HBM-view: Designing a Model Viewing Tool for Behavioral Scientists

Tylar Murray
University of South Florida
Department of Electrical Engineering
tylarmurray@mail.usf.edu

Daniel Rivera
Arizona State University
Control Systems Engineering Laboratory
Tempe, Arizona

Eric Hekler
Arizona State University
School of Nutrition and Health Promotion
Tempe, Arizona

D. Spruijt-Metz
University of Southern California
Center for Economic and Social Research
Los Angeles, California

Andrew Raij
University of Central Florida
Institute for Simulation and Training
Orlando, Florida

***Abstract*—In this paper we present definitions and design considerations relevant to human-behavior modeling software developers. The guidelines presented here are based off of a user-survey and expert-panel reviews performed in development of the BehaviorSim model-building tool. Lessons learned through iterations of this tool and unique considerations for those targeting behavioral scientists are highlighted. Our initial survey of 12 behavioral scientists reveals the diversity of opinions and approaches to behavior modeling within the community and gives context to this emerging area of research. In addition, a theory-agnostic method for defining Human Behavior Models (HBMs) is proposed and techniques for supporting different modeling paradigms within a single user interface are discussed.**

I. Introduction

A. Problem Statement

With the increasing prevalence of wearable technologies and user-behavior collection, behavioral researchers are becoming increasingly overwhelmed with data. Researchers have seen the potential for optimizing a behavioral intervention to suit not only the user, but also the context. Just-in-Time Adaptive Interventions (JiTAIs) promise to provide the optimal nudge at the optimal time to aid users looking to change their behavior [**?**]. Imagine, for example, an anti-stress app which knows not to interrupt your work meetings, but also knows when to play your favorite song to help you relax on the drive home. Or consider a smoking cessation app that knows precisely when and where you are most likely to crave, and distracts you with a game before you realize you were about to want a cigarette.

The impending wave of context-aware[**?**], affective computing[**?**] applications seems to be the holy grail of behavior change, but researchers are finding that our models of human behavior are not ready. Traditional models of behavior do not offer the granularity and specificity needed for these applications[**?**]. Existing psychological models of human behavior act as a guide for behavioral scientists looking to predict behavior, but are rarely computational in nature - making their application to adaptive systems impossible.

State-of-the-art user interface predicts user behavior in order to preload data or to customize the application. Though the machine learning approaches which enable detailed behavior-prediction can be applied to behavioral research problems, the inner workings of these models often do not tell researchers much about how the human system works. These empirically founded models, though useful for exploring specific cases in detail, also do not meet the requirements of a JiTAI. The search space of intervention design is very large, and it must be narrowed using our a priori understanding of the human system and guided by the heuristic methods of traditional modeling.

A hybridization of probabalistic and more traditional cogntive modeling may allow researchers to leverage the strength of each as they are applied in adaptive behavioral interventions. However, methods for creating computational models of human behavior are not well defined, but some early concept have been published.

B. Human Behavior Modeling

In this work we define a Human Behavior Model (HBM) as any computational model which attempts to define the human systems conversion of measurable contextual information into measurable behavior outcome. HBMs are a set which intersects the set of computational cognitive models, both sets include approximations of cognitive processes, but in contrast with cognitive models, HBMs put a greater emphasis on the measurable inflows and outflows of the human system. Cognitive models traditionally have focused on creating a comprehensible representation of cognitive processes, but HBMs have no such restriction and require only that a model define behavioral output as some function of contextual input. Thus HBMs include formulations which may not be considered a cognitive model. For instance, a model which offers little more than statistical relationship between contextual input and behavioral output may be considered an HBM.

For our purposes, we classify HBMs uncovered by our literature review into three classes based on intended application 1) Automation Agents[**?**] - models developed to act as automated decision agents, 2) Emergent Behavior Study Agents[**?**] - models developed to identify emergent behavior of a population as a system, and 3) Cognitve HBMs - models developed for prediction and understanding of an individual's

behavior. Note that these categories are not mutually exclusive - a single HBM could theoretically fufill all three roles, but the requirements for each category are quite different.

Though there is much which can be learned from models built to act as automation agents or as agent models in emergent behavior studies, our primary focus is on the emerging group referred to as Cognitive HBMs (CHBMs). CHBMs fill the gap between the heuristics laid out by existing psychological models and systems modeling, offering computational expressions of the easily understood abstractions of psychological models in computational terms which can be used in simulation.

Because CHBMs offer both numeric representations of behavior and an underlying heuristic model they can be used in both intervention design and an automated intervention-optimization system such as would be required by a JiTAI.

C. Existing Tools

Our review of existing tools and publications in the area of behavioral modeling has uncovered three types of tools: 1) Dynamical System Modeling Toolkits, 2) Cognitive Architectures, and 3) Agent Modeling Toolkits. Though each approach seeks to address the problem of human behavioral modeling, the intended use of the model is very different. Thus each approach makes an approximation of the human system from a different perspective, and the resulting models show little resemblence to one another. These three types of tools are in partial alignment with the aforementioned three classes of HBMs, but much crossover exists.

1) Dynamical System Modeling: Works in this group focus on generalized model-building. Applications include network and business analysis as well as semi-physical system modeling. One might be able to use these systems for modeling a human agent, but a more rigid model architecture for organizing the flow of information is needed. Though generalized modeling software is available [**?**], [**?**], [**?**] Specialized modeling toolkits are often developed to create a variety of detailed models focused around a single modeling goal. In our search, specialized toolkits focusing on neurological simulation[**?**], molecular dynamics[**?**], [**?**], astrophysics[**?**], and power system modeling[**?**] were all encountered. The behaviorSim model building tool explored in this publication aims to help fill this gap by creating a specialized framework in the area of human behavior modeling.

2) Cognitive Architectures: Cognitive architectures[**?**] are modeling toolkits which aim to create artificial intelligence through the modeling of cognitive processes. Cognitive architectures most often focus on the fomal logic of thought processes, taking a much less empirical approach to the task than dynamical systems modeling.

3) Agent Modeling: Agent modeling has proven useful in many industrial and commercial applications across multiple disciplines. [**?**], [**?**] Agent models are generally developed for one of two purposes: 1) to act as an intelligent actor in an automated task, or 2) to observe the emergent characteristics of many interacting agents.

II. Case Studies

Here we present findings from multiple iterations of development of a tool designed to help behavioral scientists forumlate computational models of human behavior. After casting a wide net initially to characterize the targeted use-case, evaluation at each step of this iterative process small numbers of expert testers as guided by industry practice [2].

A. Paper-prototype Model-Building Task

A paper prototype served as the first iteration of the model-builder tool. This prototype was simply a quiz-style questionairre which resulted in a model. This questionairre was tested on a small user group of both behavioral scientists and graduate students, and was evaluated informally through discussion.

This prototype allowed us to identify some major challenges early on. Namely, we found that careful choice of terminology was important, since some overlapping terms were the cause of significant confusion. Reviewers also found the tasks presented much too challenging and strongly suggested the inclusion of an example model.

B. SBM Questionairre

In the second iteration we focused on a few key elements of the model building process to greatly simplify and shorten the modeling exercise. A context and behavioral outcome based on physical activity were given, and users efforts were focused on defining the inner workings of the human system within these constraints. Reviewers of this iteration were also aked to complete several survey items about the barriers keeping modeling and sim out of behavioral science.

Approximately 50 surveys were distributed following presentations on behavioral modeling and simulation at the 35th Annual Conference of the Society of Behavioral Medicine. Out of these 50, only 12 surveys were returned complete. In general, users still had trouble with modeling exercise. Most did not stray far from the given example, and others provided very different solutions which do not fit the “engineer’s view” of the problem.

In the survey questions participants reported that the mathematics and programming concepts required for developing simulateable models were overwhelming, and nearly all participants expressed a desire for increased collaboration between disciplines and a need for software tools to help them apply and validate these methods.

C. behaviorSim Model-Builder v1

Following our paper prototypes, a software tool to further explore the concept was developed. The behaviorSim Model-Building tool is a computational model-development environment aimed at behavioral scientists. In this iteration the modeling process is divided into the following five stages:

think - Write out an explanation of your model. How does it work? What are the constructs involved? What kind of information is coming into the model?

draw - Create an ”information flow” path diagram to represent your theory. Draw the flow of information between

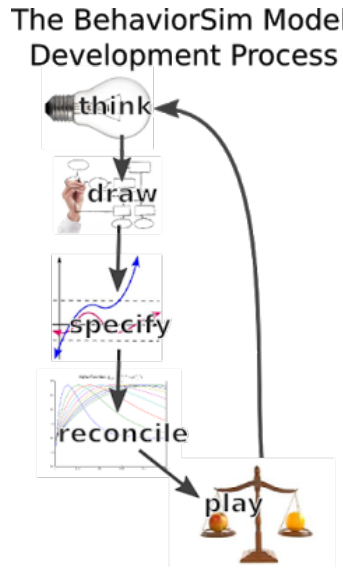

Fig. 1. The five stages implemented in v1 of the behaviorSim Model Builder.

constructs. What are the instreams to each, what are the outflows of each?

specify - How exactly can each construct be calculated from the inflows? Use a fluid-flow analogy or linear combination to define a starting function and then adjust weights, delays, etc until the construct behaves as you think it should. Do lots of thought experiments. What should each step response look like? What should each impulse response look like?

reconcile - Compare your simulation to real data. Compare individual data and group data. Compare the data at different time-scales. How can you adjust the constants in your model to match the data? Use semi-physical and grey-box model optimization methods. Specify which constructs are most important to match and which may be less important by adjusting the cost function.

play - Explore all the 'what ifs' of your model. Use system identification techniques on data to inform the next iteration of your model. Is your model missing anything? Is there anything in the data which you cannot explain?

The first iteration of the behaviorSim Model-Builder (see figure 2) took a step-wise approach to implementing these stages. This tool focuses on the first three stages (think, draw, specifiy), as the problem of data reconsiliation and analysis falls outside the scope of our basic model building tool.

In this tool, an information-flow diagram is laid out in a stepwise manner. First users are asked to list environmental inflows, internal state variables, and behavioral outflows of the model explicitly during the "think" stage. Next users are prompted to define the connections between nodes, "draw"ing the model's structure. Lastly, users are required to "specify" the functional relationships at eaach node's inflow(s).

The model builder was tested using a think-aloud protocol with 2 behavioral scientists and 1 HCI expert. Though the steps in the outlined model development process are accurate, a step-wise design is not optimal. Users who are forced to explore

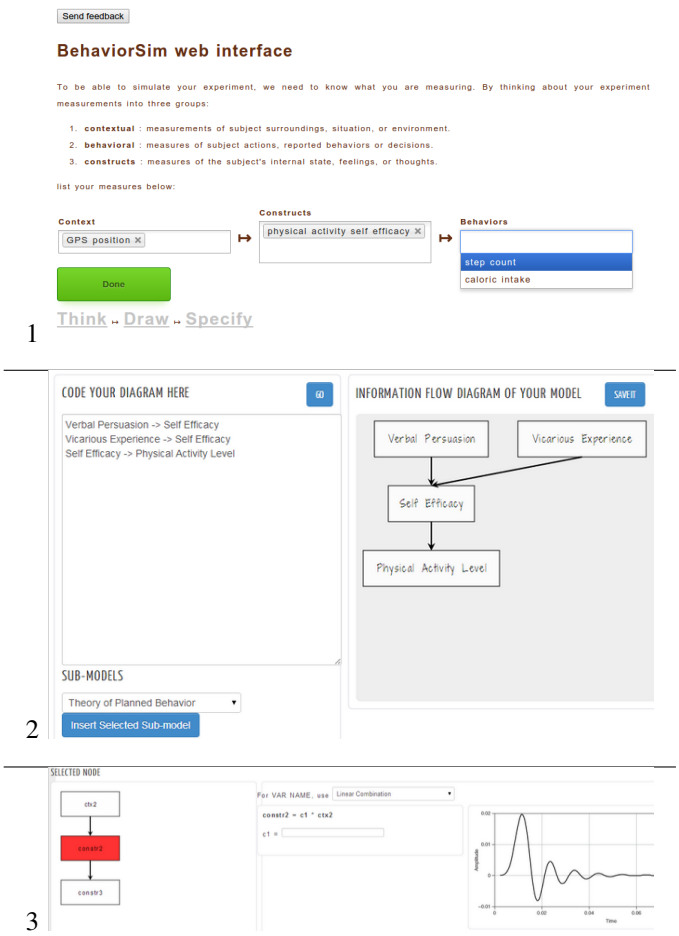

Fig. 2. behaviorSim Model Builder v0.1.x implemented three separate stages of model building as separate pages: 1) think, 2) draw, 3) specify.

the process step-by-step have difficulty understanding how earlier choices related to later results, and feel constrained by previous choices rather than backtracking to revise the model. This design does not allow for quick iteration on models, and requires the user to maintain a great deal of planning information internally. Though the information flow diagram employed in this version worked well to convey information about the model to the users, the graph was also assumed to be interactive - participants made attempts to modify the graph by clicking, and attempted to select nodes in the specification stage by clicking on them.

Our review concluded that a less literal approach to the five stages of the modeling process was needed, and a greater focus on the graphical model could greatly improve user experience.

D. behaviorSim Model-Building Tutorial v1

After reviewing v1 of the behaviorSim Model Builder tool, a tutorial was designed to help bridge the knowledge gap for new modelers looking to use the tool. In theory, the tutorial would help users see the bigger picture before diving into the step-wise process.

The tutorial was implemented as a walk-through of a simple model's internals. The tutorial also introduced a hybridized information-flow and time-series graph, wherein each

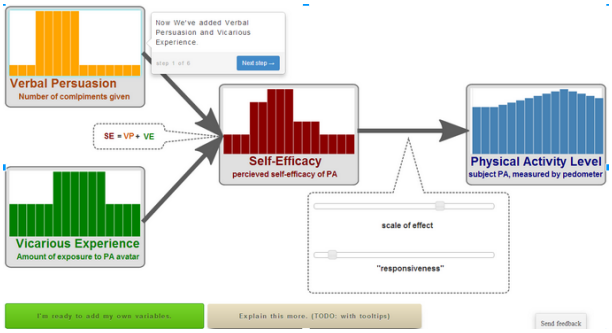

Fig. 3. behaviorSim Model Builder tutorial attempted to merge time-series and information-flow graph representations.

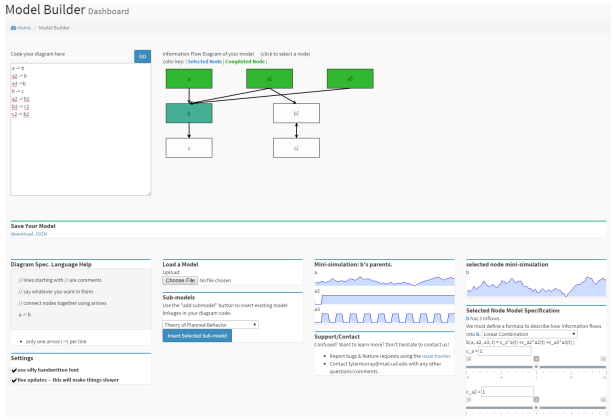

Fig. 4. behaviorSim Model Builder v2 combines the think, draw, specify, and play elements into a single view.

node of the graph contains a time-series spanning a common timeframe, and a user interface for adjusting model parameters and updating time-series values instantaneously. This real-time parameter tweaking enables some degree of reconsiliation with expectations of the data, thus addressing the reconsiliation and play stages missing from v1. These stages, while critical for the development of an accurate model, were also foud to be very valuable as pedagogical exercises for users trying to internalize model formulations.

The same think-aloud protocol as used for the evaluation of v1 of the model builder was used to evaluate this tutorial. Through this evaluation it became clear that even more explicit definition of terms was needed in order to clarify persistant diciplinary differences. Reviewers also wanted better explanation of the models' input parameters and of the functional definition of the system. They were not content with the hard-coded environmental inputs and wanted to be ablet to define how the contextual inflows changed over time. Though the hybrid graph was found helpful in conveying the connection between path diagram nodes and time series, the shared time-axis was not obvious, and reviewers expressed a need for more explicit x and y axes as well as a better explanation as to what "10 units of self efficacy" at some point in time actually means.

E. behaviorSim Model-Building Tool v2

The second major version of the behaviorSim Model Building tool attempted to unify all steps of the modeling process (figure 1) into a single-page application. In this way, users can see how choices influence the model in real-time, and thus can iterate on their design more easily. To address the terminology gap which plagued v1, a set of tooltips were added which revealed detailed definitions for key terms used in the user interface. In addition, the second version incorporates findings from the v1 tutorial, adding a "miniature simulation" to the application to allow for "reconsiliation" and "play".

In this version of the tool, users declare constructs and define the structure of their model simultaneously by specifying on the connections using a simple Diagram Specification Language (DSL). The type of each node is then inferred from the number of inflows and outflows. This approach combines stages 1 (think) and 2 (draw) in into a single user action.

The "miniature simulation" concept allows users to specify hypothetical contexts in which to explore the model dynamics, without the need to specify the full model. Users can specify environmental inflows for the "miniature simulation", choosing from adjustable presets (square wave, step function, random walk, or constant value). Selecting nodes on the graph by clicking, users can traverse the graph in any order. Time-series plots of inflows as well as the resulting outflow at each node are provided using the miniature simulation model. Internal state and behavioral nodes are specified similarly to environmental inflows through customization of function presets such as "linear combination" and "fluid flow analogy"[**?**].

This version was again reviewed using a think-aloud protocol with 2 behavioral scientists, 1 HCI expert, and 1 behavioral modeler. Though the single-page design of version two did seem to allow for increased ability to iteratively explore models, reviewers now found the user interface overwhelming. Upon starting the review, users often felt unsure where to start. Furthermore the disconnect between the information flow graphic and the related UI elements below was not obvious. Reviewers did identify the relationship after some exploration, however. The combination of the "think" and "draw" stages in this design broke down the distinction between einvironmental inflows, state variables, and behavioral outputs, leading to to some confusion when specifying the various types of nodes. Further contributing to this problem, the purpose of the mini-simulation was not clear to reviewers, though the inclusion of time-series graphs was found helpful for understanding the model functions and their parameters.

III. Design Guidelines

Using lessons learned from our attempts at developing a model-building tool for behavioral modeling we propose a graph-based HBM specification, and provide generalized guidelines for developing modeling and simulation tools for behavioral scientists.

A. HBM info-flow graph

A "standard" approach to describing, designing, and visualizing human behavior models for personal forecasting is needed. There are countless approaches to systems modeling, but concepts and methods ought to be incorporated and fused when possible in their application to behavioral science. The resulting specification will allow for the formal description of a Human Behavior Model (HBM).

We will begin with the most basic definitions and then progress into the intricacies and usage examples of the proposed model-modeling paradigm.

A model is comprised of a network of variables and their interrelations. In order to visually represent the connectivity of the model a network graph can be used. Network graphs have many applications across multiple disciplines, and though the different implementations can be visually similar they can differ significantly in their meaning. In general, however, a graph is comprised of nodes and edges which represent the variables and their connections, respectfully.

We argue that the most intuitive representation is one which uses a directed graph wherein edge arrows to represent the flow of information between nodes. Thus, a HBM directed graph edge from node A to node B indicates that information flows from node A into node B.

$A => B$

Graph 1: Can be read as "A influences B", "A informs B", or similar

This choice of notation is in agreement with graphs used in information theory, communications models, and behavioral science. In contrast, some graphing paradigms (such as probabilistic graphical models) prefer to use notation wherein an edge is used to represent dependency. In these paradigms edges may be read from tail to head as "depends on" or similar.

While the network graph does an excellent job showing the connectivity of a model, it fails to indicate the meaning of each connection. In the majority of existing applications, the mathematical form of the relationship is implied or else it is neglected completely. For instance, path diagrams from the behavioral sciences frequently denote dependence and do not specify functional form. Adding even further to the confusion is the notion that these graphs are often developed using different statistical analyses which may make different assumptions about the functional definition of inter-variate dependency. The most common analyses assess linear relationships between variables, and thus it is perhaps reasonable to assume that this is the intention of most authors. Assuming this is the case we can return to our simplistic example in Graph 1 and interpret the implied relationship as:

$$B(t) = coeff_{ab}A(t) + const_b$$

Equation 1: $coeff_{ab}$ represents the correlation coefficient which relates A to B, and $const_b$ represents a scalar constant.

For nodes with multiple inflow edges, such as node B in the following graph:

$A => B <= C => D$

Graph 2: Example; now with more nodes!

Continuing with our our assumption that node interrelations act as linear sums, the resulting formulation is simply a sum of the inflows:

$$B(t) = coeff_{ab}A(t) + coeff_{cb}C(t) + const_b$$

Equation 2: Combining multiple inflows via sum is referred to as the superposition principle and is the defining characteristic of linear systems (not to be confused with this linear formulation)

Using this formulation, the general form of our HBM is expressed via the network graph alone (perhaps along with a statement about what edges mean). To express an ideographic implementation of this general model, we must also include a table of coefficient values. This representation is useful, but also very limited. One important feature which this formulation does not take into account is the dynamics of the relationship. This is very important for human behavior modeling because the variables in a HBM will often change their value over time. Taking time into account our formulation for Graph 1 becomes:

$$B(t) = coeff_{ab}A(t) + const$$

Looking at this closely we can note that at each point in time the value of A depends only on the value of B at that same instant in time. This assumption is fine for many applications, but we argue that this is a very poor assumption for human behavior models. What if A is influenced by B but there is some lag before the effect manifests? What if A is influenced by B, but only if B is above some threshold value? What if A is influenced by the rate of change in B rather than the value of B itself? All of these scenarios cannot be expressed using this simple linear relationship.

To resolve some of these issues, a differential equations can be used to describe the relationship between variables as described by (CSEL paper which connects path diagrams and fluid-flow). Using the differential formulation our equation for Graph 1 becomes:

$$B(t) = coeff_{ab}A(t - \theta_{ab}) - \tau_b\frac{dB}{dt} + const$$

Just as before, our general model is not expressed entirely through the graph, and an ideographic example is specified by providing table of coefficient values. Our table is now quite a bit larger, but these coefficients have meaningful definitions which relate to our theory. While this formulation offers a huge improvement over the linear formulation, we can still imagine relationships which this formulation cannot express. Thus, a graph-wide assumption that each edge represents a differential equation may not be general enough for our HBM specification.

It should be noted at this point that although the linear formulation is too simple to express the dynamics of the differential formulation, the differential formulation is capable of expressing linear relationships. This is accomplished by setting coefficients of dynamical components to 0. One might think then that there is some general formula which could express any functional form and that this form should be used to express the relationships between variables in our HBM graphs. While such formulations do exist (such as Taylor or Fourier series approximations or even ANN-based relations), this usage tends to make the model difficult to understand and to simulate with. Indeed, linear and differential formulations are in such widespread use because of the relative ease with which we can understand and solve them. An additional problem raised by this approach is that of redundant formulations. Indeed formulations could even subvert the "arrow direction" by drawing information contrary to our original choice of notation. The graph can then no longer be considered a directed graph, and the meaning of an edge becomes less clear. Additionally, the table of coefficients needed to express an ideographic case of the model quickly becomes prohibitively

large, and the effect of each coefficient on the outcome is not intuitively meaningful.

Let us now consider the case where a graph-wide assumption is *NOT* made. That is, we will specify the functional form of each node individually so that each edge on the graph may be linear in form while another may be differential. This has the benefit of allowing for both complex relationships between variables as well as simplistic ones. In this way one could craft a model in which two variables are linearly related and a third is dependent on the variance of another variable (a particularly odd formulation, but one which is relevant to behavioral theory). Unfortunately, this approach also means that a table of formulations must now be included with our graph to show the meaning of each edge in the graph. Consider for example the table below for Graph 2:

$A => B <= C => D$

node	formulation
B	$coeff_{ab}A(t) + coeff_{cb}C(t) + cosnst_b$
D	$coeff_{cd}C(t-\theta_{cd}) - \tau_d\frac{dD}{dt} + const_d$

Table 1: an example formulation table for Graph 2

If a fixed number of functional forms is adhered to, the graph can be made to visually represent these functional forms through the use of different node icon shapes. This approach quickly begins to resemble applications which use flow-based programming. Indeed, they are quite similar in their approach, and the specification of a HBM is quite similar to the writing of a program.

Note also that a node can set any arbitrary formulation if needed. This is the least desirable situation, since the meaning of an inflow to each node even more convoluted. Though this usage reduces the ability of the graph to reduce processing load on the user through abstraction, there may be some cases where this type of formulation is desirable. Consider the following graph and formulation table:

$A => B <= C => D => E$

Graph 3: The final example graph.

node	formulation
B	$coeff_{ab}A(t) + coeff_{cb}sqrt(C(t)) + const_b$
D	$coeff_{cd}C(t-\theta_{cd}) - \tau_d\frac{dD}{dt} + const_d$
E	$coeff_{de}D(t) + const_e$

Table 2: an example formulation table for Graph 3

In conclusion, we propose that an HBM should be specified using the following rules:

1) use a graph-wide formula assumption if possible
2) when choosing a formulation, consistency between nodes is most important
3) when choosing a formulation, simplicity and clarity is second only to consistency
4) Thus a HBM specification must include:
   a) An information flow graph
   b) one of:
      - a graph-wide formulaic assumption
      - a formulation table

In order to express a particular solution of a general HBM, an ideographic HBM must also include a table of coefficient values.

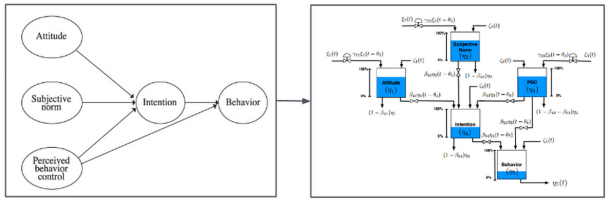

Fig. 5. Example of the rise in model complexity for the theory of planned behavior when mathematical assumptions are made explicit. (from: Ajzen 1985,1991,2002 (left), Nandola et al. 2013 (right)).

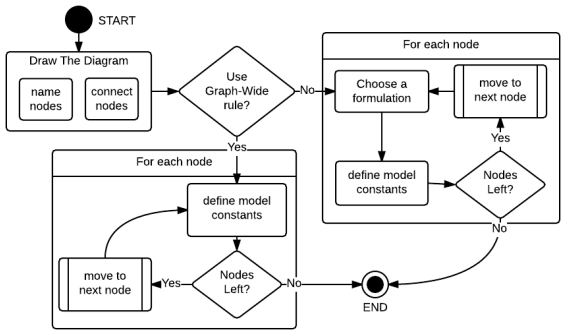

Fig. 6. Flow chart depicting the process of building a graphical HBM.

B. Tool Design Guidelines

The guidelines provided below are generalizable findings from our studies which may be useful for other developing modeling and simulation tools for behavioral scientists.

Ease of use and intuitive user interface is a primary design consideration for existing softwares for modeling and simulation in systems theory. (reference and cite vensim, etc, from here) The ease of use of a system is one of the most important factors in the adoption of the system. (cite technology acceptance model?) The trans-disciplinary nature of a human-behavior modeling toolkit may require special consideration in order to be usable by those who may be unfamiliar with modeling and simulation concepts.

1) Multiple Levels of Information Depth: * information-searching / focus and context design * use existing terminology (but also define it) * multiple terminologies from user surveys /citeontology customizable?

2) Walkthrough Instead of Tutorial: * walk-through first use instead of tutorial (see, copy, do) /cite?

3) Enable Quick Model Iteration: * multipage vs single-page (v0.1-¿v0.2)

IV. Proposed Tool: Model-Builder v3

A. User Interface Design for HBM-builder

The flow chart shown in figure 6 represents the movement of a user through the process of building an HBM, or any graphical model for that matter.

It is important to note that the process of building is pseudo-linear; users will likely want to jump back to an arbitrary point in the process and as the model takes shape.

As the user moves through the process, they will re-assess the meaning of variables and connections and their model will solidify. UI for an HBM-building tool should enable easy backtracking and should work to provide feedback on the characteristics of the model early.

While keeping the non-linearity of this process in mind, we will now walk through the diagram and address design considerations for each part of the process.

1) Draw the Diagram: This task involves the creation and connecting of all nodes in the diagram. Nodes and their connections should be primarily based in theory, so a user should be able to build a first draft of their model with little feedback. Once the formulas for each node have been specified, however, it is likely that the user will want to return to this step and reassess the network. Several paradigms diagram-drawing have been well tested, but there are essentially two prevailing options: 1) drag-and-drop node creation with drag-to-connect edges, and 2) textual "diagram specification language" (DSL).

Drag-and-drop modeling has the benefit of being intuitive, but nodes can be difficult to position to reduce edge overlap leading to users often distracted trying to optimize the layout.

The use of a DSL can be much faster than mouse-based, and also has the benefit that searching/finding nodes for modification is as easy as textual search of the DSL. On the other hand, DSL parsing errors can be a major source of confusion and users may be more frustrated as they attempt to master your markup language.

2) Use Graph-Wide Rule?: This choice can come before or after the initial drawing of the graph, since it is likely the user has a graph-wide assumption in mind when starting the process. This choice determines whether or not the interface for specifying nodes individually is shown, so it is logical to think of the "none", "node-specific", or "node-unique" option as one of the choices available here.

As an example, some graph-wide rules which may be implemented include:

1) Probabilistic Graphical Network
2) Linear Sum
3) Fluid-Flow Analogy / Differential Equation Formulation (1st order and 2nd order)

If a graph-wide rule is set, the formulation section for each node should show as locked to the rule's respective formulation. Editing an individual node's formulation will disable the graph-wide rule, and should require confirmation from the user.

The interface for choosing one of these rules may be as simple as a select-option box or may include a detailed rule description and example.

An additional feature which may be useful is to allow for user-defined graph-wide rules. In this case, users might choose "create new rule" and an interface to allow users to edit and save graph-wide rules is needed.

3) For Each Node: Items encapsulated by the "for each node" swim-lanes must be executed for each node in the graph. The graph-wide-rule-yes case differs from the graph-wide-rule-no case only in that the graph-wide-rule-no case requires the additional step of setting the formulation at each node.

4) Choose a Formulation: This item is identical to the graph-wide rule choice, except that it applies only to a single node. Similar design considerations apply, and user-defined choice is even more important at this scale.

5) Define Model Constants: The general solution of an HBM does not require definition of the constants, but a simulation cannot be run until some numerical value is assumed. These constants often have theoretical significance in that they often have meaningful influence upon system behavior. Scaling-coefficients, for instance allow for relative weighting of each inflow. Similarly, the coefficients of a dynamical equation define how quickly variables react to a change "upstream".

UI for setting these constants could be as simple as a numerical input for each coefficient's name, but additional model details provided here may be of great benefit to the model designer.

Firstly, UI should include equation-specific descriptions of the constants whenever possible. This means that a separate UI for each formulation type is necessary.

Secondly, adjustment of coefficients would ideally show changes in the dynamics of a node's value in real-time. This can take the form of a time-series chart of the selected node's numerical value over time along with (editable) time-series for the inflows. Adjustment of these inflow series may also take a number of different forms.

Thirdly, model creators may want to set these coefficients in three different ways:

a) 1) Set the coefficient value to a constant: All simulations derived from this general solution will have this exact value.

b) 2) Set a probability distribution for the coefficient: Simulations derived from this general solution will select a numerical value based on the given distribution. This distribution may be learned from a training dataset, or bounds may be set for later training.

c) 3) Leave the variable unbounded and assume a flat probability distribution: This selection is identical to option 2 with a flat distribution from -infinity to +infinity.

These different ways of defining coefficients in a HBM become very important when it is time to run simulations and compare model results to real data.

Lastly, model creators may want to use a measured value to set the value of these coefficients. In this way, one measured input (in real life), such as big-5 personality type, can be used to set the value of multiple coefficients across different variables in the model.

6) Nodes Left?: No if there are un-specified nodes remaining, yes if all are complete. It should be left to the user to confirm model-completeness, however, since a great deal of model-tweaking is likely now that simulation results can be displayed.

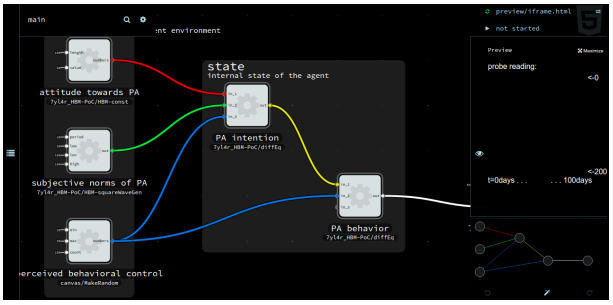

Fig. 7. behaviorSim Model Builder v3 brings increased focus onto the graph using a flow-based programming approach.

7) Next Node: Progression downstream through the model should be relatively straightforward in the case of tree-like models, but feedback loops can cause issues here. Perhaps the best paradigm here is to progress as naturally as possible, but allow the user to over-ride and select any node. This interaction mode is needed to allow for later model-tweaking anyway.

The behaviorSim Model Builder v0.2 currently does not include significant enough indicator of node ”completeness”, so that the user is sometimes unsure when they should feel free to move to the next node.

B. Implementation

Version 3 of the model builder addresses the issues uncovered in previous versions by relying on a flow-based programming interface. Flow-based programming has seen increased usage across a multiple engineering domains, and seems to fit the task of system specification particularly well. Using an open-source implementation of a flow-based programming protocol, noflo.js, the graphical representation of the model can be further connected to the formulation. Node inputs (constants, coefficients, other nodes) can be directly shown as input hubs on the node, and differing node formulations can be represented graphically.

V. Conclusion

We have provided design guidelines to aid in development of modeling and simulation tools for behavioral scientists based on lessons learned in the development process, and have outlined the design of an application which utilizes these guidelines.

References

[1] W. T. Riley, D. E. Rivera, A. A. Atienza, W. Nilsen, S. M. Allison, and R. Mermelstein, “Health behavior models in the age of mobile interventions: are our theories up to the task?” *Translational behavioral medicine*, vol. 1, no. 1, pp. 53–71, 2011.

[2] J. Nielsen, “Why you only need to test with 5 users,” 2000.