodeType == Node.ELEMENT_NODE
        && theObj.getAttribute("code") ==
            "nlx.control.runcontrol.CommandRunControl.class");
}
function inspectSelection(){
    var theDOM = dw.getDocumentDOM();
    var theNode = theDOM.getSelectedNode();
    var children = theNode.childNodes;

    var parameter = "";
    for (var i = 0; i < children.length; i++) {
        parameter = children[i].getAttribute("name");
        switch (parameter) {
            case 'logpoints' : {
                top.logpoints.value =
                    children[i].getAttribute("value");
                break;
            }
            default : {
                break;
            }
        }
    }
}
function setAppletTag(){
    var theDOM = dw.getDocumentDOM();
    var theObj = theDOM.getSelectedNode();
    var children = theObj.childNodes;

    var parameter = "";
    for (var i = 0; i < children.length; i++) {
        parameter = children[i].getAttribute("name");
        switch (parameter) {
            case 'logpoints' : {
                children[i].setAttribute('value',
                    top.logpoints.value);
                break;
            }
        }
    }
}
```

```

                default : {
                    break;
                }
            }
        }

        theDOM.setSelectedNode(theObj);
    }
function getListChoice(param, foptions) {
    var i = 0;
    while (param != foptions[i].value &&
           i < foptions.length) {
        i++;
    }
    return i;
}

function displayHelp(){
    alert("One Shot Control: This will allow the control
          of the model - one run cycle at a time.");
}
</SCRIPT>
</HEAD>
<BODY>
<SPAN ID="image" STYLE="position:absolute; width:23px; height:17px;
    z-index:16; left: 3px; top: 2px">
<H4>NLX</H4>
</SPAN>

<SPAN ID="label" STYLE="position:absolute; width:23px; height:17px;
    z-index:16; left: 44px; top: 5px">One Shot Control </SPAN>

<SPAN ID="topLayer" STYLE="position:absolute; z-index:1;
    left: 80px; top: 3px; width: 431px; height: 55px">
<FORM NAME="topLayerForm">
    <table>
        <tr>
            <td align="right">Log Points:</td>
            <td><input name="logpoints" type="text" size="10"
                onBlur="setAppletTag()"></td>
        </tr>
    </table>
</FORM>
</SPAN>
<SPAN ID="bottomLayer" STYLE="position:absolute; z-index:1;
    left: 3px; top: 58px; width: 552px; height: 39px">
<FORM NAME="bottomLayerForm">
    <table>
        <tr>
            <td align="right"></td>
            <td></td>
        </tr>
    </table>
</form>
</SPAN>
</BODY>
</HTML>

```

APPENDIX C

Graph Floating Panel Sample File Code

```
<!doctype html public "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>NLX Editor</title>
<script language="javascript" src="nlx_FL_getDefault.js"></script>
<script language="JavaScript">
//Global variables to access form and layers
var top = document.layers['layer_ObjectForm'].document.objectForm;

//Initialization function to check for applet tag and which NLX
//object is selected
function loadObject() {
    // Get the selected node
    var theDOM = dw.getDocumentDOM();
    var theNode = theDOM.getSelectedNode();

    // Variables to hold the name and success of finding an object
    var objectname = "";
    var objectfound = false;
    // Check to see if the selected node is an applet, is so, check
    //for NLX object name
    if (theNode.nodeType == Node.ELEMENT_NODE && theNode.tagName
        == "APPLET"){
        objectname = theNode.getAttribute("code");
        switch (objectname) {
            case 'nlx.view.output.graph.BSLGraph.class' : {
                objectfound = true;
                loadGraph();
                break;
            }
            default : {
                alert("No NLX object found");
                objectfound = true;
                break;
            }
        }
    }
    } else {
        alert("No NLX Object is selected");
    }
}

function updateObject() {
    // Get the selected node
    var theDOM = dw.getDocumentDOM();
    var theNode = theDOM.getSelectedNode();

    // Variables to hold the name and success of finding an object
    var objectname = "";
    var objectfound = false;

    // Check to see if the selected node is an applet, is so, check
    //for NLX object name
```

```

if (theNode.nodeType == Node.ELEMENT_NODE && theNode.tagName
== "APPLET"){
    objectname = theNode.getAttribute("code");
    switch (objectname) {
        case 'nlx.view.output.graph.BSLGraph.class' : {
            objectfound = true;
            updateGraph();
            break;
        }
        default : {
            alert("No NLX object found");
            objectfound = true;
            break;
        }
    }
} else {
    alert("No NLX Object is selected");
}
}

//-----SUB FUNCTIONS-----
function clearAll() {
    top.varcolors.value = "";
    top.graphcolors.value = "";
    top.title.value = "";
    top.subtitle.value = "";
    top.declplaces.value = "";
    top.xaxislabel.value = "";
    top.xmin.value = "";
    top.xmax.value = "";
    top.yaxislabel.value = "";
    top.ymin.value = "";
    top.ymax.value = "";
    top.y2axislabel.value = "";
    top.y2min.value = "";
    top.y2max.value = "";
    top.numofdecades.value = "";
    top.varcolors2.value = "";
    top.varvisibilities.value = "";
    top.varvisibilities2.value = "";
    top.xtickspacing.value = "";
    top.ytickspacing.value = "";
    top.y2tickspacing.value = "";
    top.numofdecades2.value = "";
}

function getListChoice(param, foptions) {
    var i = 0;
    while (param != foptions[i].value && i < foptions.length) {
        i++;
    }
    return i;
}

//-----GRAPH-----
function loadGraph() {

```

```

var theDOM = dw.getDocumentDOM();
var theNode = theDOM.getSelectedNode();
var children = theNode.childNodes;

clearAll();

//Load default values from model into listing
getDefault();

var parameter = "";
for (var i = 0; i < children.length; i++) {
    parameter = children[i].getAttribute("name");
    switch (parameter) {
        case 'varnames' : {
            var varlist = parseList(children[i].
                getAttribute("value"));
            for (var k = 0; k < varlist.length; k++) {
                for (var j = 0; j <
top.varnames.options.length;
                    j++) {
                        if (varlist[k] ==
top.varnames.options[j].value) {
                            top.varnames.options[j].selected = true;
                        }
                    }
                }
            // top.varnames.value =
children[i].getAttribute("value");
            break;
        }
        case 'varnames2' : {
            var varlist2 = parseList(children[i].
                getAttribute("value"));
            for (var m = 0; m < varlist2.length; m++) {
                for (var j = 0; j <
top.varnames2.options.length;
                    j++) {
                        if (varlist2[m] ==
top.varnames2.options[j].value) {
                            top.varnames2.options[j].selected = true;
                        }
                    }
                }
            // top.varnames2.value =
children[i].getAttribute("value");
            break;
        }
        case 'varcolors' : {
            top.varcolors.value =
children[i].getAttribute("value");
            break;
        }
        case 'varcolors2' : {

```

```

        top.varcolors2.value =
children[i].getAttribute("value");
        break;
    }
    case 'varvisibilities' : {
        top.varvisibilities.value = children[i].
            getAttribute("value");
        break;
    }
    case 'varvisibilities2' : {
        top.varvisibilities2.value = children[i].
            getAttribute("value");
        break;
    }
    case 'graphcolors' : {
        top.graphcolors.value = children[i].
            getAttribute("value");
        break;
    }
    case 'title' : {
        top.title.value =
children[i].getAttribute("value");
        break;
    }
    case 'subtitle' : {
        top.subtitle.value =
children[i].getAttribute("value");
        break;
    }
    case 'decplaces' : {
        top.decplaces.value =
children[i].getAttribute("value");
        break;
    }
    case 'showptvalues' : {
        if (children[i].getAttribute("value") ==
'true') {
            top.showptvalues.checked = true;
        } else {
            top.showptvalues.checked = false;
        }
        break;
    }
    case 'xaxislabel' : {
        top.xaxislabel.value = children[i].
            getAttribute("value");
        break;
    }
    case 'xtickspacing' : {
        top.xtickspacing.value = children[i].
            getAttribute("value");
        break;
    }
    case 'xmin' : {
        top.xmin.value =
children[i].getAttribute("value");
        break;
    }

```

```

    }
    case 'xmax' : {
        top.xmax.value =
children[i].getAttribute("value");
        break;
    }
    case 'xgrid' : {
        if (children[i].getAttribute("value") ==
'true') {
            top.xgrid.checked = true;
        } else {
            top.xgrid.checked = false;
        }
        break;
    }
    case 'yaxislabel' : {
        top.yaxislabel.value = children[i].
            getAttribute("value");
        break;
    }
    case 'ytickspacing' : {
        top.ytickspacing.value = children[i].
            getAttribute("value");
        break;
    }
    case 'ymin' : {
        top.ymin.value =
children[i].getAttribute("value");
        break;
    }
    case 'ymax' : {
        top.ymax.value =
children[i].getAttribute("value");
        break;
    }
    case 'ygrid' : {
        if (children[i].getAttribute("value") ==
'true') {
            top.ygrid.checked = true;
        } else {
            top.ygrid.checked = false;
        }
        break;
    }
    case 'y2axislabel' : {
        top.y2axislabel.value = children[i].
            getAttribute("value");
        break;
    }
    case 'y2tickspacing' : {
        top.y2tickspacing.value = children[i].
            getAttribute("value");
        break;
    }
    case 'y2min' : {
        top.y2min.value =
children[i].getAttribute("value");

```

```

        break;
    }
    case 'y2max' : {
        top.y2max.value =
children[i].getAttribute("value");
        break;
    }
    case 'y2grid' : {
        if (children[i].getAttribute("value") ==
'true') {
            top.y2grid.checked = true;
        } else {
            top.y2grid.checked = false;
        }
        break;
    }
    case 'numofdecades' : {
        top.numofdecades.value = children[i].
            getAttribute("value");
        break;
    }
    case 'numofdecades2' : {
        top.numofdecades2.value = children[i].
            getAttribute("value");
        break;
    }
    case 'showpopupmenu' : {
        if (children[i].getAttribute("value") ==
'true') {
            top.showpopupmenu.checked = true;
        } else {
            top.showpopupmenu.checked = false;
        }
        break;
    }
    case 'hide2ndset' : {
        if (children[i].getAttribute("value") ==
'true') {
            top.hide2ndset.checked = true;
        } else {
            top.hide2ndset.checked = false;
        }
        break;
    }
    case 'plotstyle' : {
        top.plotstyle.selectedIndex =
            getListItemChoice(children[i].
getAttribute("value"), top.plotstyle.options);
        break;
    }
    case 'yscaletype' : {
        top.yscaletype.selectedIndex =
            getListItemChoice(children[i].
getAttribute("value"), top.yscaletype.options);
        break;
    }

```



```

    }
    case 'y2scaletype' : {
        top.y2scaletype.selectedIndex =
            getListChoice(children[i].

getAttribute("value"),top.y2scaletype.options);
        break;
    }
    default : {
        break;
    }
}
}
top.objectName.value =
theNode.getAttribute("name").toUpperCase();
document.layers['layer_ObjectForm'].visibility = 'visible';
}

function updateGraph() {
    var theDOM = dw.getDocumentDOM();
    var theNode = theDOM.getSelectedNode();
    var children = theNode.childNodes;

    var parameter = "";
    for (var i = 0; i < children.length; i++) {
        parameter = children[i].getAttribute("name");
        switch (parameter) {
            case 'varnames' : {
                var varnames = "";
                var j = 0;
                //The j < 11 is there just in case the loop
                //will not terminate
                while (j < top.varnames.options.length && j <
11) {
                    if (top.varnames.options[j].selected) {
                        varnames = varnames +
                            top.varnames.options[j].value
+ ",";
                    }
                    j++;
                }
                varnames =
varnames.substring(0,varnames.length-1);
                children[i].setAttribute('value',varnames);
                break;
            }
            case 'varnames2' : {
                var varnames2 = "";
                var k = 0;
                while (k < top.varnames2.options.length && k <
11) {
                    if (top.varnames2.options[k].selected) {
                        varnames2 = varnames2 +
                            top.varnames2.options[k].value + ",";
                    }
                }
            }
        }
    }
}

```

```

                k++;
            }
            varnames2 =
varnames2.substring(0,varnames2.length-1);
            children[i].setAttribute('value',varnames2);
//
        children[i].setAttribute('value',top.varnames2.value)
            break;
        }
        case 'varcolors' : {
            children[i].setAttribute('value',top.varcolors.value)
                break;
            }
        case 'varcolors2' : {
            children[i].setAttribute('value',top.varcolors2.value)
                break;
            }
        case 'varvisibilities' : {
            children[i].setAttribute('value',
                top.varvisibilities.value)
                break;
            }
        case 'varvisibilities2' : {
            children[i].setAttribute('value',
                top.varvisibilities2.value)
                break;
            }
        case 'graphcolors' : {
            children[i].setAttribute('value',top.graphcolors.value)
                break;
            }
        case 'title' : {
            children[i].setAttribute('value',top.title.value)
                break;
            }
        case 'subtitle' : {
            children[i].setAttribute('value',top.subtitle.value)
                break;
            }
        case 'deplaces' : {
            children[i].setAttribute('value',top.deplaces.value)
                break;
            }
        case 'showptvalues' : {
            if (top.showptvalues.checked) {
                children[i].setAttribute('value','true')
            } else {
                children[i].setAttribute('value','false')
            }
            break;
        }
    }
}

```

```

        case 'xaxislabel' : {

children[i].setAttribute('value',top.xaxislabel.value)
            break;
        }
        case 'xtickspacing' : {

children[i].setAttribute('value',top.xtickspacing.value)
            break;
        }
        case 'xmin' : {

children[i].setAttribute('value',top.xmin.value)
            break;
        }
        case 'xmax' : {

children[i].setAttribute('value',top.xmax.value)
            break;
        }
        case 'xgrid' : {
            if (top.xgrid.checked) {
                children[i].setAttribute('value','true')
            } else {
                children[i].setAttribute('value','false')
            }
            break;
        }
        case 'yaxislabel' : {

children[i].setAttribute('value',top.yaxislabel.value)
            break;
        }
        case 'ytickspacing' : {
            children[i].setAttribute('value',
                top.ytickspacing.value)
            break;
        }
        case 'ymin' : {

children[i].setAttribute('value',top.ymin.value)
            break;
        }
        case 'ymax' : {

children[i].setAttribute('value',top.ymax.value)
            break;
        }
        case 'ygrid' : {
            if (top.ygrid.checked) {
                children[i].setAttribute('value','true')
            } else {
                children[i].setAttribute('value','false')
            }
            break;
        }
        case 'y2axislabel' : {

```

```

        children[i].setAttribute('value',
                                top.y2axislabel.value)
        break;
    }
    case 'y2tickspacing' : {
        children[i].setAttribute('value',
                                top.y2tickspacing.value)
        break;
    }
    case 'y2min' : {
children[i].setAttribute('value',top.y2min.value)
        break;
    }
    case 'y2max' : {
children[i].setAttribute('value',top.y2max.value)
        break;
    }
    case 'y2grid' : {
        if (top.y2grid.checked) {
            children[i].setAttribute('value','true')
        } else {
            children[i].setAttribute('value','false')
        }
        break;
    }
    case 'numofdecades' : {
        children[i].setAttribute('value',
                                top.numofdecades.value)
        break;
    }
    case 'numofdecades2' : {
        children[i].setAttribute('value',
                                top.numofdecades2.value)
        break;
    }
    case 'showpopupmenu' : {
        if (top.showpopupmenu.checked) {
            children[i].setAttribute('value','true')
        } else {
            children[i].setAttribute('value','false')
        }
        break;
    }
    case 'hide2ndset' : {
        if (top.hide2ndset.checked) {
            children[i].setAttribute('value','true')
        } else {
            children[i].setAttribute('value','false')
        }
        break;
    }
    case 'plotstyle' : {
        children[i].setAttribute('value',
                                top.plotstyle.options[top.plotstyle.
                                    selectedIndex].value);
    }

```

```

        break;
    }
    case 'yscaletype' : {
        children[i].setAttribute('value',
            top.yscaletype.options[
top.yscaletype.selectedIndex].value);
        break;
    }
    case 'y2scaletype' : {
        children[i].setAttribute('value',
            top.y2scaletype.options[
top.y2scaletype.selectedIndex].value);
        break;
    }
    default : {
        break;
    }
}
}
}

function parseList(list) {
    var startpos = 0;
    var stoppos = 0;
    var listarray = new Array();

    for (var i = 0; i < list.length; i++) {
        if (list.charAt(i) == ',' | stoppos == list.length) {
            listarray.push(list.substring(startpos, stoppos));
            i++;
            startpos = i;
            stoppos = i + 1;
        } else {
            stoppos++;
        }
    }

    listarray.push(list.substring(startpos, list.length));

    return listarray;
}
</script>
<body>
<div id="layer_ObjectForm" style="position:absolute; width:950px;
    height:700px; z-index:1; left: 8px; top: 11px;
    visibility: visible">
<form name="objectForm">
<table width="100%" border="1" cellpadding="0" cellspacing="0"
    id="objectTB">
<tr><td colspan="2">
<table width="100%" border="0" cellpadding="0" cellspacing="0"
    bgcolor="#CCCCCC" id="objectNameTB">
<tr><td align="left">Object Type:</td><td><input name="objectName"
    type="button" onClick="loadObject()">
        Click to load parameters...</td></tr></table></td>

```

```

<td align="center" bgcolor="#CCCCCC"><input name="Save" type="button"
    value="Save Changes..." onClick="updateObject()"></td>
</tr>
<tr align="left"><td><table width="100%" border="0" cellpadding="0"
    cellspacing="0" bgcolor="#FFFFFF" id="titleTB">
    <tr><td align="left">Title:</td>
        <td align="right"><input type="text" name="title"></td>
    </tr></table></td>
<td><table width="100%" border="0" cellpadding="0" cellspacing="0"
    bgcolor="#FFFFFF" id="subtitleTB">
    <tr><td align="left">Subtitle:</td>
        <td align="right"><input type="text" name="subtitle"></td>
    </tr></table></td>
<td><table width="100%" border="0" cellpadding="0"
cellspacing="0"
    bgcolor="#FFFFFF" id="showpopupmenuTB">
    <tr><td width="80%" align="left">Show Popup Menu:</td><td width="20%"
        align="center"><input type="checkbox" name="showpopupmenu"></td>
    </tr></table></td>
</tr>
<tr align="left"><td><table width="100%" border="0" cellpadding="0"
    cellspacing="0" bgcolor="#FFFFFF" id="decplacesTB">
    <tr>
        <td align="left">Decimal Places:</td>
        <td align="right"><input type="text" name="decplaces"></td>
    </tr></table></td>
<td><table width="100%" border="0" cellpadding="0" cellspacing="0"
    bgcolor="#FFFFFF" id="showptvaluesTB">
<tr><td width="80%" align="left">Show Point Values?:</td>
    <td width="20%" align="center"><input type="checkbox"
        name="showptvalues"></td>
    </tr></table></td>
<td><table width="100%" border="0" cellpadding="0" cellspacing="0"
    bgcolor="#FFFFFF" id="hide2ndsetTB">
<tr><td width="80%" align="left">Hide Second Set:</td><td width="20%"
    align="center"><input type="checkbox" name="hide2ndset"></td>
    </tr></table></td>
</tr>
<tr align="left">
<td><table width="100%" border="0" cellpadding="0" cellspacing="0"
    bgcolor="#FFFFFF" id="messagesTB">
<tr>
    <td align="left">Variable Names:</td>
    <td align="right"><select name="varnames" size="5" multiple
        type="select-multiple">
            <option value="">-blank-</option>
        </select></td>
</tr></table></td>
<td>
    <table width="100%" border="0" cellpadding="0"
cellspacing="0"
    bgcolor="#FFFFFF" id="varcolorsTB">
    <tr><td align="left">Variable Colors:</td>
        <td align="right"><textarea
name="varcolors"></textarea></td>
    </tr>

```

```

        </table>    </td>

    <td><table width="100%" border="0" cellpadding="0" cellspacing="0"
        bgcolor="#FFFFFF" id="varvisibilitiesTB">
    <tr><td align="left">Variable Visibilities:</td>
        <td align="right"><textarea name="varvisibilities"></textarea>
        </td></tr></table></td>
</tr>

<tr align="left">
    <td><table width="100%" border="0" cellpadding="0" cellspacing="0"
        bgcolor="#FFFFFF" id="varnames2TB">
    <tr><td align="left">Variable Names (2):</td><td align="right">
    <select name="varnames2" size="5" multiple type="select-multiple">
        <option value="">-blank-</option>
        </select>
        </td></tr></table></td>
<td><table width="100%" border="0" cellpadding="0" cellspacing="0"
        bgcolor="#FFFFFF" id="varcolors2TB">
    <tr><td align="left">Variable Colors (2):</td><td align="right">
        <textarea name="varcolors2"></textarea></td></tr></table></td>
    <td><table width="100%" border="0" cellpadding="0" cellspacing="0"
        bgcolor="#FFFFFF" id="varvisibilities2TB">
    <tr><td align="left">Variable Visibilities (2):</td>
        <td align="right"><textarea
name="varvisibilities2"></textarea></td>
    </tr></table></td> </tr>

<tr align="left">
    <td> <table width="100%" border="0" cellpadding="0" cellspacing="0"
        bgcolor="#FFFFFF" id="numofdecadesTB">
    <tr><td align="left">Number of Decades:</td>
        <td align="right"><input type="text" name="numofdecades"></td>
    </tr></table></td>
    <td><table width="100%" border="0" cellpadding="0"
cellspacing="0"
        bgcolor="#FFFFFF" id="numofdecades2TB">
    <tr><td align="left">Number of Decades (2):</td>
        <td align="right"><input type="text" name="numofdecades2"></td>
    </tr></table></td>
    <td>
        <table width="100%" border="0" cellpadding="0"
cellspacing="0"
        bgcolor="#FFFFFF" id="graphcolorsTB">
    <tr><td align="left">Graph Colors:</td>
        <td align="right">
            <textarea name="graphcolors"></textarea></td>
        </tr>
    </table>    </td>
</tr>

<tr align="left"><td><table width="100%" border="0" cellpadding="0"
cellspacing="0" bgcolor="#FFFFFF" id="xtickspacingTB">
    <tr><td align="left">X Tick Spacing:</td>
        <td align="right"><input type="text" name="xtickspacing"></td>
    </tr></table></td>
<td><table width="100%" border="0" cellpadding="0" cellspacing="0"

```

```

        bgcolor="#FFFFFF" id="ytickspacingTB">
<tr><td align="left">Y Tick Spacing:</td><td align="right">
    <input type="text" name="ytickspacing"></td>
</tr></table></td>
<td><table width="100%" border="0" cellpadding="0" cellspacing="0"
    bgcolor="#FFFFFF" id="y2tickspacingTB">
<tr><td align="left">Y2 Tick Spacing:</td><td align="right">
    <input type="text" name="y2tickspacing"></td>
</tr></table></td></tr>

<tr align="left">
<td><table width="100%" border="0" cellpadding="0" cellspacing="0"
    bgcolor="#FFFFFF" id="xaxislabelTB">
    <tr><td align="left">X-axis Label:</td>
        <td align="right"><input type="text" name="xaxislabel"></td>
    </tr></table></td>
<td><table width="100%" border="0" cellpadding="0" cellspacing="0"
    bgcolor="#FFFFFF" id="yaxislabelTB">
    <tr><td align="left">Y-axis Label:</td>
        <td align="right"><input type="text" name="yaxislabel"></td>
    </tr></table></td>
<td><table width="100%" border="0" cellpadding="0" cellspacing="0"
    bgcolor="#FFFFFF" id="y2axislabelTB">
    <tr><td align="left">Y2-axis Label:</td>
        <td align="right"><input type="text" name="y2axislabel"></td>
    </tr></table></td>
</tr>
<tr align="left">
<td><table width="100%" border="0" cellpadding="0" cellspacing="0"
    bgcolor="#FFFFFF" id="xminTB">
    <tr>
        <td align="left">X-axis Min Value</td>
        <td align="right"><input type="text" name="xmin"></td>
    </tr></table></td>
<td><table width="100%" border="0" cellpadding="0" cellspacing="0"
    bgcolor="#FFFFFF" id="yminTB">
    <tr><td align="left">Y-axis Min Value:</td>
        <td align="right"><input type="text" name="ymin"></td>
    </tr></table></td><td><table width="100%" border="0" cellpadding="0"
        cellspacing="0" bgcolor="#FFFFFF" id="y2minTB">
    <tr>
        <td align="left">Y2-axis Min Value:</td>
        <td align="right"><input type="text" name="y2min"></td>
    </tr></table></td>
</tr>
<tr align="left">
<td><table width="100%" border="0" cellpadding="0" cellspacing="0"
    bgcolor="#FFFFFF" id="xmaxTB">
    <tr><td align="left">X-axis Max Value:</td>
        <td align="right"><input type="text" name="xmax"></td>
    </tr></table></td>
<td><table width="100%" border="0" cellpadding="0" cellspacing="0"
    bgcolor="#FFFFFF" id="ymaxTB">
    <tr><td align="left">Y-axis Max Value:</td>
        <td align="right"><input type="text" name="ymax"></td>
    </tr></table></td>
<td><table width="100%" border="0" cellpadding="0" cellspacing="0"

```



```

        bgcolor="#FFFFFF" id="y2maxTB">
<tr><td align="left">Y2-axis Max Value:</td>
  <td align="right"><input type="text" name="y2max"></td>
</tr></table></td>
</tr>

<tr align="left">
  <td><table width="100%" border="0" cellpadding="0" cellspacing="0"
    bgcolor="#FFFFFF" id="xgridTB">
    <tr><td align="left">X-axis Grid?:</td>
    <td align="center"><input type="checkbox" name="xgrid"></td>
    </tr></table></td>
    <td><table width="100%" border="0" cellpadding="0"
cellspacing="0"
      bgcolor="#FFFFFF" id="ygridTB">
      <tr><td align="left">Y-axis Grid?:</td>
      <td align="center"><input type="checkbox" name="ygrid"></td>
      </tr></table></td>
      <td><table width="100%" border="0" cellpadding="0"
cellspacing="0"
        bgcolor="#FFFFFF" id="y2gridTB">
        <tr><td width="80%" align="left">Y2-axis Grid?:</td>
        <td width="20%" align="center">
          <input type="checkbox" name="y2grid"></td>
        </tr></table></td></tr>
<tr align="left">
<td><table width="100%" border="0" cellpadding="0" cellspacing="0"
  bgcolor="#FFFFFF" id="plotstyleTB">
  <tr><td align="left">Plot Style:</td>
  <td align="right"><select name="plotstyle">
    <option value="point" selected>Point</option>
    <option value="line">Line</option>
  </select></td>
  </tr></table></td>
  <td><table width="100%" border="0" cellpadding="0" cellspacing="0"
    bgcolor="#FFFFFF" id="yscaletypeTB">
    <tr><td align="left">Y Scale Type:</td>
    <td align="right"><select name="yscaletype">
      <option value="linear" selected>Linear</option>
      <option value="log">Log</option>
    </select></td>
    </tr></table></td>
  <td><table width="100%" border="0" cellpadding="0" cellspacing="0"
    bgcolor="#FFFFFF" id="y2scaletypeTB">
    <tr><td align="left">Y2 Scale Type:</td>
    <td align="right"><select name="y2scaletype">
      <option value="linear" selected>Linear</option>
      <option value="log">Log</option>
    </select></td>
    </tr></table></td>
  </tr>
</table>
</form>
</div>
</body>
</html>

```

APPENDIX D

NLX Code for Insertbar.xml

```
<category id="DW_Insertbar_NLX" name="NLX" folder="NLX">
  <button id="DW_DefaultRunControl"
    name="Default Run Control" image="NLX\DefaultControl.gif"
    file="NLX\DefaultControl.htm"/>
  <button id="DW_ContinuousRunControl" name="Continuous Run
Control"
    image="NLX\ContinuousControl.gif"
    file="NLX\ContinuousControl.htm"/>
  <button id="DW_OneShotControl" name="One Shot Control"
    image="NLX\OneShotControl.gif"
    file="NLX\OneShotControl.htm"/>
  <separator />
  <button id="DW_DefaultPanel" name="Default Panel"
    image="NLX\DefaultPanel.gif" file="NLX\DefaultPanel.htm"/>
  <button id="DW_SimplePanel" name="Simple Panel"
    image="NLX\SimplePanel.gif" file="NLX\SimplePanel.htm"/>
  <button id="DW_ResetPanel" name="Reset Panel"
    image="NLX\ResetPanel.gif" file="NLX\ResetPanel.htm"/>
  <separator />
  <button id="DW_Slider" name="Slider"
    image="NLX\Slider.gif" file="NLX\Slider.htm"/>
  <button id="DW_Knob" name="Knob" image="NLX\Knob.gif"
    file="NLX\Knob.htm"/>
  <button id="DW_ItemSelector" name="Item Selector"
    image="NLX\ItemSelector.gif" file="NLX\ItemSelector.htm" />
  <button id="DW_DialInput" name="Dial Input"
    image="NLX\DialInput.gif" file="NLX\DialInput.htm"/>
  <button id="DW_ParameterInit" name="Parameter Init"
    image="NLX\ParameterInit.gif"
file="NLX\ParameterInit.htm"/>
  <separator />
  <button id="DW_Graph" name="Graph" image="NLX\Graph.gif"
    file="NLX\Graph.htm"/>
  <button id="DW_Legend" name="Legend" image="NLX\Legend.gif"
    file="NLX\Legend.htm"/>
  <button id="DW_DialOutput" name="Dial Output"
    image="NLX\DialOutput.gif" file="NLX\DialOutput.htm"/>
  <button id="DW_ConditionalText" name="Conditional Text"
    image="NLX\CondxText.gif" file="NLX\CondxText.htm"/>
  <button id="DW_NumericOutput" name="Numeric Output"
    image="NLX\NumericOutput.gif"
file="NLX\NumericOutput.htm"/>
  <button id="DW_SVGAnimation" name="SVG Animation"
    image="NLX\SVG.gif" file="NLX\SVGAnim.htm"/>
  <button id="DW_CondGraphic" name="Conditional Graphic"
    image="NLX\CondGraphic.gif" file="NLX\CondGraphic.htm"/>
</category>
```

APPENDIX E

NLX Code for Menus.xml File

```
<menu name="_NLX" id="DWMenu_NLX">  
  <menuitem name="Graph Editor" enabled="true"  
    command="dw.toggleFloater('graphEditor')"  
    checked="dw.getFloaterVisibility('graphEditor')" />  
</menu>
```

APPENDIX F

Instructions for User Study

1. Start Dreamweaver and open the HW Model template file.
2. Save the file with a new filename.
3. Insert each object below into a separate layer and set parameters:
 - a. Default Run Control – Set stoptime to “100” and numbreakpoints to “100”.
 - b. Knob – Set parameter parname to “k”.
 - c. Default Panel
 - d. Numeric Output – Set varname to “U”.
 - e. Numeric Output – Set varname to “T”.
 - f. Numeric Output – Set varname to “S”.
 - g. Graph – Set varnames to “time,S,U,T” and xmax to “100”.
4. Update the numapplets parameter of Starter applet with the total number of NLX objects (Java applets) on the page.
5. Save the page and open it in the Internet Explorer browser. If there are problems, return to the page to fix and retry.