Simulation Modelling Practice and Theory 47 (2014) 28–45

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

Insight Maker: A general-purpose tool for web-based modeling & simulation

Scott Fortmann-Roe

University of California, Berkeley, Department of Environmental Science, Policy, and Management, 130 Mulford Hall, Berkeley, CA 94720-3114, United States

ARTICLE INFO

Article history:
Received 29 April 2013
Received in revised form 23 March 2014
Accepted 26 March 2014
Available online 14 June 2014

Keywords:
Modeling
Simulation
Web-based technologies
System Dynamics
Agent-Based Modeling

ABSTRACT

A web-based, general-purpose simulation and modeling tool is presented in this paper. The tool, Insight Maker, has been designed to make modeling and simulation accessible to a wider audience of users. Insight Maker integrates three general modeling approaches – System Dynamics, Agent-Based Modeling, and imperative programming – in a unified modeling framework. The environment provides a graphical model construction interface that is implemented purely in client-side code that runs on users' machines. Advanced features, such as model scripting and an optimization tool, are also described. Insight Maker, under development for several years, has gained significant adoption with currently more than 20,000 registered users. In addition to detailing the tool and its guiding philosophy, this first paper on Insight Maker describes lessons learned from the development of a complex web-based simulation and modeling tool.

1. Introduction

The field of modeling and simulation tools is diverse and emergent. General-purpose modeling tools (e.g. MATLAB's Simulink or the Modelica language [1]) sit beside highly focused and domain-specific applications (e.g. [2] for modeling network control systems, [3] for simulating the behavior of wireless network routing protocols, or [4] for the simulation and control of turbines). Interest in and published works on such tools has grown over time. The ISI Web of Knowledge reports a substantial growth in papers published on modeling or simulation tools with 299 such papers published in the span of 1985–1989, 1482 published from 1995 to 1999, and 3727 published from 2005 to 2009.[1]

For end-users, simulation and modeling tools are generally designed as executables to be run on a consumer operating system such as Windows or Mac OS X. With the expansion of the Internet and the ubiquity of the World Wide Web, new possibilities to harness this communication technology and platform to develop rich collaborative modeling tools have become available. One major review of the opportunities and issues faced when utilizing web-based technologies as part of a simulation or modeling application has been [5]. However, technology has advanced rapidly in the period since the publication of that review. Competition among technology companies such as Apple, Microsoft and Google has led to a rapid increase in the availability of advanced features in web browsers and significant performance enhancements in these same browsers (these improvements are documented at sites such as http://AreWeFastYet.com and http://html5test.com). Such improvements have changed what is effectively possible in the browser. Where [5] note that web-based technologies were

E-mail address: ScottFR@berkeley.edu

[1] The exact search query used was '"modeling tool" OR "simulation tool"' in the Topic field.

http://dx.doi.org/10.1016/j.simpat.2014.03.013
1569-190X/

not well suited to "local simulation and visualization" – running all simulation code on an end-user's machine – that is no longer the case as the work presented here demonstrates.

Other work on the intersection between simulation and modeling tools and web-based technologies includes but is not limited to, [6] who develop a simulation–optimization platform for industrial applications, [7] who propose a framework for online collaborative modeling and simulation an approach to environment construction adaptable across multiple domains, [8] who study the execution of simulations in cloud-based platforms, and [9] who explore collaborative model integration in a distributed environment. What is missing from earlier work, however, is a realization of a general-purpose simulation environment developed using web-based technologies targeted first at a general audience.

Web-based technologies are especially beneficial to non-experts as they can significantly reduce the costs required for a new user to experiment with and learn about a simulation and modeling tool. Simulation and modeling tools have been shown to be useful instructional devices (e.g. [10] who used a simulation tool in the economics classroom or [11] for examples of work in primary schools). By developing web-based simulation and modeling tools, the accessibility of these tools can be increased both within the classroom and outside it. A layperson considering modeling may be adverse to downloading and installing modeling and simulation software on a personal computer (due to fear of viruses and other concerns). On the other hand, the same person may be more willing to open a simple web page that contains a simulation model or embedded model construction interface. Lowering barriers to entry make it possible to engage more people in modeling and simulation, potentially leading to societal benefits as a result of an increased usage and understanding of modeling.

The rest of this paper presents the simulation and modeling tool Insight Maker (available at http://InsightMaker.com). It is a free, open-source modeling and simulation tool developed using web-based technologies and supports graphical model construction using multiple paradigms. The software is designed primarily to be as accessible to the layperson as possible, but also contains significant advanced modeling tools such as embedded scripting capabilities and an optimization toolset. Insight Maker was made public at the end of 2009 and has undergone continual development and evolution since then. Since its release, the tool has gain over 20,000 registered users. The feedback from these users has helped set the course for the direction and development of the tool.

The remaining sections of this paper first describe the philosophy and design principles of the simulation and modeling tool. Next the architecture of the tool is detailed. As a web-based technology, the tool includes both server-side and client-side components and this architecture is discussed. Particular attention is paid to the benefits offered and compromises required to successfully implement a complex tool such as Insight Maker in a web-based environment. Finally, given that the tool has undergone multiple years of iteration and development based on user feedback and benchmarks of user adoption, conclusions and lessons learned from this evolution are also presented.

2. Guiding principles

It can be argued that three basic criteria should be used when assessing a tool or environment used to develop simulation and models: performance, features, and accessibility. A high-performing environment is one that executes simulations quickly with minimal resource requirements. Features are capabilities and high-level functionality in the simulation and modeling tool. Accessibility indicates how easy it is to learn or use an environment. There are generally tradeoffs among these criteria in the face of a fixed amount of development resources:

- *Performance versus features:* Adding features to the simulation tool may often result in a negative impact on performance. In addition to software "bloat", certain features may require computational overhead during simulations or make it more difficult to implement some types of optimizations.
- *Performance versus accessibility:* Making a simulation program accessible to users often requires utilizing higher-level concepts divorced from the underlying machine hardware. An environment's ability to convert higher-level concepts to efficient machine code vary, but in general, accessible higher-level environments will have reduced performance when compared to less accessible, lower-level environments.
- *Accessibility versus features:* A baseline of features is important to make a simulation program accessible for users (e.g. a Graphical User Interface). However, many features added to simulation software are targeted at specific subsets of users and offer little value to the average user. Excess features that lead to "bloat" may result in user frustration or the feeling of being overwhelmed [12].
- *Performance versus accuracy:* One final tradeoff is between performance and accuracy, as they are often inversely correlated. This tradeoff may often be exposed directly to users allowing them to determine how to balance these competing goals.

Although tradeoffs may be mitigated through increased development time or resources, the choices developers make about these tradeoffs are key in defining their simulation programs and environments (Fig. 1). As an example, models and simulations may be developed in raw assembly as machine processor instructions. Such models should have excellent performance, but their development would be highly difficult as assembly is inaccessible and has a minimal built-in feature set. At the cost of some performance, higher-level development environments such as C++ are alternatives. Higher-level tools

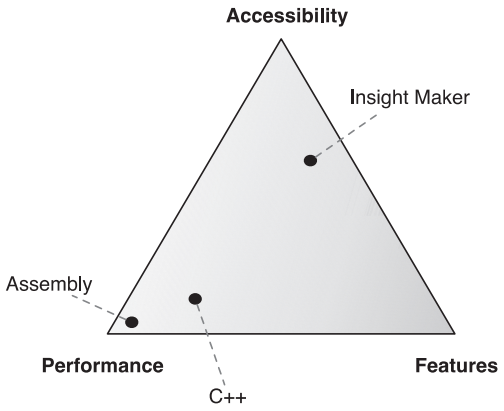


Fig. 1. Conceptual illustration of priority tradeoffs for simulation and modeling tools.

offer increased accessibility and features, but in exchange are often not able to match the performance of a highly optimized, lower-level alternative.

Insight Maker prioritizes tradeoffs among performance, accessibility and features in the following way. Accessibility is given primary priority, followed by features, with performance assigned the lowest priority. Accessibility is the primary goal of Insight Maker and is central to the design of the tool from the perspective of the overall program architecture (namely, the choice of implementation as a web application to increase application availability), to small language design choices (such as the use of case-insensitivity in the language), to cosmetic choices.

The inclusion of a rich feature set is Insight Maker's second priority. Many tools and capabilities are included in the program such as a built-in optimizer and scripting language. In adding features, attempts are made not to significantly increase the apparent complexity of the software; this has meant making those features available for users who desire them but highlighting a basic set of functionality for newer users who are not immediately interested or able to use the more complex features. An instance of this approach is Insight Maker's support for measurement units. The simulation engine contains the ability to associate units with each number in the simulation that are then automatically checked and validated when the simulation runs. This useful capability can catch a class of equation errors and improve the reliability of models. However, it can also decrease the accessibility of the software as it requires that users specify the units upfront when constructing a model, which can be an unnecessarily complex burden for many applications. Thus, the simulator makes units a purely optional feature in model construction, which the user can ignore or fully engage with depending on their individual skill level and needs.

The priority that is least focused on in Insight Maker is performance. Significant work has gone into optimizing the application. Nevertheless, choices made as a result of prioritizing accessibility first and features second make it impossible to achieve speeds that would be possible if performance had been the primary priority. For instance, the choice of a web application architecture imposes significant limitations on the application's performance. As a pure web application, the simulator must use ECMAScript as its primary programming language[2] which has good, but not excellent, performance characteristics, being roughly 2–8 times slower than C++ for numerical work [13]. Similarly, specific features may have a performance cost. For instance, the support of units in Insight Maker requires an aggregate data object to be maintained representing the results of calculations. This data object has significant performance costs compared to the usage of generic numbers.

3. The application

Insight Maker as a web application means that users run it by accessing a URL through their web browser. The application provides a number of different services that both mirror what would be found in a traditional desktop application – model construction and model simulation – but also extend beyond into areas more specific to a web environment – user account management as well as model searching and sharing. Although the latter services are important to the proper functioning of the application, they serve a purely bookkeeping role and are not intellectually interesting in the context of modeling and simulation. Instead, it is the novel simulation and modeling aspects of the application that are the focus of this section and paper.

Fig. 2 illustrates the primary model construction interface of Insight Maker running within the Google Chrome web browser. The interface is divided into three primary components: the model diagram, the toolbar and the configuration panel. The model diagram is an interactive illustration of the current model's structure along with additional UI features such as textual descriptions, pictures and interactive buttons. Users may select, position and resize elements in the model diagram using an

[2] Other languages such as Java and Flash are often used in the browser. However, usage of these technologies require that users have plug-ins installed to support them. Many users may not.

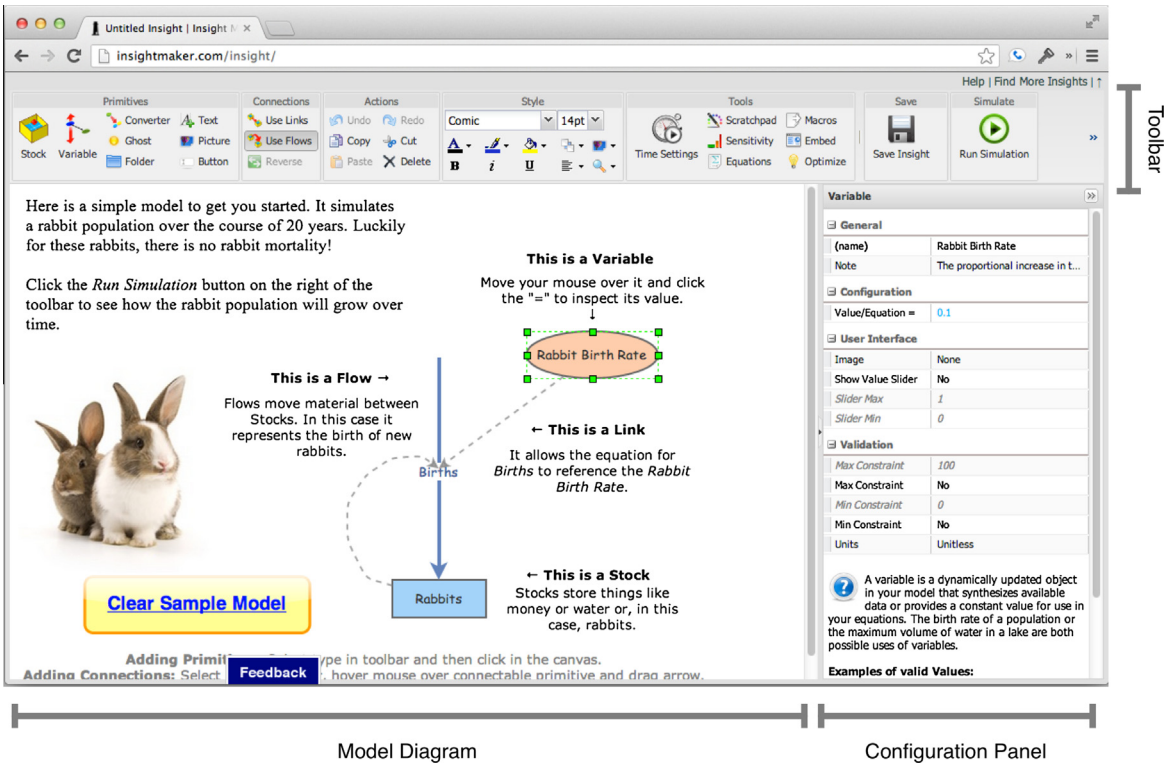


Fig. 2. The primary model construction interface running within the Google Chrome web browser.

input device such as a mouse or, on touch devices such as tablets, their finger. Each item in the model diagram has a set of attributes associated with it that control its behaviors. The attributes can be configured in the configuration panel. The toolbar at the top of the window provides a number of functions. It contains a selector to add different types of objects to the model diagram (both for simulation and for descriptive purposes), standard tools for document editing (undo/redo, copy and paste, etc.), styling options to adjust the appearance of objects in the diagram (coloring and font characteristics), tools to interface with the simulation, and buttons to save the completed model or start a simulation of the model.

The entire model building and simulation aspect of the application runs within a single web browser window. When a simulation completes, results are displayed within the same browser window. A number of different display types for results are available including time series charts, tables, histograms, scatterplots, and two-dimensional maps (Fig. 3). For offline analysis, users can export results to CSV files. In addition to the display of results, a number of other windows and configuration dialogues open within this main window. By way of example, dialogues to configure an object's mathematical equation, set simulation time settings, and carry out optimization runs or sensitivity tests all open within this primary window. The entire contents of this window are defined using standard web technologies such as Hypertext Markup Language (HTML) and Cascading Style Sheets (CSS).

4. System architecture

The application uses a client–server architecture (Fig. 4), where clients (user computing devices) are connected to the server over the Internet. Users may use the web browser on their machine to connect to the server and load the model construction and simulation environment. The server-side components are responsible for managing user accounts, storing user data and models, and providing search infrastructure. The client-side components provide model-editing capabilities along with model simulation and analysis functionalities. Clients do not communicate with one another directly and instead all communication happens through the centralized server.

It is important to note that the client-side code runs the simulation. An alternative would be to place the simulation logic and code on the server and simply use the client for model construction and the reporting of results. This latter structure was, in fact, how the application was originally designed. There are tradeoffs between these two approaches. In general, optimized server-side simulations should be faster than client-side, ECMAScript-powered simulations. For small models, however, the latency of sending data back and forth to the server may result in a server-side simulation effectively being slower than client-side simulations. Other trade-offs between server-side and client-side simulations involve issues with proprietary models (client-side simulation gives the client access to the model, making it impossible to release a model

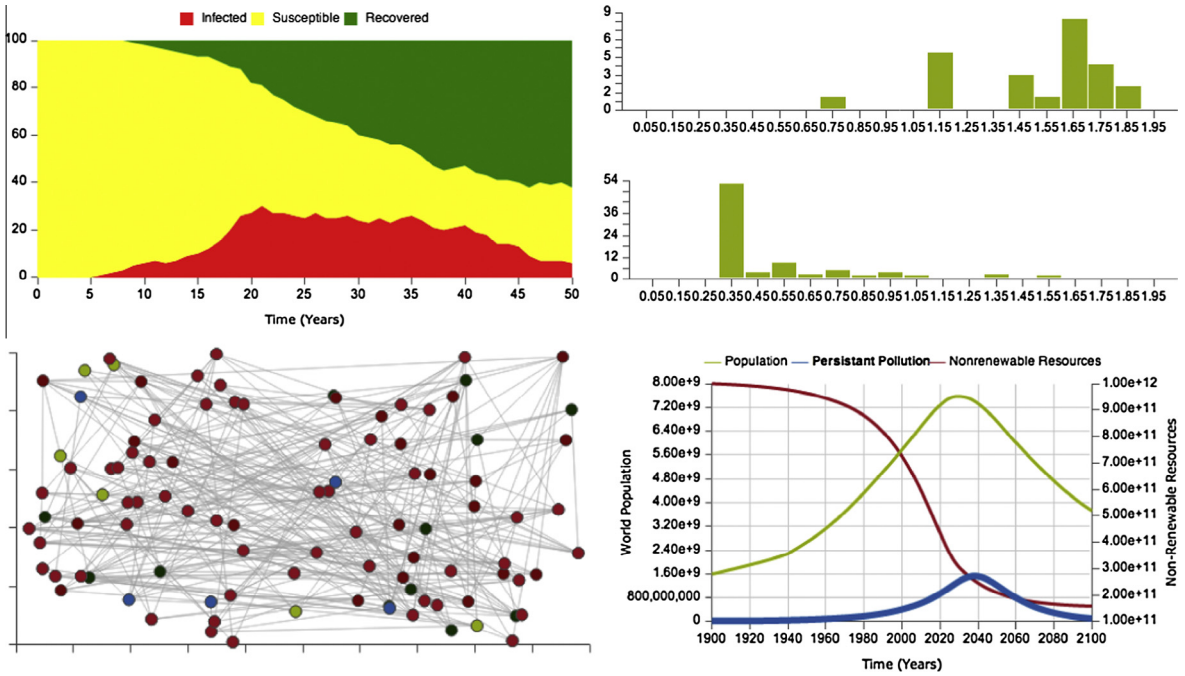


Fig. 3. Example of simulation output displays. Clockwise from upper left: the simulation of the spread of disease in a community, the distribution of two attributes of individual agents in a simulation, a simulation of world population growth, the simulation of location and connections between individual agents in a model.

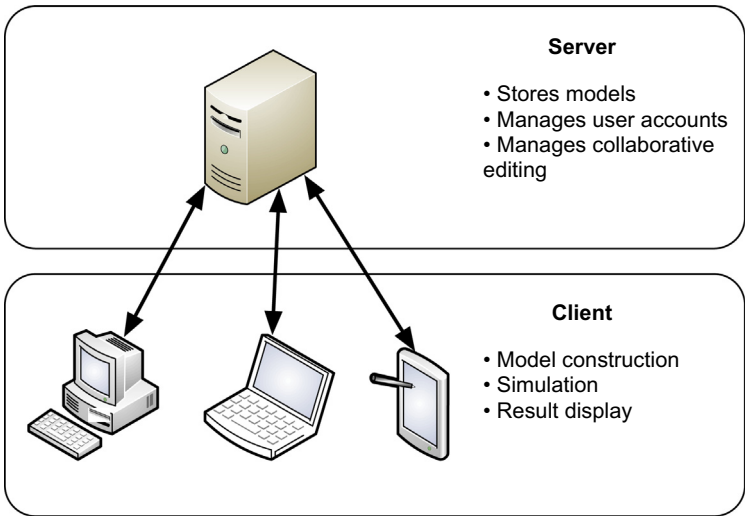


Fig. 4. Overview of tool's client–server architecture.

without also releasing access to the model details) and the effects of software popularity (server-side simulations can become costly and require the purchasing of additional servers, if the software experiences significant demand; client side simulation reduces this pressure). Ideally, both server-side and client-side simulation options would be available for the user to choose between. Unfortunately, this would most likely require maintaining two simulation code bases, which is currently beyond the scope of the resources available to this project.

In terms of technological choices, the tool uses standard open-source technologies and runs on a generic Linux/Unix server. As much as possible, existing open-source technologies and solutions were used to minimize costs and increase the portability of the system. Data are stored on the server in the open-source MySQL database, while the open-source language PHP and the open-source content management system Drupal are used to store data and implement server-side application logic. The open-source application Lighttpd is used as the actual server software.

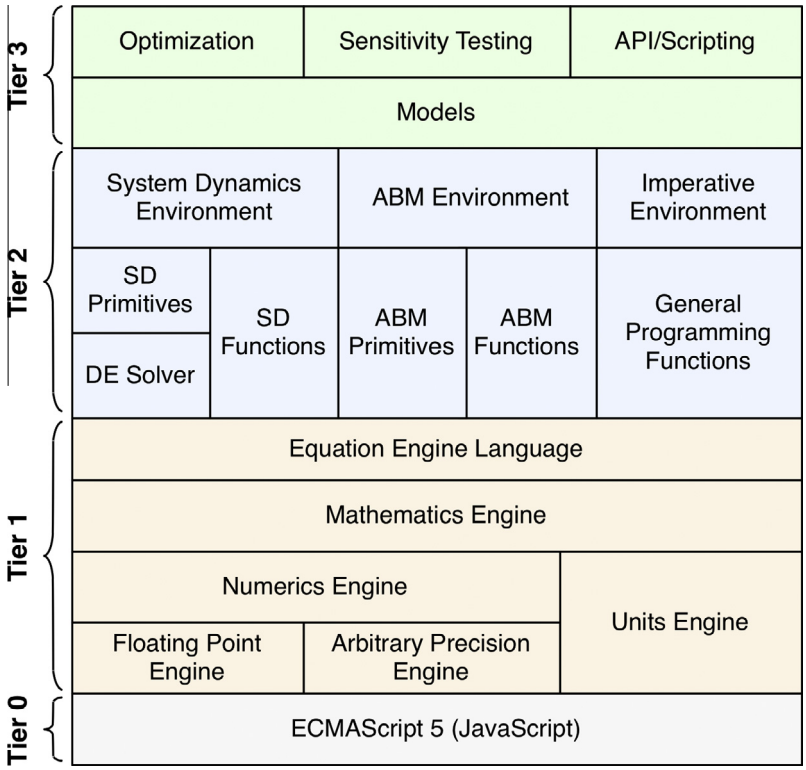


Fig. 5. Overview of the simulator architecture.

5. Simulator architecture and capabilities

The simulator portion of Insight Maker is architected modularly in multiple layers, where the higher-level layers depend on functionality provided by the lower levels (Fig. 5). The layers may be grouped into separate tiers that represent broad classes of functionality. Tier 0, the lowest level tier that provides the foundation for the simulator, is the ECMAScript programming language (colloquially referred to as "JavaScript"). ECMAScript is a dynamic programming language with high performance implementations available in every modern browser. The simulator uses ECMAScript 5 or higher and is supported in all modern browsers.

5.1. Tier 1

Tier 1 provides mathematics and equation language engines to evaluate equations entered by users. The engines themselves are generic, computational tools and do not provide simulation and modeling specific features. Such features are provided by the higher tiers that will build on this tier.

5.1.1. Mathematics engine

The mathematics engine parses and evaluates mathematical statements. There are two distinct underlying numerical libraries that the simulator may use. One generic calculation engine is provided by the ECMAScript environment itself. This is a purely double precision floating-point library with no integer, fractional or decimal data types. The second library is an arbitrary precision mathematics library implemented in ECMAScript. This second library supports precise decimal mathematics and numbers of arbitrary size with no numerical errors or imprecisions. The tradeoff between the two libraries is one of performance versus accuracy. The arbitrary precision library requires a large performance penalty over the native ECMAScript mathematics. On the other hand, the native library will have small errors due to the inherent inaccuracies in floating point numbers.[3] The open-source simulator can be configured to use either library. For the deployment of the Insight Maker code base on InsightMaker.com, only the native ECMAScript numeric library is made available to users in order to maximize the performance of user models (at the cost of some precision).

[3] An illustration of this lack of precision in double precision floating point math engines (utilizing the IEEE 754 standard [14]) is that the summation of "0.1" and "0.2" will evaluate to "0.30000000000000004". Many environments (such as Insight Maker) will attempt to round off the imprecisions when displaying the results thus hiding the issue from the user. Applications that require high precision, however, render such imprecisions potentially problematic.

Table 1
Example usage of equation engine to define and utilize a custom function.

```
Function Fib(n)
  If n = 1 or n = 0 Then
    1
  Else
    Fib(n − 1) + Fib(n − 2)
  End If
End Function

# Calculates the tenth Fibonacci number
Fib(10)
```

Table 2
Modeling paradigms supported by the simulator.

	Basic principle	Strengths	Weaknesses
System Dynamics	Models system on a highly aggregate level using rates of changes and state variables	Results are generally easy to interpret. Models can be simulated rapidly	Not well suited to modeling heterogeneity
Agent-Based Modeling	Models individuals each with their own properties and the interactions between individuals	Highly relatable results useful for communication. Can model systems in great detail	Potentially slow and can be hard to interpret results due to stochasticity
Imperative Programming	Develop logic using standard programming concepts	Highly flexible. Potentially very fast model execution	Limited abstraction. May require significant development effort

Additionally, the mathematics engine supports tagging numbers with arbitrary measurement units (e.g., instead of stating the length of an object in the model as "5", it can be declared as "5 meters"). The engine enables the transparent conversion between equivalent units and also the raising of exceptions when incompatible unit operations are attempted. This is primarily useful for end-users in that it provides an additional check that equations are entered correctly.

5.1.2. Equation engine language

The simulator includes a general equation engine language that sits above the mathematical engine. This equation engine language is a general-purpose programming language with domain specific features targeted toward modeling and simulation utilization. Language features include: the usage and creation of block-scoped variables; the usage and creation of functions; flow control, including If-Then-Else statements, While loops and For loops; and the construction of vectors (arrays).

Although the equation engine has a rich feature set, in most practical cases it serves as a thin layer over the mathematics engine in order to allow users to enter simple mathematical expressions. The more complex features are generally only required when constructing complicated Agent-Based Models. An example of the equation engine used for a simple recursive calculation is shown in Table 1 and provides the reader a feel for the language characteristics. Again it is important to underscore that the equation engine language is implemented purely in ECMAScript (including the equation engine language's tokenizer, parser and interpreter) and can be run in a web browser.

5.2. Tier 2

Tier 2 of the simulator is a multi-paradigm modeling environment that supports several different approaches to modeling (Table 2). Each approach has a different tradeoff between flexibility and ease-of-use. System Dynamics is highly abstract and aggregate. Agent-Based Modeling is often much more granular, allowing increased control over the system but at the potential cost of decreased performance and increased modeling effort. Imperative programming, on the other hand, has no modeling-specific constraints and can be flexible to address any modeling task but potentially at a high cost in development time. The simulator allows for the seamless integration of the three approaches within a single model, enabling different resolutions and methods for different portions of the model as required.

5.2.1. System Dynamics environment

System Dynamics is a modeling paradigm developed in the 1950s to study industrial systems [15]. The technique is general and has since been applied to a range of different systems including, notably, the development of urban systems [16] and forecasting world wide trends [17]. Mathematically, System Dynamics models are often sets of nonlinear differential equations. What sets System Dynamics apart from the standard analytical analysis of differential equations is that the System Dynamics community focuses on easy-to-use graphical tools and numerical analysis of simulation results. System Dynamics modelers are also primarily focused on feedback loops and the roles they play in the evolution of a system.

Table 3
Overview of key System Dynamics primitives in the tool.

Primitive	Use	Example usages
Stock	To store a material. Equivalent to a state variable in a differential equation	A lake that stores water. A bank account that stores money. A population that stores people
Flow	To move material between stocks. Equivalent to a derivate term in differential equations	A river into a lake. Withdrawals from a bank account. Deaths in a population
Variable	To store a parameter value or dynamically calculate a feature of the model	The flow rate for a river. The interest rate for a bank account. The death rate for a population
Converter	To load tabular data. To carry out a non-parametric transformation function	A schedule of flow rates by time of year. A non-parametric function relating population death rate to population size

5.2.1.1. Primitives. System Dynamics models are built graphically out of primitives: building blocks each with a unique function. The basic paradigm is that of a stock-and-flow model (referred to in some fields as a compartmental model), where material is moved by flow primitives between stock primitives. This maps conceptually and directly onto standard differential equation or dynamical systems models used in many fields. In this mapping, stocks represent state variables and flows represent the derivatives (or rates of change) of these state variables. The key primitives provided by the simulator are described in Table 3.

5.2.1.2. Functions. Although not strictly formalized, the System Dynamics community has evolved conventions for functions and patterns of model construction. These numerous conventions are generally shared among different System Dynamics simulation software tools along with System Dynamics education resources. The conventions are followed within the simulator's System Dynamics environment by including standard System Dynamics functions, such as those for generating canonical inputs (e.g. RAMP, STEP, and PULSE input functions) and functions for causing various forms of delays and feedback (e.g. DELAY1 and DELAY3, being first and second order exponential delays respectively). Such conventions facilitate the movement of models and ideas between System Dynamics software environments and they are supported in the implementation of the simulator.

5.2.1.3. Numeric differential equation solver. The simulator includes two numerical solvers to estimate solutions to user-defined differential equations underlying the System Dynamics models: Euler's method and a 4th Order Runge–Kutta method. Generally, the 4th Order Runge–Kutta method should be the preferred choice over Euler's method due to its increased accuracy per unit of computational effort. Nonetheless, Euler's method may be preferred in two cases – first, when there are a number of discontinuities in the rates of change for the systems state variables. In this case, the averaging process of the Runge–Kutta method might not be desirable. The second usage case for Euler's method is for learning and educational environments, as it is conceptually simpler and easier to understand as compared to the 4th Order Runge–Kutta method.

5.2.2. Agent-Based Modeling environment

Agent-Based Modeling simulates the individual actors or agents in a model [18]. Each agent has the potential ability to independently modify its state, based on rules or algorithms. Agents may modify their state in relation to other agents, forming systems and developing complex system-level behaviors [19,20]. This represents a significant difference from System Dynamics modeling, which addresses aggregated behavior and results. Simulating individual agents allows for finer grained models and greater flexibility, but it comes at a cost of increased computational effort and potentially leads to results that are more difficult to interpret.

5.2.2.1. Primitives. The simulator includes an approach to constructing Agent-Based Models that, in terms of user-interface elements, is very similar to that used to construct System Dynamics models. As in System Dynamics models, Agent-Based Models are built graphically using a set of primitives, each of which fulfills a specific modeling function. Whereas in the System Dynamics modeling paradigm the primary product was the stock and flow diagram, in Agent-Based Modeling it is a state transition chart where an agent may occupy one or more states and transition between them as a function of mechanistic or stochastic rules. Table 4 summarizes the primitives used to construct Agent-Based Models in the simulator.

Table 4
Overview of key Agent-Based Modeling primitives in the tool.

Primitive	Use	Example usages
Agent population	A collection of agents of a given type	A population of people. A forest of trees
State	Defining the status of a given agent.	Given a person, are they a male or female?
	A binary attribute[a]	Given a disease simulation, is the person in the healthy or infected state?
Transition	Moves agents between states	The process of infection moving a person from the healthy state to infected state
Action	Executes arbitrary actions for an agent	An action to move the agent away from infected agents. An action to change a state based on a global condition

[a] For continuous state variables, stocks may be used within an agent.

Although Agent-Based Modeling can be conceived as distinct from the System Dynamics paradigm, they are in fact fully integrated in the simulator and the distinction between them is made solely to facilitate discussion in regards to existing paradigms. System Dynamics primitives can readily be included in primarily Agent-Based Models (for instance, a stock and flow model can be created within an individual agent) and vice versa. Thus, while states only represent binary values, stocks can be included within agents to track continuous state variables.

5.2.2.2. Functions. A number of functions included in the simulator are specifically targeted at building Agent-Based Models. One category of functions deals with selecting agents matching certain criteria from a population. For instance, the FINDSTATE function will return a vector of all the agents in a model that have activated a given state. Find statements can be nested enabling binary AND logic, and the results from separate find-statements may be merged enabling binary OR logic.

The Agent-Based Modeling in the simulator supports two forms of geographic relationships between agents: two-dimensional geography and network geography. Agents may be given a physical position within the plane. Their position may be queried or used as a selector (e.g. the FINDNEAREST or FINDFURTHEST functions to select agents that are, respectively, close to or far from a given target) and the position may also be modified (e.g. the MOVE or MOVETOWARDS functions). With regard to network geometry, each agent may have one or more connections to other agents. Each connection is binary and may be queried or modified as the simulation progresses (e.g. with the CONNECT or FINDCONNECTED functions).

5.2.2.3. Imperative programming environment. The equation engine language is an imperative programming language. This language will generally be used to write elementary mathematical equations. It can also be used to implement algorithms or model logic that does not fit within the System Dynamics or Agent-Based Modeling paradigms. These custom functions and code developed using the imperative environment can augment the System Dynamics or Agent-Based Modeling environments by providing additional functionality within them. For instance, a function could be defined in the imperative environment to carry out some numerical algorithm and that function will automatically become available for use by equations in the other environments.

5.3. Tier 3

The third tier consists of the model itself along with tools and services that operate on a given model. Three primary tools are offered by the simulator: an optimizer that maximizes or minimizes a goal, a sensitivity testing tool, and an API that can be used to script model behavior or analyze model results.

5.3.1. Optimizer

The simulator includes an optimizer to determine the set of parameter values that minimize or maximize an objective function. The user specifies the objective function, the parameters that should be adjusted, and the respective ranges of these parameters. The optimizer then explores the specified parameter space in search of the parameter set that achieves the objective. The study of optimization methods is a wide, multi-discipline field with numerous techniques and approaches available. Optimizations of the arbitrary user specified models in the simulator are, however, constrained by three primary features:

- *Non-convex objective functions:* Although some models and associated optimization tasks may result in convex optimization problems, in general it should be assumed that the parameter space for a given model has multiple local minima.
- *Non-smooth objective functions:* Given the use of logical functions in a model (e.g. if-then-else statements), it is quite possible for the objective function to have sharp discontinuities. Even without the usage of such functions, the numerical limits of the floating-point mathematics engine could create discontinuities on a fine scale.
- *No analytic derivatives:* Although the derivatives or Hessian matrix at a given point in the parameter space may be estimated numerically, it is generally impossible to calculate them directly.

A wide class of techniques used to address such optimization problems are known as "Direct Search" methods [21–23]. This class includes methods which do not rely upon differentials, such as the classic Nelder-Mead Simplex search [24], genetic algorithms [25] and simulated annealing [26]. Published comparisons between techniques exist (e.g. [27]); however, the results can often be context-dependent and vary across applications. For the simulator, a form of direct search based on Powell's method [28] is implemented.

Briefly, Powell's method takes an initial starting point and a set of search vectors that span the parameter space. The objective function is evaluated at the starting point. Then for each search vector the objective is in turn evaluated at a new position that is a given search step size in the positive direction along that vector and, if an improved solution is not found, subsequently in the negative direction. As soon as an improved solution is identified, the center of the search is relocated along that search vector to a distance twice the search step size (the optimistic assumption being that if an improvement is found in a direction, the solution will continue to improve if the optimizer continues in that direction). After a full iteration through all the search vectors is completed, a combined search vector is constructed as a composite from all the moves that were taken in that iteration. This composite search vector then replaces one of the existing vectors in the set of search vectors. If no move is made, the step size is reduced or the search can be terminated according to a stopping condition.

Note that the search algorithm as described is purely deterministic. The algorithm will find a single minimum value that, if multiple minima exist, may or may not be the global minimum. Many optimization techniques address this issue by introducing stochasticity into the optimization process (e.g. simulated annealing or genetic algorithms). Given convex problems, such stochasticity may, however, reduce efficiency in finding the minimum. As such, the simulator uses the deterministic algorithm as described, but deals with local minima by offering random starting locations for the search pattern. For instance, given an optimization problem where local minima are known or expected to exist, the optimizer could be run ten times, each time starting at a different random location. If all ten optimization sequences converge to the same minimum, it is highly likely the global minimum has been found. If all ten optimization sequences converge to separate minima, then little evidence is there to conclude that the global minimum has been located.

5.3.2. Sensitivity testing

The simulator's sensitivity testing tool repeatedly runs a simulation and aggregates results. It can display the aggregated result by plotting results for each run or by averaging the runs and calculating their distribution. The sensitivity-testing tool has several general use cases:

- *Average responses and variances for stochastic models:* Given a modification to a model's parameter values, most models will exhibit a changed trajectory of results. For stochastic models, it can be difficult to ascertain when changes in the trajectory were caused by the parameter modifications and when changes were caused by the model's inherent stochasticity. The sensitivity-testing tool allows the simulation to be run many times and the results averaged to numerically estimate a response independent of the stochastic variations.
- *Parameter sweeps:* Parameters sweeps can determine the model's response to a range of different parameter values. They can be useful for exploring model behavior, and the sensitivity-testing tool can automate the process of this exploration.
- *Uncertainty in parameter values:* Model parameter values are often known with limited certainty. The data used to estimate them may be limited or in some cases non-existent. The simulator's sensitivity-testing tool allows the formulation of a probability model to define the uncertainty for one or more parameters and then obtain a numerical estimate of the resulting distribution of outcomes given this uncertainty. This form of analysis is recommended as part of the model validation and verification process to determine whether model results and conclusions are robust to parametric uncertainty [29,30]. The simulator includes a set of functions to generate random numbers according to a range of common distributions (e.g. the normal distribution, the uniform distribution, and the log-normal distribution, among others) to model the uncertainty of knowledge about a parameter. The simulation is then repeated, each time sampling a set of a parameter values from their distributions. This Monte Carlo method enables the propagation of the uncertainty from the parameter values to the results.

5.3.3. API/scripting

The simulator includes an API that can be used to programmatically build models and analyze their results. The API is implemented in ECMAScript and there are several mechanisms by which it can be accessed. The most basic is through a special primitive called a Button that can be added to the model. When a user presses a button primitive, the associated code and API commands are executed. Another mechanism that can be used to access the API is the browser's console that allows users to directly enter API commands in an interactive manner.

The range of API functions includes those for model construction (e.g. CREATEPRIMITIVE or SETVALUE), those for styling the model (e.g. SETLINECOLOR or SETSIZE), those for running the model (RUNMODEL), and those commands for input and output (e.g. SHOWDATA or PROMPT). As the commands are implemented in ECMAScript, any standard ECMAScript data analysis or communication functions may, of course, also be used. These capabilities allow model scripts to, for instance, download parameter data from an external server on the fly, run the model, analyze the results, and/or upload the results summary to another server.

6. Integration and portability

Given the usage of standard web technologies (HTML, CSS, and ECMAScript) by the simulator and model construction interfaces, these aspects of the application are portable and may be deployed in any environment that supports rendering and displaying web pages. The most common usage for this capability is to deploy content within other web pages. Models built within Insight Maker can be embedded within external web pages and can retain full interactivity within these external environments. Users embedding models can also harness the simulator's API to develop a custom user interface for a model and simulation. Such custom user interfaces can be used to develop "serious games" (e.g. [31], a game exploring the Israel-Palestine conflict) or "flight simulators" (e.g. [32,33] for discussions of flight simulators in relation to management education and training).

7. Validation and verification

Although the application's development process does not follow testing approaches such as Test Driven Development [34], an automated suite of internal tests has a key role in ensuring proper simulation behavior and the prevention of

regressions. An extensive suite of over 1000 individual tests is part of the simulator that comprehensively evaluates the correctness of all aspects of the simulator's behavior. The tests range from the trivial (does the equation engine correctly evaluate that 2 + 2 equals 4?) to higher-level aggregate tests (does a given model containing multiple equations and primitives evaluate to the correct result?). Tests are written as new functionality is added to the program and also in response to reports of issues from users. When an issue is found, the issue is first corrected and then a test for the issue is written to ensure that it does not reappear in a later regression. The development of this test suite has proven invaluable in ensuring the proper operation of the tool.

8. Case study

8.1. Overview

To illustrate the applied use of the tool, a case study is presented in which Insight Maker was used by a group of laypeople as part of a collaborative modeling development process to explore the impacts of economic policies. Given Insight Maker's primary focus on accessibility and collaboration – especially among laypeople – this example is well suited for illustrating Insight Maker's strengths and weakness in achieving this primary goal. The case study is a review of work done by an ad-hoc group formed under the auspices of Systems Thinking World [35], a non-profit organization focused on exploring the practical application and realization of systems thinking, a holistic approach to analysis and problem-solving that centers on understanding the relationships and interactions between the components of a system. Out of the larger organization, fifteen laypeople organized to better understand the 2007–2008 financial crisis and its aftermath using a modeling-based approach.

The group used Insight Maker for this modeling due to its feature set and web-based nature, allowing them to share models easily. This latter capability was critical for the group as they were geographically dispersed and all their communication and coordination occurred over the Internet. In the following sections of the case study, we first present the group's modeling process, then detail their final model, and last, explore results and conclusions. While the case study only utilizes a portion of Insight Maker's features (it makes no use of the Agent Based Modeling features, the optimizer, or the sensitivity testing tool, for instance), it does speak to the accessibility and collaborative potential of Insight Maker that, as underscored above, is the tool's primary purpose.

8.2. Modeling process

The group was organized non-hierarchically. After formation, group progress followed a "diverging/converging" pattern. Group members would independently research or study a topic before returning and sharing their findings with the rest of the group, in turn addressing what the rest of the group had produced during that period. The group found this protocol to be highly effective; so much so that in the almost seven month period their work took, there was not a single group meeting among the members. After having developed some dozen models, the group discovered a sector model of the economy developed by Stephanie Kelton, chair of the Department of Economics at the University of Missouri–Kansas City [36]. This model had a profound impact on the group's thinking and in the remaining three months of the group's work, they took this base model and experimented with modifications to it.

8.3. Model details

The final version of the group's Investment versus Austerity model is presented in Fig. 6.[4] This figure depicts a stock and flow model combined with a number of buttons that users may use to run different scenarios or unfold the model in a structured revealing of the key components of the model. The model is a conceptual representation of a single nation's economy. Because most of the members of the group were located in the United States of America, the group used the term "dollars" to represent monetary units. However, the model was not calibrated to represent a specific economy and was instead focused on generalized dynamics. There is not a specific currency associated with the model and therefore the term "dollars" as used here means notional dollars. The rectangular primitives in the model diagram represent stocks. For instance, the *firm dollars* stock represents the quantity of money currently held by firms in the economy. The solid lines are flows and in this case model the movement of dollars between different stocks in the economy. The direction of the arrow indicates the direction of the flow. The ovals are variables that in this model are primarily used to represent model parameters.

Table 5 shows the unfolding of the model along with descriptions of the model's key connections and the logic behind these connections.

Full model primitives and equations are given in Appendix A. One challenge in implementing this model in Insight Maker was that Insight Maker does not include support for saving various combinations of parameter values. To emulate this support, the Investment versus Austerity model makes use of Insight Maker's API/scripting capabilities to develop scenarios and

[4] This model may also be accessed in runnable form at: http://insightmaker.com/insight/3023. Note that in this diagram some primitives have the same name (e.g. there are two *ftax* primitives). In such cases, the two primitives refer to the same underlying object using an Insight Maker feature called 'Ghosting'. These duplicates are created in order to simplify the complexity of the diagram and reduce the need for connection lines that intersect other lines.

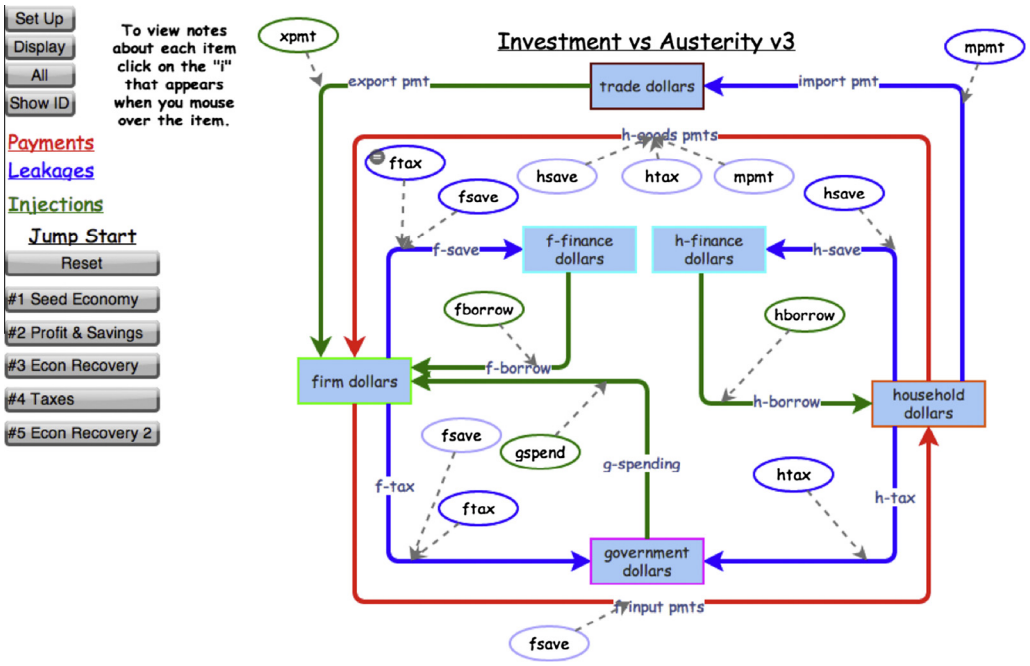


Fig. 6. Schematic diagram of the final version of the Investment versus Austerity model. Stocks are represented by rectangles; variables by ovals; and flows by solid lines. The shaded rectangles on the left are interactive buttons added by the modelers.

experiments that can be switched between by end-users. For instance, the code illustrated in Table 6 is executed when a user clicks the "Seed Economy" scenario button in the model. This code configures the value of the *gspend* variable and then runs the model.

8.4. Simulations and results

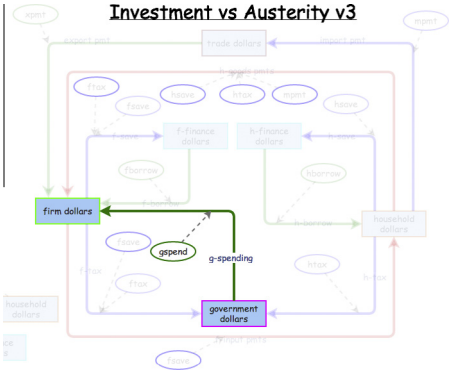
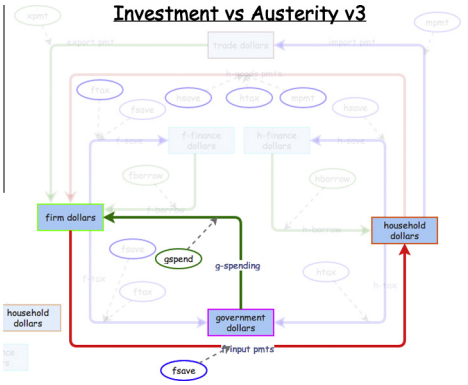
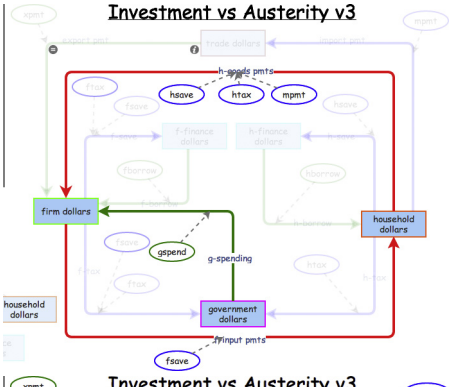
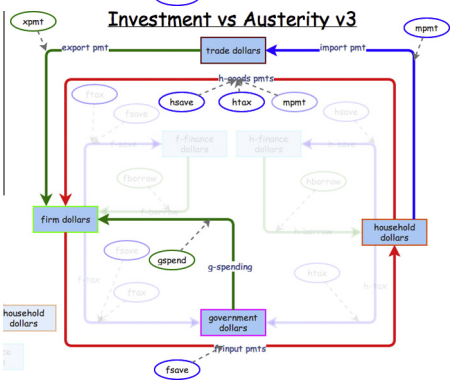
A number of simulation experiments and scenarios were run by the group with different parameter values and configurations in order to better understand the behavior of the modeled economy. Table 7 lists the primary scenarios developed by the group. Each scenario was defined by a model parameterization that differed from the other scenarios, thereby leading to different behaviors. The comparison of the behaviors between these scenarios increased modelers' understanding of the simulated economy. The left column displays the scenario name and motivation, the second column indicates the equation changes that were made to the model to implement the scenario, and the third column lists the key results from running the scenario. The overrides in each scenario are new equations for a primitive that are used in place of the initial model equation for that primitive.[5] The overrides in each scenario in the table are cumulative with the scenarios listed above it. Thus, for instance, the Econ Recovery scenario also includes the overrides from the Profit & Savings scenario.

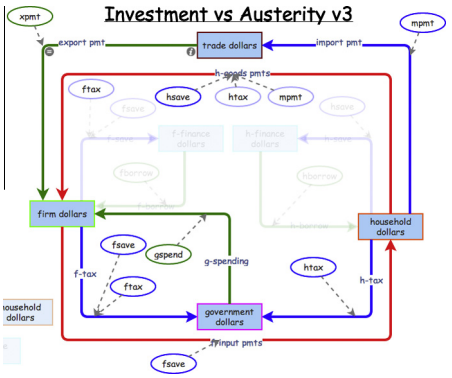
Example simulation results for two of the scenarios are presented in Fig. 7. The left panel illustrates the Seed Economy scenario, where the government undertakes deficit spending for a few years to prime the economy. The solid black line shows additional government spending as a pulse function over three years, that results in funds circulating between firms (dashed line) and households (dotted line) in a steady state. The accumulated government deficit spending (solid gray line) reflects the national debt. If nothing else changed, government would not have to inject additional funds into the economy and the dollars circulating between firms and households would keep the economy in a steady state. The right panel shows the more realistic Econ Recovery scenario where savings are non-zero. When firms and households save a portion of their income, they essentially subtract it from the economy. This extraction reduces the funds available for both firms and households to circulate in the economy (dashed and dotted lines). Unless the government then conducts deficit spending, the economy will decline, by definition, into a recession and then into a depression. Economic recovery requires continued deficit spending on the part of the government to offset the funds extracted by firms and households, other things being equal (for example, it is assumed no exogenous increases in factor productivity are occurring). The money put back into circulation to ensure the recovery further adds to the national debt (solid gray line).

These findings along with the entire modeling process were met with enthusiasm among the members in the group. Gene Bellinger, the group organizer, reports that the group members felt they now understood better how the economy works.

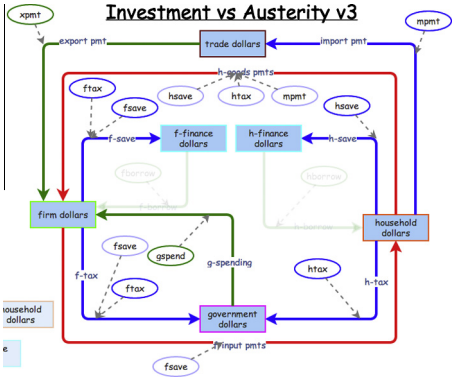
[5] These initial equations were designed such that the economy was in steady state with no money flowing.

Table 5
Unfolding of the Investment versus Austerity model.

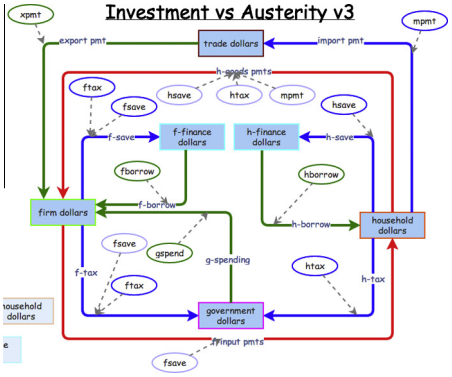
	This first connection is based on the simplified perspective that government spending injects money into the economy via purchases from firms. While the government injects money into the economy via other streams such as Medicare, Medicaid, and Social Security; purchases from firms were used as a first approximation of how money enters the economy from the Government
	This second flow represents the fact that firms employ household members to produce the goods and services purchased from firms. Firms do not purchase services with all of their funds as some are used to pay taxes and some are saved and distributed out to stockholders when appropriate
	This portion of the model represents the fact that households create demand on firms though the funds they have for purchases are less than their income as some of the funds are saved (*hsave*), some are used to pay taxes (*htax*) and some of their funds are used to purchase imports (*mpmt*) which creates demand in trade dollars
	These flows represent the fact that some of the household dollars are used to make foreign purchases resulting in imports (*import pmt*). At the same time, foreign governments, companies, and individuals purchase from the country's firms, which results in exports paid for with export payments



These added flows represent firms' tax payments (*f-tax*) on their profits and households' tax payments (*htax*). Both of these taxes represent income to the government and are used to support government spending (*g-spending*). When government income is less than government spending, a deficit results. The accumulation of this deficit spending over time is represented as the national debt



These flows indicate that households tend to save a portion of their income (*hsave*) for future use. This reduces the funds they spend on purchases from firms. Also when firms keep retained earnings (*fsave*), it reduces the funds they have available to purchase labor from households



This final segment of the model represents the fact that if households require additional funds, they borrow from their savings or from financial institutions (*hborrow*). When firms require additional funds they borrow them from their savings or from financial institutions (*fborrow*)

Table 6
Script to configure a scenario and then run the model. The *findName* function takes a primitive name and returns a reference to the primitive object with that name. The *setValue* function takes a primitive object and sets its equation to a given value. The *runModel* function starts a simulation and displays the results to the user. The equation passed to *setValue* results in a pulse output that is 0 until $t = 1$ years and then is 0.5 for the next 3 years before becoming 0 again after $t = 4$ years.

```
setValue(findName(''gspend''), ''Pulse(1,0.5,3)'');
runModel();
```

One can speculate that the increase in understanding the model facilitated among the group's participants was due directly to the fact that they were the ones building it rather than it being a model handed down to them by the experts in the field.

9. Discussion and conclusions

This study has been a complex and long-running one that accomplished four primary goals to date:

Table 7
Primary scenarios developed by the group. The overrides in the second column implement the changes needed to explore the questions in the first column.

Scenario	Equation overrides	Key results
Seed Economy – Base Scenario	gspend = Pulse(1,0.5,3)	
Sets government deficit spending to seed the economy with funds to prime the simulation	*Sets government spending (gspend) to a non-zero value starting at year one and proceeding for 3 years.*	For the economy to operate, the government has to initially conduct deficit spending which accumulates as the national debt
Profit & Savings	fsave = Step(6,0.05)	Firms and households tend to save a portion of their income, which in turn takes money out of circulation resulting in an ongoing decline in the economy
What is the implication of firms sitting on profits and households saving funds?	hsave = Step(7,0.1) *Starting at years 6 and 7, have both firms (fsave) and households (hsave) save a fraction of their capital*	
Econ Recovery	gspend = Pulse(1,0.5,3) + Step(7,0.15)	To offset the savings by firms and households, the government must continue deficit spending at a level equivalent to what firms and households are taking out of the economy. The problem is that the deficit spending also increases the national debt
How does deficit spending offset the savings by firms and households?	*Increase government spending (gspend) beyond the initial pulse by spending a constant amount starting at year 7*	
Taxes	ftax = Step(6,0.05)	The government requiring the payment of taxes reduces the level of deficit spending, other things being equal, but also reduces the funds in circulation thus depressing the economy
What effect does the government levying taxes have on the economy?	htax = Step(7,0.1) *Add a non-zero tax burden to both firms (ftax) and households (htax) starting in years 6 and 7*	
Econ Recovery 2	gspend = Pulse(1,0.5,3) + Step(7,0.2)	
What is the effect of increased deficit spending given the presence of taxes?	*Increase the government spending level (gspend) beyond that of the first Econ Recovery scenario level*	This scenario increases the long run government deficit spending by 33% to correct the economic decline created from the levy of taxes

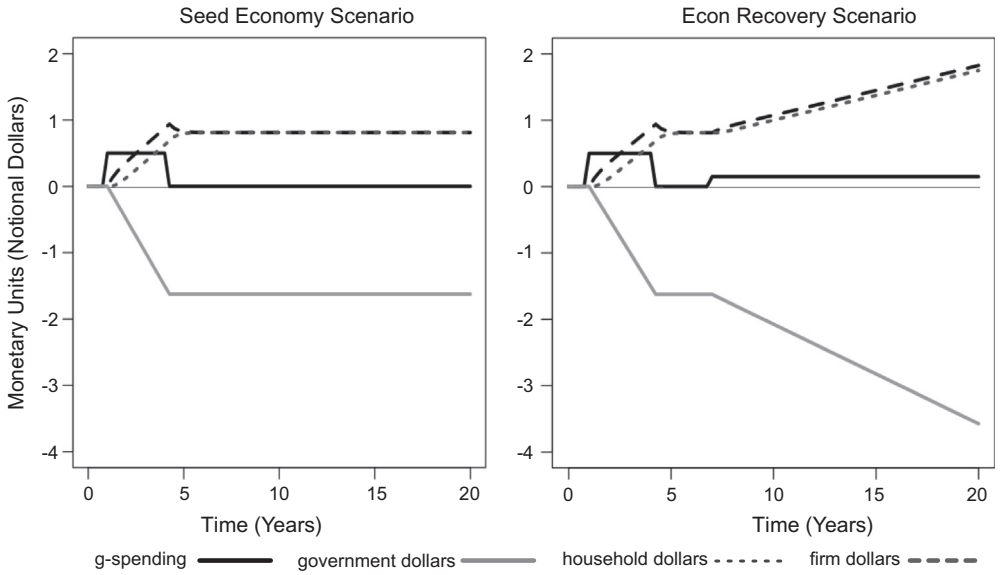


Fig. 7. Left-Panel, Seed Economy scenario results; Right-Panel, Econ Recovery scenario results. Notional dollars is the abstract unit of money used by the group in their model of a general economy. In keeping with common System Dynamics modeling practice, the group focused on dynamics and trends rather than specific quantitative magnitudes and values.

1. A general-purpose simulation and modeling tool was built using web-based technologies that supports multi-paradigm modeling using a graphical model construction interface.
2. The simulation and modeling tool combined both features targeted at laypeople and individuals new to the modeling field (e.g. a graphical interface) with those targeted at more advanced users (e.g. API and scripting, the optimization toolset) within a unified interface.
3. The tool was iterated based on user feedback and its direction was shaped by the needs and problems faced by its users. This led to an evolution of the software that, though now somewhat different from the original vision, better met the needs of the majority of users interested in using a general web-based modeling tool.

4. Usage of the tool continues to grow, indicating the existence of a need and desire for a general web-based modeling tool and validating the web as a platform for developing complex simulation and modeling tools. As of February 2014, the software has some 20,000 registered users with over 12,000 models constructed and saved in its database.

Along the way, challenges and points of resistance from users were encountered in the development of the application that stemmed from its nature utilizing web-based technologies. Four key challenges have been:

1. *Security of intellectual property:* Models are stored on a centralized server not controlled by the users.[6] For most users this is acceptable (and they have the ability to back-up the model to their local machines). However, for certain users – especially those within a corporate environment – lack of control over the model may be viewed as a security risk. Although Insight Maker incorporated fine-grained settings to control model access, the fact that the model is stored on an external server made it inappropriate for some potential users.
2. *Performance:* Web-based technologies are fundamentally limited in the performance they can obtain compared to their counterparts developed in more traditional languages. That said, the performance of Insight Maker is quite good in practice and more than adequate for its main target usage cases.
3. *Unfamiliarity of web-based environments:* Many users at present are not comfortable using a simulation and modeling tool within their web browser. It is expected that this will change as people become more familiar with using rich applications within browsers. Such a trend was seen historically with email clients where once Desktop clients were the norm, but now web-based clients are ubiquitous.
4. *Perceived application quality:* A final challenge centers on a perception that web-based applications are inferior to their desktop counterparts. In the case of the modeling and simulation tool presented here, it is in some ways superior in capabilities and features to even commercial counterparts. However, the stigma and experience of the restrictive web applications that users may have come into contact with in the past is difficult to overcome. Again, this is a difficulty that is expected to diminish over time as users become more familiar with high-quality interactive web-based applications.

Despite the challenges, modeling and simulation tools developed using web-based technologies have a distinctly positive future. The application presented in this paper demonstrates the possibilities to create extensive modeling and simulation tools in a web-based environment. Given the steady improvement in web technologies, the possibilities available to build such applications will only grow over time. The benefits offered to developers using web-based technologies – increased accessibility, ability to push updates to users immediately, a rich toolkit of open-source libraries, etc. – are starting to outweigh the costs of web-based technologies in cases other than those where performance is the key factor. It is not hard to imagine a future where web-based technologies are the primary platform for developing simulation and modeling tools outside that of performance critical applications.

Acknowledgements

This work was supported by a United States National Science Foundation Grant number DGE 1106400. The two anonymous reviewers are thanked for their feedback and numerous suggestions that greatly improved this paper. Special acknowledgement should be given to the users of Insight Maker and the invaluable feedback they have provided that has shaped its evolution, notably, Gene Bellinger and Geoff McDonnell. Great thanks is given to the group who developed the Investment versus Austerity model described in the case study, and to Gene Bellinger for organizing and encouraging the group.

Appendix A

This appendix presents the full set of equations used in the Investment versus Austerity model and their values given the application of the scenarios. Primitive names may appear within multiple equations, if the same primitive is referenced multiple times. This is seen, for instance, when applying the effect of tax. The fraction of firm capital paid in tax – *ftax* – is used to calculate both how much is paid to the government by firms and how much is consequently available for saving by firms. The following Insight Maker functions and notations were used in the model:

Step(Start, Height): Returns 0 until $t = Start$, after which it returns *Height*.
Pulse(Start, Height, Width): Returns 0 until $t = Start$ and after $t = Start + Width$. In between these bounds, returns *Height*.
INTEGRATE(Equation): Integrates the differential equation using the integration method and time step specified for the model. This is not written within Insight Maker, however, it is included here to make clear the integration process being carried out during simulation. Applies to stocks only.

[6] When InsightMaker.com is used, all models must be saved on the centralized server. The Insight Maker software is open-source and users can download it and set it up on their own computers. But doing so may be quite difficult for users who are not highly technically proficient.

Full model equations:

Primitive name	Description	Initial value/equation (scenarios may change equations)
fborrow	Fraction of *f-finance* dollars borrowed	0
fsave	Fraction of after tax firm dollars saved	Step(6, 0.05)
ftax	Fraction of tax paid by firm	Step(6, 0.05)
gspend	Variable used to alter Government spending during the simulation run	Pulse(1, 0.5, 3) + Step(7, 0.15)
hborrow	Fraction of household finance (*h-finance* dollars) borrowed	0
hsave	Fraction of household income saved	Step(7, 0.1)
htax	Fraction of household income paid in taxes	Step(7, 0.1)
mpmt	Fraction of household dollars spent on imported goods and services	0
xpmt	Fraction of trade dollars that result in firm dollars for exported goods and services	0
export pmt	Payments for exports	0
f-finance dollars	Balance between firm and household savings less firm and household investments	INTEGRATE([f-save] − [f-borrow]) Value $t_0 = 0$
firm dollars	Funds available from export payments, household goods payments, and investments which are available for input payments and savings	INTEGRATE([f-borrow] − [f-input pmts] − [f-save] − [f-tax] + [g-spending] + [export pmt] + [h-goods pmts]) Value $t_0 = 0$
government dollars	Difference between government spending and tax receipts	INTEGRATE([f-tax] + [h-tax] − [g-spending]) Value $t_0 = 0$
h-finance dollars	Balance between firm and household savings less firm and household investments	INTEGRATE([h-save] − [h-borrow]) Value $t_0 = 0$
household dollars	Funds available for household spending	INTEGRATE([f-input pmts] + [h-borrow] − [h-save] − [h-tax] − [h-goods pmts] − [import pmt]) Value $t_0 = 0$
trade dollars	Difference between import payments and export payments	INTEGRATE([import pmt] − [export pmt]) Value $t_0 = 0$
f-borrow	Funds borrowed as investment	[f-finance dollars] ∗ [fborrow]
f-input pmts	Firm input payments. This is firm inputs less savings	[firm dollars] ∗ (1 − [fsave])
f-save	Portion of firm dollars used to pay down debt or retained earnings	[firm dollars] ∗ [fsave] ∗ (1 − [ftax])
f-tax	Firm taxes paid on retained earnings	[firm dollars] ∗ [fsave] ∗ [ftax]
g-spending	Funds injected into the economy by the government. This is as direct purchases, program support, and social programs	[gspend]
h-borrow	Funds borrowed to supplement household spending dollars	[h-finance dollars] ∗ [hborrow]
h-goods pmts	Household spending producing demand. This is household dollars less household taxes and savings	[household dollars] ∗ (1 − [hsave] − [htax] − [mpmt])
h-save	Portion of household dollars used to pay down debt or saved	[household dollars] ∗ [hsave]
h-tax	Household taxes	[household dollars] ∗ [htax]
import pmt	Payments for imports	0

References

[1] Modelica Association, Modelica – A Unified Object-Oriented Language for Systems Modeling Version 3.3, 2012, p. 1–282.
[2] E. Eyisi, J. Bai, D. Riley, J. Weng, W. Yan, et al, NCSWT: an integrated modeling and simulation tool for networked control systems, Simul. Model. Pract. Theory 27 (2012) 90–111, http://dx.doi.org/10.1016/j.simpat.2012.05.004.

[3] N. Saquib, M.S.R. Sakib, A.-S.K. Pathan, ViSim: a user-friendly graphical simulation tool for performance analysis of MANET routing protocols, Math. Comput. Model. 53 (2011) 2204–2218, http://dx.doi.org/10.1016/j.mcm.2010.08.026.
[4] W. Li, L. Vanfretti, Y. Chompoobutrgool, Development and implementation of hydro turbine and governor models in a free and open source software package, Simul. Model. Pract. Theory 24 (2012) 84–102, http://dx.doi.org/10.1016/j.simpat.2012.02.005.
[5] J. Byrne, C. Heavey, P.J. Byrne, A review of Web-based simulation and supporting tools, Simul. Model. Pract. Theory 18 (2010) 253–276, http://dx.doi.org/10.1016/j.simpat.2009.09.013.
[6] A. Syberfeldt, I. Karlsson, A. Ng, J. Svantesson, A web-based platform for the simulation–optimization of industrial problems, Comput. Indust. Eng. 64 (2013) 987–998, http://dx.doi.org/10.1016/j.cie.2013.01.008.
[7] A. Levytskyy, H. Vangheluwe, L.J.M. Rothkrantz, H. Koppelaar, MDE and customization of modeling and simulation web applications, Simul. Model. Pract. Theory 17 (2009) 408–429, http://dx.doi.org/10.1016/j.simpat.2008.10.004.
[8] X. Liu, Q. He, X. Qiu, B. Chen, K. Huang, Cloud-based computer simulation: towards planting existing simulation software into the cloud, Simul. Model. Pract. Theory 26 (2012) 135–150, http://dx.doi.org/10.1016/j.simpat.2012.05.001.
[9] H. Wang, H. Zhang, Using collaborative computing technologies to enable the sharing and integration of simulation services for product design, Simul. Model. Pract. Theory 27 (2012) 47–64, http://dx.doi.org/10.1016/j.simpat.2012.05.002.
[10] E. Motahar, Teaching modeling and simulation in economics: a pleasant surprise, J. Econ. Educ. 25 (1994) 335–342.
[11] G. Ossimitz, Teaching system dynamics and systems thinking in Austria and Germany, in: System Dynamics Conference. Bergen, Norway, 2000.
[12] J. McGrenere, G. Moore, Are We All In the Same "Bloat"? Graphics Interface 2000, Montreal, Canada, 2000, pp. 187–196.
[13] J. Bezanson, S. Karpinski, V.B. Shah, A. Edelman, Julia: A Fast Dynamic Language for Technical Computing, 2012. arXiv, preprint arXiv:12095145.
[14] Society MSCOTIC, IEEE Std 754™-2008 (Revision of IEEE Std 754–1985), IEEE Standard for Floating-Point Arithmetic, 2008, pp. 1–70.
[15] J.W. Forrester, Industrial Dynamics, MIT Press, Cambridge, Massachusetts, 1961.
[16] J.W. Forrester, Urban Dynamics, MIT Press, Cambridge, Massachusetts, 1969.
[17] D. Meadows, J. Randers, D. Meadows, Limits to Growth: A report for the Club of Rome's Project on the Predicament of Mankind, Universe Books, New Yotrk, 1972.
[18] M. Wooldridge, An Introduction to MultiAgent systems, John Wiley & Sons, 2009.
[19] E. Bonabeau, Agent-based modeling: methods and techniques for simulating human systems, Proc. Nat. Acad. Sci. 99 (Suppl. 3) (2002) 7280–7287, http://dx.doi.org/10.1073/pnas.082080899.
[20] V. Grimm, Pattern-oriented modeling of agent-based complex systems: lessons from ecology, Science 310 (2005) 987–991, http://dx.doi.org/10.1126/science.1116681.
[21] R. Hooke, T.A. Jeeves, "Direct Search" solution of numerical and statistical problems, J. ACM (JACM) 8 (1961) 212–229.
[22] M. Powell, Direct search algorithms for optimization calculations, Acta Numer. 7 (1998) 287–336.
[23] T.G. Kolda, R.M. Lewis, V. Torczon, Optimization by direct search: new perspectives on some classical and modern methods, SIAM Rev. 45 (2003) 385–482, http://dx.doi.org/10.1137/S0036144502428893.
[24] J.A. Nelder, R. Mead, A simplex method for function minimization, Comput. J. 7 (1965) 308–313.
[25] J. Duggan, Equation-based policy optimization for agent-oriented system dynamics models, Syst. Dyn. Rev. 24 (2008) 97–118, http://dx.doi.org/10.1002/sdr.393.
[26] D. Bertsimas, J. Tsitsiklis, Simulated annealing, Stat. Sci. 8 (1993) 10–15.
[27] J.C. Meza, R.S. Judson, T.R. Faulkner, A.M. Treasurywala, A comparison of a direct search method and a genetic algorithm for conformational searching, J. Comput. Chem. 17 (1996) 1142–1151.
[28] M.J. Powell, An efficient method for finding the minimum of a function of several variables without calculating derivatives, Comput. J. 7 (1964) 155–162.
[29] J.W. Forrester, P.M. Senge, Tests for Building Confidence in System Dynamics Models. System Dynamics Group, Sloan School of Management, 1979.
[30] Y. Barlas, Formal aspects of model validity and validation in system dynamics, Syst. Dyn. Rev. 12 (1996) 183–210.
[31] M. Maybury, O. Stock, W. Wahlster (Eds.), PeaceMaker: A Video Game to Teach Peace, Intelligent Technologies for Interactive Entertainment, Springer, Berlin Heidelberg, 2005, pp. 307–310.
[32] J.D. Sterman, Flight simulators for management education, OR/MS Today 19 (1992) 40–44.
[33] B. Bakken, J. Gould, D. Kim, Experimentation in learning organizations: a management flight simulator approach, Eur. J. Oper. Res. 59 (1992) 167–182.
[34] E.M. Maximilien, L. Williams, Assessing test-driven development at IBM, in: 25th International Conference on Software Engineering, 2003, pp. 564–569.
[35] G. Bellinger, About Systems Thinking World, n.d. <http://www.systemswiki.org/index.php?title=About_Systems_Thinking_World> (accessed 16.02.14).
[36] S. Kelton, Implications of Modern Sovereignty and Stock-Flow Consistency. Modern Money and Public Purpose 2, Columbia Pre-Law Society, 2012.