Improving System Dynamics Teaching Using Online Surveys and Exercises

by

Nicholas A. Behrens

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degrees of

Bachelor of Science in Computer Science and Engineering

and Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

May 26, 2006

Author
Departme Science
26, 2006

Certified by
John Sterman
namics Group
s Supervisor

Accepted by
ur C. Smith
Chairman, Department Committee on Graduate Theses

Improving System Dynamics Teaching Using Online Surveys and Exercises

by

Nicholas A. Behrens

Submitted to the
Department of Electrical Engineering and Computer Science

May 26, 2006

In Partial Fulfillment of the Requirements for the Degree of
Bachelor of Science in Computer Science and Engineering
and Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

As the world becomes increasingly connected, traditional problem solving and decision-making skills becomes less effective. Complex systems found in nature and society exhibit long time delays between cause and effect, feedback, and non-linearity making it difficult to reason effectively about system behavior. Recent studies have shown even highly educated graduate students lack basic systems thinking skills indicating a need for improved system dynamics education. This paper describes the development of a two new tools for improving system dynamics education: a stock-flow simulator that allows users to experiment with simple stock-flow systems, and a web application framework for building system dynamics surveys. This framework is used to build a survey capable of evaluating systems thinking skills and compare the effectiveness of online teaching methods.

Thesis Supervisor: John Sterman
Title: Jay W. Forrester Professor of Management

Table of Contents

List of Figures

Introduction

The ability to reason about complex systems concepts such as stocks, delays, and feedback are crucial to successful decision making. As business, economic, and government systems become more interconnected, interactive complexity grows. New policies affect all system components and often result in feedback effects with long time horizons. Attempting to affect targeted changes in such an environment is risky and unpredictable. Only by leveraging systems thinking skills, can one understand such systems and consistently achieve desired outcomes. In short, these skills are crucial to successful decision making.

Despite their clear importance, research shows few people have sufficient intuition to apply these skills. In an inventory of system thinking skills, Booth Sweeney and Sterman (2000) found consistently poor performance among highly educated business school students suggesting "violations of basic principles, not merely calculation errors". In an experiment investigating mental models of climate change, Sterman and Booth Sweeney (2006) show "widespread misunderstanding of the fundamental stock and flow relationships" among similarly educated graduate students at MIT.

This lack of system thinking skills requires more focused and effective education. A key component of developing system dynamics education is a way to compare the effectiveness of new and existing teaching methods. Comparisons need to be done via controlled experiments where the teaching method is the independent variable. Experiments, however, generally require significant amount of time to receive approval, collect data, and analyze data. If there was a way to speed up one or more of these processes, experiments could become cheaper, allowing for more iterative improvement in teaching methods in the same amount of time.

This paper describes the development of a software framework for delivering system dynamics surveys and exercises online. This framework will help improve system dynamics teaching methods by facilitating rapid experimental evaluation of new and existing teaching methods

A stock-flow simulator was also developed as a new teaching tool for both students and instructors. This simulator allows students to visualize the relationships between stocks and flows and how changes to each component affect the rest of the system.

Background

Previous work relevant to online system dynamics education falls into three main categories:

1) Formal modeling tools
2) Commercial survey applications
3) Management flight simulators

Sastry and Sterman list *Dynamo*, *Dysmap*, *Stella/iThink*, *Vensim*, and *Microworld Creator/S*^{4} as available modeling and simulation tools in their survey of system dynamics literature (1992). These software tools enable users to build and analyze models of complex systems. Unfortunately, they often require significant skill to use effectively, and thus are ill-suited for learning basic stock and flow concepts. *Stella*, developed by *iseesystems*, is an exception. This software package is specifically designed for education and research. It is a commercial product, however, and not available over the Internet, limiting its usefulness in exploratory research and education.

Online survey applications are available from a variety of vendors. An Internet search for "online surveys" yields the following applications: *QuestionPro*, *Surveyz!*, *WebSurveyor*, *Zoomerang*. These applications serve marketing, business, government, and educational organizations. None of them is free, limiting their usefulness for research and education, and none of them allow construction of custom survey inputs. For example, it is not possible to create a Java applet that collects graphical responses; only the classic HTML input mechanisms (i.e. text, checkboxes, and radio buttons) are supported. The survey framework described in this paper is distinguished from these systems by supporting custom input modes (in addition to the classic HTML inputs) via well-defined programmatic interfaces. Researchers are thus able to extend the system to suit their needs.

Management flight simulators from the business dynamics world are the best examples of web-based teaching of systems concepts. Sterman's beer distribution game was the first flight simulator (1992). Since then, a handful of companies have formed offering services and tools for the creation and hosting of management simulators. Two examples are *Forio Business Simulations* and *Management Simulations Inc.* Most simulators are turn-based exercises, allowing users to make decisions affecting the underlying model and viewing the results at regular intervals. The simulator described in this paper provides an intuitive interface supporting increased interactivity over previous flight simulator. Also, the simulator's focus on teaching just the fundamental system concepts should allow it to be an effective teaching tool for a larger group of people.

Project Goals

The primary project goal was to develop software tools to improve system dynamics education and enable future research.

This goal is achieved with the development of the following software tools:

1. A framework for developing and managing survey-based experiments online
2. A framework for developing and managing online exercises
3. A simulator for visualizing and exploring stock-flow systems

The first two tools comprise the Survey Framework. The last tool is the Stock-Flow Simulator. The following sections describe the design and development of these tools.

As the Survey Framework was developed, the Greenhouse Gas Survey was also developed. This survey is an online version of an existing systems thinking skills survey developed and used by Sterman and Booth Sweeney in their Bathtub Dynamics paper (2000). Creating this online version helped identify required features in the Survey Framework. Experiments are planned that will use this survey to compare and evaluate system dynamics teaching methods.

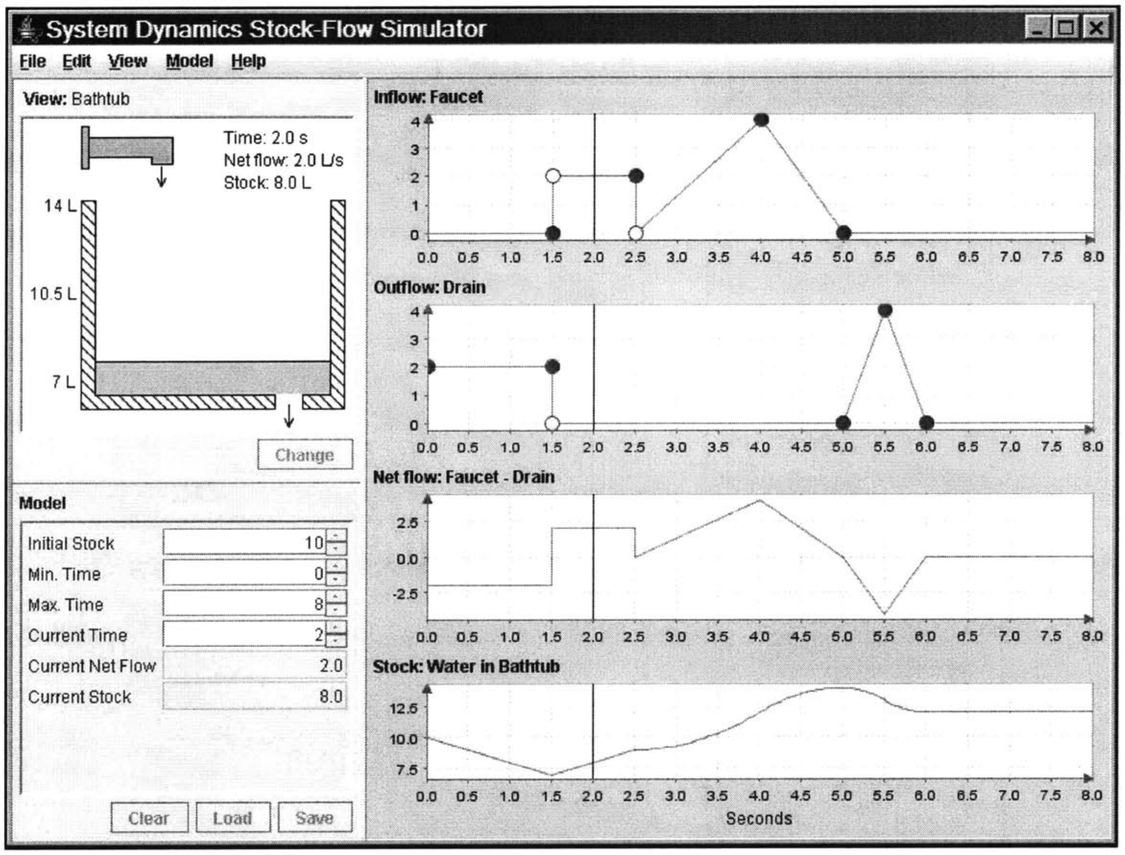
Stock-Flow Simulator

This section provides an overview of the Stock-Flow Simulator, the requirements influencing the design of the simulator, and notable design and implementation issues.

Overview

The Stock-Flow Simulator allows users to experiment with a simple stock-flow system containing a single inflow and a single outflow. This simple system is designed to be as basic as possible, with no feedbacks, time delays, or other complexity so learners can develop strong intuition for the fundamental principles of accumulation.

A screenshot of a simulator with user-defined inflow and outflow functions is presented below:



Time: 2.0 s
Net flow: 2.0 L/s
Stock: 8.0 L

Initial Stock	10
Min. Time	0
Max. Time	8
Current Time	2
Current Net Flow	2.0
Current Stock	8.0

Figure 1: The stock-flow simulator

The Simulator contains three main components:

1. Charts Panel
2. Model Panel
3. View Panel

The following sections provide additional details about these three main panels.

Charts Panel

The charts panel is displayed on the right. It contains four charts representing the system Inflow, Outflow, Net flow, and Stock:

1. The Inflow chart shows the rate at which units enter the stock. It shows the rate at which items are added to the stock at any moment.
2. The Outflow chart shows the rate at which units leave the stock. It shows the rate at which items are removed from the stock at any moment.
3. The Net flow chart is the difference between the Inflow and Outflow. It shows the net rate at which the stock is changing at any moment (Net flow = Inflow - Outflow).
4. The Stock chart shows the quantity in the stock at any moment resulting from the flows above, starting at the initial quantity.

The Inflow and Outflow charts may be edited. The "User Interface" section that follows this overview section describes the interface for editing functions.

Model Panel

The model panel is displayed on the bottom left. It displays system parameters such as the initial stock, the time range, and the current time in a simple, numerical format.

The following system parameters may be edited:

1. Initial Stock - The stock value at the minimum time, indicating how much is in the stock when the time range begins
2. Minimum Time - The lower bound of the system time range
3. Maximum Time - The upper bound of the system time range
4. Current Time - The time at which the Current Net Flow and Current Stock parameters are evaluated

View Panel

The view panel is displayed on the top left. It contains system information similar to the model panel. This information, however, is represented by a metaphor or symbol. These metaphors can often be easier to understand or relate to than the raw numbers in the model panel.

The default view metaphor is the bathtub. This metaphor displays system state using a tub of water. The quantity of water in the tub is the system stock. The faucet at the top represents flow into the system ("Inflow"). The drain at the bottom represents flow out of the system ("Outflow").

As a user changes the current time in the model panel, the level of water displayed in the tub will update to show its current value. In this way, the simulator can be used to animate the stock in a system over time. By clicking and holding over the up or down arrows that change the current time, the level of water displayed in the tub will trace out the pattern shown in the Stock chart.

Design and Implementation

This section begins by outlining the primary requirements guiding the design and implementation of the Stock-Flow Simulator. It also includes a brief description of the implementation technologies, a high-level overview of the Simulator's implementation, and a section devoted to numerical issues encountered during development and testing.

Requirements

The Simulator's target audiences are students and teachers with diverse technical backgrounds. The Simulator must therefore be intuitive and easy to use.

Students and teachers comprising the target audience likely use a wide variety of computing platforms (e.g. Windows, Mac, and Linux). The Simulator should be capable of running on any of these platforms.

At the very least, the Simulator must support construction of step functions for flows. Constant rate flow functions are the simplest flows to explain. As a teaching tool, the Simulator must support these flows. If possible, the Simulator should also allow construction of piecewise-linear functions for system flows.

The Simulator should support different system metaphors. Example metaphors are a bathtub with a faucet and drain, a bank account with deposits and withdrawals, and a department store with people leaving and entering.

Technologies

The Simulator is implemented in Java. Java allows the simulator to be run as a stand-alone application or delivered over the Internet as an applet.

Java is mature and portable. There is a rich library of available development tools for the language, and user interface development is simplified by the Swing libraries.

The chart panels use JFreeChart, an open-source charting library for Java, to render the stock and flow functions. While there were additional challenges integrating the user interface with the charting programming interface, the benefits of not having to re-implement charting significantly outweighed this cost.

Modules and Interfaces

The simulator is composed of four main modules:

1. Interface
2. Metaphor
3. Model
4. Utilities

These modules as well as their primary components are shown in the diagram below. Dotted lines represent modules. Boxes represent components. Arrows indicate dependencies. Implementation details are omitted where possible to improve readability.

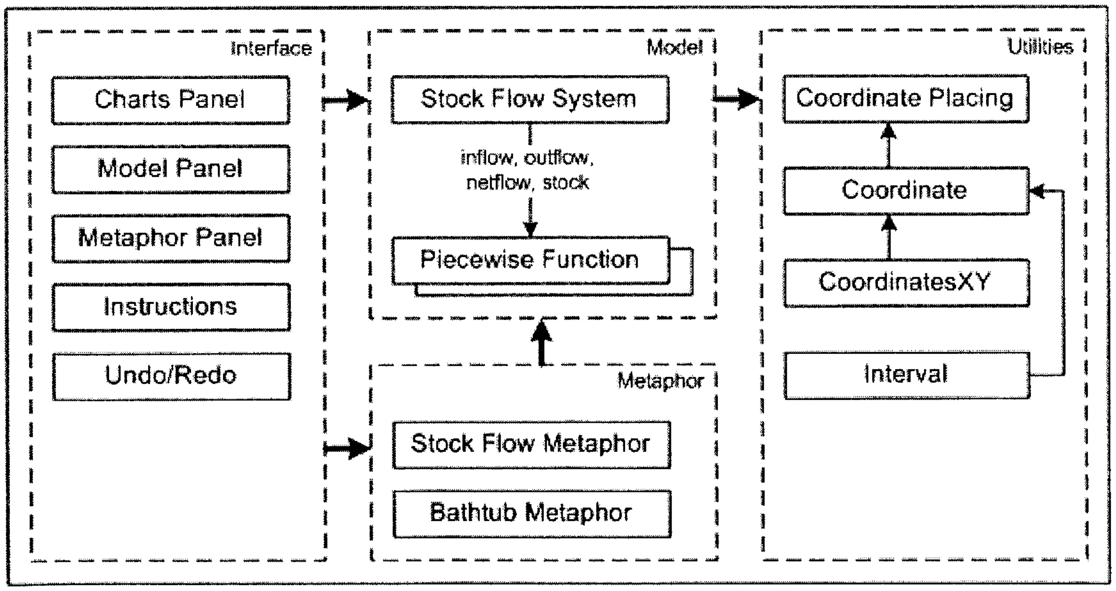


Figure 2: Stock-Flow Simulator modules and their dependencies

The Interface module defines the user interface components such as the charts panel, the model panel, and the metaphor panel. This module also contains the instruction components and support for undo and redo operations.

The Metaphor module defines the interface for all stock-flow metaphors. A default metaphor implementation—the bathtub metaphor—is also contained in this module.

The Model module is the core of the simulator, defining the stock-flow system representation. A system contains a single inflow, a single outflow, a net flow, and a stock, all represented as piecewise functions. The flow functions are piecewise linear, whereas the stock function is piecewise quadratic.

The Utilities module defines data types such as coordinates and intervals needed by the model. See the "Numerical Issues" section for a discussion of the coordinate components.

One Model, Many Views

The charts, the metaphor, and the system parameters displayed in the model panel are all views into a single stock-flow system instance maintained by the application. This stock-flow system allows these components to register to be notified when the system changes. This event system (the Observer pattern) allows the model to stay decoupled from the various components that depend on it, while also ensuring dependent components are always updated to reflect the latest system state. For example, when the metaphor panel receives a system-changed event, the metaphor can query the model to determine which features have changed and update the visual representation (e.g. the water level in the bathtub).

Metaphors

The metaphor interface simplifies the process of creating and integrating new metaphors for a stock-flow system. By implementing the interface, new metaphors can simply be "plugged-in". In addition to providing a visual description of the system state, a metaphor defines name and units for the stock and flow functions. The bathtub metaphor is the default metaphor.

Numerical Issues

Early simulator implementations used standard floating-point values to represent coordinates in the system functions. This representation was poorly suited to representing step functions, though; floating-point values only have magnitude, no concept of coordinate placing. To remedy this, a coordinate placing is associated with floating-point values to define formal coordinate objects. The placing value may be "above", "below" or "on" corresponding to the mathematical notion of $x+$, $x-$, and x, respectively. By representing function coordinates in this manner, the simulator is able to represent step functions more effectively.

Numerical integration also proved to be challenge in early simulator versions. Floating-point numbers have limited precision due to their fixed-length binary encoding. Standard numerical integration techniques that rely on subdividing intervals and summing interval areas exhibit small but noticeable errors that accumulate as the number of summations increase.

To avoid these issues, the simulator limits flow functions to piecewise linear functions. Limiting the flow functions to piecewise linear functions guarantees the net flow function will be piecewise linear as well. These piecewise linear functions can be integrated exactly using area formulas from geometry. In addition to greater precision and accuracy, this method tends to be much more efficient than integration by subdivision, since it only accumulates area in the intervals between control points.

User Interface

The Simulator user interface strives to be as usable as possible. Interface development leveraged an spiral development model. Design began with low-fidelity paper prototypes. Digital imaging programs were later used to create static images for each interface component. User testing with these prototypes helped work out usability issues before significant resources were invested in implementation. This iterative development model is largely responsible for the quality of the current user interface.

The following section provides an overview of important user interface features.

Editing Functions

System flow functions (inflow and outflow) may be edited in the charts panel on the right.

These flow functions are defined by a set of points called control points. At a control point, the value of the function is equal to the control point. Between control points, the function's value is interpolated linearly (connected by a straight line). Thus, by inserting, removing, and moving control points, users can define a wide range of patterns for the Inflow and Outflow.

Function editing requires four basic operations:

1. Insertion
2. Removal
3. Modification
4. Selection

Inserting Control Points

New control points are inserted by double-clicking with the mouse on the Inflow or Outflow chart. The location of the mouse cursor will determine the location of the new control point.

A control point can only be inserted where there is no existing control point. Attempting to insert a control point at the same location as an existing control point will have no effect.

Moving Control Points

Control points may be moved by pressing and holding the mouse over a control point. While keeping the mouse button down, move the mouse to re-position the control point. When the desired position is reached, release the mouse button.

Note, the previous and next control points bound horizontal control point movement. This restriction is necessary to prevent the construction of invalid functions. Attempts to move a control point outside these bounds have no effect. This functionality means users do not need to have sufficient mathematical training to understand what a valid function is; the software prevents them from constructing infeasible paths for the flows.

Control point movement is also bounded by the chart's limits. A point may not be placed outside the chart's bounds. Attempts to move a point outside the charts bounds will move the point to the valid location closest to the desired location.

Visual feedback provided while dragging a control point takes on two forms:

1. The point representation under the mouse cursor indicates the proposed destination
2. Invalid destination regions are filled with a translucent gray color

The point representation under the mouse cursor depends on whether the proposed move is valid. The screenshots below compare the visual feedback:

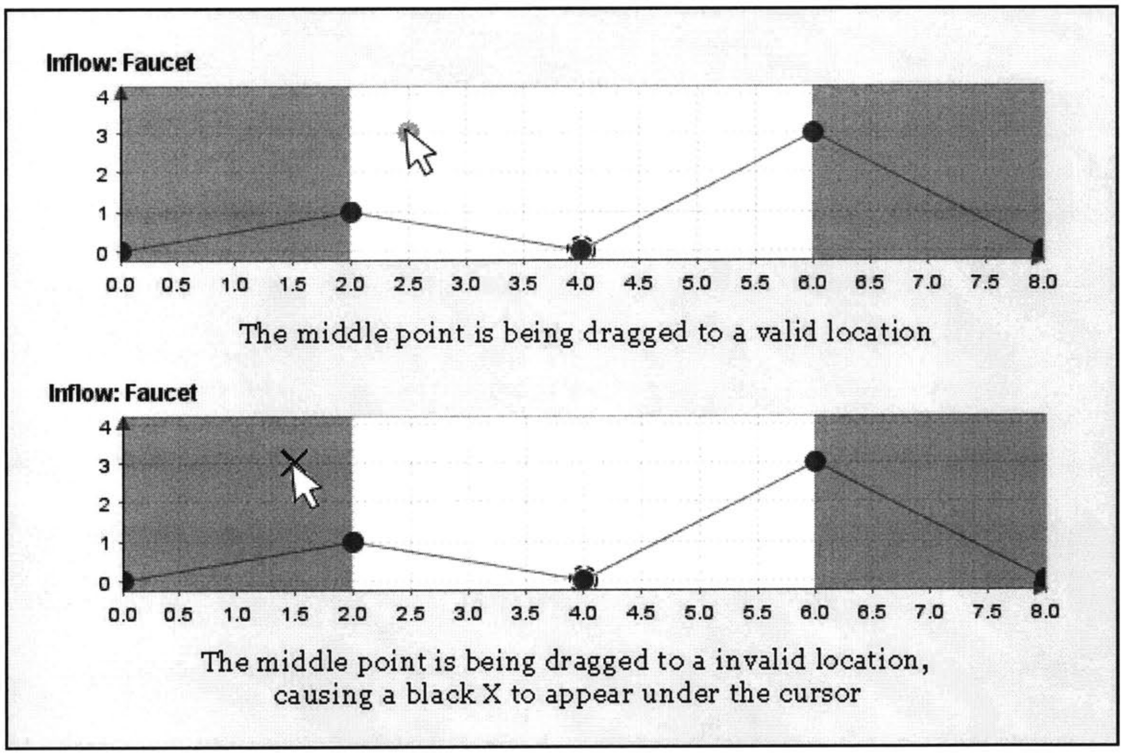


Figure 3: A comparison of visual feedback for valid and invalid control point moves

When a point is dragged into an invalid region, the point under the cursor changes to a black "X" to provide additional visual feedback.

Removing Control Points

Control points are removed by pressing the *Delete* key or *Backspace* key while one or more points are selected.

Selecting Control Points

Clicking on an individual control point will select it. Clicking and dragging a selection box will select all points contained in the box.

Selected control points have a dashed outline.

Undo Support

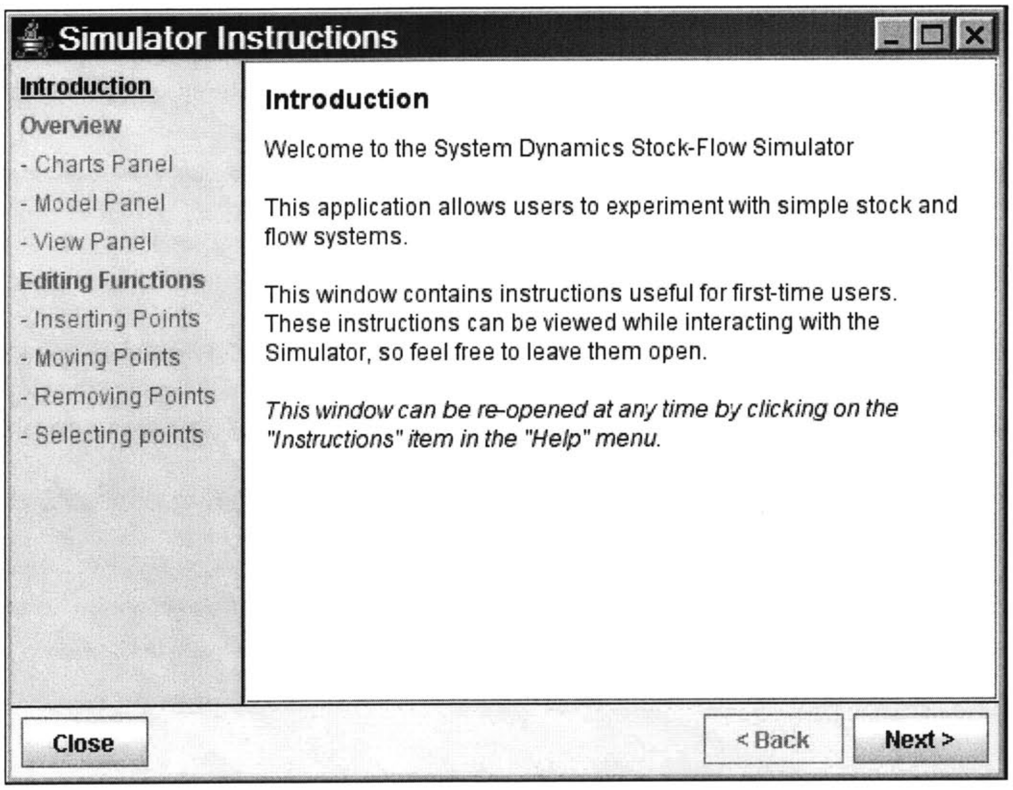
The simulator supports undo and redo operations for all changes to the underlying system. These changes include edits made to the function charts as well as edits made to system parameters such as the initial stock or system time range.

Undo functionality greatly enhances the simulator's usability, encouraging users to explore the interface and experiment with the system.

Instructions

The simulator displays an instructions window on startup. This instruction window is independent of the main window, allowing users to keep it open while using the main interface.

A screenshot showing the instructions window is included below:



Introduction

Welcome to the System Dynamics Stock-Flow Simulator

This application allows users to experiment with simple stock and flow systems.

This window contains instructions useful for first-time users. These instructions can be viewed while interacting with the Simulator, so feel free to leave them open.

This window can be re-opened at any time by clicking on the "Instructions" item in the "Help" menu.

Figure 4: The simulator instructions window

The instructions window contains an overview of the simulator's functionality as well as detailed instructions explaining function editing.

Instruction topics are listed in an outline format in a column on the left. Users may click on instruction topics to jump to information about that topic. Alternately, users may choose to view instruction pages sequentially, using the Next and Back buttons at the bottom of the window.

Limitations and Future Work

Many lower-priority or optional features in the original Simulator design have not yet been implemented. For example, while there are buttons for loading and saving systems to files, they are currently disabled, since this functionality is not yet implemented. The Simulator interface also lacks a mechanism for changing the system metaphor. While the simulator is designed to support multiple metaphors, only the bathtub metaphor is implemented. Much of the code for the bathtub metaphor, though, could be re-used to construct new metaphors, simplifying the process of creating new ones. Examples of metaphors that might be developed for the simulator:

1. Bank account—the balance represents the stock, deposits represent the inflow and withdrawals represent the outflow
2. Department store—the number of people in the store represents the stock, the rate of people entering represents the inflow, the rate of people leaving represents the outflow

Survey Framework

This section begins with a short overview of the survey framework. It continues by outlining the primary requirements guiding the design and implementation of the Survey Framework. It concludes with a brief description of the implementation technologies and a high-level overview of the implementation.

Overview

The Survey Framework facilitates the creation, deployment, and maintenance of online surveys. These surveys collect data as part of online system dynamics experiments.

The diagram below illustrates the relationships between the different users of the framework:

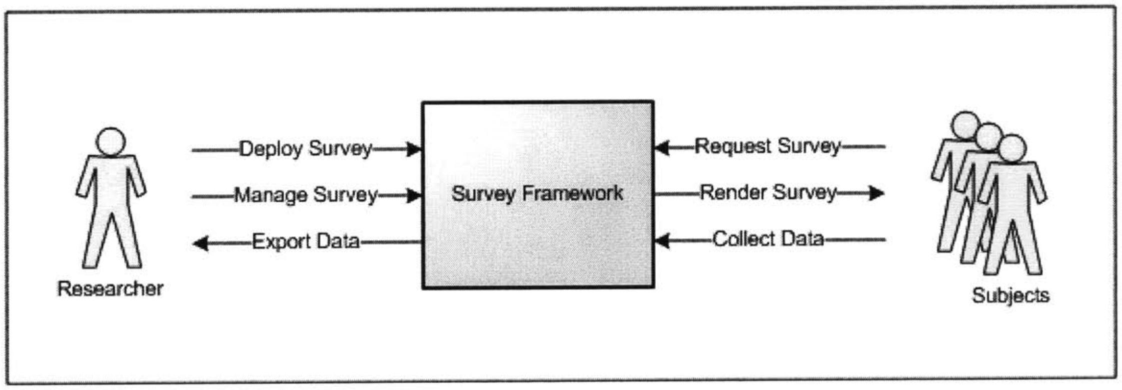


Figure 5: The relationships between different survey framework users

A typical use case for the survey framework is as follows:

1. A researcher creates a survey, defining content for pages and the types of input to collect

2. The researcher uploads a description of the survey to the web server that will host the survey
3. The researcher configures survey properties via a web interface in preparation for data collection
4. Subjects browse to a particular URL on the hosting server that presents the survey to users. Subjects respond to survey questions and responses are persisted to a database.
5. The researcher completes the experiment and asks the survey to export its data for analysis.

Online surveys may also be used as educational tools by disabling data collection and supplying expected responses to survey inputs. These features, however, are not yet supported by the management interface or the survey runtime.

The diagram below summarizes the primary components of an online survey supported by the framework:

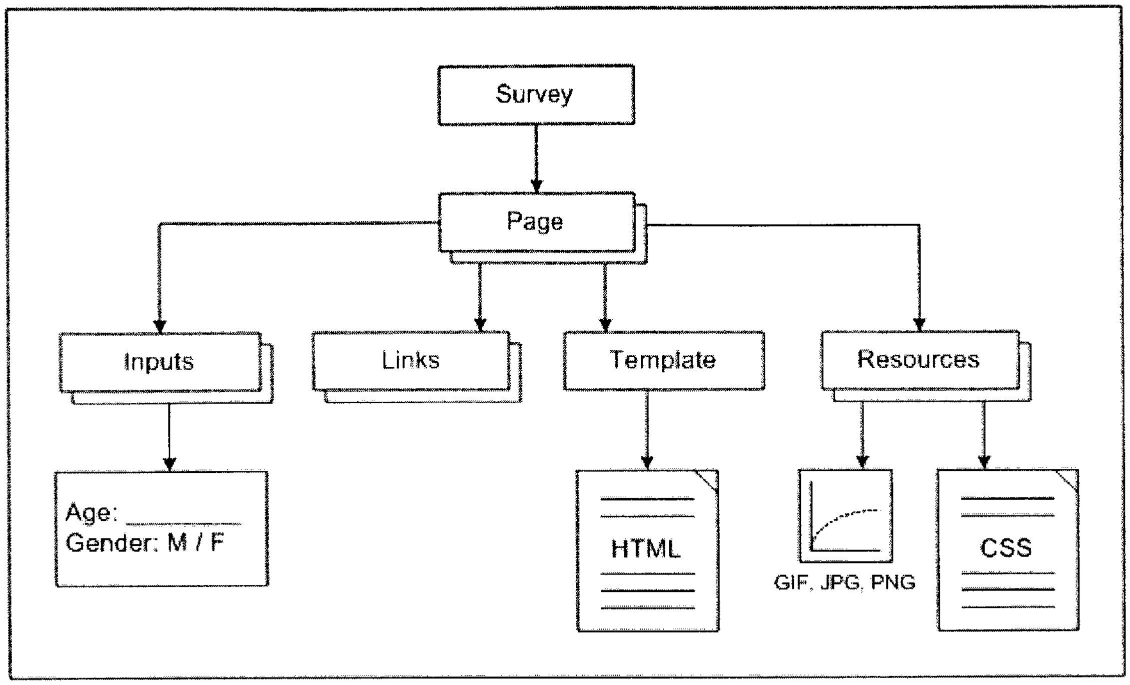


Figure 6: The components of an online survey

A survey is collection of pages with links between them. Each page may contain zero or more inputs that collect responses from users. Each page has an associated HTML template that defines the visual representation of the page. Page templates may require resources such as images or style sheets for proper rendering.

Requirements

The Survey Framework must facilitate the creation, deployment, and maintenance of online surveys.

The framework must support all common web inputs: text, select one of many, select many of many. Additionally, the framework should support collection of graphing responses from users, as systems thinking assessment often involves interpreting graphs.

The framework must provide a web interface for deploying surveys. The web interface should also allow administrators to export survey responses to a format suitable for import into data analysis software such as Microsoft Excel.

Survey developers should be able to restrict access to surveys via passwords. A researcher may wish to limit the number or type of subjects with access to an experiment. This should be enforceable by defining the set of users authorized to view and access the survey.

As a research and educational tool, the framework should be as portable and easy-to-setup as possible.

Framework Operations

At a high-level, the survey framework supports five core operations:

1. Survey creation
2. Survey deployment and management
3. Survey rendering
4. Survey data collection
5. Survey data export

Survey creation allows a user to define a new survey in a format compatible with the survey deployment operation. Newly created surveys may then be deployed and configured via a web interface. Survey rendering creates HTML pages from survey pages and survey inputs. Data collection validates and records subject responses. Data export formats the response data for easy import into data analysis software.

The following sections describe each of these operations in detail.

Survey Creation

A combination of XML, HTML, and images define a survey. Individual survey pages are written in HTML, and a single XML survey descriptor links them together.

The diagram below identifies which survey components are described by the XML descriptor.

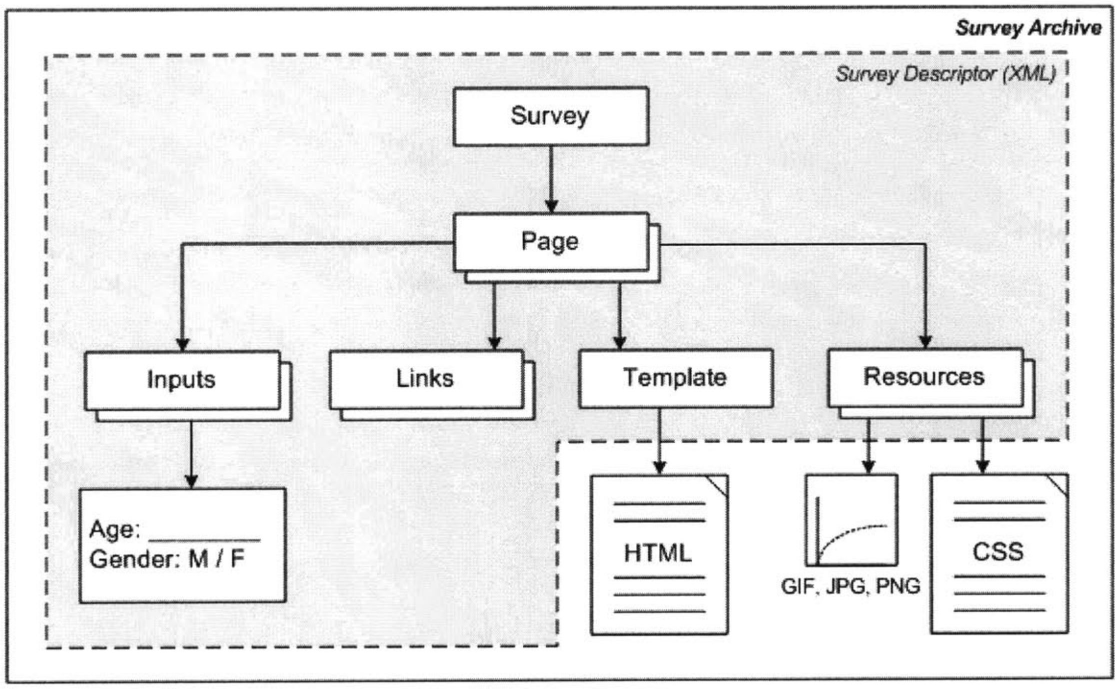


Figure 7: Survey descriptor components vs. survey resources

The remaining components exist as separate resource files for the survey.

The format of the XML descriptor is detailed in the Appendix using an example survey.

No graphical editor exists for creating survey descriptors. The schema is straightforward, however, and it is not difficult to create a descriptor manually for a small or medium-sized survey. Future work should explore creating a graphical editor to obviate the need for manual XML editing.

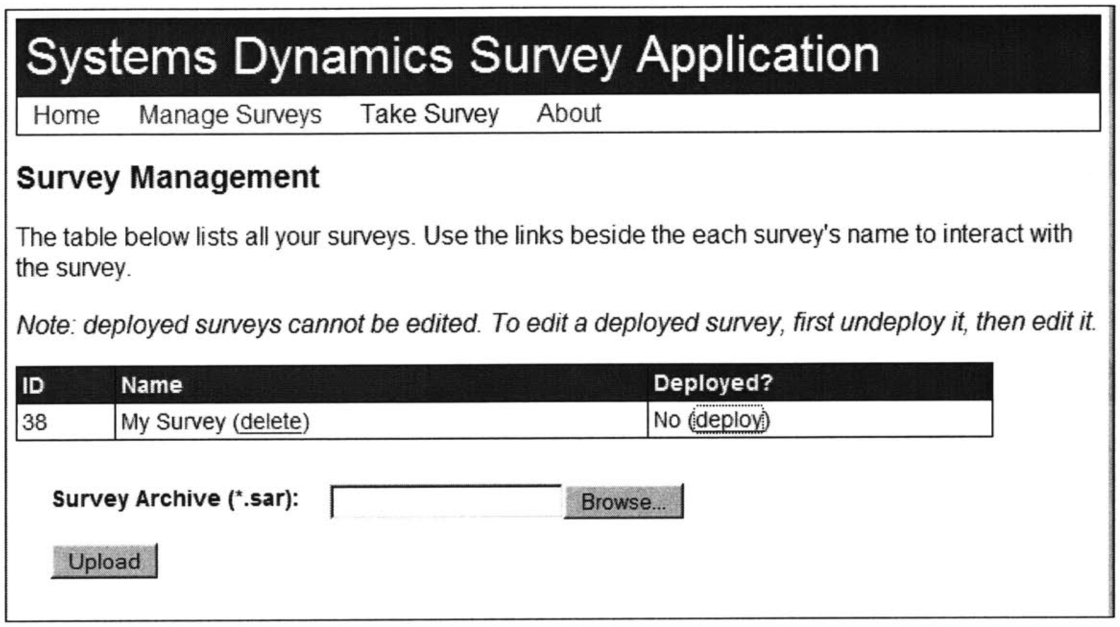
Survey Deployment and Management

A survey is completely defined by its XML descriptor and its resource files. These files are grouped together into a survey archive for deployment. The files are compressed using the ZIP compression algorithm. In this way, survey archives are very similar to Java archives (JAR files). The compression reduces the survey's total size, making survey deployment more efficient. Grouping all the resources into a single file also makes it simpler to upload to the hosting server.

Like the survey descriptor, creation of the survey archive should be automated via a graphical editor. Due to time constraints, an editor was not implemented. Tools to archive multiple files into a ZIP archive, however, are commonly available and straightforward to use.

The survey framework offers a web interface for managing surveys. This web interface is password protected, allowing researchers to limit administrative access.

The main management page displays a list of uploaded surveys along with basic status information. New surveys are uploaded by selecting a survey archive file and pressing the upload button. When a survey is uploaded, the server decompresses the archive, parses the survey descriptor, and validates all required resources are present. If successful, the survey will be added to the list of uploaded surveys; otherwise, an error message explaining the problem will be displayed.



Survey Management

The table below lists all your surveys. Use the links beside the each survey's name to interact with the survey.

Note: deployed surveys cannot be edited. To edit a deployed survey, first undeploy it, then edit it.

ID	Name	Deployed?
38	My Survey (delete)	No (deploy)

Survey Archive (*.sar):

Figure 8: The survey management interface for the survey framework

Surveys may be removed from the server by clicking "delete" links next to the survey's name. The management interface will prompt for confirmation before removing the survey. Removing a survey also removes any stored survey responses associated with that survey.

Clicking on a survey's name lets administrators view and edit survey details. These details include the survey's name, when the survey was last updated, the number of responses received, whether the survey is password protected, whether data collection is enabled, and whether the survey is available. This details page also provides links for exporting survey response data.

Survey Rendering

When the survey framework receives a request for a survey page, the framework uses a combination of services provided by the Click application framework and the Velocity templating language to produce HTML for the client. See the "Technologies" section for more information about Click and Velocity.

The page rendering process is illustrated in the diagram below:

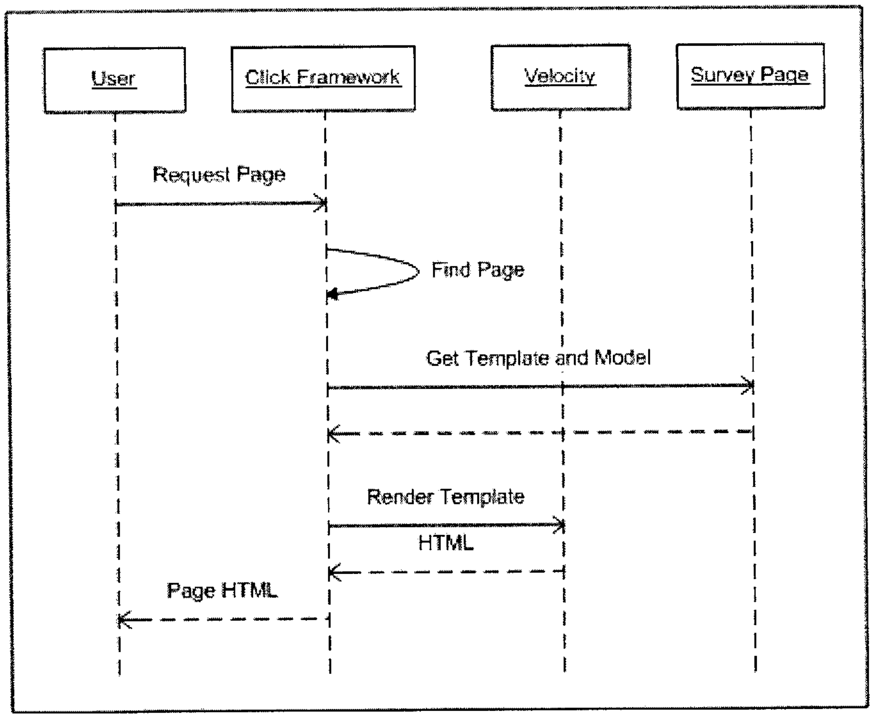


Figure 9: The page rendering process in the survey framework

Each survey page has an associate page template composed of HTML and Velocity. To handle a page request, the survey framework first constructs a model containing data needed by the page template for rendering. This model includes the paths to required resources such as images or style sheets as well as page links and survey inputs. Page links and survey inputs are implemented in the survey framework as special classes capable of rendering themselves to HTML when referenced in a page template.

Once a page's model is constructed, the model and template are passed to the Click application framework that delegates page rendering to Velocity's templating engine. The templating engine replaces variables and processes templating commands using data in the model. The resulting HTML is forwarded to the client requesting the survey page.

See the Appendix for an example survey with page templates and Velocity commands.

Survey Data Collection

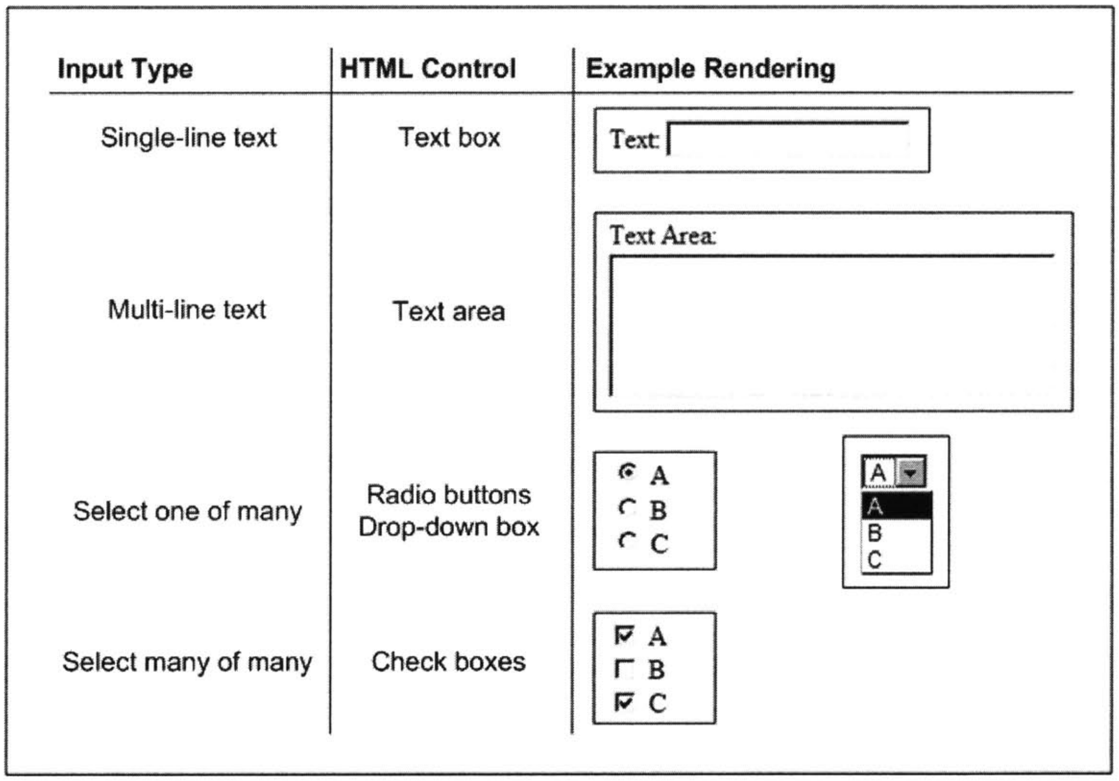
The survey framework defines a common interface for survey input objects, defining the functionality required of all input types.

The most important input requirements are as follows:

1. Inputs must have unique names so they can be uniquely referenced within a page
2. Inputs must be capable of rendering themselves to HTML as form input elements
3. Inputs must be capable of parsing and validating user-submitted values from HTML form submissions
4. Inputs must support importing and exporting values to and from strings to enable uniform data export and data persistence mechanisms
5. Inputs must specify whether they are required
6. Inputs may optional specify default values

The survey framework provides input implementations for all common web input formats (single and multi-line text, select one of many, select many of many) as well as a special graphing input.

Text inputs correspond to single-line and multi-line text boxes in HTML pages. Select-one-of-many inputs map to radio buttons or drop-down boxes in HTML pages. Select-many-of-many inputs map to checkboxes buttons in HTML pages. The figure below summarizes the available input types and their corresponding HTML controls.

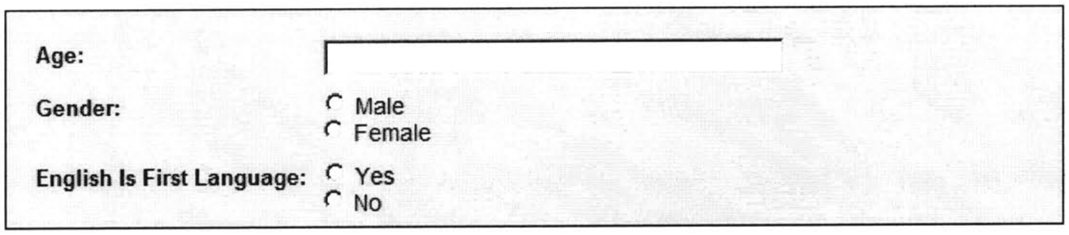


Input Type	HTML Control	Example Rendering
Single-line text	Text box	Text:
Multi-line text	Text area	Text Area:
Select one of many	Radio buttons Drop-down box	● A ○ B ○ C A ▾ A B C
Select many of many	Check boxes	✓ A ☐ B ✓ C

Figure 10: Basic survey framework input types

All of the common web inputs include flexible support for input validation. Text input values may be restricted to numeric or non-numeric data, within a certain range of values, within a fixed set of values, whole or fractional numbers, among other options. Select input values may be restricted to a minimum or maximum number of selections or a subset of available values.

If an input fails validation, the survey framework adds an error message for that input to the page model. The page template can detect this error message and place it near the offending input field to alert the user.

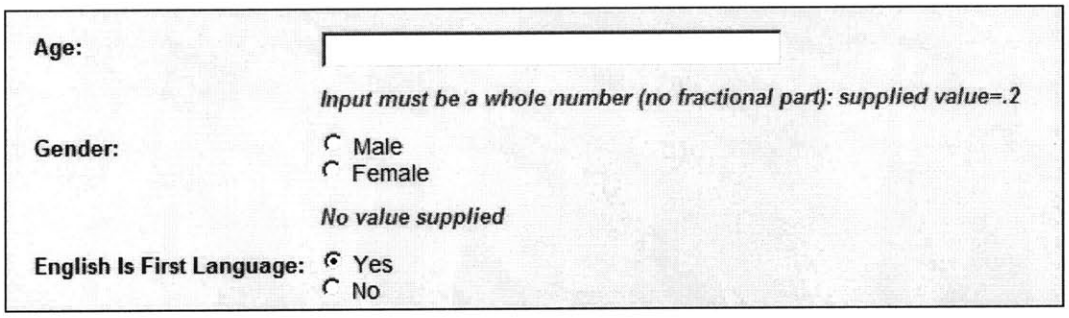


Age:

Gender: ○ Male ○ Female

English Is First Language: ○ Yes ○ No

A survey input form with three fields



Age:

Input must be a whole number (no fractional part): supplied value=.2

Gender: ○ Male ○ Female

No value supplied

English Is First Language: [✓ Yes] ○ No

If an age value of 0.2 is entered and "Yes" is selected,
the form will be returned with error messages

Figure 11: Examples of survey input validation

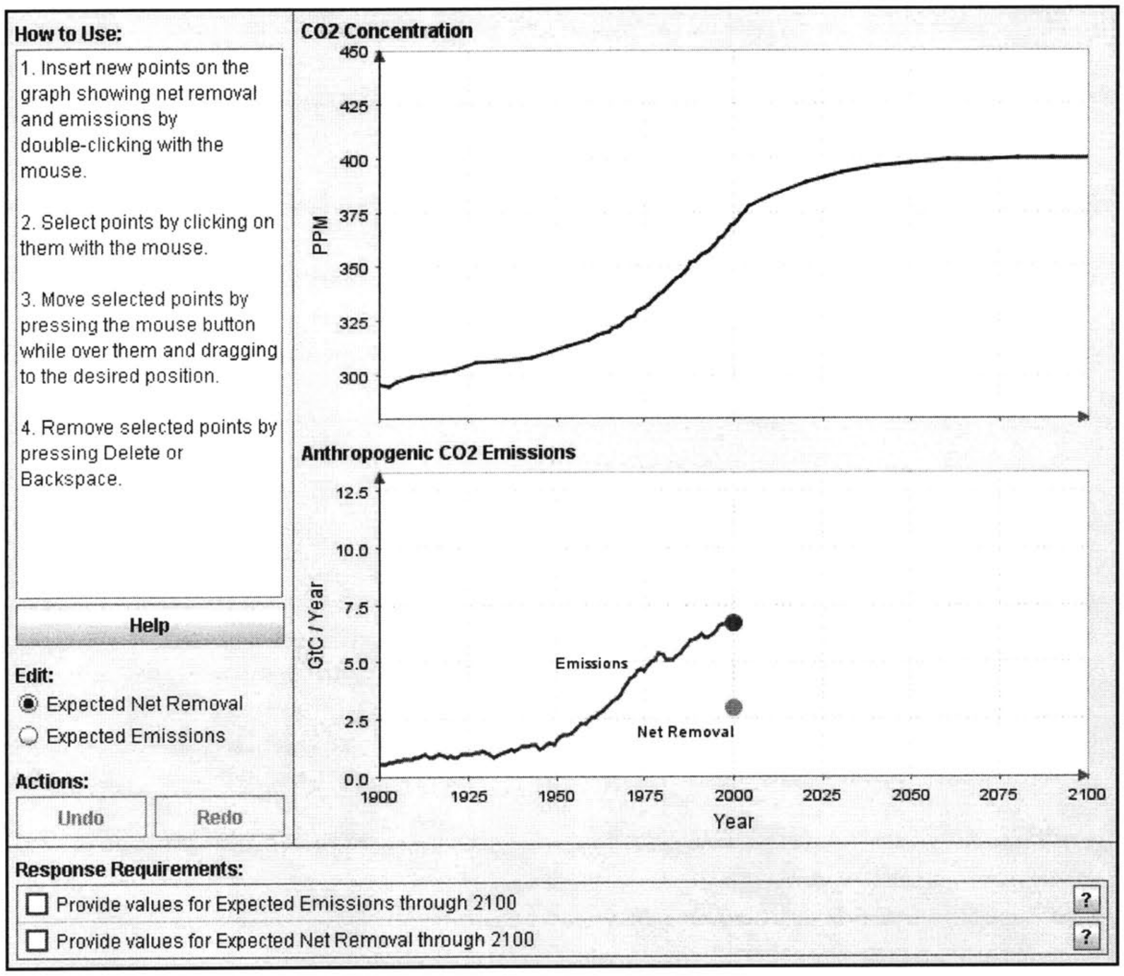
The figure above shows a simple survey form with three fields. The age input fields is configured to require whole numbers in the range 18-99. If a user submits a fractional value (e.g. 0.2), input validation fails. Since the age input is required, the survey framework returns the user to the form with an error message explaining the problem. Validation for the gender input field is similar, though simpler. The gender field is simply required, and by failing to pick an option, the survey framework returns the user to the form with an error message for that field.

Graphing Input

Interpreting graphs is a fundamental system dynamics skill. Experiments that seek to assess system dynamics performance, such as the Greenhouse Gas survey discussed later in this paper, require collection of user responses on graphs.

To serve these needs, the survey framework provides a survey input for collecting graphing responses. In its current state, this input is fairly specialized to meet the needs of the Greenhouse Gas survey. Much of its implementation, however, is independent of the Greenhouse Gas survey, making it straightforward to generalize it to other applications.

The graphing input component appears to a user on a survey page as a Java applet. A screenshot of the graphing input applet is shown below:


How to Use:

1. Insert new points on the graph showing net removal and emissions by double-clicking with the mouse.

2. Select points by clicking on them with the mouse.

3. Move selected points by pressing the mouse button while over them and dragging to the desired position.

4. Remove selected points by pressing Delete or Backspace.

Edit:

[✓ Expected Net Removal]
Expected Emissions

Response Requirements:

☐ Provide values for Expected Emissions through 2100

☐ Provide values for Expected Net Removal through 2100

Figure 12: The survey framework's graphing input applet

The applet contains three main panels:

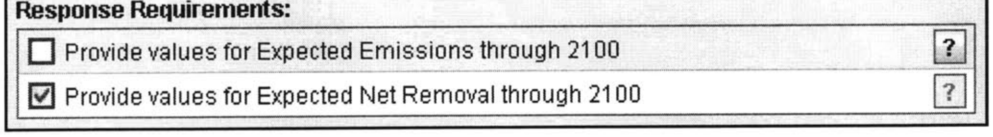
1. Charts panel
2. Sidebar panel
3. Requirements panel

The charts panel presents a user with two graphs. The top graph is the reference graph. The bottom graph is the input graph. The reference graph provides task-specific data for a given experimental condition; it is not editable. The input graph allows the user to plot a response to the task question associated with the input.

In this example, the user is being asked to define future values for two flow functions (CO_2 Emissions and Net CO_2 Removal) to achieve a desired stock function (CO_2 Concentration) shown in the reference graph.

The sidebar panel contains tools to help the user respond. A large textbox gives an overview of the user interface, and the nearby "Help" button brings up additional instructions. Below the instructions are radio buttons allowing the user to select which input function to edit. In this example, there are two functions to edit. Below the radio buttons, are button to undo and redo edits to the input functions. While undo and redo functionality is also available through the standard *Control-Z* and *Control-Y* key strokes, user feedback reported the inclusion of buttons helped make it more apparent this functionality was available.

The requirements panel provides visual feedback about the validity of the current response data. The panel lists the set of requirements the graphing response must satisfy similar to a checklist. Satisfied requirements are checked off; unsatisfied requirements are highlighted in yellow to warn the user that there is still work to be done. The screenshot below shows two requirements. The first one is unsatisfied and the second one is satisfied.

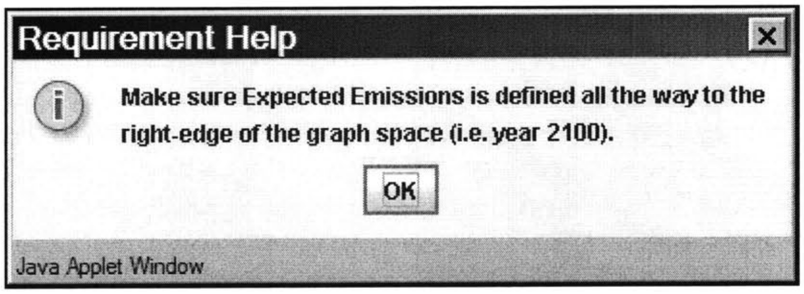

Response Requirements:

☐ Provide values for Expected Emissions through 2100 ?

☑ Provide values for Expected Net Removal through 2100 ?

Figure 13: Comparing satisfied and unsatisfied graphing input requirements

Each requirement also includes a help button that pops a dialog box explaining the requirement in more detail. The screenshot below shows an example of one of these help dialogs:


Requirement Help

Make sure Expected Emissions is defined all the way to the right-edge of the graph space (i.e. year 2100).

Figure 14: A graphing input requirement help dialog box

Initial graphing input implementations did not include this requirements panel. If users did not supply valid responses, the survey framework would return to the task page and simply print an error message explaining why the input was invalid. In a situation like the example, where there is more than one input requirement, a user trying to fix only what was mentioned in the error messages might have to re-submit the page multiple times to learn about all the input requirements. This piecemeal approach leads to a frustrating user experience.

The requirements panel addresses these issues by presenting all input requirements together. It is clear when requirements are satisfied and additional help is available if users are confused.

Survey Data Export

Survey response data is exported via the survey management interface. Users with administrative privileges may select a survey from the main management page to view details about that survey. On the details page, there are options for selecting one of the following export formats:

1. HTML tables
2. Comma-Separated Value (CSV)
3. Tab-delimited

Once an output format is selected, clicking the export button will export all survey responses on a new page. This page may then be saved locally for later analysis.

Unfortunately, this export mechanism does not offer much security. While the survey management interface is password protected, the subject response data is sent unencrypted from the server to the user requesting the data.

Encrypting the management interface connection via SSL is the obvious solution to this problem. Almost all online stores have this feature, allowing customers to submit credit card information securely. Due to time constraints, however, this feature was not implemented.

Technologies

The survey framework is implemented as a web application using Java. Java provides a wide range of deployment options. Popular open source web servers such as Apache support it, and there are a variety of open source Java servlet containers for all common operating systems.

The web application is implemented using the open source Click framework. This framework simplifies web application development by abstracting away many of the details of web requests and responses. The framework encourages object-oriented web development, providing page-level abstractions for web requests and responses. The framework was selected for its simplicity and ease of use. It requires minimal configuration and has a clean programming interface.

Survey data is persisted to a database. A PostgreSQL database was used during development and testing, though the application does not require it. Any JDBC-compliant database (e.g. Microsoft SQL Server, MySQL, Oracle) should be supported. The web application accomplishes this flexibility using the Hibernate persistence framework. This framework manages storage and retrieval of data in an object-oriented fashion. It is the de facto open source framework for database persistence in Java.

Surveys are defined through a combination of XML, HTML, and Velocity. A combination of HTML and Velocity is used to define pages. XML is used to tie all the pages together and configure the survey prior to deployment.

Velocity is a popular templating language for Java web programming. These templating languages allow developers to define page templates, combinations of HTML and scripting. The templating language renders page templates given a page model that maps variables to values. The survey framework makes extensive use of templating to render survey pages.

Ant build scripts automate the compilation and deployment of the survey framework. Ant is a build scripting language for Java. The use of build scripts ensures a consistent and automated build process across supported platforms.

Limitations and Future Work

While the current survey framework satisfies many of the original design goals, there is still plenty of work to be done.

Framework usability is one particular area that could use additional work. Installing the survey framework is not as easy as it could be. A graphical installer program with clear instructions for various hardware and software platforms would help encourage use by researchers and educators. Manually creating surveys from XML and HTML requires a thorough knowledge of these languages, creating an unnecessary barrier to entry. A graphical survey editor could make it easier to develop survey for users with less HTML experience.

Insecure data collection and data export may also be a significant limitation. In situations where sensitive subject data is being collected, research policy may require this information be secured over encrypted connections, preventing use of this framework. Future work to support secure connections via SSL should be a high priority.

Greenhouse Gas Survey

The Greenhouse Gas (GHG) survey is an online version of an existing survey developed by Booth Sweeney and Sterman to assess systems thinking skills (2000). This online version served as a proof-of-concept for the survey framework, helping to identify required features and testing the implementation. There are also plans to use this survey to compare subject performance in the online version with the paper version, and to evaluate the effectiveness of different system dynamics teaching methods.

This section provides a brief overview of the GHG survey, explains issues encountered while developing the survey, and discusses plans for using the survey to improve system dynamics education.

Overview

The Greenhouse Gas (GHG) survey presents subjects with a brief, non-technical background of global warming, indicating a direct relation between atmospheric CO_2 concentration and average global temperature. Subjects are then given a graph showing atmospheric CO_2 concentration for the last 100 years and the next 100 years. An example CO_2 concentration scenario is shown below:

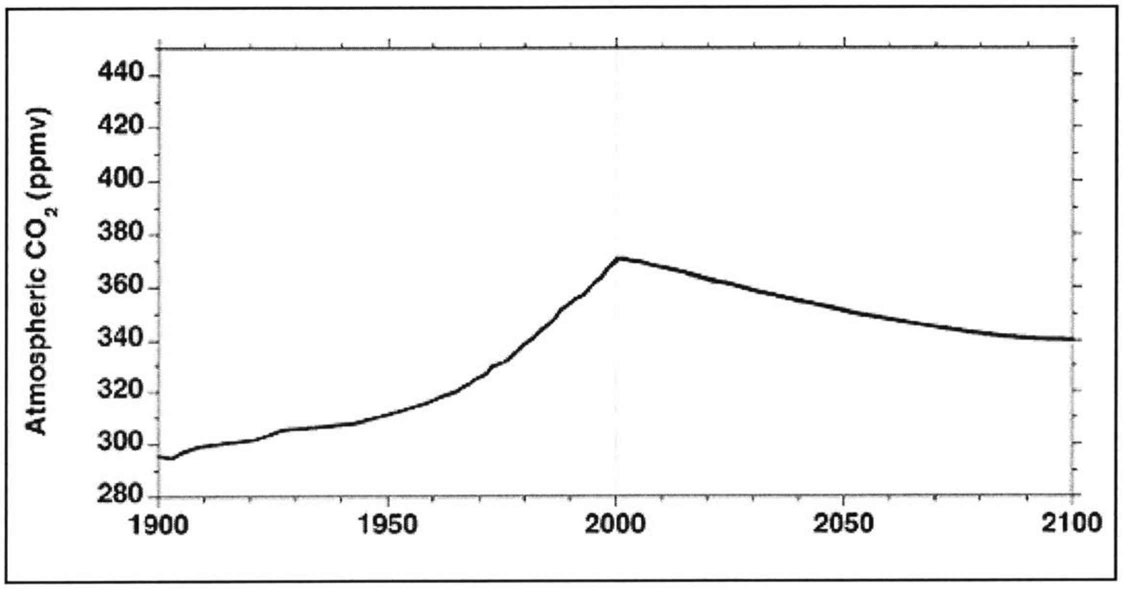


Figure 15: An example CO_2 concentration scenario for the GHG survey

Given this scenario, subjects are asked to respond to three questions whose format depends on the experimental condition:

1. Given CO_2 emissions and net CO_2 removal rates for the past 100 years, estimate the rates for the next 100 years required to produce the atmospheric CO_2 concentrations presented above
2. Given the atmospheric CO_2 concentrations presented above, what would happen to average global temperature?
3. Briefly explain your responses

There are two CO_2 concentration scenarios each containing the same data for the last 100 years. In the first scenario, atmospheric CO_2 concentrations rise by about 8% over the next 100 years to 400ppm. In the second scenario, atmospheric CO_2 concentrations fall by about 8% over the next 100 years to 340ppm.

There are two formats for the first question as well. In the first format, subjects are asked to select a single response from multiple choices that describe expected rates using words. In the second format, subjects are presented with partially completed graphs for the emissions and net removal rates. Subjects must finish drawing the rate functions for the next 100 years according to the given CO_2 concentration scenario.

The online survey's task background and task questions are the same as those in the original paper survey. The only difference is in the graphing response question. The online survey implements a custom Java applet to collect this response. This applet is described in the Survey Data Collection subsection within the Survey Framework section.

See the Appendix for screenshots of online GHG survey pages.

Using the Survey Framework

The online version of the GHG survey successfully leverages available framework features such as required inputs, input validation, and data export to streamline the process of collecting survey responses.

To collect data using the online GHG survey, a researcher may conduct either a public or a private experiment. A public experiment could be conducted by publishing a link to the survey's first page. The subject set could be restricted using survey password protection. Using a computer lab is probably the easiest way to conduct a private experiment. The researcher could set up some number of computers with web browsers pointed at a private survey address. Clearly, there is a trade-off between researcher control and ease of enlisting a large subject set. A publicly published survey link would likely attract significantly more subject responses at the cost of researcher control over who responds and under what conditions.

In summary, the survey framework offers the following advantages to the online GHG survey:

- Automatically and randomly assigns subjects to conditions
- Validates inputs, showing detailed errors under fields with problems
- Data export automatically tabulates subject responses in a format compatible with data analysis software

Converting the survey to an online version, though, required an initial time investment by a developer with experience writing HTML and XML. The survey framework also suffers from some limitations as detailed in the Survey Framework section. In particular, the lack of a secure (encrypted) mechanism for transferring subject responses to and from the survey server may limit deployment opportunities.

Next Steps

Two experiments are currently planned to use the online GHG survey. The first experiment will replicate the climate change experiment by Booth Sweeney and Sterman to compare subject performance in the online survey with the paper survey. Subject performance is expected to be similar to the results of the paper survey.

A second experiment will add an independent, pre-task variable to the first experiment. Subjects will be assigned to one of three pre-task conditions:

1. No pre-task information

2. Non-interactive pre-task information
3. Interactive pre-task information

The first condition will result in the same setup as the first experiment. Subjects are simply directed to the task questions.

The second and third conditions will direct subjects to a pre-task page containing instructional material explaining stocks and flows. The second condition explains these concepts using text and static images. The third condition contains everything in the second condition as well as the stock-flow simulator embedded into the page as an applet. In the third condition, users are encouraged to experiment with the simulator prior to continuing to the task questions.

This second experiment will compare subject performance across the three pre-task conditions. Using the first condition as a reference, we should be able to study how changes to the pre-task instructional material affect subject performance. These studies will hopefully lead to insights into how system dynamics teaching may be improved.

Conclusion

This paper described the development of a two new tools for improving system dynamics education: a stock-flow simulator that allows users to experiment with simple stock-flow systems, and a framework for building online system dynamics surveys. The simulator helps students and teachers visualize interactions in simple stock-flow systems. The survey framework facilitates the construction of online surveys that are more efficient to conduct than traditional paper surveys.

The survey framework was used to build an online version of an existing survey developed to assess systems thinking skills. This online survey demonstrates the capabilities of the survey framework.

Both the simulator and the survey framework have plenty of opportunities for future work. Formal usability testing and interviews with student and teacher users will likely yield valuable insights into how the simulator may be improved. The survey framework would benefit from a graphical editor for survey construction to eliminate the need for a specialized survey developer with experience in HTML and XML. Experiments need to be conducted using the Greenhouse Gas survey to compare subject performance in online surveys versus paper surveys. With this research complete, the GHG survey provides a flexible platform for comparing the effectiveness of system dynamics teaching methods.

References

Booth Sweeney, L. and J. D. Sterman (2000). "Bathtub Dynamics: Initial Results of a Systems Thinking Inventory." System Dynamics Review 16(4): 249-294.

Forio Business Simulations. 26 May 2006. <http://www.forio.com>.

iseesystems. Developer of STELLA and iThink. 26 May 2006. <http://www.iseesystems.com>.

Logo Computer Systems, Inc. Developer of Microworlds Creator and S^4. 26 May 2006. <http://www.microworlds.com>.

Management Simulations, Inc. 26 May 2006. <http://www.capsim.com>.

QuestionPro (2004). 23 November 2005. <http://www.questionpro.com>.

Sastry, M. A. and J. D. Sterman (1992). "Desert Island Dynamics: An Annotated Survey of the Essential System Dynamics Literature." 26 May 2006. System Dynamics Group, MIT. <http://web.mit.edu/jsterman/www/DID.html>.

Sterman, J.D. People Express Management Flight Simulator. 26 May 2006. <http://web.mit.edu/jsterman/www/SDG/MFS/PE.html>.

Sterman, J.D. October 1992. Teaching Takes Off: Flight Simulators for Management Education, "The Beer Game". Sloan School of Management, MIT. 26 May 2006. <http://web.mit.edu/jsterman/www/SDG/beergame.html>.

Sterman, J. and L. Booth Sweeney (Working paper). "Adults' Mental Models of Climate Change Violate Conservation of Matter." Forthcoming, Climatic Change.

Sterman, J. D. and L. Booth Sweeney (2002). "Cloudy Skies: Assessing Public Understanding of Global Warming." System Dynamics Review 18(2): 207-240.

Surveyz! The Ultimate Survey Software Solution (2005). 22 November 2005. <http://www.surveyz.com>.

Ventana Systems, Inc. Developer of Vensim. 26 May 2006. <http://www.vensim.com>.

WebSurveyor. 26 May 2006. <http://www.websurveyor.com>.

Zoomerang. 26 May 2006. <http://info.zoomerang.com>.

Appendix

Technologies

The following is a list of technologies used to develop the Stock-Flow Simulator and Survey Framework:

- Apache Tomcat. <http://tomcat.apache.org>.
- Click. <http://click.sourceforge.net>.
- Extensible Markup Language (XML). <http://www.w3.org/XML>.
- Hibernate. <http://www.hibernate.org>.
- HTML 4.01 Specification. <http://www.w3.org/TR/html4>.
- JFreeChart. <http://www.jfree.org/jfreechart>.
- Java Platform, Standard Edition. <http://java.sun.com/javase>.
- PostgreSQL. <http://www.postgresql.org>.
- Velocity. <http://jakarta.apache.org/velocity>.

Example Survey

This section walks through the construction of a trivial example survey using the survey framework. The example survey tries to answer the question:

> *Does option ordering matter when people are asked to pick their favorite color from a set of options?*

Subjects in the example survey will be randomly assigned to one of two scenarios. The first scenario will present colors in the order red, orange, yellow, green, blue, purple. The second scenario will present colors in the opposite order (purple, blue, green, yellow, orange, red). In each scenario, subjects will be asked to select their favorite color.

Constructing this survey will require two classes of data:

1. An XML survey descriptor
2. HTML describing the survey pages

The following sections explore these classes of data in more detail.

Survey Descriptor

All XML survey descriptors contain the following structure:

```
<?xml version="1.0" encoding="UTF-8"?>

<survey id="...">
    <name>...</name>
    <description>...</description>

    <resources>
        <!-- INSERT RESOURCE DECLARATIONS HERE -->
    </resources>

    <pages start-page="...">
        <!-- INSERT RESOURCE DECLARATIONS HERE -->
    </pages>

    <scenarios assignent="random">
        <!-- INSERT SCENARIO DECLARATIONS HERE -->
    </scenarios>

    <page-groups>
        <!-- INSERT PAGE GROUP DECLARATIONS HERE -->
    </page-groups>

    <page-links>
        <!-- INSERT PAGE LINK DECLARATIONS HERE -->
    </page-links>
</survey>
```

Figure 16: An overview of an example survey descriptor

The "id" attribute of the root "survey" tag must be a non-empty string unique among all surveys. This identifier tells the survey framework whether an uploaded survey represents a new survey or a new version of an existing survey.

The "name" and "description" tags are self-explanatory.

The remaining tags are discussed in separate sections below.

Resources

Resources are files such as images or style sheets needed during page template rendering. Each resource must be assigned a unique name and declared between the "resources" tags as shown below.

```
<resources>
    <resource name="styles.common" path="common-styles.css" />
    <resource name="styles.forms" path="form-styles.css" />
    <resource name="styles.tables" path="table-styles.css" />
</resources>

```

Figure 17: Resource tags in an example survey descriptor

When a page template is rendered, it may reference the *$resources* variable in the page model. This variable contains a table mapping resources names to resources paths. The "Shared Template Header" section below shows the use of resources in page templates.

Pages

Each page in a survey must have be declared via page tags in the survey descriptor. Each page is must have a unique name and an associated page template.

A page may have zero or more inputs. There are input tags for each supported input type (single line text, multi-line text, select-one-of-many, select-many-of-many, graphing). A full specification of all input tags is beyond the scope of this document. The example below shows a select-one-of-many input.

```
<pages start-page="welcome">
    <page name="welcome" path="welcome.htm">
        <inputs />
    </page>
    <page name="consent" path="consent.htm">
        <inputs />
    </page>
    <page name="task (red->purple)" path="task.htm">
        <inputs>
            <choose name="question1" label="Question 1" required="true">
                <description>Pick favorite color (red->purple)</description>
                <options>
                    <option label="Red" />
                    <option label="Orange" />
                    <option label="Yellow" />
                    <option label="Green" />
                    <option label="Blue" />
                    <option label="Purple" />
                </options>
                <param name="validate.min-selected" value="1" />
                <param name="validate.max-selected" value="1" />
            </choose>
        </inputs>
    </page>
    <page name="task (purple->red)" path="task.htm">
        <inputs>
            <choose name="question1" label="Question 1" required="true">
                <description>Pick favorite color (purple->red)</description>
                <options>
                    <option label="Purple" />
                    <option label="Blue" />
                    <option label="Green" />
                    <option label="Yellow" />
                    <option label="Orange" />
                    <option label="Red" />
                </options>
                <param name="validate.min-selected" value="1" />
                <param name="validate.max-selected" value="1" />
            </choose>
        </inputs>
    </page>
    <page name="debrief" path="debrief.htm">
        <inputs />
    </page>
</pages>
```

Figure 18: Page tags in an example survey descriptor

One page must be designated as the starting page. This is the page users are shown when accessing the survey for the first time. In the example above, users are directed to a welcome page.

Scenarios

Scenarios correspond to experimental conditions. Each scenario must have a unique name. Scenarios contain one or more pages. Each scenario defines a path through the survey page graph.

```
<scenarios assignent="random">
    <scenario name="red->purple">
        <page name="task (red->purple)" />
    </scenario>
    <scenario name="purple->red">
        <page name="task (purple->red)" />
    </scenario>
</scenarios>

```

Figure 19: Scenario tags in an example survey descriptor

The example survey has two scenarios. In the first scenario, users are sent to a task page showing colors ordered from red to purple. In the second scenario, users are sent to a task page showing colors ordered from purple to red.

Page Groups

Page groups define named collections of pages. An example page group in created in the XML below.

```
<page-groups>
    <page-group name="my group">
        <page name="welcome" />
        <page name="consent" />
        <page name="debrief" />
    </page-group>
</page-groups>

```

Figure 20: Page group tags in an example survey descriptor

Page groups are an optional feature in the survey descriptor. While the example above creates a group, it is not used in the rest of the descriptor. Groups are useful when generating one-to-many or many-to-one page navigation structures. For example, most surveys have a final debrief or survey-complete page after subjects finish submitting responses. A page group could be defined containing all pages that lead to the final page. Only a single page link declaration would be required to link all pages in the page group to the final page. Without page groups, a page link declaration would be required for each page that linked to the final page.

Page Links

Page links support navigation between survey pages with side effects. A page link has a label that is displayed when the link is rendered. A link is defined from a page to another page.

Page links may have side effects related to data collection. In the example below, when a user accepts the consent form, the "begin" task action is executed. This tells the survey to assign the user to a scenario and redirect to the appropriate task page. If a user quits the survey from a task page, the "abort" task action discards the users responses. If a user submits their responses, the "commit" task action and "record-inputs" attributes tell the survey to validate and record the subject's responses to the survey data set.

```
<page-links>
    <page-link text="Take the survey!">
        <from page="welcome" />
        <to page="consent" />
    </page-link>

    <page-link text="I accept">
        <from page="consent" />
        <to page="" task-action="begin" />
    </page-link>

    <page-link text="I decline">
        <from page="consent" />
        <to page="welcome" />
    </page-link>

    <page-link text="I'm finished" task-action="commit" record-inputs="true">
        <from page-group="scenarios" />
        <to page="debrief" />
    </page-link>

    <page-link text="Quit" task-action="abort">
        <from page-group="scenarios" />
        <to page="welcome" />
    </page-link>
</page-links>

```

Figure 21: Page link tags in an example survey descriptor

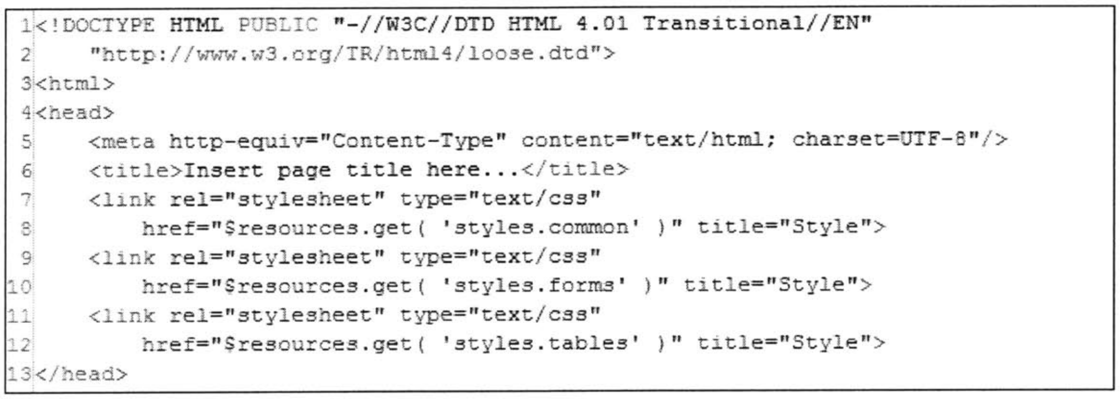
Pages may also be connected using traditional HTML links (e.g. "<a href="..." />). Page links in the survey descriptor are only necessary to for scenario assignment and data collection.

Page Templates and Rendered Output

Page templates define the visual appearance of survey pages using a combination of HTML and Velocity. Velocity commands are prefixed with the '#' symbol. Velocity references to the page model begin with the '$' symbol.

Shared Template Header

All page templates in this example survey have a common header. This header is presented once below to conserve space. All other page templates contain comments in the header's place.



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
    <title>Insert page title here...</title>
    <link rel="stylesheet" type="text/css"
        href="$resources.get( 'styles.common' )" title="Style">
    <link rel="stylesheet" type="text/css"
        href="$resources.get( 'styles.forms' )" title="Style">
    <link rel="stylesheet" type="text/css"
        href="$resources.get( 'styles.tables' )" title="Style">
</head>
```

Figure 22: The header shared by all example survey page templates

The header shows the use of survey resources for importing style sheets. Resources are accessed from the page model's *resources* variable using the names defined in the survey descriptor.

Shared Input and Link Form

All page templates in this example survey also share an input and link form. This form renders page links, survey inputs, and input error messages (if present). This form is presented once below to conserve space. All other page templates contain comments in the form's place.

```
<div class="form_container">
$form_begin <!-- <form ...> -->
    <div class="form_input_set">
        <table border="0" cellpadding="2" cellspacing="10" width="100%">
            #foreach ($input in $inputs)
                <tr>
                    <td align="left" valign="top" nowrap="nowrap">
                        #if ($input.isRequired)
                        <label class="form_field_label_required"
                            for="$input.name">${input.label}:</label>
                        #else
                        <label class="form_field_label"
                            for="$input.name">${input.label}:</label>
                        #end
                    </td>

                    <td align="left" valign="top" width="99%">
                        <span class="form_field">
                            $input.renderToHTML()
                        </span>
                    </td>
                </tr>

                #if ($input.lastError)
                <tr>
                    <td>&nbsp;</td>
                    <td class="form_field_error" align="left"
                        valign="top">$input.lastError</td>
                </tr>
                #end
            #end
            <tr>
                <td colspan="2" align="left">
                    #foreach ($link in $links)
                        $link &nbsp;
                    #end
                </td>
            </tr>
        </table>
    </div>
$form_end <!-- </form> -->
</div>
```

Figure 23: The input and link form shared by all example survey page templates

Welcome Page

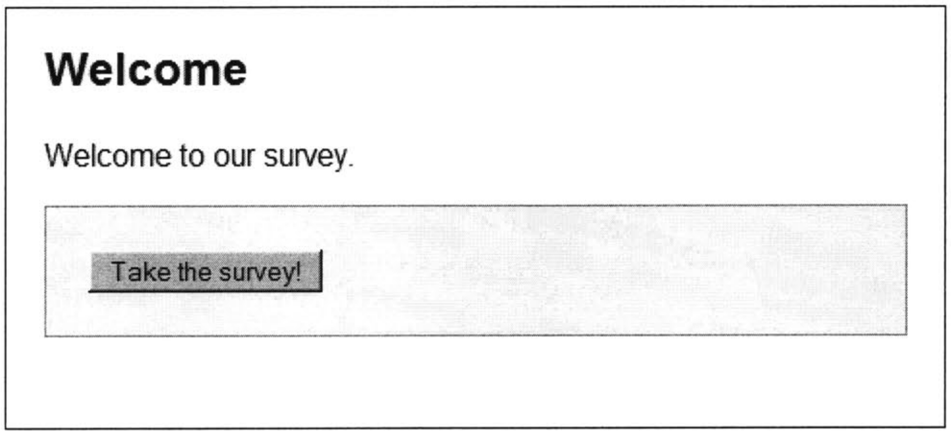
The welcome page is the first page in the example survey. This page greets users and presents a link to the consent page for users that want to take the survey.

The figure below shows the welcome page template.

```
<!-- INSERT TEMPLATE HEADER HERE... -->
<body>
    <h2>Welcome</h2>
    <p>Welcome to the color survey.</p>

    <!-- INSERT INPUT AND LINK FORM HERE... -->

</body>
</html>

```

Figure 24: The page template for the example survey's welcome page

The figure below shows the welcome page rendered is a user's web browser. Notice how the page link specified in the survey descriptor is rendered as a button.



Welcome

Welcome to our survey.

Figure 25: The example survey's welcome page as seen by a user

Consent Page

The consent page is the second page in the example survey. This page presents the user with terms of consent prior to collecting data.

The figure below shows the consent page template.

```
<!-- INSERT TEMPLATE HEADER HERE... -->
<body>
    <h2>Consent</h2>
    <p><i>Insert request for consent here...</i></p>

    <!-- INSERT INPUT AND LINK FORM HERE... -->

</body>
</html>

```

Figure 26: The page template for the example survey's consent page

The figure below shows the consent page rendered in a user's web browser.

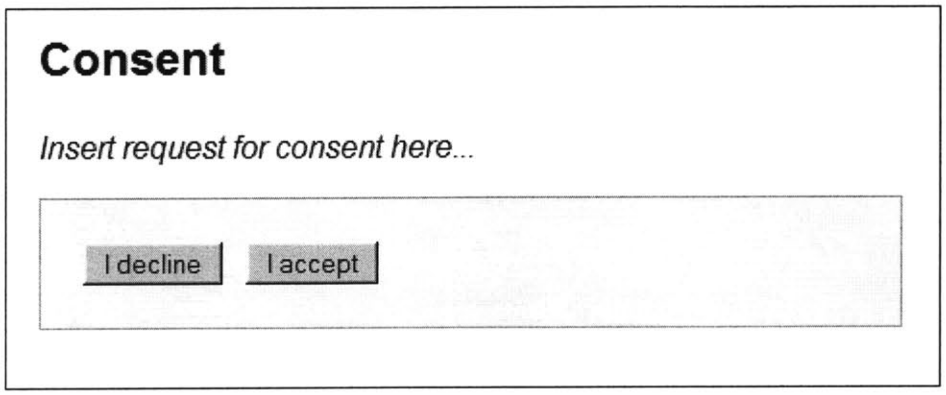
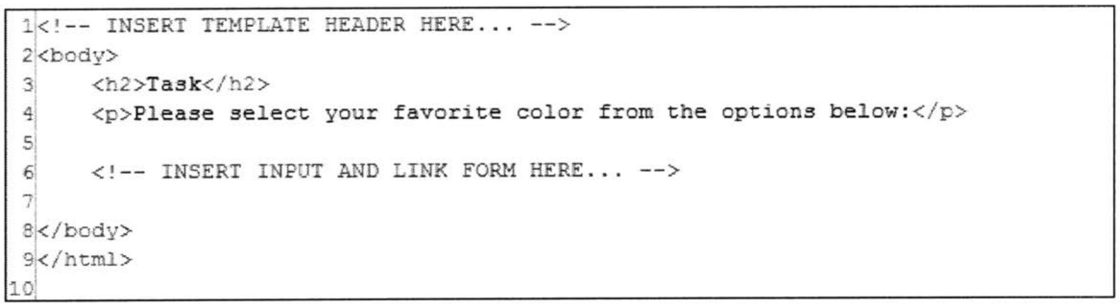


Figure 27: The example survey's consent page as seen by a user

Task Page

The task page is the third page in the example survey. There are actually two task pages, one for each scenario. Both scenarios, however, share the same page template. The only difference is the ordering of the input options. This difference is captured in the survey descriptor, independent of the page template.

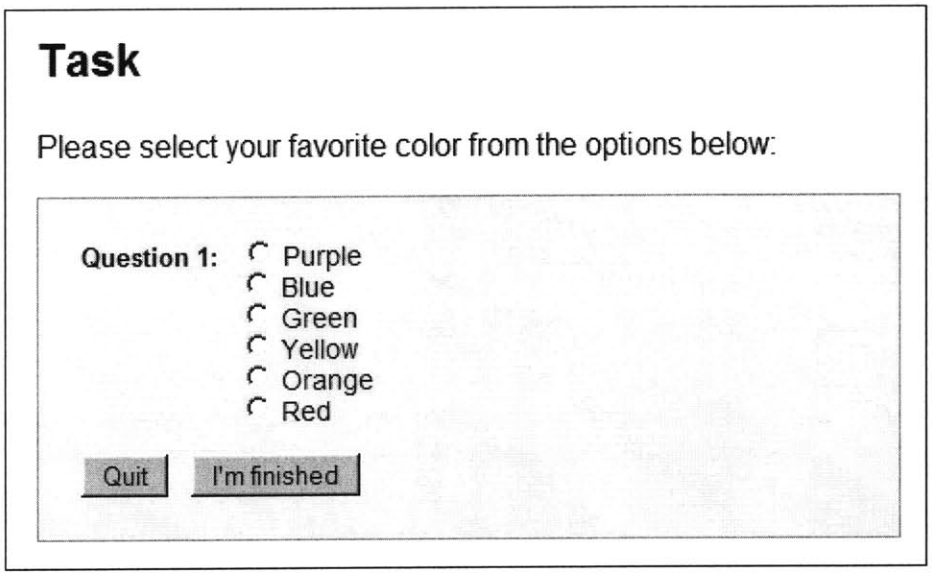
The figure below shows the task page template.



```
<!-- INSERT TEMPLATE HEADER HERE... -->
<body>
    <h2>Task</h2>
    <p>Please select your favorite color from the options below:</p>

    <!-- INSERT INPUT AND LINK FORM HERE... -->

</body>
</html>

```

Figure 28: The page template for the example survey's task page

The figure below shows the task page rendered in a user's web browser.


Task

Please select your favorite color from the options below:

Question 1:

- Purple
- Blue
- Green
- Yellow
- Orange
- Red

Figure 29: The example survey's task page as seen by a user

Debrief Page

The debrief page is the final page in the example survey.

The figure below shows the debrief page template.

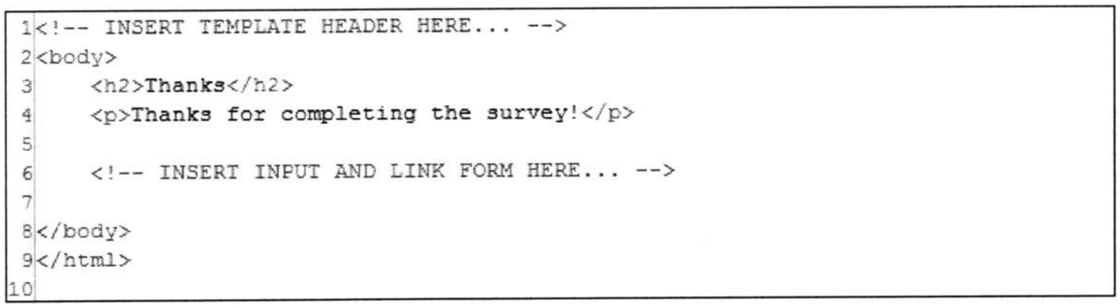
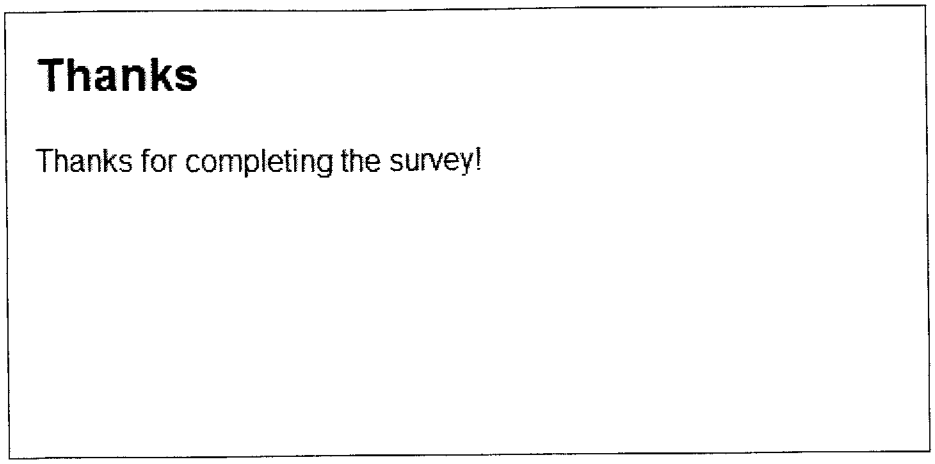

```
<!-- INSERT TEMPLATE HEADER HERE... -->
<body>
    <h2>Thanks</h2>
    <p>Thanks for completing the survey!</p>

    <!-- INSERT INPUT AND LINK FORM HERE... -->

</body>
</html>

```

Figure 30: The page template for the example survey's debrief page

The figure below shows the debrief page rendered in a user's web browser.



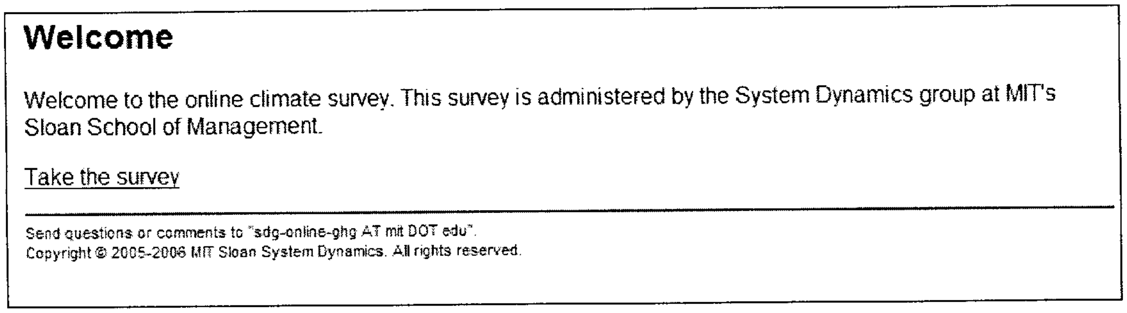
Thanks

Thanks for completing the survey!

Figure 31: The example survey's debrief page as seen by a user

GHG Survey Screenshots

This section includes screenshots of the entire Greenhouse Gas survey.



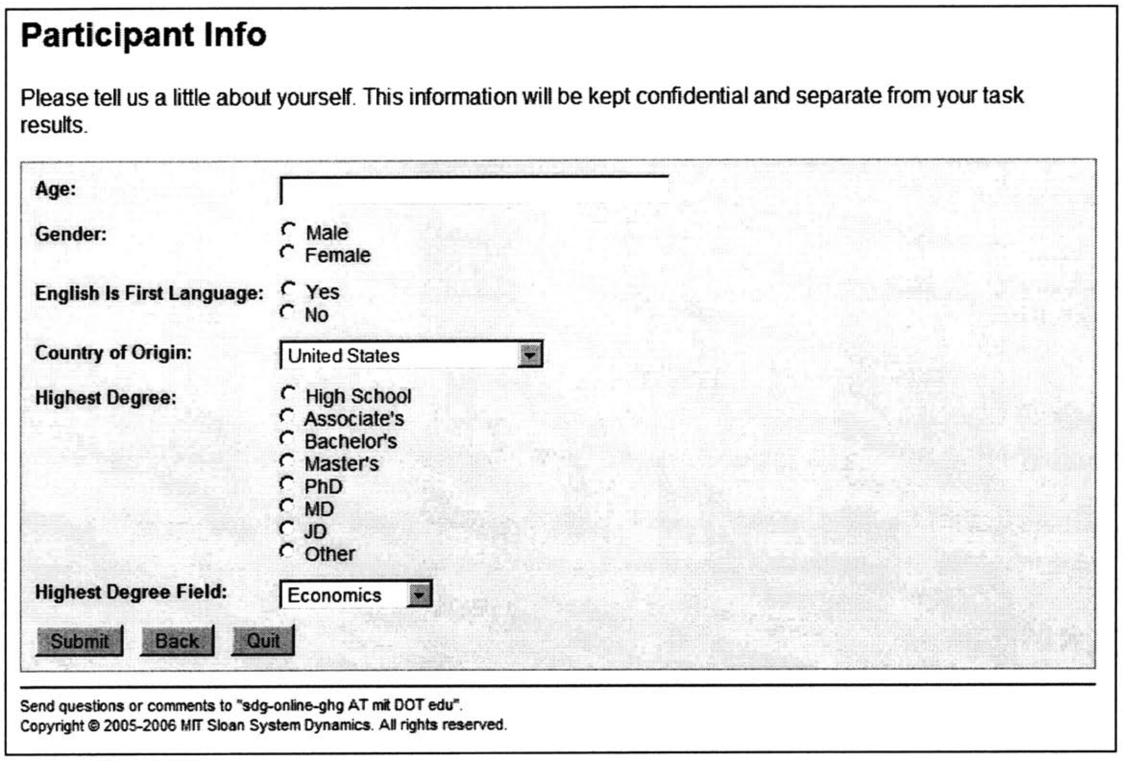
Welcome

Welcome to the online climate survey. This survey is administered by the System Dynamics group at MIT's Sloan School of Management.

Take the survey

Send questions or comments to "sdg-online-ghg AT mit DOT edu".
Copyright © 2005-2006 MIT Sloan System Dynamics. All rights reserved.

Figure 32: A screenshot of the GHG survey's welcome page



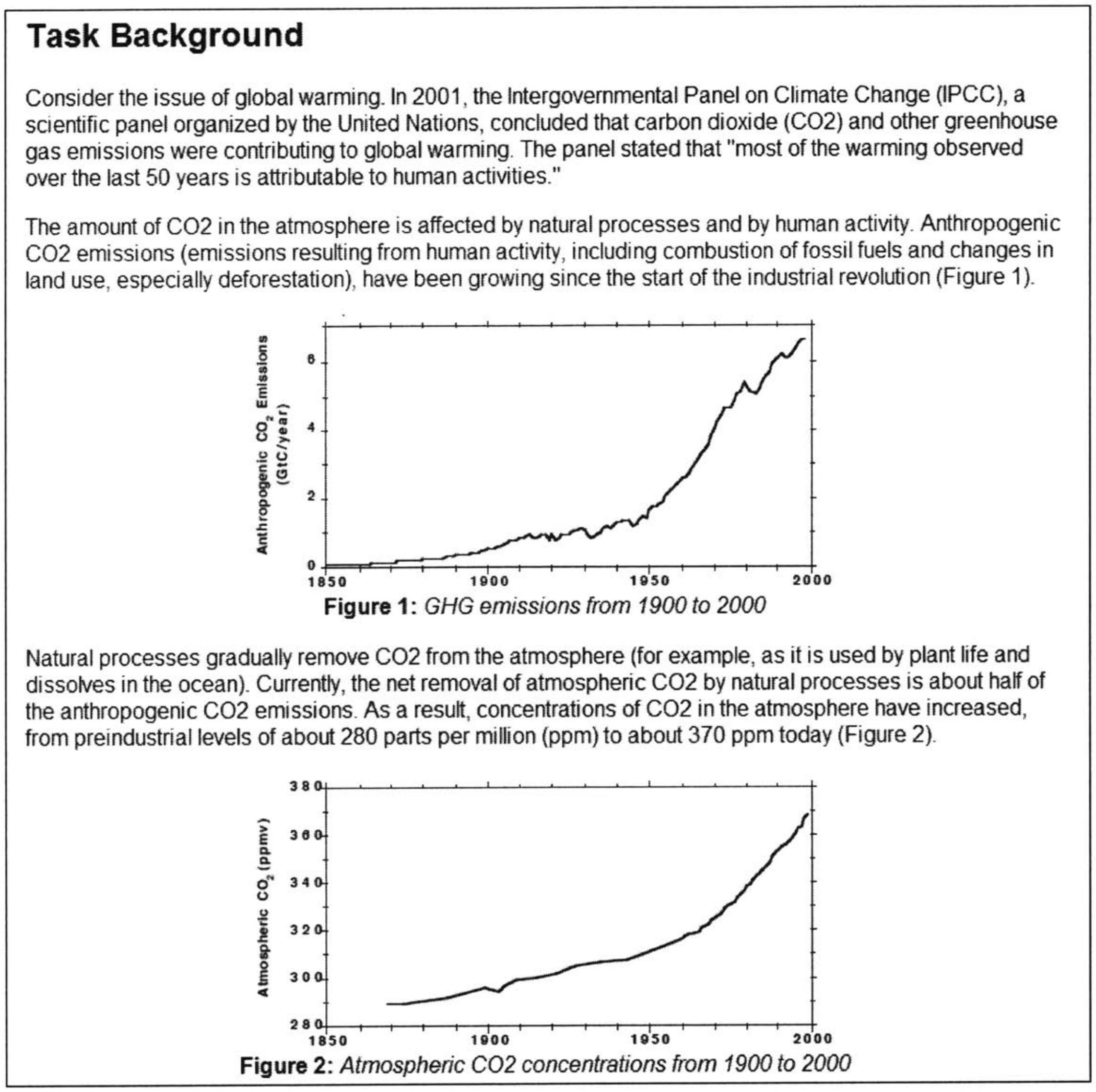
Participant Info

Please tell us a little about yourself. This information will be kept confidential and separate from your task results.

Age:	
Gender:	○ Male ○ Female
English Is First Language:	○ Yes ○ No
Country of Origin:	United States
Highest Degree:	○ High School ○ Associate's ○ Bachelor's ○ Master's ○ PhD ○ MD ○ JD ○ Other
Highest Degree Field:	Economics

Send questions or comments to "sdg-online-ghg AT mit DOT edu".
Copyright © 2005-2006 MIT Sloan System Dynamics. All rights reserved.

Figure 33: A screenshot of the GHG survey's participant info page



Task Background

Consider the issue of global warming. In 2001, the Intergovernmental Panel on Climate Change (IPCC), a scientific panel organized by the United Nations, concluded that carbon dioxide (CO2) and other greenhouse gas emissions were contributing to global warming. The panel stated that "most of the warming observed over the last 50 years is attributable to human activities."

The amount of CO2 in the atmosphere is affected by natural processes and by human activity. Anthropogenic CO2 emissions (emissions resulting from human activity, including combustion of fossil fuels and changes in land use, especially deforestation), have been growing since the start of the industrial revolution (Figure 1).

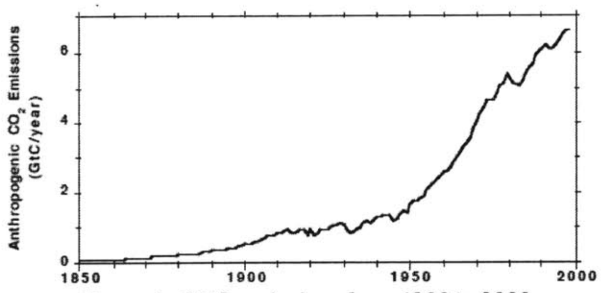


Figure 1: *GHG emissions from 1900 to 2000*

Natural processes gradually remove CO2 from the atmosphere (for example, as it is used by plant life and dissolves in the ocean). Currently, the net removal of atmospheric CO2 by natural processes is about half of the anthropogenic CO2 emissions. As a result, concentrations of CO2 in the atmosphere have increased, from preindustrial levels of about 280 parts per million (ppm) to about 370 ppm today (Figure 2).

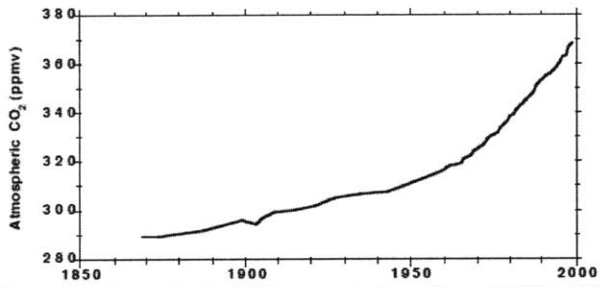
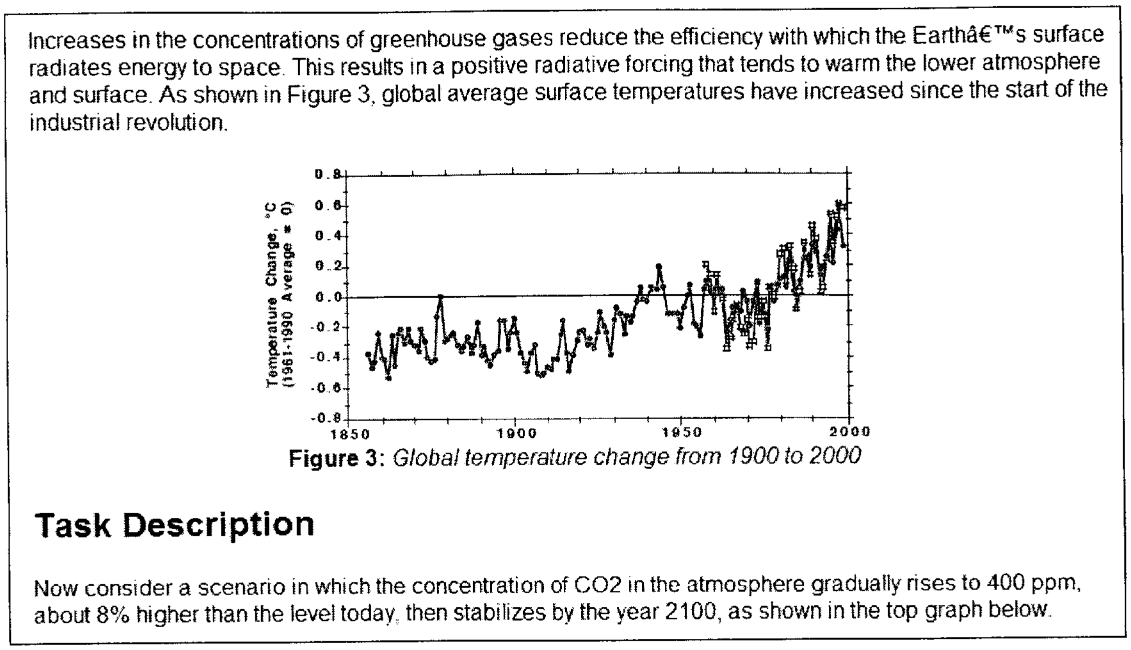


Figure 2: *Atmospheric CO2 concentrations from 1900 to 2000*

Figure 34: The first part of a screenshot of the GHG survey's task page



Increases in the concentrations of greenhouse gases reduce the efficiency with which the Earthâ€™s surface radiates energy to space. This results in a positive radiative forcing that tends to warm the lower atmosphere and surface. As shown in Figure 3, global average surface temperatures have increased since the start of the industrial revolution.

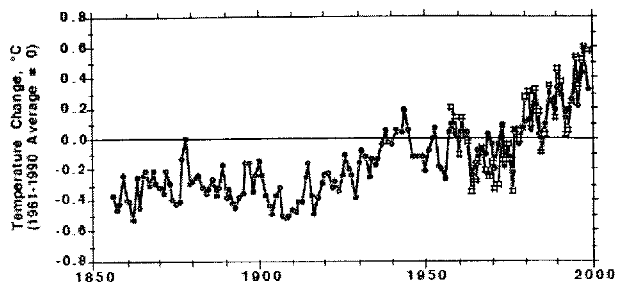
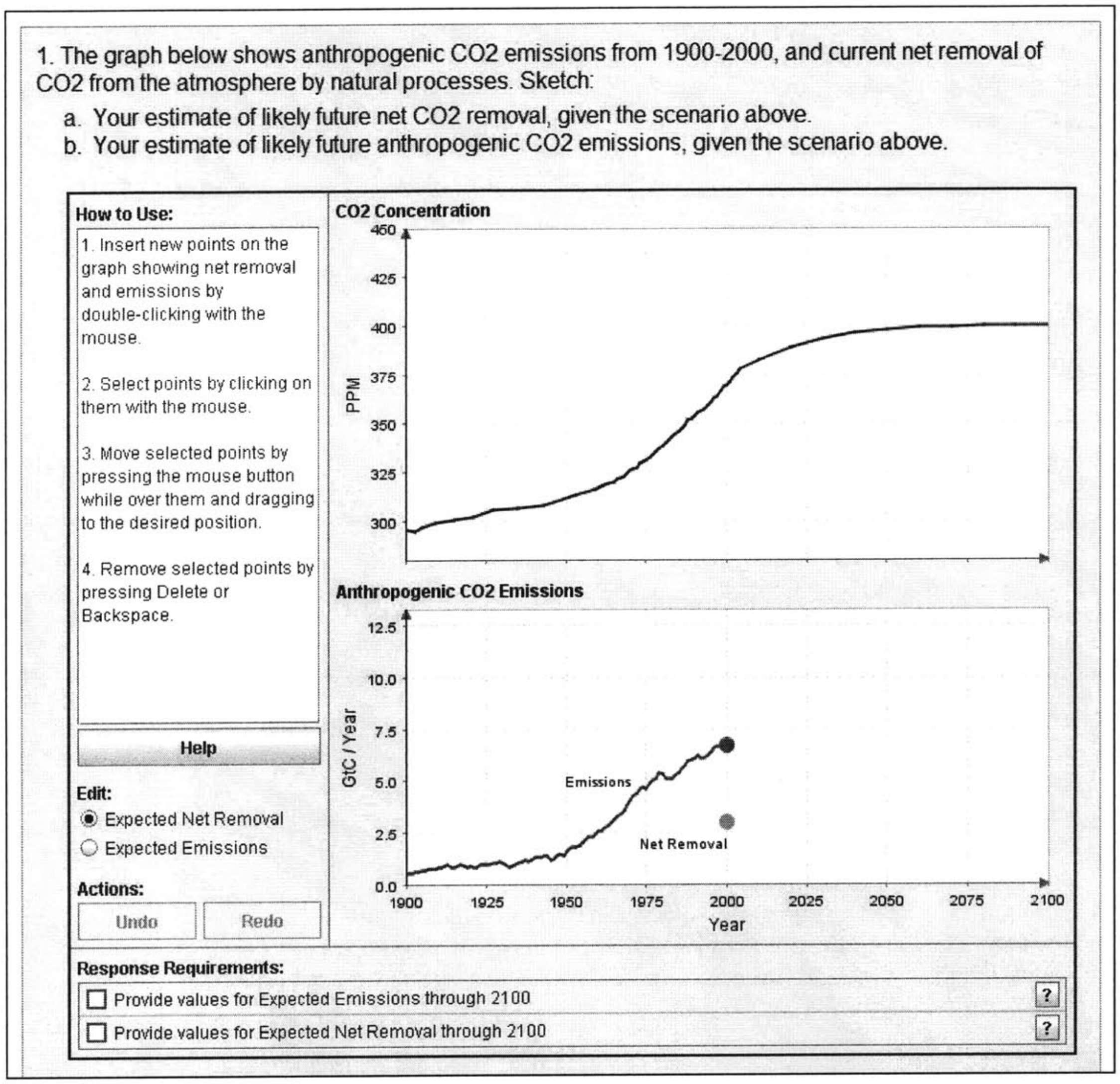


Figure 3: *Global temperature change from 1900 to 2000*

Task Description

Now consider a scenario in which the concentration of CO2 in the atmosphere gradually rises to 400 ppm, about 8% higher than the level today, then stabilizes by the year 2100, as shown in the top graph below.

Figure 35: The second part of a screenshot of the GHG survey's task page



1. The graph below shows anthropogenic CO2 emissions from 1900-2000, and current net removal of CO2 from the atmosphere by natural processes. Sketch:
   a. Your estimate of likely future net CO2 removal, given the scenario above.
   b. Your estimate of likely future anthropogenic CO2 emissions, given the scenario above.

How to Use:

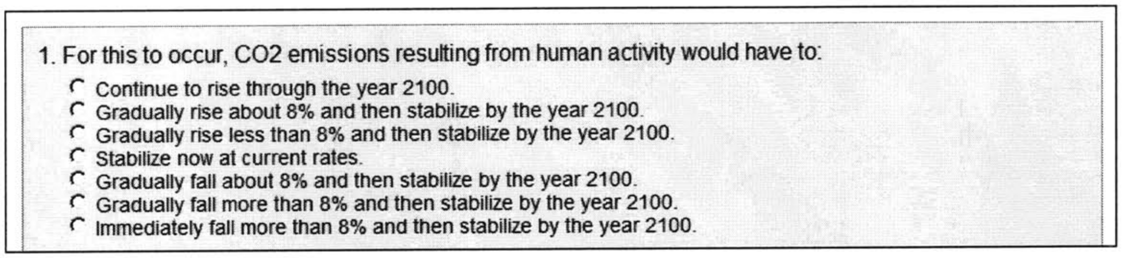
1. Insert new points on the graph showing net removal and emissions by double-clicking with the mouse.
2. Select points by clicking on them with the mouse.
3. Move selected points by pressing the mouse button while over them and dragging to the desired position.
4. Remove selected points by pressing Delete or Backspace.

Response Requirements:

☐ Provide values for Expected Emissions through 2100

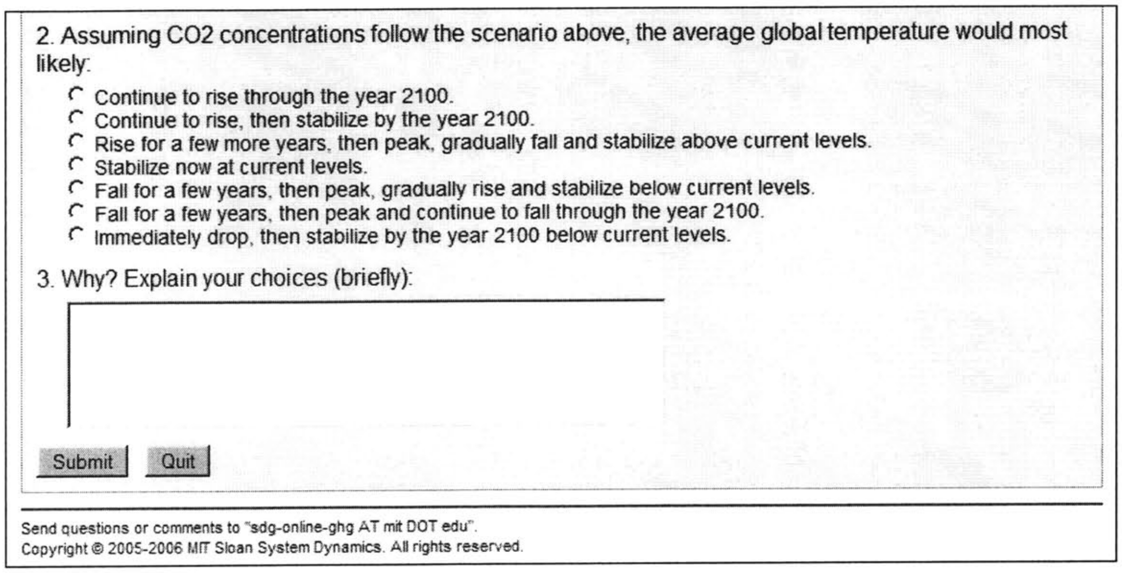
☐ Provide values for Expected Net Removal through 2100

Figure 36: A screenshot of the GHG survey using the graphing input applet



1. For this to occur, CO2 emissions resulting from human activity would have to:
   - ○ Continue to rise through the year 2100.
   - ○ Gradually rise about 8% and then stabilize by the year 2100.
   - ○ Gradually rise less than 8% and then stabilize by the year 2100.
   - ○ Stabilize now at current rates.
   - ○ Gradually fall about 8% and then stabilize by the year 2100.
   - ○ Gradually fall more than 8% and then stabilize by the year 2100.
   - ○ Immediately fall more than 8% and then stabilize by the year 2100.

Figure 37: A screenshot of the GHG survey's non-graphing inputs

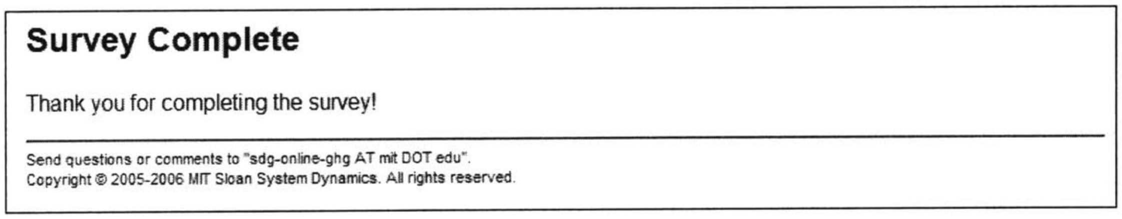

2. Assuming CO2 concentrations follow the scenario above, the average global temperature would most likely:

- ◯ Continue to rise through the year 2100.
- ◯ Continue to rise, then stabilize by the year 2100.
- ◯ Rise for a few more years, then peak, gradually fall and stabilize above current levels.
- ◯ Stabilize now at current levels.
- ◯ Fall for a few years, then peak, gradually rise and stabilize below current levels.
- ◯ Fall for a few years, then peak and continue to fall through the year 2100.
- ◯ Immediately drop, then stabilize by the year 2100 below current levels.

3. Why? Explain your choices (briefly):

Send questions or comments to "sdg-online-ghg AT mit DOT edu".
Copyright © 2005-2006 MIT Sloan System Dynamics. All rights reserved.

Figure 38: A screenshot of the GHG survey's non-graphing inputs


Survey Complete

Thank you for completing the survey!

Send questions or comments to "sdg-online-ghg AT mit DOT edu".
Copyright © 2005-2006 MIT Sloan System Dynamics. All rights reserved.

Figure 39: A screenshot of the GHG survey's final page