Erasmus Mundus European Master in System Dynamics

Automated Model Conceptualization and Interactive Modeling Environment: A Software Prototype

By

Wang Zhao

Thesis submitted in partial fulfillment of the requirements of

Master of Philosophy in System Dynamics (University of Bergen)
Master of Science in System Dynamics (New University of Lisbon)
and
Master of Science in Business Administration (Radboud University Nijmegen)

Supervised by

Prof. Dr. Pål I. Davidsen

System Dynamics Group
Department of Geography
University of Bergen

June 2019

Acknowledgement

It is a great pleasure to acknowledge my deepest gratitude to Prof. Pål Davidsen for his patient guidance and valuable advices throughout this research and development project. Many thanks for all the time for helping me and all the fruitful discussions in this semester.

I would also like to acknowledge the guidance and advices I received from Prof. Yaman Barlas. It was because of the wonderful discussions we had in Nijmegen that I finally decided to develop a software to test my hypotheses.

In this course of research, I was very often helped and advised by Dr. David Lara-Arango, Ph.D. candidate William Schoenberg (Billy), Ph.D. candidate Anaely Aguiar Rodriguez, and Prof. Birgit Kopainsky at University of Bergen. The discussions we had brought up great ideas. I specially appreciate their professional interest in my work and personal encouragement to me.

I would like to extend my sincere appreciation to Prof. Hubert Korzilius and Prof. Dr. Inge Bleijenbergh at Radboud University Nijmegen, for their instructions in the course 'Research Methodology' on developing a research topic, and specially to Prof. Korzilius for being my second reader from Radboud University.

Many thanks to Prof. David Lane for his helpful comments on this project.

It has been a great experience to participate in the 8th cohort of EMSD. I am grateful for the chance to spend these two amazing years with friends from all over the world.

Wang Zhao

June 2019
Bergen

Abstract

In system dynamics modeling process, modelers retrieve information from various sources to come up with a model that is able to reproduce a problematic behavior, then based on the model, policies are designed to alleviate this problem. Aiming at facilitating or automating this process, researchers have been focusing on integrating this modeling process with technologies from computer science, data science, and artificial intelligence.

Model conceptualization is a key step in this process. Sometimes it is also called 'structure generation'. It is about coming up with model structures based on available information on the problem or the situation where the problem is manifested. A series of research projects have been focusing on automating this conceptualizing step.

In the first part **'Introduction'**, the author will introduce basics of system dynamics modeling process, with a focus on the step 'model conceptualization' and information needed for this step.

In the second part **'Literature review'**, the author will guide a tour of important researches in the past 20 years on automated model conceptualization. In the end of this part, the author will discuss their methods and information sources they used for model conceptualization.

In the third part **'A platform for interactive model conceptualization'**, the author will propose a method for automated model conceptualization. The method is able to utilize information sources which have not been used in existing researches. The method has been preliminarily implemented as a demo software: A Python-based interactive modeling platform.

In the fourth part **'Experiments and results'**, features of the software will be demonstrated by carrying out three case-studies step-by-step.

In the last part **'Discussions'**, the author will summarize this project, discuss the innovations in this work, how these innovations could help the research of automation of model conceptualization, limitations to this work, and how the author plans to overcome these limitations in the future.

Table of Contents

List of Figures

List of Tables

1. Introduction

"System dynamics is a computer-aided approach to policy analysis and design. With origins in servomechanisms engineering and management, the approach uses a perspective based on information feedback and circular causality to understand the dynamics of complex social systems." (Richardson, 1991, pp. 144) Models are center to this methodology, as they are used to represent the understanding of a particular system in the real world in which a problematic behavior is observed and to be alleviated by policies. Model building, analyzing, and model-based policy design are key parts of system dynamics practice.

1.1 The system dynamics modeling process

This work is about automation of system dynamics modeling process. It is therefore necessary to delineate this modeling process at beginning. Although there is still not a sole definition accepted by all, throughout decades' research and practice, some opinions have been widely accepted, of which three highly followed ones are elaborated and discussed as follows.

In the context of working with clients, Sterman (2000, pp.86) defines the system dynamics modeling process as five steps:

Table 1 Steps in modeling process summarized from Sterman (2000)

No.	Step	What to do in this step
1	Problem Articulation	Make clear what to consider and time horizon, define the model boundary, and find reference modes.
2	Formulation of Dynamic Hypothesis	Come up with a dynamic hypothesis and represent it with tools such as diagrams.
3	Formulation of a Simulation Model	Make clear more details, build a simulation model by specifying equations and parameters.
4	Testing	Compare simulation behavior with reference modes, carry out extreme condition tests, sensitivity test, and so forth.
5	Policy Design and Evaluation	Design and analyze policies based on the simulation model.

Moreover, Sterman (2000, pp.87) explicitly states that this modeling process should be iterative, which means modelers are not to follow these five steps only once, but iteratively, and could go from any step to another.

Sterman (2000)'s division of the modeling process and his iterative perspective are shared by others. For example, J. W. Forrester, in his 1994 publication, defined modeling process as 6 steps (figure 1), of which steps from 1 to 5 are similar to Sterman (2000)'s five steps. Moreover, the arrows between steps indicate the possibility for the modeler to jump from one step to another.

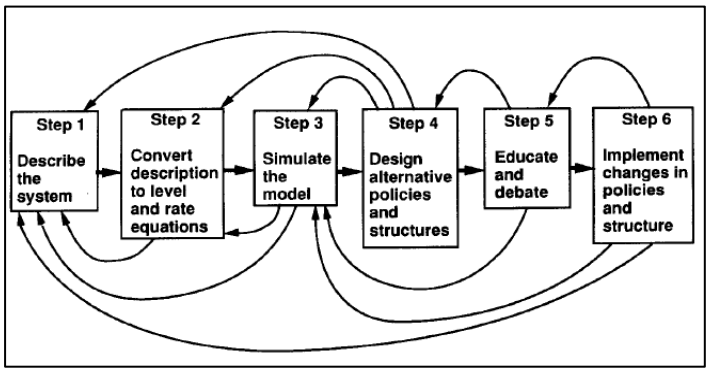


Figure 1 System dynamics steps from problem symptoms to improvement, from Forrester (1994)

Another example is the modeling paradigm taught in University of Bergen, which is named P'HAPI and practiced in almost all system dynamics modeling courses in that school. Although having not been published formally, P'HAPI[1] can be briefly explained as the following steps:

Table 2 Steps in the P'HAPI modeling process

No.	Step	What to do in this step
1	P (Problem)	Identify the problem and illustrate the problem by reference mode(s).
2	H (Hypothesis)	Put the system structure that is believed to create the reference mode behavior into a formal model such as stock-and-flow diagram, as hypothesis.
3	A (Analysis)	Test the model's structure and behavior to make sure it reproduces the right behavior (the reference mode) from an endogenous perspective.
4	P (Policy)	Based on the model, formulate hypotheses about policies that could alleviate the problem.
5	I (Implementation)	Consider the costs of implementation of policy in the context of the real world.

P'HAPI also stresses iteration. Iteration in P'HAPI often happens between H (Hypothesis) and A (Analysis).

Given the fact that different definitions of system dynamics modeling process have a big part in common, and to make it simple for discussions in the coming chapters, the author would propose a simplified division of this modeling process:

Table 3 A simplified division of system dynamics modeling process

No.	Step
1	Problem definition
2	Model conceptualization (including structure generation and parameter calibration)
3	Model analysis
4	Policy design
5	Implementation of policy

[1] Since P'HAPI has not been formally published, this table is summarized from the lectures given by Professor Erling Moxnes.

1.2 Model conceptualization

1.2.1 Common understandings

As ‘model conceptualization’ is the focus of this work, the table below further compares steps relevant to ‘formulation of a system dynamics model’ from all three divisions.

Table 4 Comparison of steps related to model formulation

	Sterman (2000)	**Forrester (1994)**	**P’HAPI**
Related step(s)	Step 2 and 3	Step 2	Step 2
	Formulation of Dynamic Hypothesis; Formulation of a Simulation Model	Convert description to level and rate equations	Hypothesis
Details	1) Examine the current theories of the problematic behavior; 2) Formulate a dynamic hypothesis that can explain the problematic behavior from an endogenous perspective; 3) Map the hypothesis into a formal representation; 4) Formulate the formal representation into a simulation model, including specifying equations, parameters, and initial conditions.	1) Translate the system description into the level and rate equations (stock and flow diagrams, to put in a more modern way), which requires- 2) Make the general description of system more explicit;	1) Put the system structure that is believed to create the reference mode behavior into a formal model; during which it would be beneficial to- 2) Identify if the hypothesis belongs to a class of problems such that one can benefit from previous research and such that the results can be generalized.

Although formed in different times and under different contexts, what could be found in common about ‘model conceptualization’ in these three interpretations are:

1. It requires problematic behavior obtained from a real-world system as reference mode;
2. It requires a preliminary informal understanding of the system or the structure that generates the problematic behavior;
3. It could benefit from relating the hypothesis (the suspected structure or theory) to an existing class of problems;
4. The outcome is in a formal representation, often a simulation model.

These four points could be put into the following diagram:

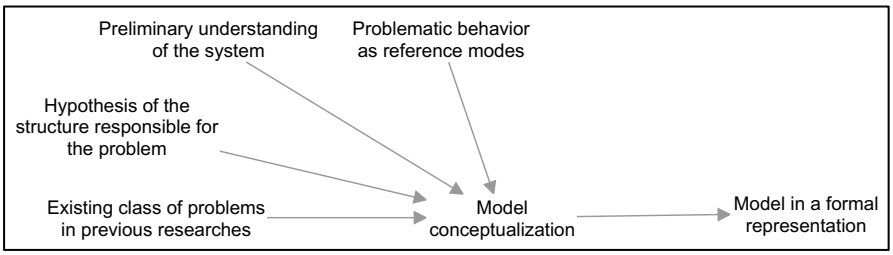


Figure 2 Model conceptualization in diagram (1)

1.2.2 Structure and parameters

Generally, 'model structure' refers to all variables in a model and connections between them. In this sense, parameters that are constants are also included in model structure. Since no causal relationships go into constants, they are usually seen as a model's boundary.

Forrester (1980a) provides a different perspective which is worth to discuss. In Forrester (1980a), a diagram is used to explain the process of creating a system dynamics model:

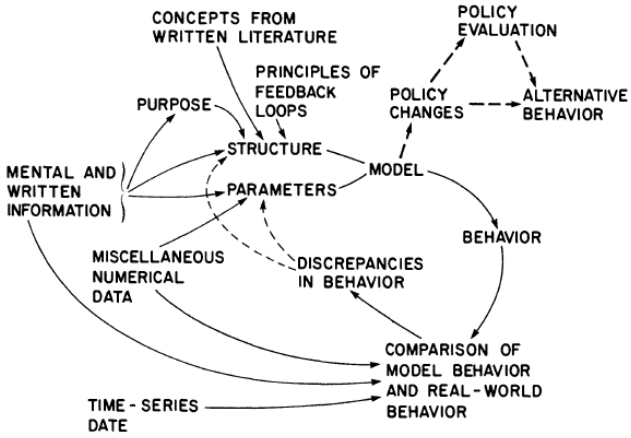


Figure 3 Creating a system dynamics model, from Forrester (1980a), pp.559

It is noteworthy that in this figure, 'model' comprises 'structure' and 'parameters'. From this perspective, 'structure' is different from 'parameters. This division is important to this work. Theoretically, it inspires a new perspective to see a model. Assume in a simulation model we have the following equation:

$y = a * x_1$, where x_1 is the input, y is the output, and a is a parameter.

If we accept the division put forward in Forrester (1980a), in this equation, a would be a 'parameter' and the 'structure' would only include the function, which is the operation of multiplication.

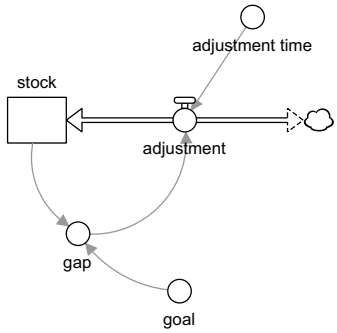


Figure 4 Goal-gap model

Moreover, when looking at a causal loop in a simulation model, this differentiation between structure and parameters could help us to see loops in a different way. Take the goal-gap model (figure 4) as an example. The flow 'adjustment' in this model is calculated through the following equations:

$$adjustment = \frac{gap}{adjustment\ time}, \text{ where}$$

$$gap = goal - stock.$$

We have known that 'adjustment time' and 'goal' are constants and therefore parameters. Following the division in Forrester (1980a), the above equations could be transformed into:

$$adjustment = g\big(f(stock)\big), \text{ where}$$

$$f = Subtraction(parameter_2, stock),\ g = Division(gap, parameter_1);$$

$$parameter_1 = goal,\ parameter_2 = adjustment\ time$$

From this perspective, 'coming up with model structure' is be different form 'calibrating parameters' since the former is only about selecting functions and the latter is only about tuning parameters. The author names these two different processes **structure generation** and **parameter calibration**. Although many researches use 'structure generation' for both, in our discussions they are referred to differently, and are collectively called **model conceptualization**.

To sum up, **structure generation** means to come up with the right functions to use as 'backbones' of computation, while **parameter calibration** means to tune parameters in an existing model so that the simulation behavior will fit the reference mode.

1.2.3 Definition of model conceptualization

According to Forrester (1980a) and learning from figure 3, to conceptualize a model one needs at least the following inputs:

- Principles of feedback loops;
- Concepts from written literature;
- Purpose (as a part of mental and written information);

- Other mental and written information;
- Miscellaneous numerical data;
- Time-series data.

As for now, we have had a list of inputs for model conceptualization from analysis of different divisions of the system dynamics model process, and another list of inputs for model conceptualization from Forrester 1980a). In the same paper, Forrester also proposed a taxonomy of information sources that could be used for modeling: mental data, written data, and numerical data. This taxonomy is used here to categorize these inputs. Putting everything in one diagram, we have:

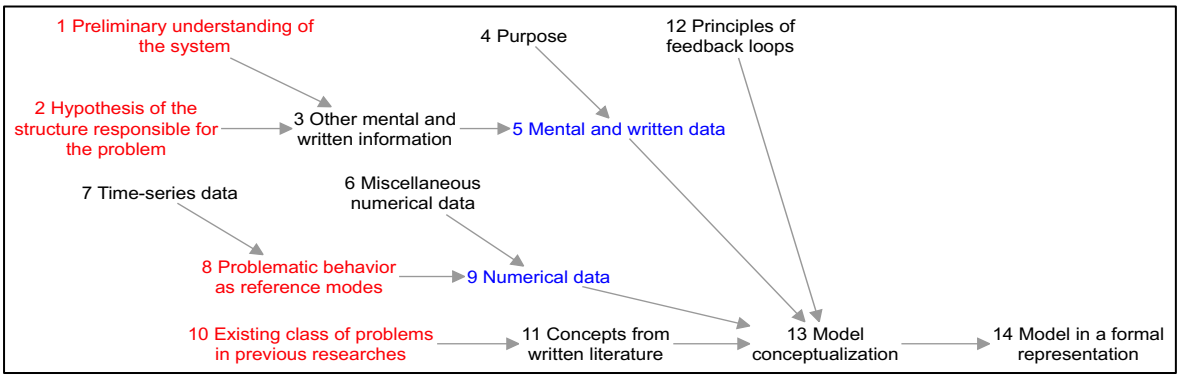


Figure 5 Model conceptualization in diagram (2)

Blue elements: Taxonomy proposed in Forrester (1980a); Red elements: Inputs from analysis of divisions of system dynamics modeling process; Black elements: Necessities for building a model in Forrester (1980a).

In summary, to conceptualize a system dynamics model in a formal representation, a modeler needs four types of inputs.

First, the modeler needs to know basics of system dynamics, such as the concepts of stock, flow, feedback loops, and how they work with one another.

Second, the modeler needs to have mental and written data about the problem at hand, such as basic understandings of the system in the real world ('the real system').

Third, the modeler needs numerical data, such as time-series data which is used for reference mode.

Fourth, the modeler could be helped by written literature, from which concepts and existing class of problems could be borrowed to model the new problem at hand. As commented by Forrester, "the conceptualization phase in system dynamics has rested heavily on past modeling experience gained from working with 'canned' models, from apprenticeship in working with experienced modelers, and from trial-and-error learning." (Forrester 1994, pp.254)

Last but not least, as the modeling process is iterative, the above necessities may not be all available at the same time. Some information may come after others, and very often, only in a certain stage will the modeler become aware that a specific part of information is needed.

Based on discussions so far, the author would propose a definition for model conceptualization:

Model conceptualization is a step in the system dynamics modeling process, where with the help of previous study and following the system dynamics modeling paradigm, understanding of the real system and speculated structural causes of the problem are formulated into a formal model that is able to reproduce the reference mode.

However, such a definition does not provide insights into how conceptualization actually works. What has been clarified are the input and output of this process, while the underlying mechanism – how these inputs are utilized to get the output still remains unclear. Richardson and Pugh have pointed out that, "Problem identification and model conceptualization are the apparently less technical stages of a system dynamics study." (Richardson and Pugh 1981, pp.61)

2. Literature review

In the past 20 years, there have been multiple attempts to automate model conceptualization, by at least 4 research teams. In this chapter, after an overview, the author will discuss the methods used by them in detail and compare them with the concept of model conceptualization.

2.1 Attempts to automate model conceptualization

Four research teams have made attempts to automate model conceptualization over the past 20 years. They are referred to as Barlas team, Chen & Jeng team, Drobek team, and Abdelbari and Shafi team. Using the simplified division of the system dynamics modeling process proposed in Chapter 1 (Table 5), works by these four teams are listed and categorized in Table 6, of which shaded entries are directly related to model conceptualization.

Table 5 A simplified division of system dynamics modeling process

No.	Step
1	Problem definition
2	Model conceptualization (including structure generation and parameter calibration)
3	Model analysis
4	Policy design
5	Implementation of policy

Table 6 Researches on automation of system dynamics modeling process since 2000

Year	Researchers	Step automated
2000	Barlas & Kanar	Model analysis
2002	Chen & Jeng	Structure generation Parameter calibration
2004	Chen & Jeng	Policy design
2006	Jeng, Chen, & Liang.	Parameter calibration
2007	Yücel & Barlas	Parameter calibration
2011	Chen, Tu, & Jeng.	Structure generation Parameter calibration
2011	Yücel & Barlas	Parameter calibration
2014	Drobek, Gilani, & Soban.	Structure generation
2015	Drobek, Gilani, Molka, & Soban.	Structure generation
2015	Yücel & Barlas	Model analysis
2015	Abdelbari, Elsawah, & Shafi	Structure generation
2016	Abdelbari & Shafi	Structure generation
2017	Abdelbari & Shafi	Structure generation
2018	Abdelbari & Shafi	Structure generation

Barlas team started on this direction from 2000 and their research continued until recent years. They haven't been directly focusing on generating model structure but made considerable progress on parameter calibration based on recognition of behavior pattern.

Research led by Chen and Jeng also has a long history. Different from Barlas team, they had the objective to automate model conceptualization - especially structure generation - from the very beginning, and developed an artificial neural network-based methodology, where system dynamics models are represented not in stock-and-flow diagrams but in neural networks. They also used genetic algorithm as the core mechanism for structure generation.

A similar method is later adopted by Abdelbari, Shafi, and Elsawah in 2015. Benefited from the fast development in machine learning (especially neural network-based deep learning), they are able to use more advanced neural networks to approximate a system dynamics model.

Drobek et al. also use neural network techniques but in a way different from Chen & Jeng team and Abdelbari & Shafi team. While the other 2 teams use a single neural network to represent a whole system dynamics model, Drobek et al. represent equations for variables each with a neural network and train all of them in the same time.

2.2 Techniques used for automation

2.2.1 Essence of a model conceptualization task

Before moving to detailed analysis of techniques, it would be helpful to generally discuss the essence of a model conceptualization task. As discussed in Chapter 1, the expected outcome of model conceptualization is a formal model. In the field of system dynamics, two most often used forms to represent a model are stock-and-flow diagram (SFD) and causal loop diagram (CLD). Model conceptualization can have any of them as outcome.

The essence of a model conceptualization task depends on which form is chosen for the outcome. If CLD is chosen, the outcome will include only variables and causal links between them, and the nature of this task will be to estimate causal relations between variables and their directions. However, if the conceptualized model is to be in SFD, the outcome will comprise both the structure (which is a set of functions) and the parameters. The essence of this task will therefore be to come up with a set of functions and to calibrate their parameters.

2.2.2 Techniques used by existing researches

With the objective of generating a preliminary CLD, Drobek et al. (2014) uses Pearson product-moment correlation coefficient, which is a statistical tool to estimate dependencies between variables in a system. As time series data of all variables are provided, the task is to estimate their dependencies by calculating correlation. The estimated dependencies are later used as indication for causal relationships. Although correlation could not be seen as causality, the goal of this method is to generate a correlation graph for later incorporation of expert knowledge to finally get the CLD, and the method fits this goal well.

Except for Drobek et al. (2014), all other attempts aim for a simulation model. They all use and only

use time-series data as information source. As mentioned above, their tasks are two-fold: coming up with a set of functions and calibrating parameters. Their objective is to generate simulation models that are able to reproduce the reference behaviors.

Abdelbari and Shafi (2016, 2017, and 2018) chose to represent system dynamics models with a type of artificial neural networks (ANN), because the structural similarity between these two architectures and ANN's ability to learn by training with time-series data. In a neural network, neurons (nodes) are connected by edges, and each edge has a weight. Through a training process, the weights will be adjusted, which altogether can make the neural network produce the desired output.

In Abdelbari and Shafi's works, the chosen type of neural network is Echo State Network (ESN) because it is structurally similar to stock-and-flow diagram. Through training, an ESN is able to formulate some structures that resemble stocks and loops, which creates opportunity for this neural network to morph toward an SFD. However, because the similarity is still limited, we could find stocks and loops in the trained structure, but not counterparts for functions or equations that we normally use in an SFD. Therefore, the authors set their goal to be 'generating a CLD that can run'. As in Drobek et al. (2014), the outcome is supposed to be used subsequently by experts to elicit SFDs.

Drobek team changed their method in their 2015 work. They used neural networks not to represent an entire model but to represent functions in every single variable. Through a training process, each neural network will approximate the equation in one variable. The learning outcome will be a set of approximated equations. This is an advancement, but the approximated equations are still different from the equations used in SD models, since the generated functions are still in a form of neural network and therefore not analytic. Analytic functions are those we would use when manually building a system dynamics model, such as subtraction and multiplication. Modelers use these functions to represent causal relations in the real world, while a neural network, even if able to reproduce a behavior, could not be used to map such a causal relation.

Moreover, it is noteworthy that in Drobek et al. (2015) and Abdelbari and Shafi (2016, 2017 and 2018), time series for all variables in the system are readily available, which means there is no need to suggest new variables. The algorithm only needs to suggest ways to connect them. This is different from another type of structure generation, in which not all variables are pre-defined, and the algorithm needs to suggest new variables by itself. The latter therefore demands a more flexible method that is able to construct model structure.

Chen and Jing used artificial neural networks in their 2002 work to represent model structure and in the following years their focus was policy design based on parameter optimization. Later, in Chen et al. (2011) a new method was proposed. This method enables a complete translation of SFDs into neural networks. That means, equations used in an SFD such as subtraction and multiplication can be translated into neural networks and still stay analytic. It is therefore different from the method in Drobek et al. (2015) and Abdelbari and Shafi (2016, 2017, and 2018).

Such an advancement makes it possible to generate 'real' system dynamics model structure, but it puts forward a new problem. Since analytic functions could not be obtained through training a neural network, they need a new mechanism to build model structures in a humanly way. Abdelbari and Shafi (2015) and Chen et al. (2011) solve this problem by using genetic algorithm to build analytic equations.

In their methods, functions, operators, and operands are seen as ‘elements’ or ‘blocks’ of equations. Using genetic algorithm, a great amount of possible combinations of these basic elements are formulated and tested, and the performance of one combination is measured by how well its simulation behavior fits the reference mode.

Genetic algorithm is an adaptive algorithm widely used in generative tasks. It needs only basic elements and rules that tell how these elements could be combined. Then it can try huge amount of possible combinations and test their performance against pre-defined criterion. Although mechanisms such as ‘mutation’ and ‘selection’ are used to help this ‘evolution’ of combinations by making it more efficient, time consumption is still very often a problem for this algorithm, because it highly relies on brutal computational force to try different combinations. Adaptivity and flexibility of this method is at cost to efficiency.

Because of its high adaptivity, genetic algorithm is also used in parameter calibration tasks to try enormous combinations of parameter values. Chen & Jeng (2004), Jeng et al. (2006), and Yücel & Barlas (2007 and 2011) all use genetic algorithm for this purpose.

Barlas team never tried to directly generate model structure. Instead, they focused on parameter calibration, and their purpose was not to reproduce exactly the reference mode, but to reproduce the dynamic pattern of the reference mode. The significance of this pattern-based method to categorize behavior will be elaborated in the next chapter.

Overall, the above explanations could be summarized into the following table:

Table 7 Techniques used in automatic model conceptualization

		Automatic model conceptualization	
		Structure generation	Parameter calibration
Method	Representation of SD models	Graph network (as model) Artificial neural network (as model) Artificial neural network (as equation) Combination of basic elements (as model)	Artificial neural network (as model)
	Searching for optimal solution	Genetic algorithm Training of artificial neural network	Genetic algorithm

2.3 Discussions on existing methods

2.3.1 On simplifying model conceptualization into a mathematical task

Taking a close look at one time step in the simulation of a system dynamics model, one would find that all calculations will converge into the calculation of flow(s). In the end of this time step, stocks are updated by the flows they are connected with. In a word, initial level of stocks, values of parameters, and all the functions determine the outcome of the simulation.

Since we can aggregate all functions into flow functions, one could compare ‘model conceptualization’

to ‘coming up with equations for flows’, which makes it a mathematical problem. And the judge of ‘how correct the equations are’ will be the fitness between the model’s behavior and the reference mode. However, such a mathematical simplification of model conceptualization may not always be valid. Different models can fit the data equally well but produce different predictions and respond to a policy differently, because they might be different in structure. Therefore, the ability to fit the historical data does not necessarily indicate which hypothesis about the feedbacks in the real system is the one to select. (Sterman, 2000, pp. 330)

For example, assume we have the following reference mode for a stock (figure 6), and we want to find a model that can explain this behavior.

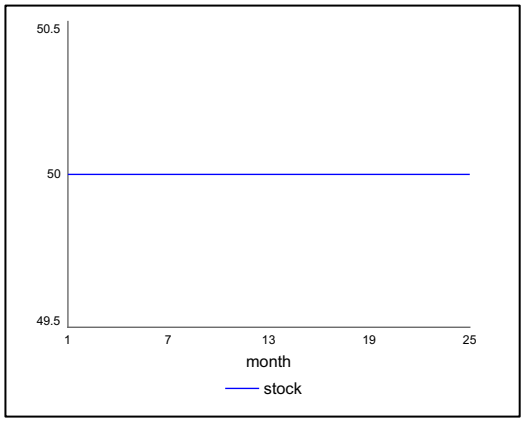


Figure 6 Reference mode for a stock

Because the reference mode shows that the stock stabilizes at a level, one may come up with the following structure (Figure 7), which is an isolated stock with initial value of 50.

Figure 7 Suggested structure (1)

However, it is also possible for the underlying structure to be the one in figure 8, where a stock is connected to two equal flows.

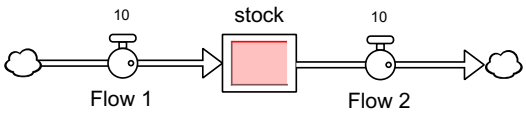


Figure 8 Suggested structure (2)

In this case, it is not important which one of the hypotheses is correct; the point is, there might be multiple explanations for one reference mode, if we only look at the time-series data. It is easy for one to forget this when the situation is much more complicated, when the time-series data is abundant and of high quality, and when the generated model structure is convincing.

2.3.2 On information used in model conceptualization

Bring back the definition made for model conceptualization in the first chapter:

> *Model conceptualization is a step in system dynamics modeling process, where with the help of previous study and following the system dynamics modeling paradigm, understanding of the real system and speculated structural cause of the problem are formulated into a formal model that is able to reproduce the reference mode.*

One would find that almost all works analyzed above, whatever the technique, only use time-series data as the only source of information. If mapping this to the system dynamic modeling process illustrated by Forrester (1980a) (figure 9) or to the inputs for model conceptualization (Figure 10), one would find that the essence of the attempts is the 'giant leaps' marked in the diagrams.

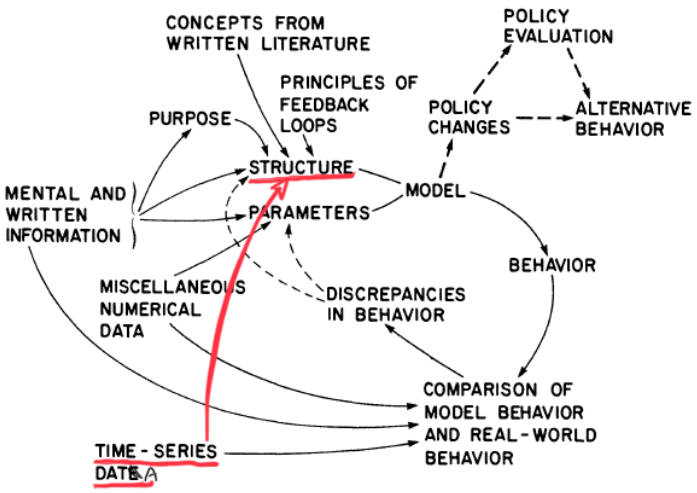


Figure 9 Generating system dynamics model directly from time-series data with the 'giant leap' marked

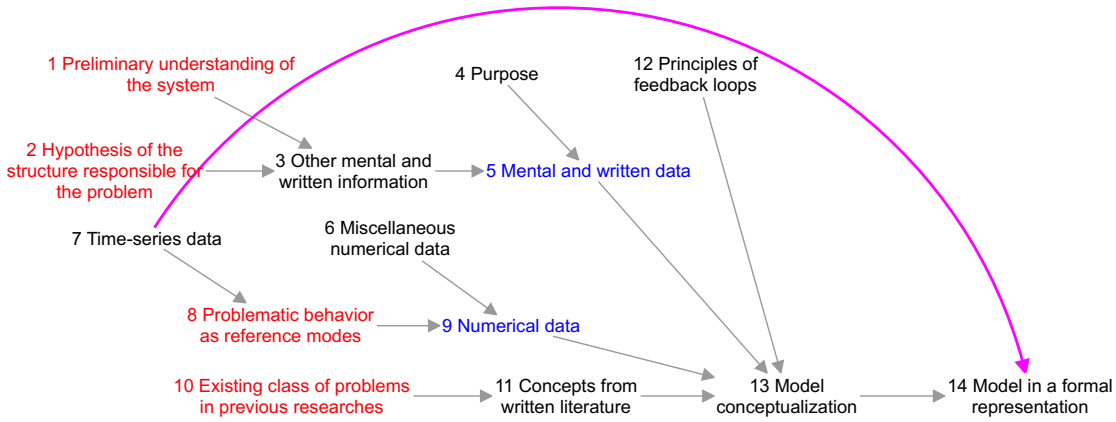


Figure 10 Model conceptualization with the 'giant leap' marked

Blue elements: Taxonomy proposed in Forrester (1980a); Red elements: Inputs from analysis of theories of modeling process; Black elements: Necessities for getting a model in Forrester (1980a).

Considering the example in figure (6, 7, and 8), it would be very possible that the learned model structures in the analyzed works are subject to great uncertainly. This is not because their mathematical algorithms are not advanced enough, but because their algorithms are not able to ascertain which possible structure is the one in the real world. And when these possible structures are many, the outcome will be ambiguous. This is actually what happened: in their many results, redundant causal relations still exist in great amount. Efforts by human modelers and expert knowledge are therefore needed to make the outcomes clear.

On the other hand, exclusion of any information sources in figure 10 may lead to negative consequences. First, if not including mental and written data, understandings of the system and hypotheses of the structure owned by clients would not be sufficiently considered and engagement of the stakeholders would therefore be compromised. Second, not following the principles of feedback loops, one may build models that could not produce meaningful result. Moreover, not relating the hypothesized structure to an existing class of problems may slow down the conceptualization process and make it more difficult, because humans highly rely on previous experience when conceptualizing a model:

> *"The conceptualization phase in system dynamics has rested heavily on past modeling experience gained from working with 'canned' models, from apprenticeship in working with experienced modelers, and from trial-and-error learning." (Forrester 1994, pp. 254)*

It is therefore not only necessary but also beneficial to consider using more information sources than only time series data. However, the current situation is still understandable, because up to now algorithms are best at – if not only good at – analyzing numerical data. It is still impossible for computers to fully understand a dialogue or a text, although we already have seen great advancement in the field of natural language processing (Young et al. 2018). Under current condition, relying solely on numerical data is clearly a compromise that is made due to the limitations of technologies on handling mental and written data.

3. A platform for interactive model conceptualization

To improve the performance of existing methods for automatic model conceptualization, additional information sources need to be included. Due to technical limitations, it is currently not possible to incorporate all of them in one time. Therefore, instead of aiming at an ideal automatic method, this work intends to take one step forward and explore the possibility of incorporating two of them.

The first one is mental data. They include preliminary understanding of the system, hypotheses about the structure beneath the problem, and purpose of building this model. These are information specific to a particular problem or situation, and usually come from the problem owners. This part of information is utilized by allowing the modeler to interact with the conceptualization algorithm during the model building process.

Second, existing class of problems in previous researches. They are generic structures and behaviors, often extracted from models that are already built, and could be adapted for new situations and problems. This part of information is utilized by building a knowledge base for the algorithm.

To achieve these goals, a new framework to integrate information and a new algorithm to carry out model conceptualization are proposed. They are developed with lessons drawn from the way we conceptualize models, as humans are the only ones so far that could utilize all those information sources.

3.1 Learning from humans: how do we build a model?

3.1.1 Iteratively making and testing hypotheses

Conceptualizing a system dynamics model is to a large extent about working with hypotheses. One makes new hypotheses by building structures and test these hypotheses by comparing structures and their behaviors with the real world.

As the P'HAPI workflow says, a modeler needs to iteratively go through Hypothesis and Analysis to get a plausible model. Qualitative and quantitative information are utilized in both forming and testing hypothesis (Forrester, 1980a; Barlas, 1996). The iterative process allows the modeler to take up information little by little, for one cannot focus on too much information at one time. Similar observation could be made from figure 3, in which a loop is in the center of the diagram.

The new framework follows this iterative process. In one iteration, different information could be taken up and utilized in different steps. Moreover, as hypotheses are made and tested alternately, some steps in one iteration could be used to generate possible structures while others used to test and validate them.

3.1.2 Following rules and previous experience

A modeler could benefit his or her previous experience obtained from practice. Every modeler maintains a stock of models in memory as well as the situations the models were built for. A modeler does not always need to build new models from scratch. Instead, he/she will be glad to take a shortcut by adapting an existing model built for a previous problem to a new situation. As Forrester wrote in his 1980 work:

> *In the process of finding valuable insights in the mental data store, one talks to a variety of people in the company […]. The discussion is filtered through one's catalog of feedback structures into which the behavioral symptoms and the discussion of structure and policy might fit. (Forrester, 1980a, pp.560).*
>
> *Interrogation must be guided by a knowledge of what different structures imply for dynamic behavior. (Forrester, 1980a, pp. 556).*

This process happens so often that generic structures could be used as generalized insights to facilitate one's searching for inspiration (Lane, 1998). In contrast, if a modeler does not have this 'catalog of feedback structures', he/she has to re-invent all models even if it is a generic structure familiar to many.

On the other hand, there are times when we cannot find a proper generic structure or a previous model ready for use, which means we have to from time to time build some structures on our own. Even if we are able to find one, the adaptation will include modifications to the model structure. In these occasions, we need to follow a certain set of rules for building a system dynamics model, such as 'a stock needs to be adjusted by a flow' or 'a subtraction function takes two parameters'. This does not rely heavily on previous experience but the ability to map relationships in the real world into model structures.

Moreover, the two mechanisms are not conflicting but often cooperating: after adding a few elements to a structure, one suddenly realizes that there might be a generic feedback structure. Such alternation of mechanisms happens subconsciously and is therefore often not noticed.

The new framework reproduces these two mechanisms. The one guided by previous experience and generic structures is called 'top-down', while the other one that builds structure with basic elements is called 'bottom-up'.

3.1.3 Exploring and exploiting possibilities

Humans are able to keep multiple possibilities in mind at the same time, but without paying equally much attention to each of them. Usually only one or two are focused on, with the rest being inactive. However, this allocation of attention is dynamic. As we only have limited patience for a specific possibility, if one possibility takes too long time to try or some inactive possibility is suddenly found promising, we would re-allocate our attention to focus somewhere else.

This mechanism is termed as 'parallel terraced scan' (Hofstadter and Mitchell, 1994) and has a bionic origin, since it was inspired by the behavior of ants. When searching for food, a group of ants will neither focus solely on one path, nor evenly distributed on all possible paths. They start by exploring many paths, and once food is found on one path, through a feedback mechanism, more ants will gather to this path for exploiting. Meanwhile, there are still ants exploring other paths.

Paths are comparable to different ways to build a model structure, and our search for an optimal structure is comparable to ants' search for food in terms of their ability to balance exploration and exploitation. Comparing to genetic algorithm which has been used in many previous works, this method relies less on computational force because it has a mechanism that gradually allocates computational capacity to the most promising candidates as they emerges. This mechanism of prioritization is a feature of the new framework.

3.2 The workflow

This interactive modeling platform is designed to work in an iterative way. It allows the system, which is the automatic model conceptualization algorithm, to interact with the modeler in order to generate model structures. The main workflow of it is therefore a circle (1-2-3-4), as shown in figure 11. What runs through the circular path are partially-built models, which are called candidate structures. Through iteration, candidate structures are constantly generated, modified, validated, and selected. Inside this circle are mechanisms that could influence this iterative process.

In this diagram, the owner of the problem could interact with the system through managing reference modes (5), manually modifying model structure (7) and accepting structures generated by the system (8). The rest of the processes are automatically done by the system.

Once a reference mode (time-series data) is added through managing reference mode (5), it will stay (6) unless manually deleted. Every time before a new loop begins, the system will check if all reference modes have their stock/variables in the model structure. If not, the system will build them through (9). Problem owner can also manually modify a structure (7) before and after a new loop begins.

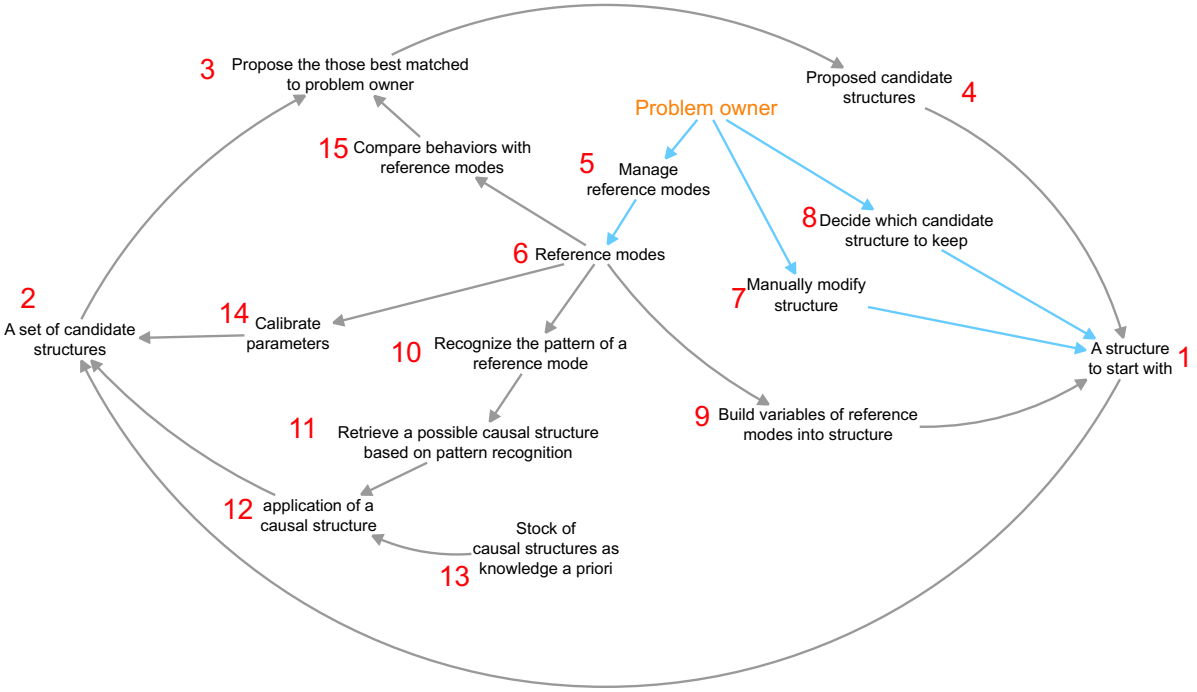


Figure 11 Main workflow of the interactive modeling platform

Reference modes (6) play an important role in this system and could influence the model conceptualization process through multiple ways. First, they could be used for measuring the performance of a candidate structure through comparison of behaviors (15). Second, it could be used to suggest a possible feedback structure and apply this feedback structure to a candidate structure (10-11-12-13), which is the 'top-down' mechanism. Third, they are used as reference behaviors for calibrating the parameters (14) of a candidate structure.

A new iteration begins with structure expansion (1-2), through which the base structure (1) could be modified through one or more mechanisms. The diagram only shows one mechanism, which is the 'top-down' mechanism (10-11-12-13). Modification is not working on the base structure itself but on a copy of it, which makes the base structure co-exist with the modified structure.

Like structure expansion (1-2), parameter calibration (6-14-2) is also a part of model conceptualization. Through this process, the system will adjust the parameters in candidate structures, trying to make their behavior closer to the reference mode.

Candidate structures whose behavior matches the reference mode the best (15-3) are ranked top (2-3), and they will be proposed to the problem owner for selection (3-4). The problem owner could accept and keep (8) the one he or she finds convincing. It is also possible for him/her to manually modify one or more candidate structures (7), and to add or remove reference modes (6) through (5) before the next iteration begins.

3.3 Key mechanisms

To implement the workflow discussed above, the modeling platform is equipped with a number of key mechanisms. They are revealed by the following diagram and will be explained in detail.

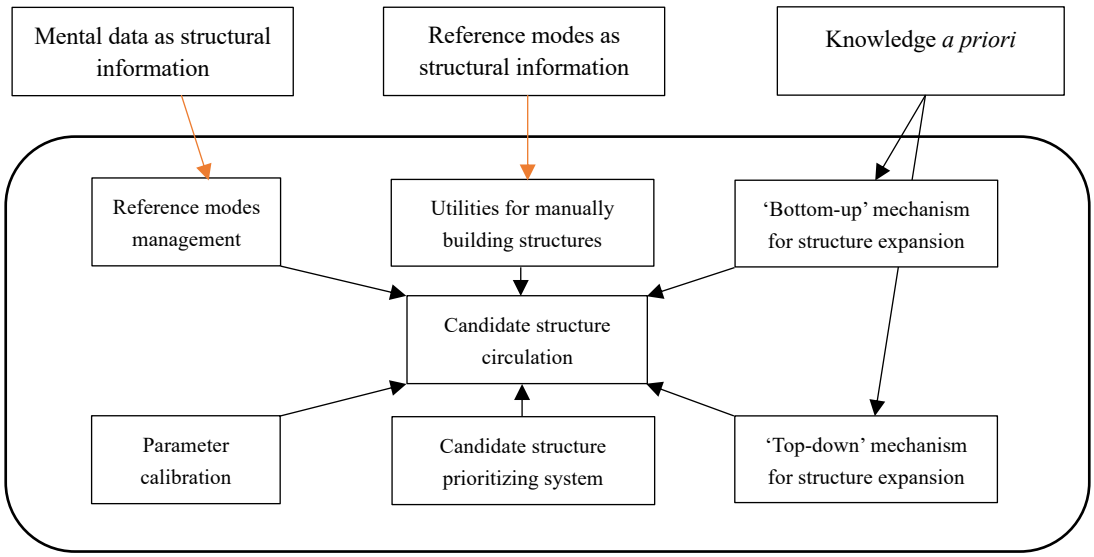


Figure 12 Key mechanisms of the modeling platform

3.3.1 Incorporation of mental data

The modeling platform incorporates mental data into the conceptualization process by interacting with the problem owner. Mental data include the problem owner's understanding of the situation and the speculated structures responsible for the problematic behavior. There are two ways for the problem owner to input his/her mental data into the platform. First, the problem owner could build a model structure (7) for the system to begin with, which would help the structure expansion (1-2) to go in a more directed way. Second, by the end of each iteration (1-2-3-4), the problem owner is able to accept,

reject or modify any candidate structure through (8), influencing the next iteration.

3.3.2 The top-down mechanism for structure generation

The top-down mechanism is guided by the experience a modeler accumulates throughout his/her practice and learning, which is a type of knowledge *a priori*. It comes from building and analyzing huge amounts of models, but it is not just models. For example, a tea-cup model (whose core structure is a first order negative feedback loop) will contribute to this knowledge base not only by itself, but also by the idea of first order negative feedback loop, which is represented as a chain of functions discussed in Section 1.2.2.

These feedback loops can be used to guide the generation of new candidate structures (10-11-12-13). First, the reference mode is categorized (6-10) as one of the generic dynamic patterns defined in Barlas and Kanar (2000). Most of the generic dynamic patterns are linked with feedback loops. For example, Decline (c) in figure 13 is linked to a first order negative feedback loop.

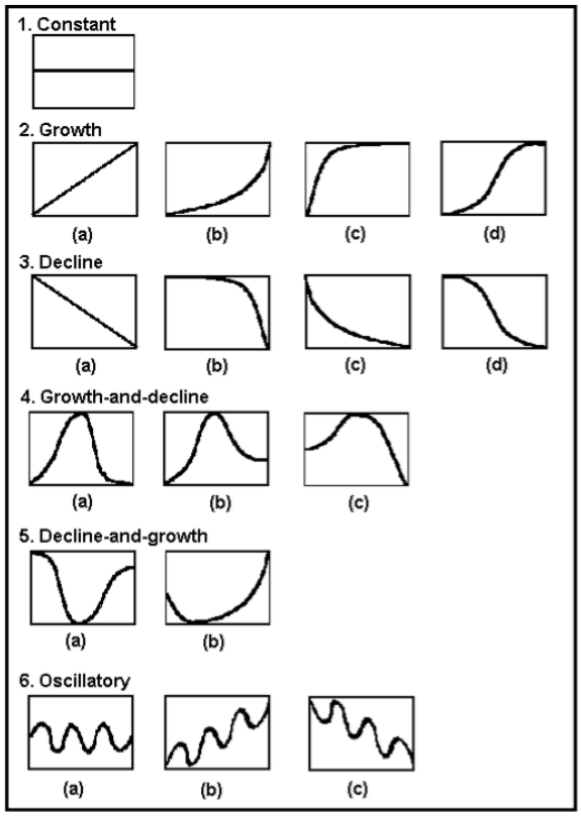


Figure 13 Typical dynamic patterns from Barlas and Kanar (2000)

When a reference mode is categorized as Decline (c), the concept of 'first order negative feedback loop' is activated. Then through process (13-12), a first order negative feedback structure is retrieved from a particular model (e.g. a tea-cup model) and added to a candidate structure (12-2).

The following example is used to elaborate this mechanism.

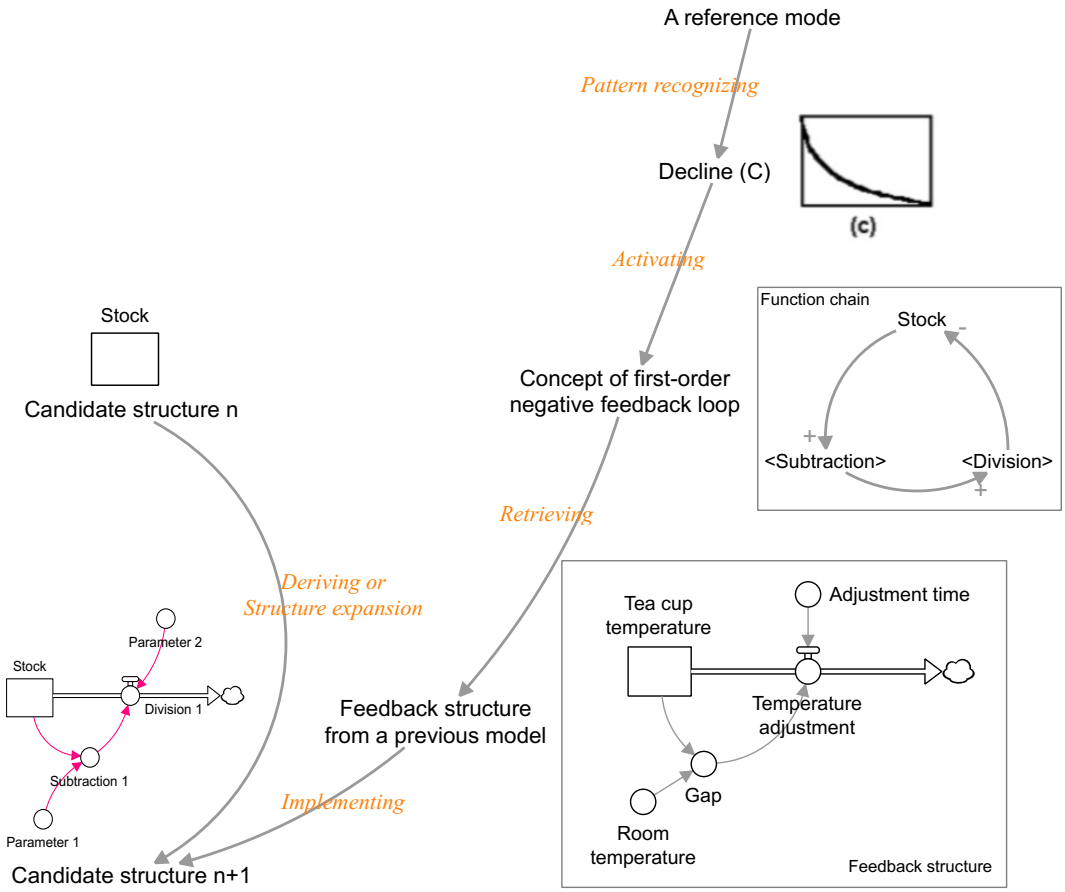


Figure 14 An example for the 'top-down' mechanism

The example shows how a reference mode could be used to derive a new candidate structure through the ‘top-down’ mechanism.

3.3.3 Managing all candidate structures in a tree

As we use ‘candidate structures’ to represent the many possible ways a model could be built, there are often multiple candidate structures (2) existing in the system at the same time. They all originate from the root candidate structure which is the first one that comes to (1). Each candidate structure has a pathway that traces back to the root. Candidate structures are therefore managed in a tree-shape data structure, in which each node represents a candidate structure, and to derive a new candidate structure from an existing one is done by adding a node and point to it with an arrow from the existing node.

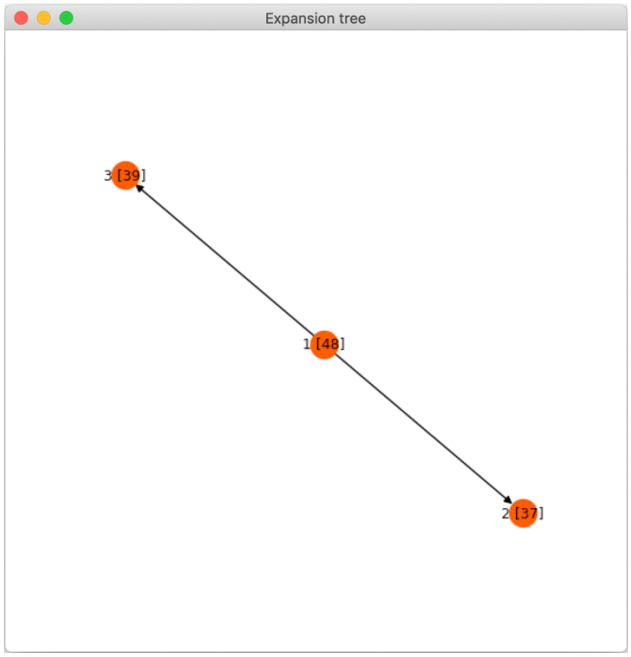


Figure 15 Example of an expansion tree

Figure 15 shows an expansion tree with three candidate structures. Candidate structure No.1 is the root, from which all other candidate structures are derived. Candidate structure No.2 and No.3 are different but both modified based on No.1, through structure expansion (1-2).

As described before, deriving a new candidate structure from an existing one does not overwrite the old one, but keeps both of them. In figure 15, this means that after Candidate structure No.2 is derived from No.1, No.1 is still considered as an existing candidate structure and therefore can be used to derived Candidate structure No.3 afterward. This is because a structural modification may not always be beneficial. In the case of figure 15, Candidate structure No.2 is different from No.1 in structure, and this difference may make its behavior less fitting the reference behavior. If so, Candidate structure No.2 will be less likely to survive in later selections. Keeping the old one (base structure) is to keep the possibility for it to be expanded on a different direction, as in figure 15, keeping Candidate structure No.1 makes possible the later derivation of No.3.

Comparing with the way ants searching for food, this mechanism to search for promising candidate structures is trying to keep a balance between exploration and exploitation when there are only limited computational resources to allocate: building new candidate structures is to explore new pathways, while the pathway on which candidate structures are most promising would be prioritized for further exploitation, by deriving even more candidate structures from those existing. However, no matter how promising one pathway seems to be, exploration on other pathways will not completely stop; just as in ants' searching, there are always a few ants scouting around the area.

3.3.4 A scoring system to identify promising candidate structures

A scoring system is designed to decide which candidate structure to prioritize. It is implemented through a scoring system that grades candidate structures (2-3). The indicator used for score is 'likelihood', which calibrates how possible one candidate structure could become the expected model structure. A

candidate structure with higher likelihood would receive more attention from the system, which makes its further expansion easier to happen. Adjustment of likelihood is based mainly on the similarity between the candidate structure's behavior and the reference mode.

However, there is not a fixed mapping between 'similarity' and 'likelihood'. Instead, the adjustment of likelihood is carried out through competition. Every time, two candidate structures are picked and each of them is compared with the reference mode to calculate similarity. The one with higher similarity will get some score from the other one. This mechanism is implemented through the Elo rating algorithm. The entire prioritizing process can be summarized in the following causal diagram:

Figure 16 The prioritizing process

4. Experiments and results

In this chapter, the author will further demonstrate both the features and the user interface of the interactive modeling platform by running three experiments. Features to demonstrate are:

- Suggesting model structure to explain a given time-series;
- Suggesting model structure to explain a given time-series while incorporating structural information from the user;
- Suggesting model structure while being adaptive to new information during the modeling process.

4.1 Case study 1: Tea cup case with time-series data only

4.1.1 Case description

This experiment is set up to test the platform's ability to automatically suggest a model structure from only time-series data.

The situation in this case is a cup filled with hot water cooling down in a room. The time series shows the cup's temperature going down over time (figure 17). The manually measured temperature data is obtained from Wagon and Portmann (2005) and converted into degrees Celsius from original degrees Fahrenheit (figure 18). No additional information is provided to the system.

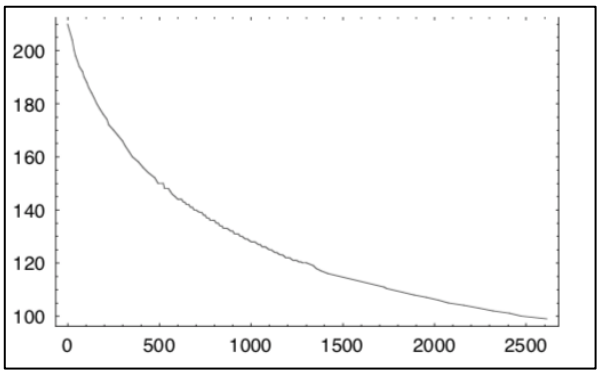


Figure 17 Time series of the cup's temperature (degrees Fahrenheit), x-axis in seconds.

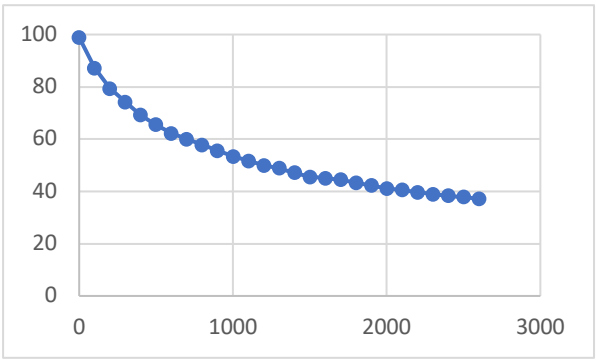


Figure 18 Time series of the cup's temperature (degrees Celsius), x-axis in seconds.

The expected outcome of this experiment is a model structure that could explain this reference behavior.

4.1.2 Experiment process and result

Step 1: Load reference mode

The platform is able to load reference modes from file. In this experiment, the time-series data of the cup's temperature is stored in a csv file. The platform loads the csv file and displays time-series data. The user can preview a reference mode by selecting it in the interface (figure 19).

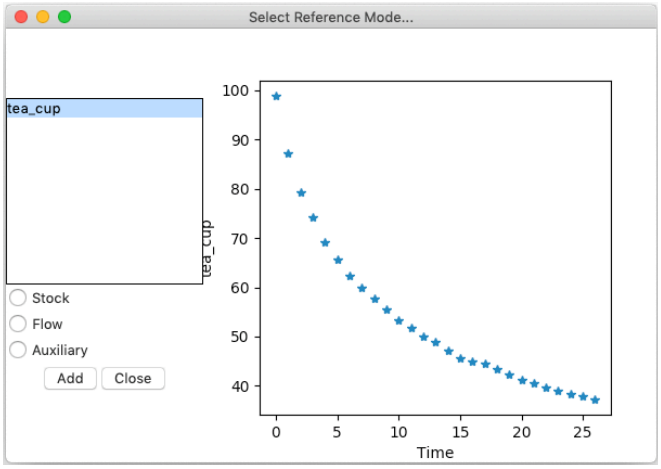


Figure 19 Reference mode loader

The user needs specify the type of variable for this reference mode. In this case, the temperature takes time to change, so it needs to be represented by a stock. By selecting 'stock' and clicking the 'add' button, a reference mode is added to the system's reference mode manager. After adding all reference modes, the window could be closed by clicking the 'close' button.

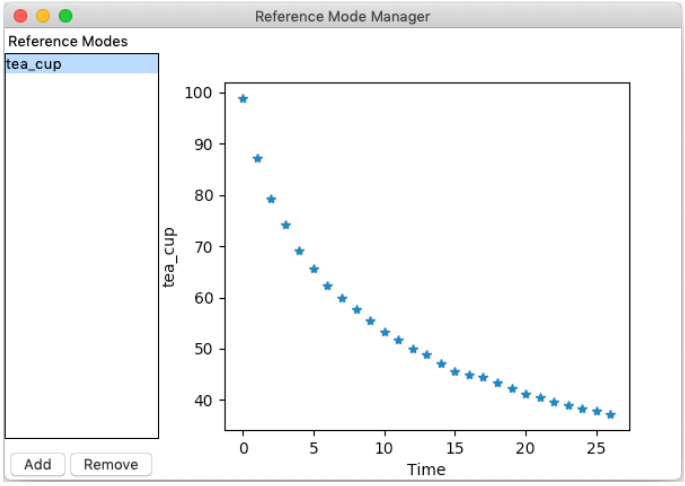


Figure 20 Reference mode manager

Reference mode manager of the system has a graphic user interface to preview, add, and remove reference modes (figure 20).

Step 2: Start a conceptualization loop

The user can control the conceptualization process with the platform controller.

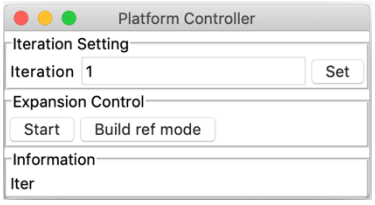


Figure 21 Platform controller

When reference modes have been loaded, the system will generate its first candidate structure as a starting point. In this candidate structure, variables are created for each reference mode. In this case, we only have the stock 'tea_cup' in the model, because only one reference mode is added. The first value in the reference mode is extracted and used as the initial value of this stock, which is about 98.0. The model is simulated to generate an initial behavior. The simulation time is 25 time-units. With a DT of 0.25, 100 time-steps are calculated.

Through the graphic user interface, a candidate structure can be displayed in both a stock-and-flow diagram and a causal loop diagram, with its behavior shown in line graph.

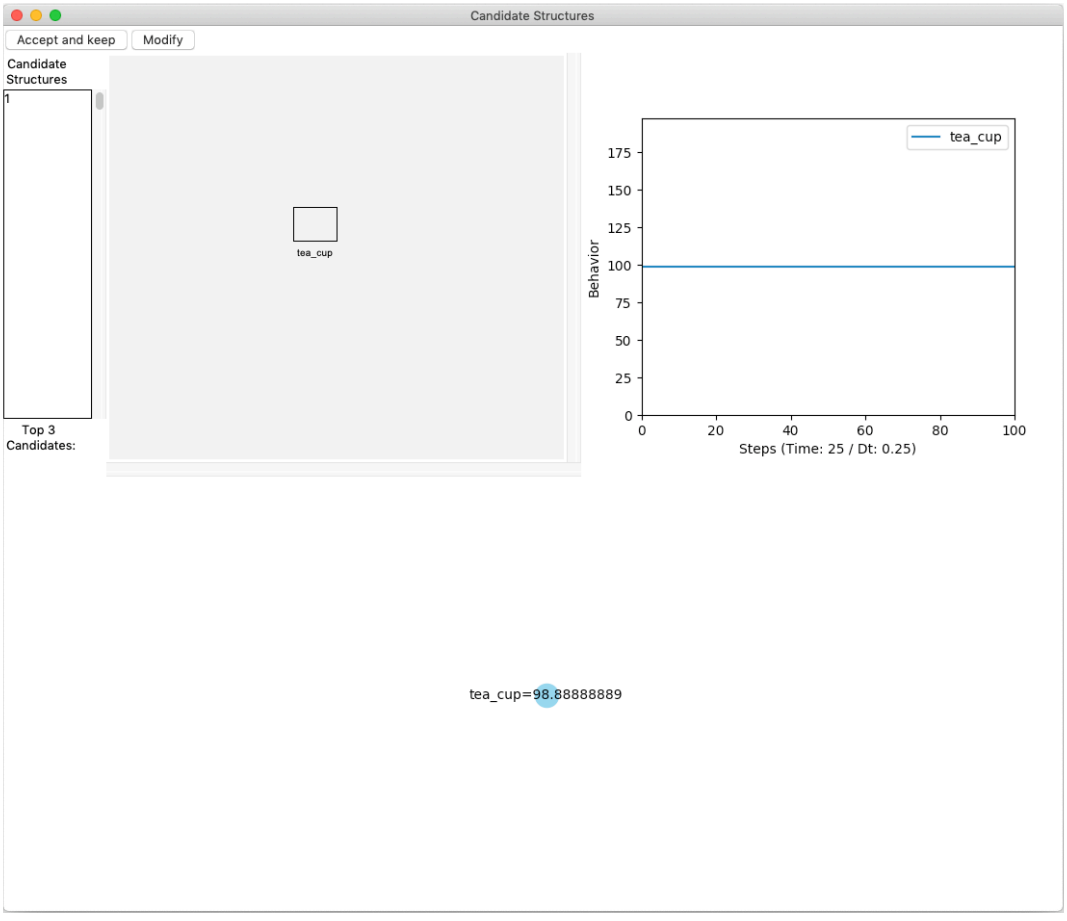


Figure 22 Candidate structures (1)

Step 3: Building structure by applying blocks

Structure expansion is the process to derive a new candidate structure from an old one by adding more structures to it. After derivation, the old candidate structure is still in the memory, and a tree-shape data structure is used to track the derivation path. The structure is called ‘expansion tree’.

For example, the expansion tree shown below has only one branch, on which there are three candidate structures, namely No.1, No.2, and No.3. The arrows indicate the derivations’ direction. Numbers in ‘[ ]’ after the structure’s serial number is the structure’s likelihood to reproduce the reference behavior.

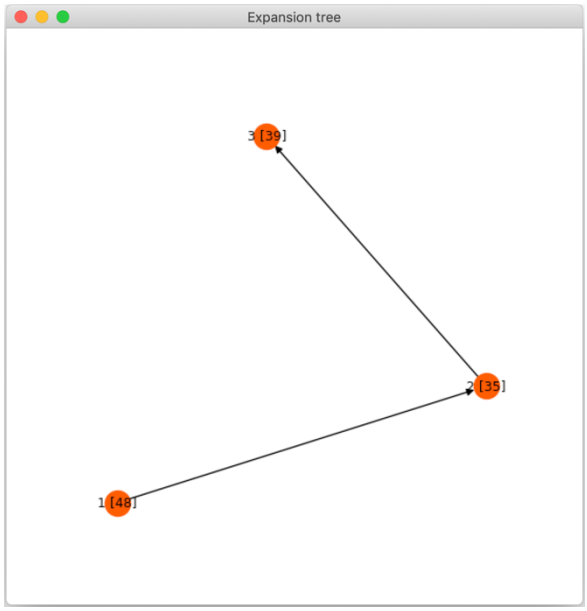


Figure 23 Example of an expansion tree

As discussed in Section 3.1.2, structure expansion is done through two different mechanisms. One is ‘bottom-up’, the other is ‘top-down’. ‘Bottom-up’ is more random than ‘top-down’, since it only follows basic system dynamics modeling rules, while ‘top-down’ is to apply a generic causal structure, for example, a first order negative feedback loop, to a candidate structure.

In this run, ‘top-down’ comes in first. The system recognizes the reference mode for ‘tea cup’ as ‘Decline (c)’, which in Barlas and Kanar (2000)’s dynamics patterns is the third one from the left.

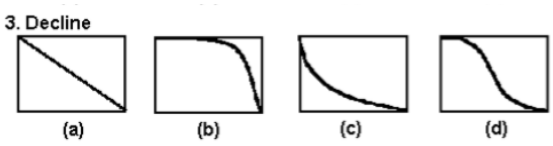


Figure 24 The decline family of behaviors in Barlas and Kanar (2000)

The generic causal structure associated with ‘Decline (c)’ is a first order negative feedback loop. As mentioned before, it is not a discrete model structure, but the abstract concept of this loop. A chain of functions is used to represent this concept. In the platform’s denotation, the feedback loop looks like:

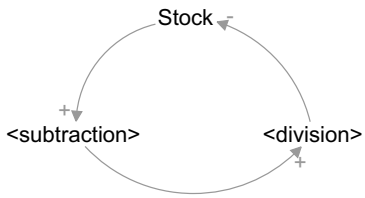


Figure 25 Function chain as the concept of a feedback loop

It means this loop starts with a stock, then calculates the difference between this stock and some other variable. The gap between them is be divided by yet another variable. The quotient calculated from the division is used as a flow to influence the starting stock. This is more like a 'backbone' of a feedback loop than a feedback loop itself.

Applying this concept of feedback loop to candidate structure No.1, a new candidate structure – No.2 is derived. As shown in figure 26, 'variable_1' and 'variable_2' are created with subtraction and division as their functions respectively. Parameters are set to their default value, because they have not been calibrated.

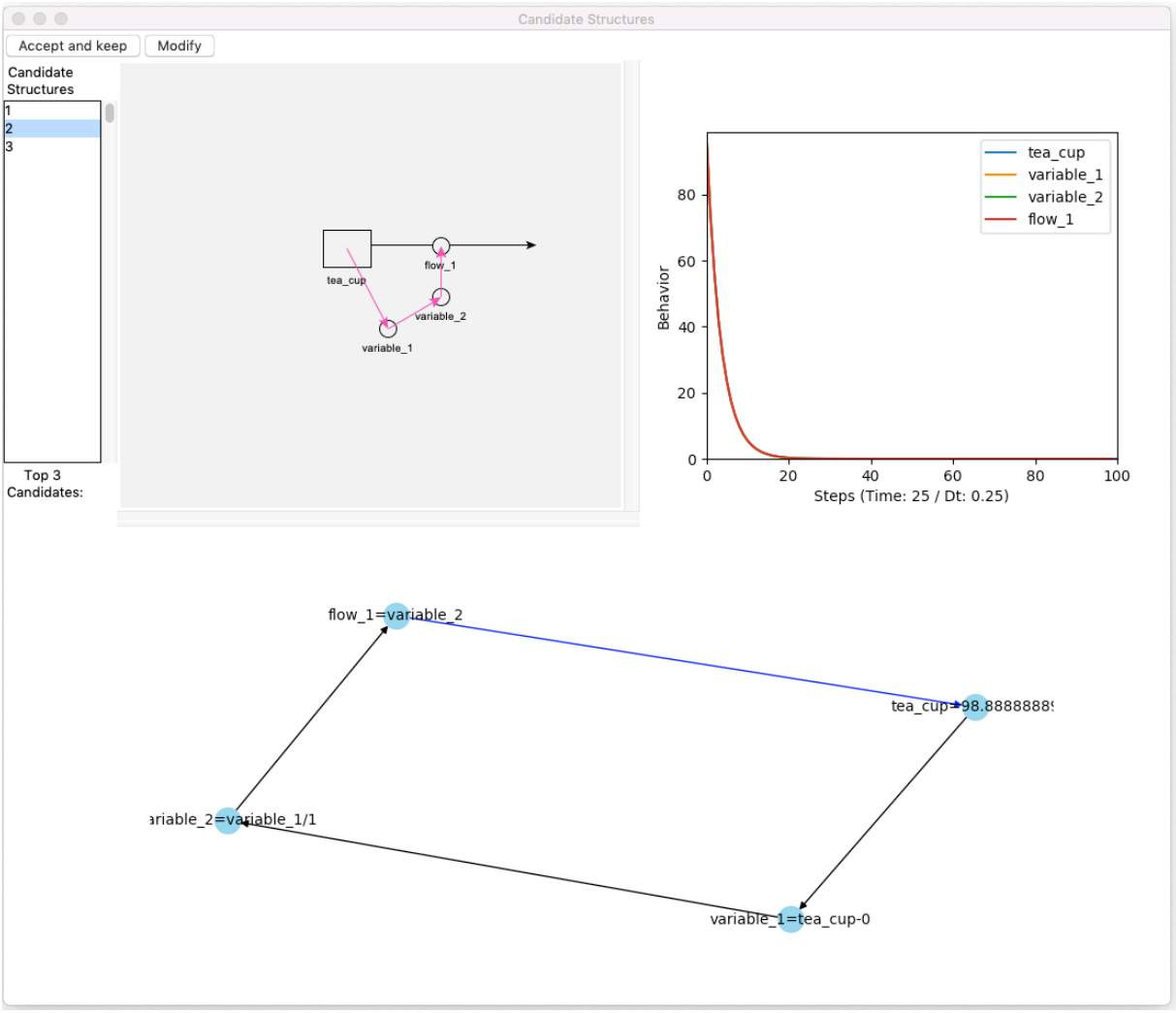


Figure 26 Candidate structures (2)

Step 4: Calibrating parameters

Parameter calibration comes after structure expansion. In this step, the system goes over the variables in a model to look for parameters and calibrates them. For example, the equation in 'variable_1' is:

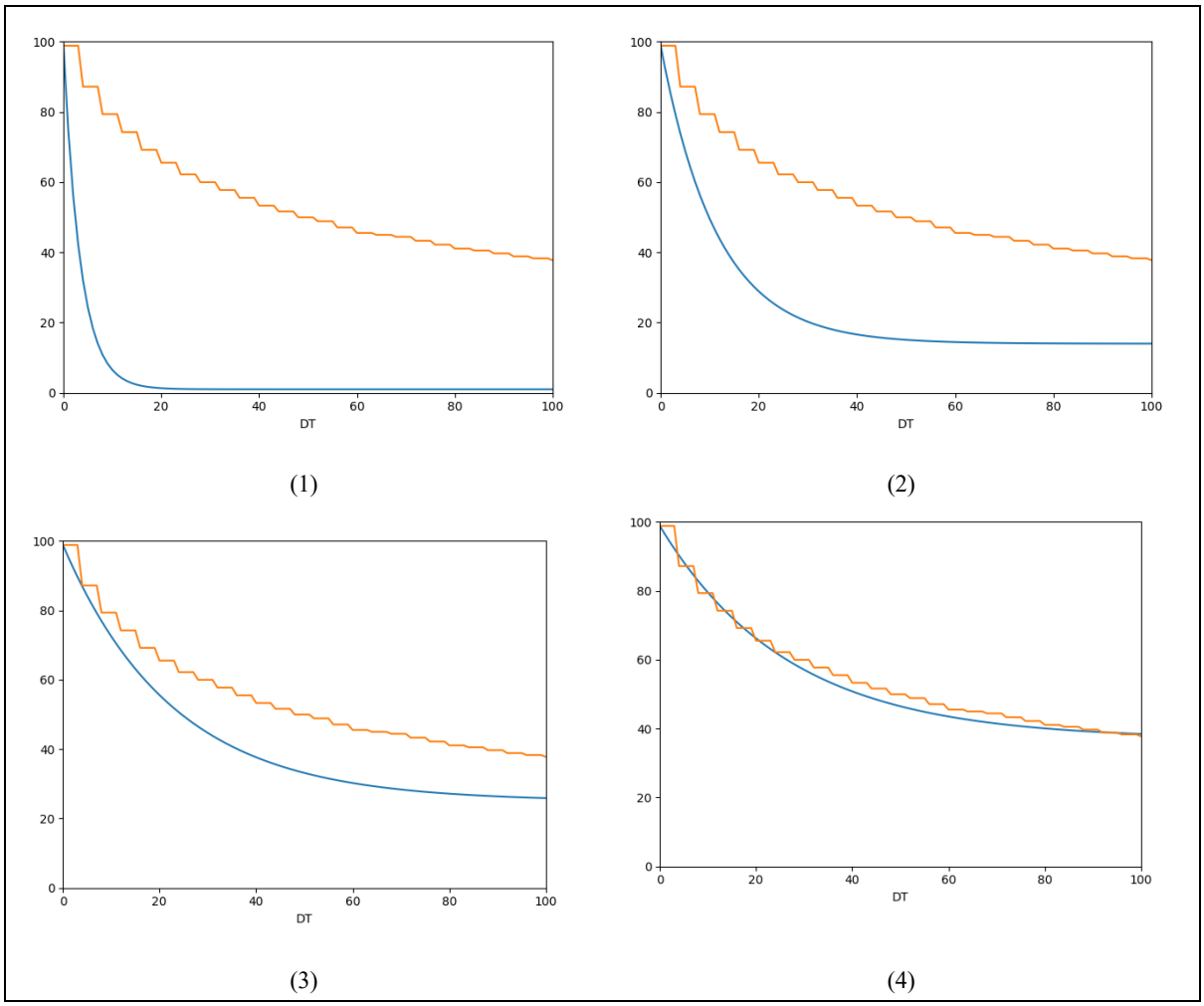
$$variable_1 = tea\_cup - 0$$

In this case, 0 would be the parameter to adjust. Following this process, the system identifies 2 parameters in candidate structure No. 2, one in the 'subtraction' function, another in the 'division function', whose equation is:

$$variable_2 = variable_2 \div 1$$

The goal of the calibration algorithm is to minimize the distance between the model's behavior and the reference mode. The distance is calculated through 'dynamic time warping' (DTW), an algorithm designed for calculating similarity between time-series (Berndt & Clifford, 1994). Inspired by the gradient descending algorithm, a similar algorithm is developed to minimize this distance by gradually adjusting parameters.

Table 8 Process of parameter calibration



As calibration goes on, the model's behavior gets closer to the reference mode. In the meantime, changing history of the two parameters is also recorded. They stabilize at 36 and 6 respectively after an adjusting phase.

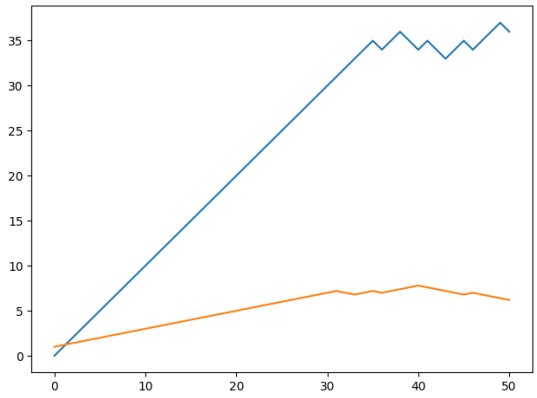


Figure 27 Changing history of the two parameters

Figure 28 shows the calibrated candidate structure No.3, which is derived from No.2.

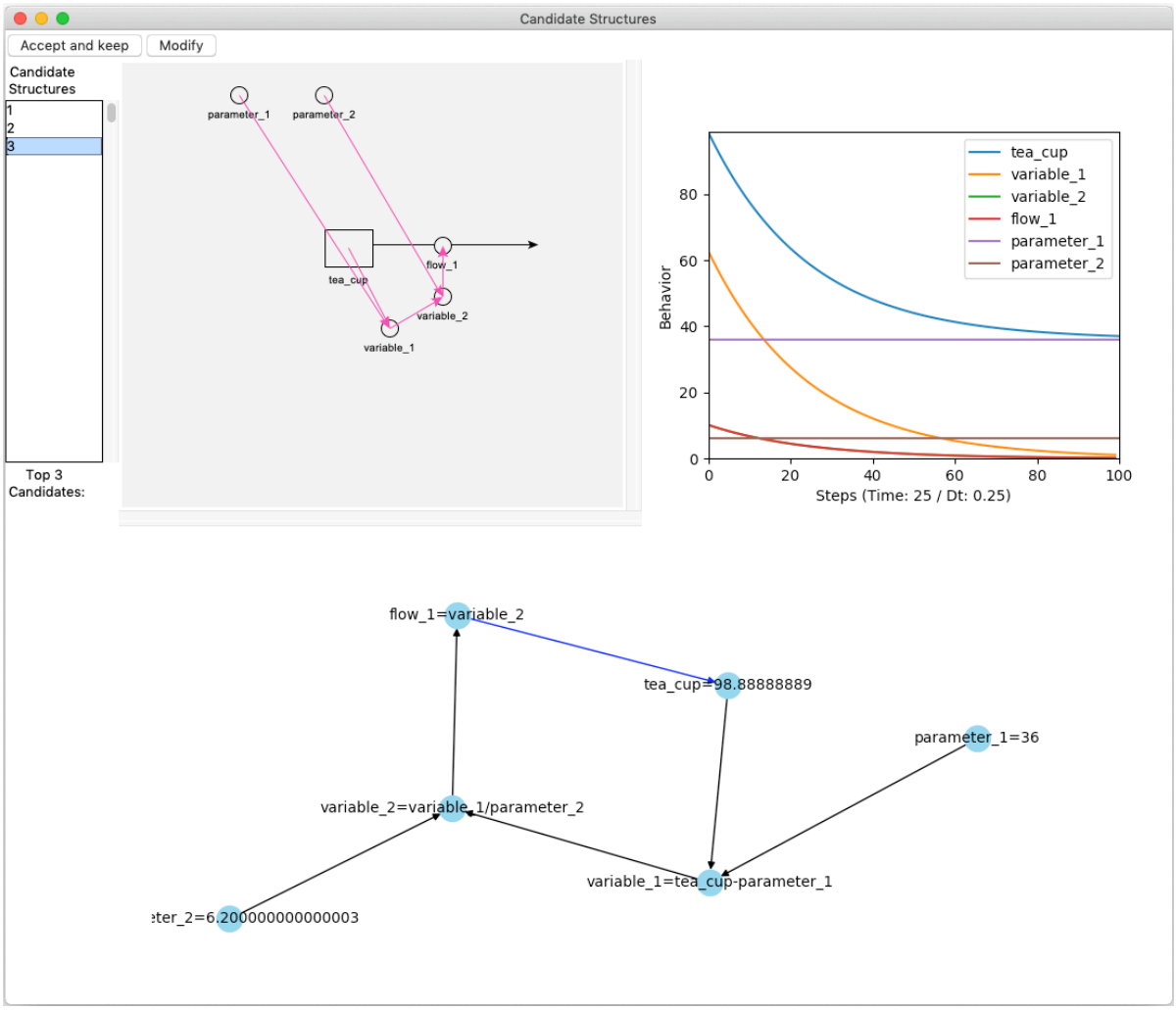


Figure 28 Candidate structures (3)

4.2 Case study 2: Tea cup case with time-series data and structural information

4.2.1 Case description

This experiment is set up to test the platform's ability to conceptualize a simulation model from time-series data while incorporating structural information provided by the problem owner.

In system dynamics modeling process, there are very often mental and written information for model conceptualization. They include both understandings of the situation in which the problem happened and preliminary hypotheses about the structure that could explain the problematic behavior (Forrester, 1980a). As discussed before, given that it is still difficult for algorithms to extract such information automatically from mental and written data, the modeling platform is designed to be interactive so that the user can build a piece of structure they know before or during a conceptualization process.

In this case, in addition to time-series data provided as in case study 1, additional information about the structure is given as:

"The cup's cooling down speed might have to do with the gap between the cup's temperature and something else."

This information implies a hypothesis about a part of the target model structure, but still far from a complete model. The platform is supposed to begin the conceptualization with an incomplete model that already contains structure built by the user, which reflects the above sentence.

4.2.2 Experiment process and result

Step 1: Load reference mode

The reference mode is loaded in the same way as in case study 1 and a root candidate structure is generated as a starting point.

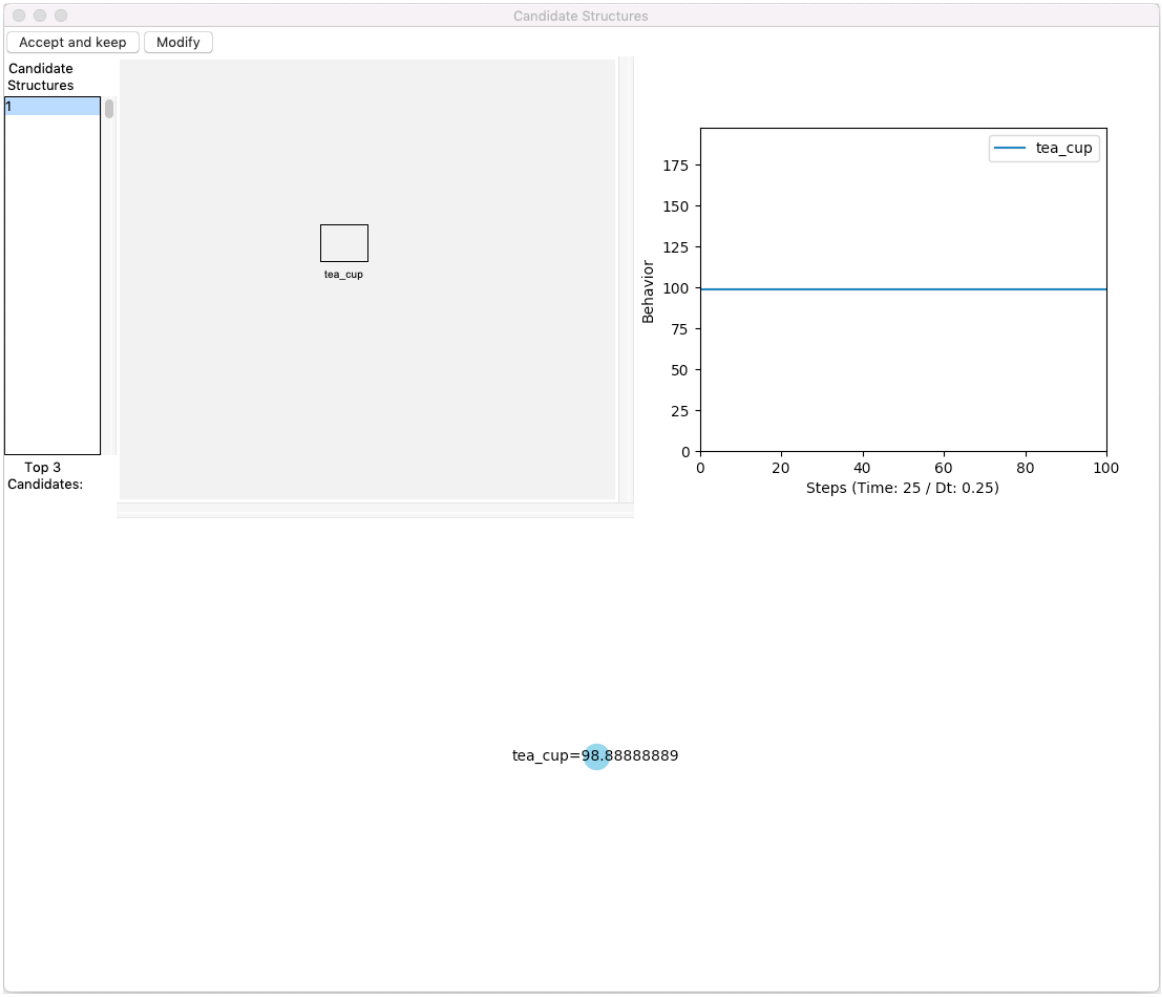


Figure 29 Candidate structures (4)

Step 2: Incorporate the structural information from the problem owner into the model

In this case, in addition to the time-series, additional information about the model structure is provided as knowledge *a priori*:

"The cup's cooling down speed might have to do with the gap between the cup's temperature and something else."

From this information, a modeler could infer that the cup's temperature, which is represented by stock 'tea_cup', needs to be fed into a subtraction function to calculate a gap between itself and another unknown variable. Such a structure, although incomplete, still reflects some preliminary understanding of the feedback structure underlying the problematic behavior, and therefore needs to be provided to the platform before starting conceptualization.

After loading the reference mode, an initial candidate structure is generated (figure 29). In the candidate structure window, the user can modify a candidate structure by clicking 'modify'. Multiple tools are provided for modification, as shown in figure 30.

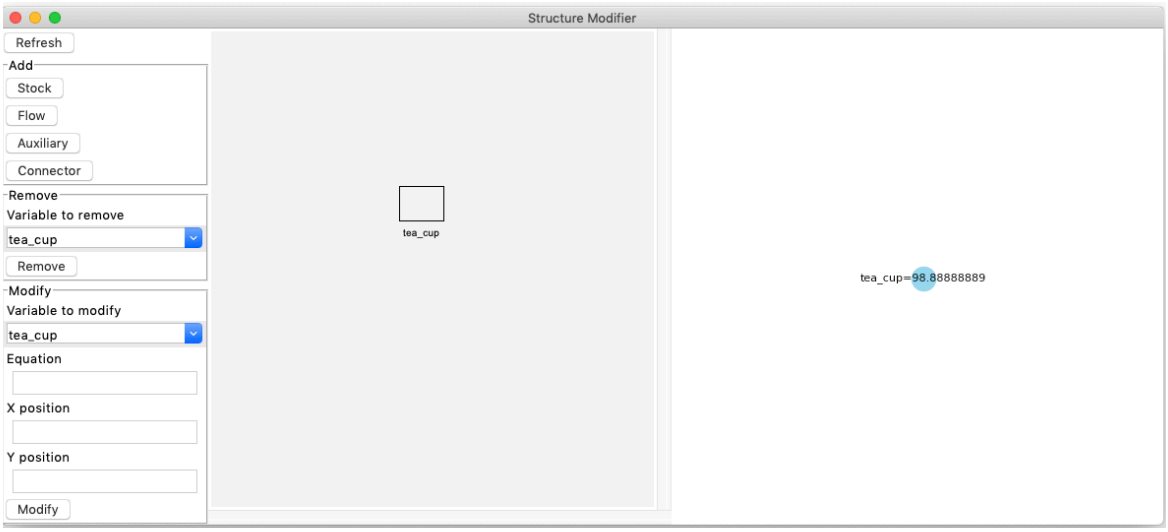


Figure 30 Structure modification

A variable is needed to represent the function 'subtraction'. It can be added through clicking the 'Auxiliary' button (figure 31). The value of this variable should be a subtraction function, in this case '*tea_cup – 0'*. The '0' is used as a default value for a parameter in the subtraction function when information about this parameter is not available. Finally, because this function indicates a dependency on 'tea_cup', a causal relation from 'tea_cup' to 'gap' is added through 'Connector' (figure 32).

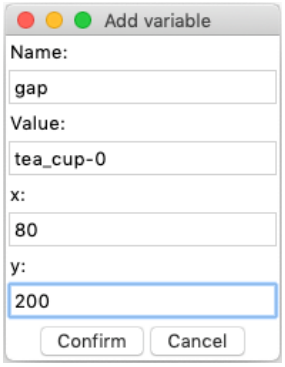


Figure 31 Adding variable

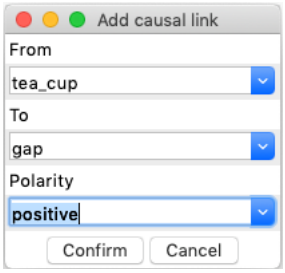


Figure 32 Adding connector

The modified initial candidate structure to begin with is shown in figure 33.

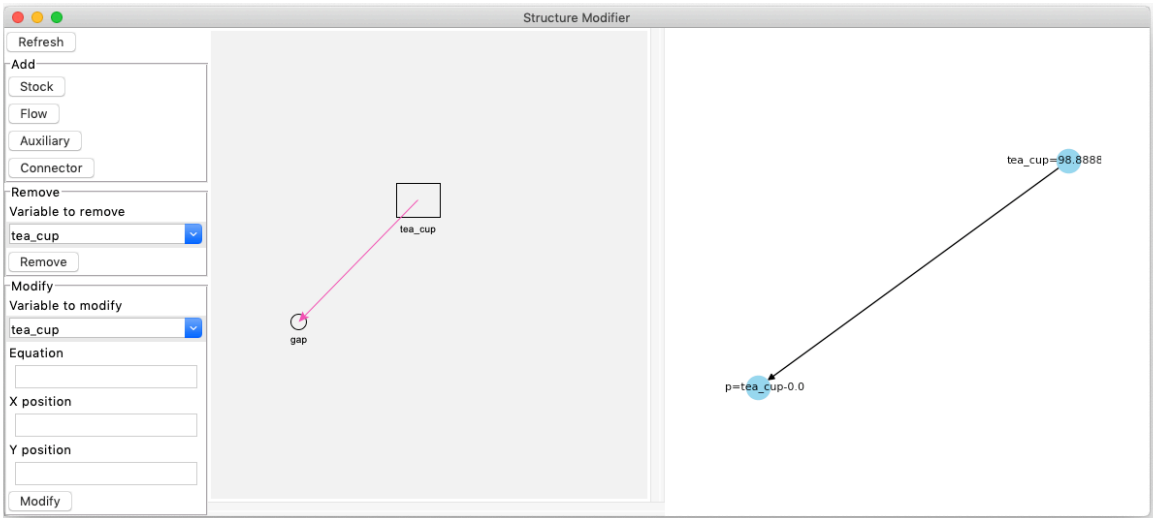


Figure 33 Modified candidate structure

Step 3: Building structure by applying blocks

As in case study 1, the ‘top-down’ mechanism comes first when structure expansion begins, trying to apply a first order negative feedback loop to this candidate structure (No. 1). This mechanism is designed to be able to search for needed functions in the existing structure before building new ones. In other words, if a variable containing the needed function already exists, the algorithm will try to use it. Such a feature effectively enables the incorporation of structural information provided by the problem owner. As shown in figure 34, the algorithm recognizes the subtraction function in the variable ‘gap’ and includes it in the feedback loop.

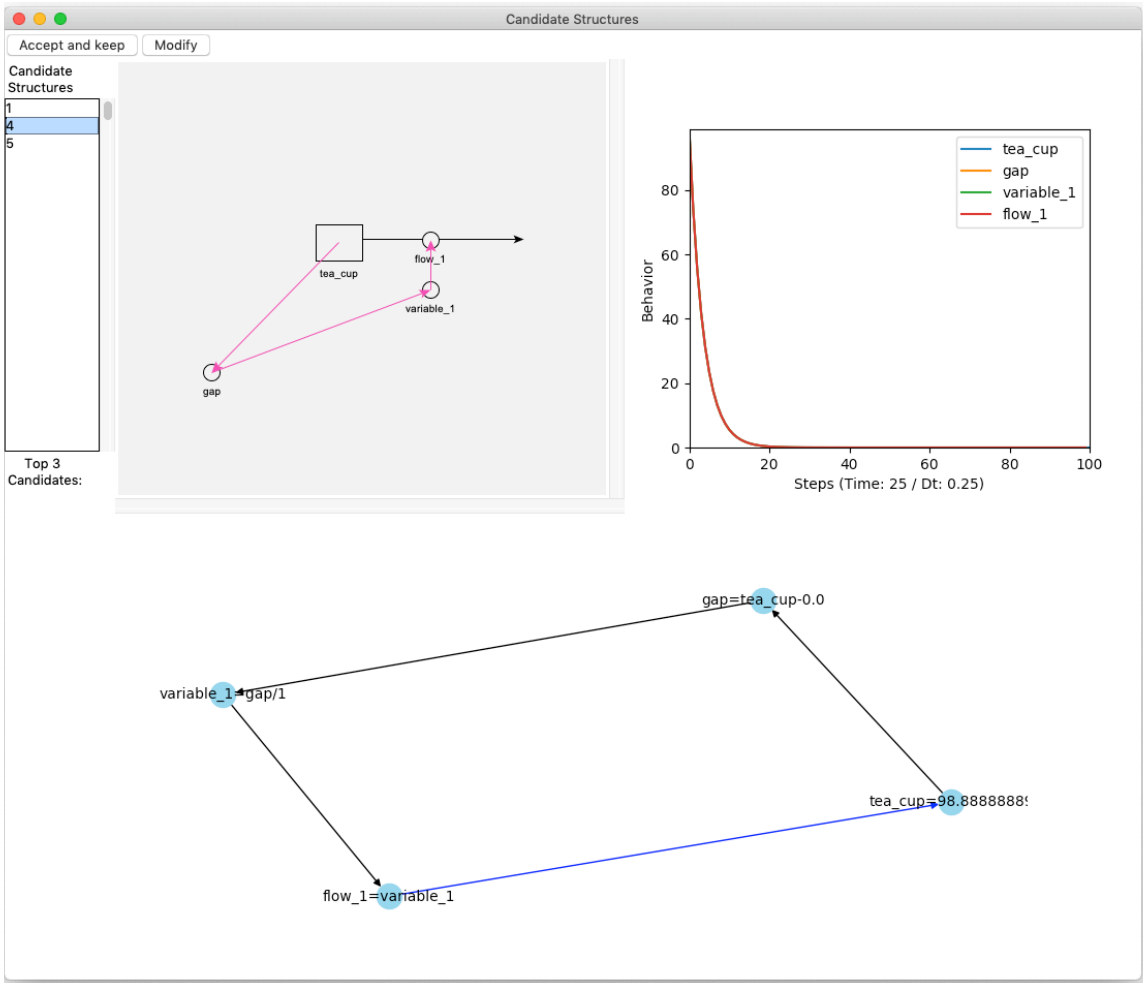


Figure 34 Candidate structures (5)

Step 4: Calibrating parameters

The rest of this experiment is identical to case study 1. Parameters for the subtraction function (parameter_1) and the division function (parameter_2) are calibrated to make the behavior fit the reference mode as shown in figure 35.

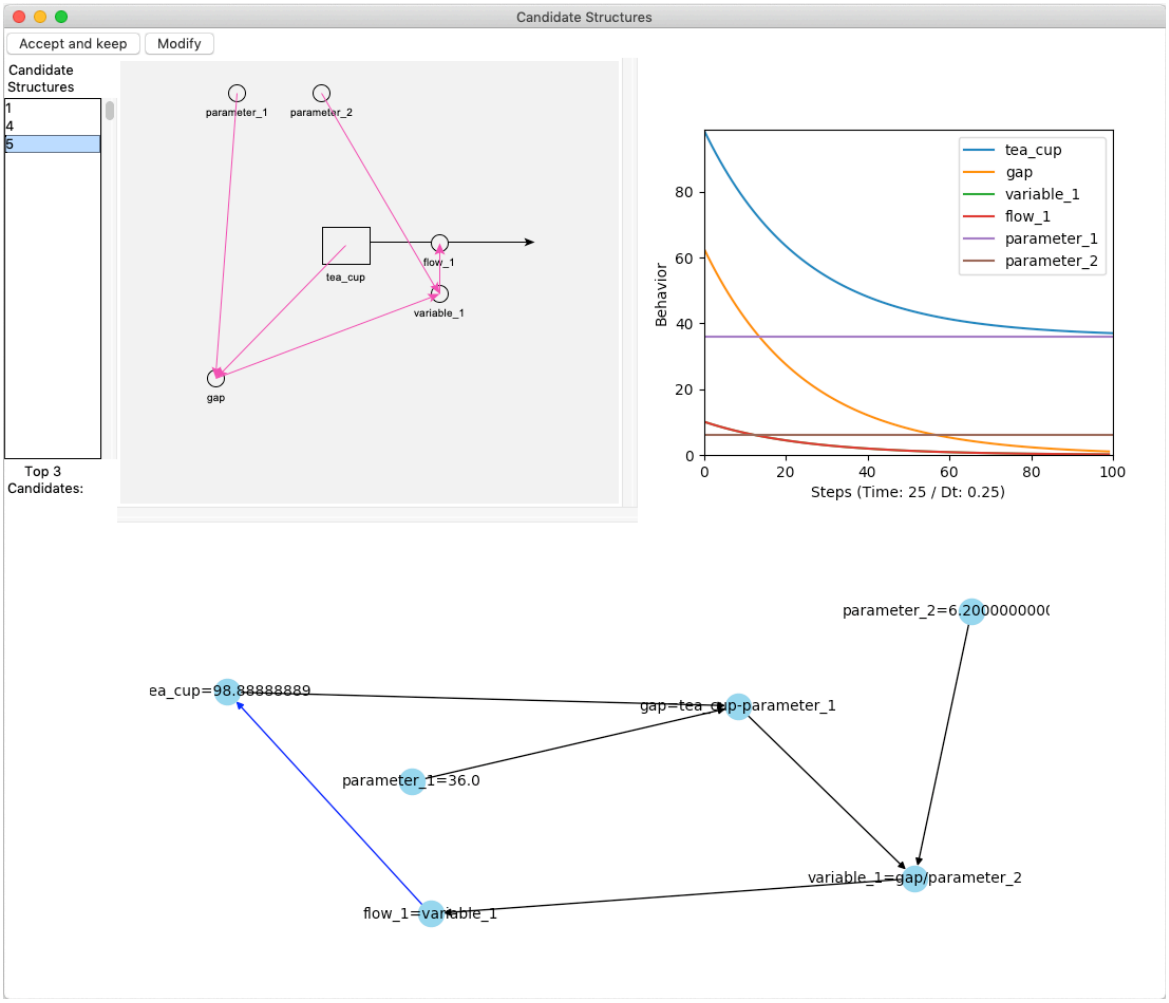


Figure 35 Candidate structures (6)

4.3 Case study 3: Water sink case with additional reference mode provided during conceptualization

4.3.1 Case description

In system dynamics modeling process, not all the information will always be readily available from the very beginning. Due to the iterative nature of this process, one could assume that new information can come up to the modeler throughout the entire modeling process. Additional information, whether it provides structural or behavioral insights, can be used to confirm or reject existing hypotheses. In the case of rejection, a modeler would think of a new possible structure. Such a maneuvering process reflects adaptivity, which is required for an automated model conceptualization algorithm.

This experiment is therefore set up to test the platform's adaptivity to additional behavioral information provided in the course of conceptualization.

The case used here is a water sink. In this situation, water is added to a sink from a faucet and in the same time drained through a pipe. The complete target model structure is:

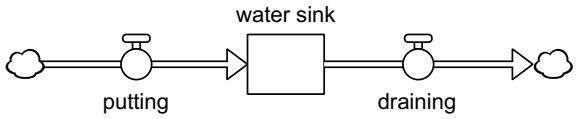


Figure 36 Complete SFD for the water sink case

However, information for conceptualization is only partly available in the beginning. Available behavioral information is the reference modes for water level in the sink:

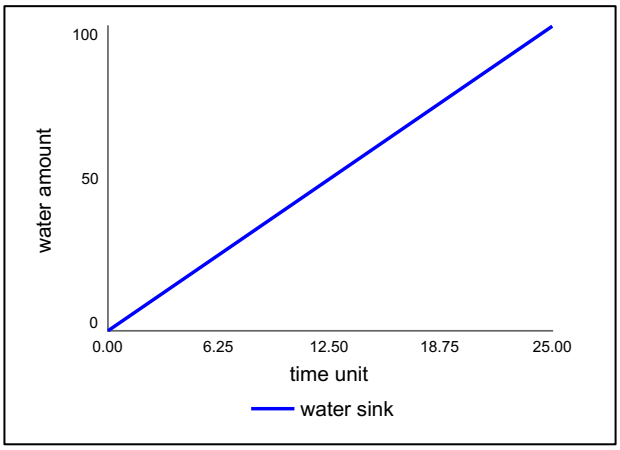


Figure 37 Reference mode for water sink

Available structural information is about the faucet:

"Water is added to the sink through a faucet."

Sometime after the model building process begins, additional behavioral information about the faucet becomes available, shown as the following reference mode:

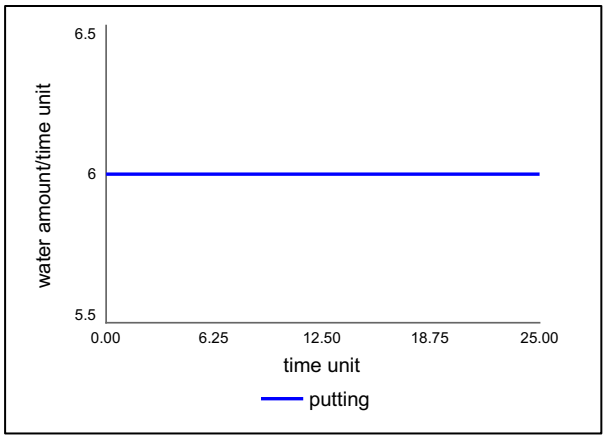


Figure 38 Reference mode for faucet

A modeler will then take this additional information into consideration and find out that this inflow could not itself generate the stock's behavior, because the reference mode for water sink indicates a net

flow of 4 *water amount per time unit* but this inflow is 6, according to its reference behavior. Such contradiction will lead the modeler to a new hypothesis that there could be an outflow as well.

This experiment is to test if the platform can reproduce this deliberating process.

4.3.2 Experiment process and result

Step 1:

The time series data for the water sink is imported to the platform.

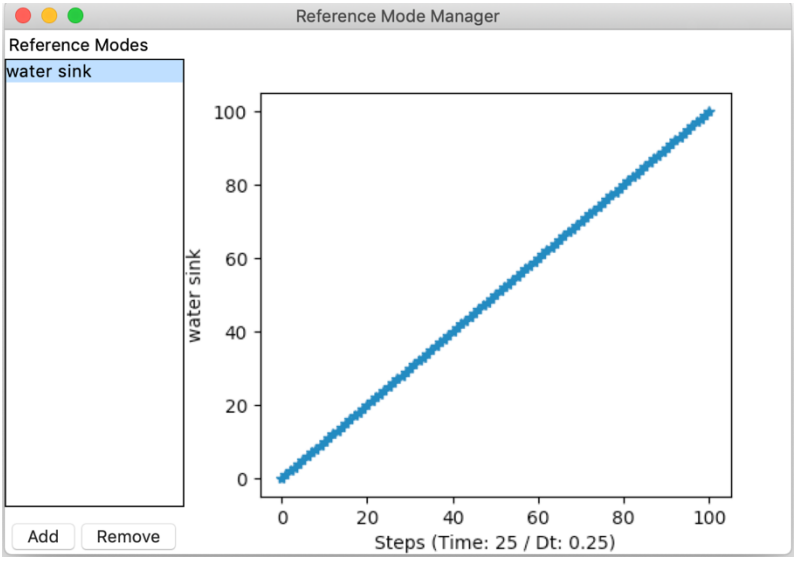


Figure 39 Adding reference mode for water sink

As described above, the modeler has known that water is added to the sink through a faucet. This could be represented by an inflow connected to the stock. Before conceptualization starts, the inflow is manually built to the initial candidate structure by the user, with a default value of 0, as shown in figure 40, 41, and 42.

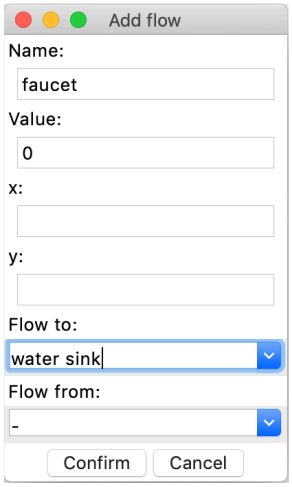


Figure 40 Adding inflow 'faucet' to water sink

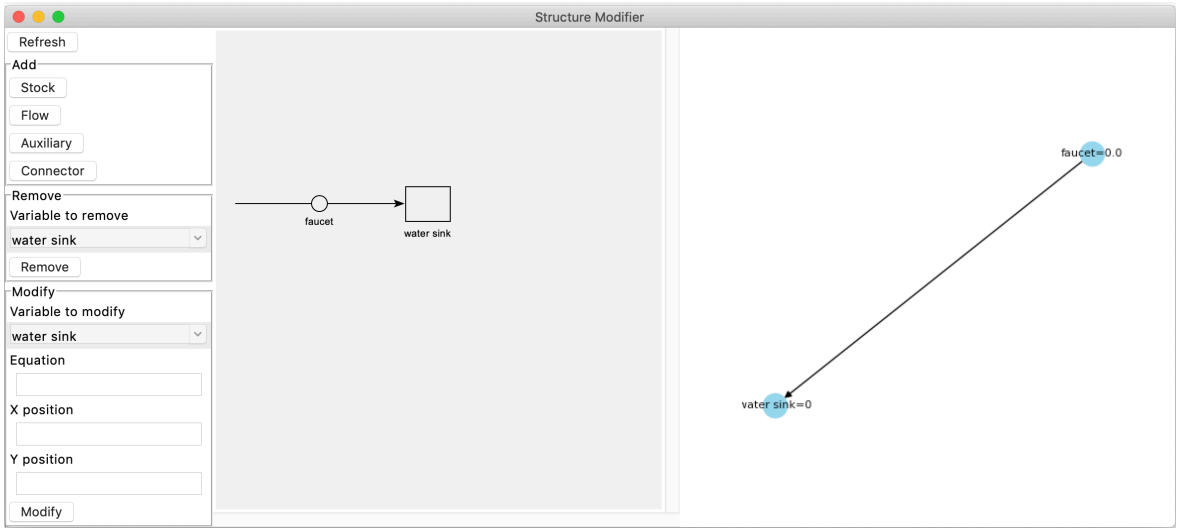


Figure 41 Inflow 'faucet' connected to water sink

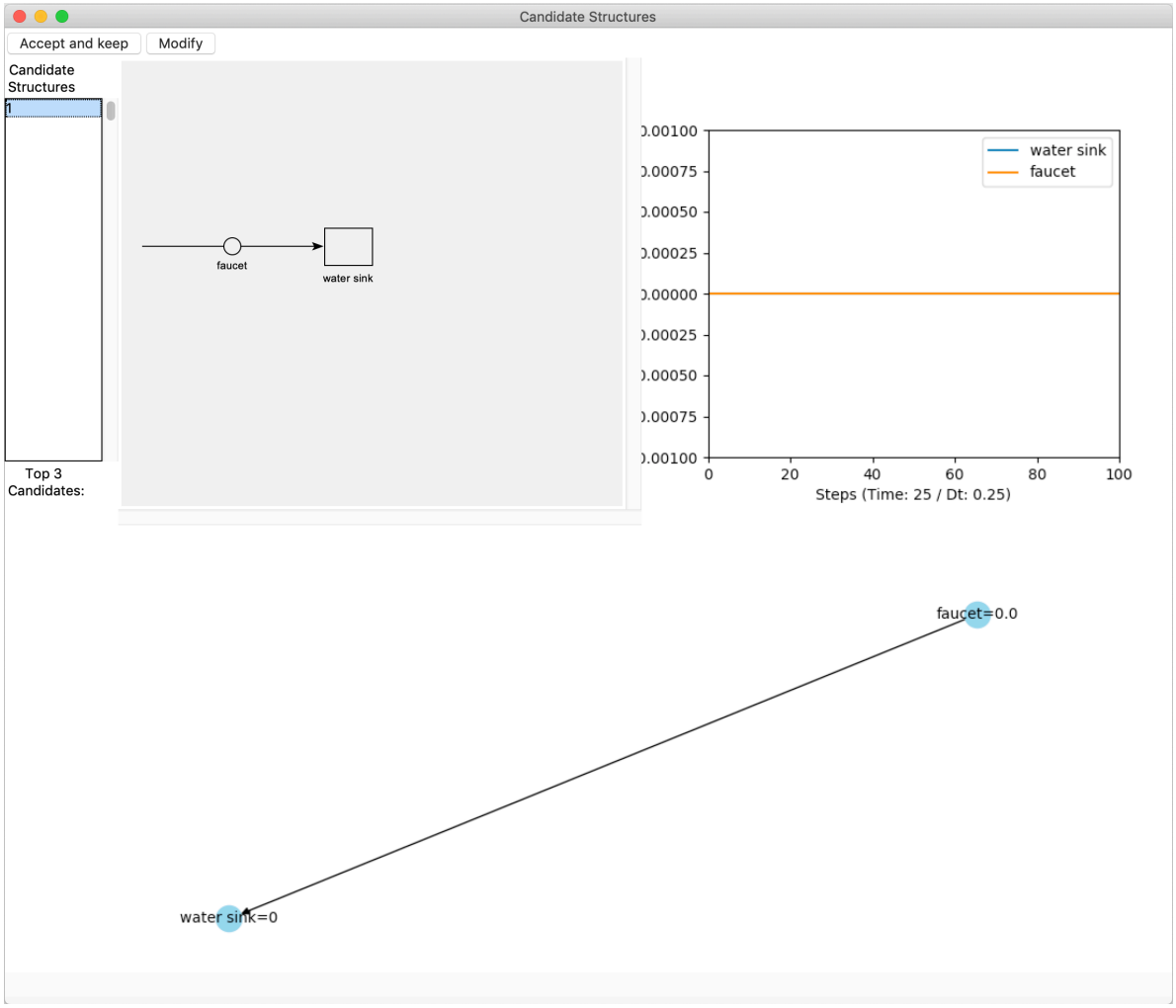


Figure 42 Candidate structure to begin with

As all knowledge *a priori* has been provided to the platform, automatic structure generation and parameter calibration are performed. It does not take a long time for the system to reproduce the

reference mode by calibrating parameter:

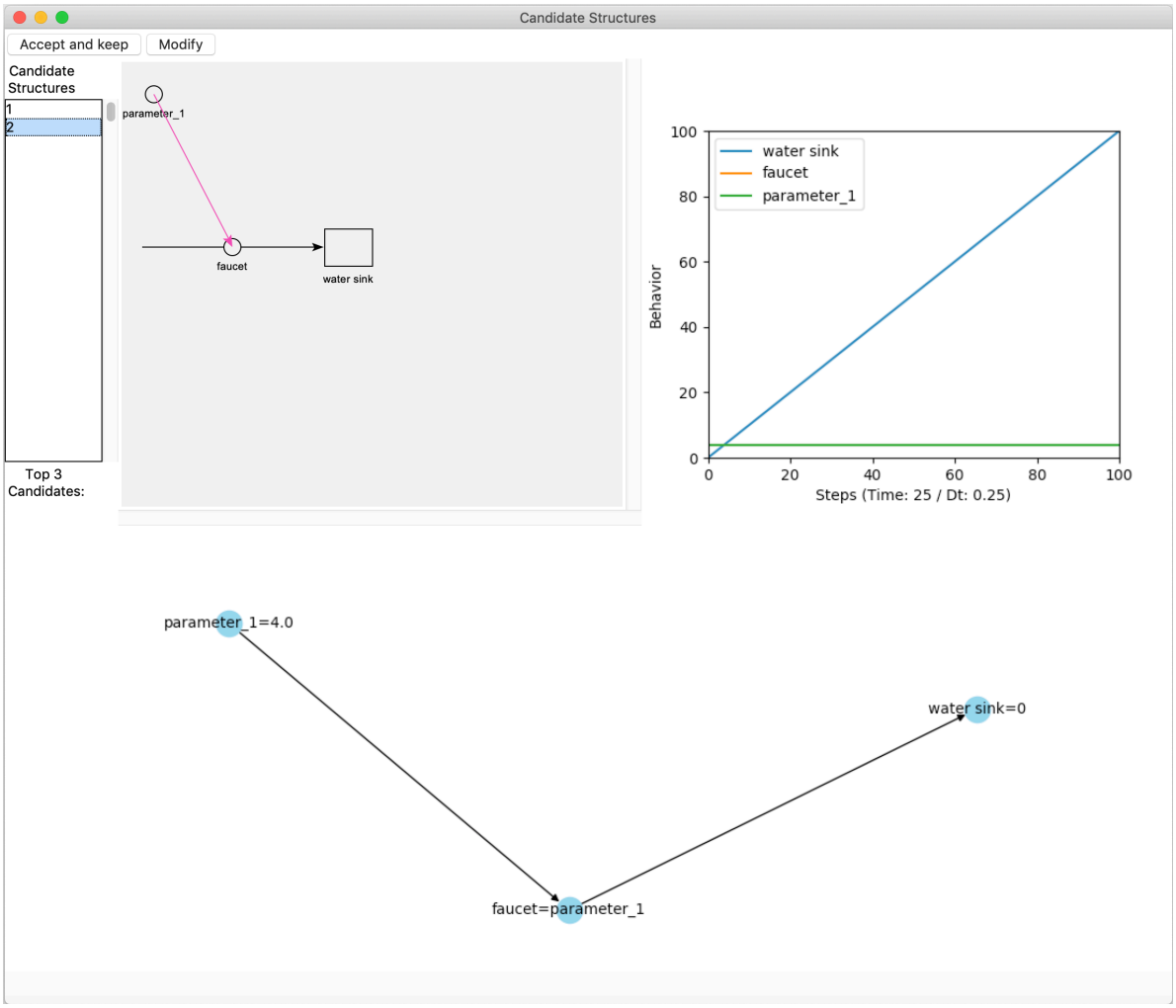


Figure 43 Conceptualization result based on information provided so far

As shown in figure 43, in candidate structure No.2, the default value 0 in the inflow 'faucet' is replaced by a parameter, and the value of the parameter has been calibrated to 4, exactly the net flow required for reproducing the reference mode.

Step 2:

Reference mode for the inflow 'faucet' is now provided as additional information. In this case, it is loaded from file and managed by the reference mode manager of the platform (figure 44) as 'filling'.

After adding the reference mode, it is still needed to let the platform know which variable the reference is bound to. In other words, the platform should know 'this is the reference mode for which variable'. A 'binding management system' is therefore implemented. On startup of the platform, when an initial reference mode is loaded and a variable is built for it, a binding is automatically created. Now a new reference mode needs to be assigned to an existing variable (in this case it is the inflow 'faucet'). The user can manually do this with the binding manager (figure 45).

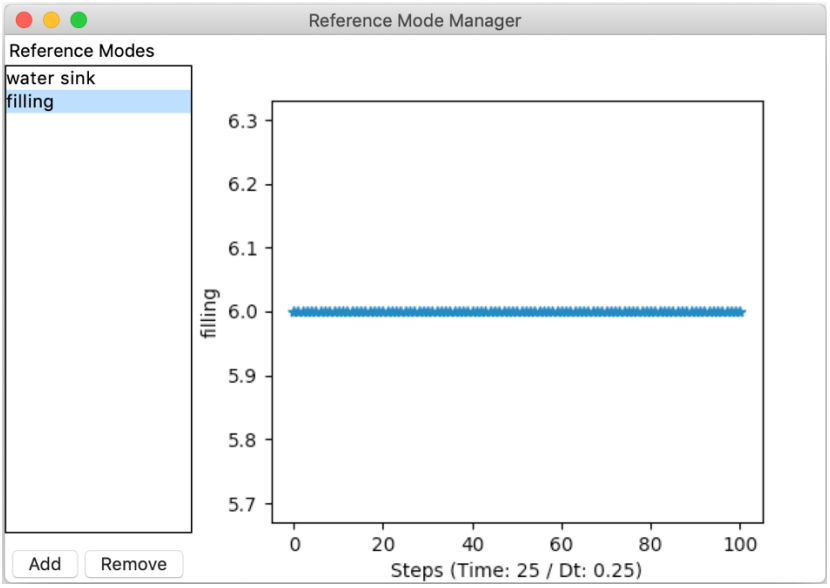


Figure 44 Adding reference mode for inflow 'faucet'

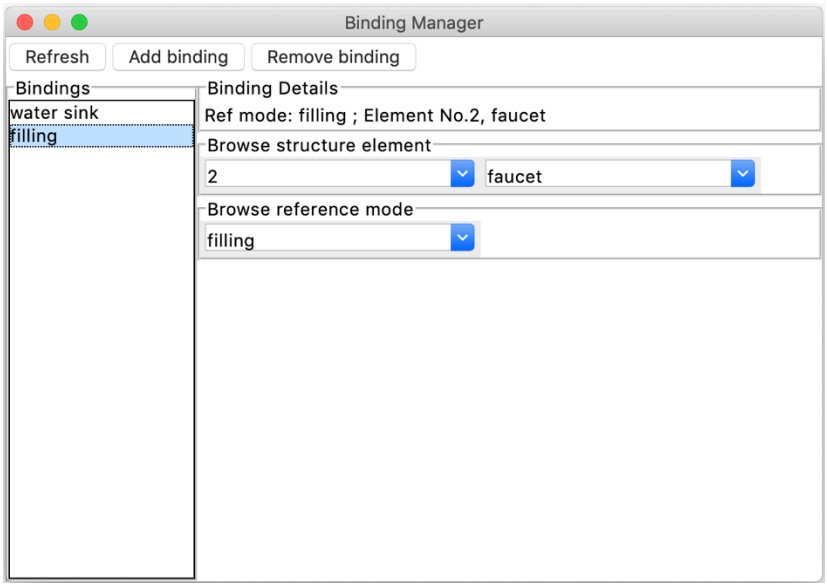


Figure 45 Binding 'faucet' to its reference mode

In the binding manager, a list of existing bindings in the system is shown on the left side. The details of a selected binding are displayed in the section 'binding details' on the right side. For example, '*Ref mode: filling; Element No. 2, faucet*' means the inflow 'faucet' is bound to reference mode 'filling'. Such a binding can be created by:

1) Browse through all candidate structures and choose the one that contains the variable to be bound (in this case it is candidate structure No.2);
2) Browse through all variables in this candidate structure and select the variable to be bound (in this case it is 'faucet');
3) Browse through all reference modes managed by the system and find the one to use (in this case it is 'filling');
4) Click the 'Add binding' button on the top.

Step 3:

With the updated information, a new iteration of conceptualization begins. Because there are now two reference modes (‘water sink’ and ‘filling’) bound to two variables respectively, the performance of a candidate structure is measured by the fitness of both reference modes. Therefore, the suggested candidate structure in step 1 will no longer be a good solution, because while the reference mode for stock ‘water sink’ is matched, the behavior of ‘faucet’ (which is 4) does not match its reference mode (which is 6).

A drop in overall fitness is deemed by the system as a ‘rejection’ of a candidate structure. It is noteworthy that the system does not use the concept ‘rejection’ directly; instead, it is a dynamic process in the prioritizing system. As described in Chapter 3, the likelihood of one candidate structure to become the target structure is measured mainly by the fitness of its behavior to reference modes takes a big part. A likelihood number is used to represent a candidate structure’s performance. Candidate structures with higher likelihood will receive more attention from the system, which means they are more likely to be chosen for further structural expansion and parameter calibration.

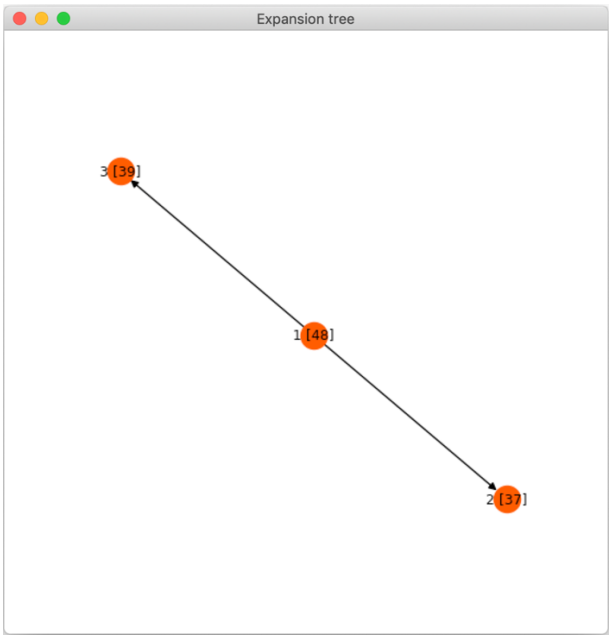


Figure 46 Expansion tree of this conceptualization task

In case study 3, this process is revealed by the ‘expansion tree’ (figure 46). Candidate structure No.1 is the initial candidate structure, and No.2 is the candidate structure suggested in step 1. As a drop in behavioral fitness of No.2 happens as a result of the additional reference mode, more attention is paid again to No.1, making it more likely to be chosen for further expansion. The expansion happens and a new candidate structure (No. 3) is derived from No.1 by adding an outflow (flow_1) to the stock ‘water sink’, as shown in figure 47. This structure is different from No.2 as it contains two flows while No.2 contains only one. Parameter calibration is then carried out for this new candidate structure, through which values of the 2 flows are adjusted, and both of the reference modes (‘water sink’ and ‘filling’) are successfully reproduced. A high behavioral fitness makes No.3 more likely to be the target structure, and the user can see this from the expansion tree.

The outcome of step 3 is shown in figure 47. The value of inflow ‘faucet’ is calibrated to 6 and the value of outflow ‘flow_1’ is calibrated to 2. They jointly make the stock ‘water sink’ increase from 0 to 100 in 25 time-units, perfectly reproducing the reference mode. As it also complies with the provided structural information (‘water is added through a faucet’), it could be considered as one explanation of the observed situation. Moreover, it provides a hypothesis that there might be water flowing out from the water sink, and this hypothesis is to be validated in the real world by the user. For example, the user can go to check if the water sink really has a drain.

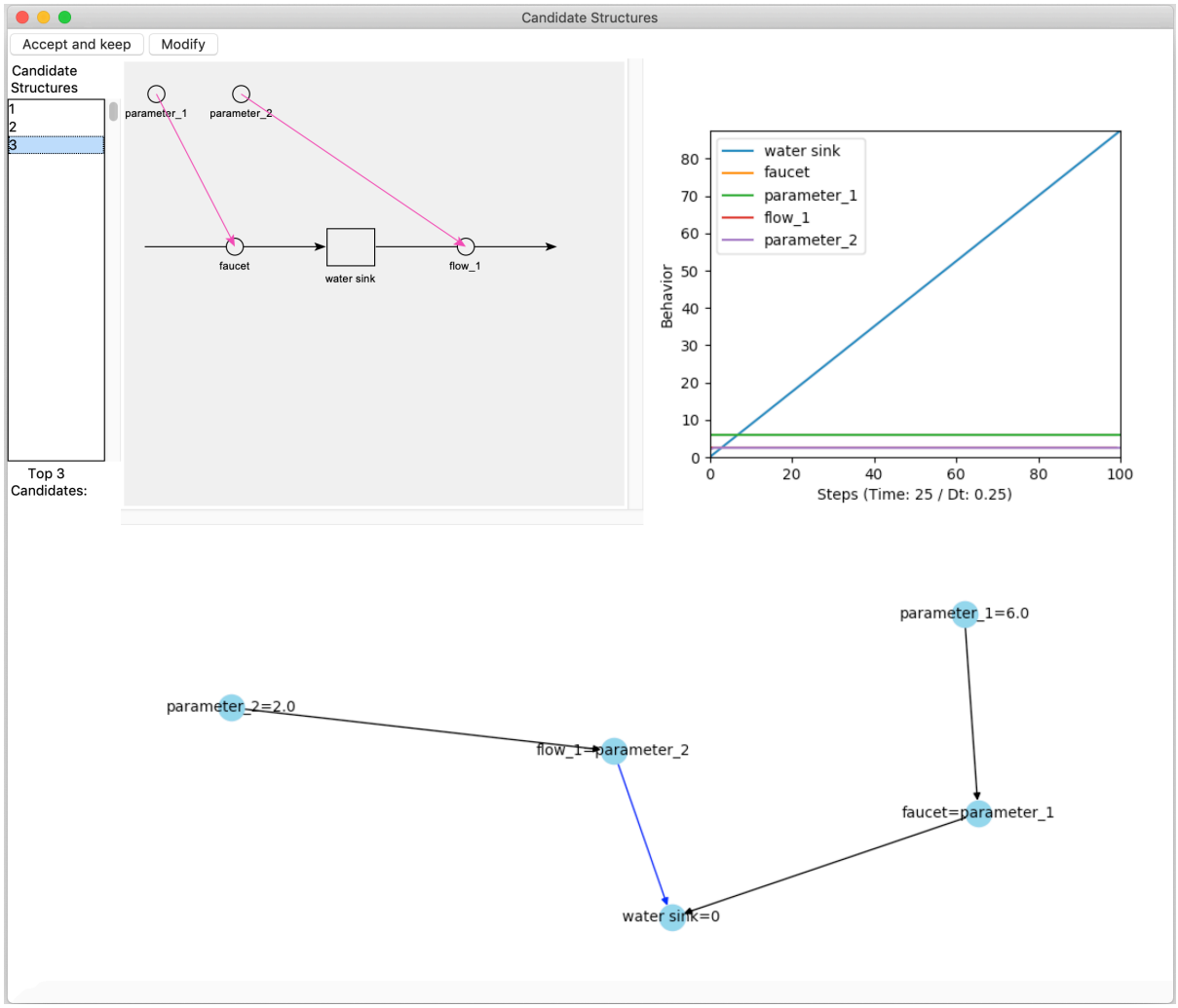


Figure 47 Conceptualization based on information available so far

4.4. Observations from the experiments

In this chapter, features of the modeling platform are demonstrated and tested through three case studies. A summary of the experiments is provided below.

Table 9 Summary of experiments

	Case 1	Case 2	Case 3
Features	Application of generic structure	Incorporating existing structure	Adaptivity to new information
Available information	Behavioral	Structural and behavioral	Structural and behavioral
Expansion mechanism	Top-down	Top-down	Bottom-up
Platform modules	1. Reference mode management 2. Dynamic pattern categorization 3. Structure expansion 4. Parameter calibration	5. Structure modification	6. Binding management 7. Behavioral fitness and 8. likelihood-based 'rejection'

Both structural information and behavioral information are utilized in model conceptualization. Structural information is provided to the platform by manually building stock-and-flow diagrams. Behavioral information is provided in the form of time-series data. The platform is able to use the provided information both before and during the conceptualization process.

Two components of model conceptualization, namely structure generation and parameter calibration are carried out by the system. The system also shows its ability to generate new model structures through two different mechanisms: the 'top-down' mechanism, which implements feedback structures guided by generic dynamic patterns, and the 'bottom-up' mechanism, which adds new variables to an existing model structure following the system dynamics modeling rules. Moreover, in case study 2, the platform shows its capability to combine the two mechanisms: variables added to the structure through the 'bottom-up' mechanism are able to be integrated or utilized by the 'top-down' mechanism.

To sum up, the results from the experiments prove that the designed features are able to work as expected.

5. Discussions

5.1 An overview of this work

This research-and-development project is based on previous works on automatic model structure generation and aims to advance the research in this direction. It follows the hypothesis-validation paradigm by putting forward a new theory of automatic model conceptualization and testing it by developing a model building software and subsequently carrying out analytic experiments on it.

Though varying in techniques, existing researches on automatic model structure generation are mainly taking a numerical data-driven perspective. The dominant method is to 'learn' model structures from time-series data. Beside the exclusive reliance on time-series data that constitutes only a small part of all types of information available for modeling, the outcome of the existing methods very often demands extensive human work to make it usable.

Having provided an analysis of model conceptualization, the author defined it as the process a modeler derives conceptual or simulation model from multiple data sources. The existing automatic methods are aiming at the same objective as model conceptualization while using much more limited sources of data because mental and written data are excluded. The author therefore postulated that beside refining the existing methods, another possible method could be to bring broader sources of data into this automated conceptualization process.

Accommodating mental and written data requires a new framework for information process, since the existing artificial neural network-based methods are suitable for processing numerical data but not for data in other forms. Moreover, the iterative characteristics of modeling process calls for an iterative workflow to allow structure generation and validation to take place iteratively.

Learning from how a human modeler conceptualizes models, the author characterized the model conceptualization process as a process that has multiple iterations, that takes information from multiple sources, that benefits from previous modeling experience, and that ranks candidate structures to find the most promising ones. The author then proposed an interactive and iterative workflow in which the participation of human modeler is included, and previous modeling experience is consulted. In one iteration of the workflow, new hypotheses of structure (candidate structures) are automatically generated through 'top-down' and 'bottom-up' mechanism and validated by both comparing behavior to reference mode and by modeler's selection. The hypotheses-validation process observed in human modeling is therefore reproduced in this human-machine interaction, with non-numerical information handled.

The workflow is implemented with a Python-based modeling platform and experiments are carried out to further demonstrate the features and test its performance.

5.2 Innovations of this work

A framework to utilize information from multiple sources

Forrester (1980a) categorized information for system dynamics modeling into three types, namely mental data, written data, and numerical data. This work intends to bring more types of data than only numerical to the automatic model conceptualization process. The additional types of data are mental data from the problem owner and knowledge *a priori*. As they have not been included in automatic model conceptualization before, a new framework is proposed to handle all these data.

Different from previous works in this field, the new framework conceptualizes models in a humanly way, reproducing the iterative hypothesis – validation workflow. Data are used to inform all steps of this workflow, both for making new hypotheses and for validating them.

It is arguably that this new framework is no longer fully automatic, for it requires participation of the user to provide structural input, as a result of that current technology is still not able to extract model structures from verbal information.

However, this expedient measure does not compromise the objective of automation. Although input of structural information is currently manually done by the user, the modeling platform is designed flexible enough to take input from automated data sources in the future. This feature is benefited from the high-level application interface used in the structure modifier of the platform (figure 41). High-level application interface allows the system to take order from its user in a natural language-like format. For example, 'flow A is connected to stock B' looks like a sentence in natural language, and there is already a function built in the platform that can connect a flow to a stock, requiring only two parameters – flow A and stock B. In the future, once the natural language process technology – which is the 'automatic data source' mentioned above – is smart enough to extract 'there is a flow A connected to stock B' from a specific text (for instance, a paragraph that describes a bathtub), this readily built application interface could be called to make the corresponding structural change to the model.

In a word, this work contributes not only by the automation it realized, but also by the framework that could accommodate future automation.

A graph network-based representation of system dynamics models

The platform is able to both add new variables to and modify existing variables in a model structure following a humanly way. This feature guarantees the transparency of the model conceptualization process, so that every step taken by the algorithm could be observed and understood by the modeler. It is made possible by graph network, a data structure that is adapted to store, represent, modify, and simulate system dynamics models.

Graph network comes from graph theory. It is a data structure that can store a broad range of information in 'nodes', and relationships between them in 'edges'. Concepts familiar to us in system dynamics such as variables, connectors, and loops all have their counterparts in graph networks. A wide range of algorithms to operate graph networks have already been developed, making it easier to operate a graph network-based system dynamics model: for example, detection of feedback loops in the model structure

can be done through one simple command.

In addition to this new representation of SD models, the author also developed a simulation engine that could simulate such a model without using other modeling software. This is not the first Python-based system dynamics simulator, as the PySD project has become a reliable simulation engine that could run system dynamics models under a Python environment (Houghton and Siegel, 2015). However, what has not been seen is an engine that integrates Python and system dynamics closely enough so that not only simulation, but also model building can be done without switching to another platform. A by-product of this work is a Python-based system dynamics modeling environment. As data science is becoming more significant and much of its applications are in Python, a Python-based modeling environment for system dynamics will be helpful to connect data science to system dynamics.

Previous models as knowledge base

One feature of the platform is a stock of existing models, mostly generic structures. They are used here as a knowledge base. Knowledge *a priori* plays an important role in conceptualizing models, according to Forrester (1980a and 1994), for it allows a modeler to search through and adapt an existing model to a new situation, instead of modeling every time from scratch. There might be multiple methods for carrying out this 'filtering' process in computer, and the one used in this work is through comparison between behaviors.

Moreover, a stock of existing models implicitly contains rules to build system dynamics models, and if this stock is large enough, a good part of the system dynamics modeling rules could be elicited from it. For example, if in a model a stock is connected to a flow, it would be possible for the system to import this flow to a candidate structure and connect it to an existing stock. In this case, 'a flow can be connected to a stock' is not an explicit rule, but a precedent to follow. The advantage of this method is that sometimes there are so many rules to define and defining them is so time-consuming that a set of pre-built examples would work better.

To sum up, previous models can be used for two purpose: adaptation for new situations and sources of system dynamics modeling rules.

A way to operationalize generic structures

Paich (1985) defines generic structures as 'dynamic feedback systems that support particular but widely applicable behavioral insights'. Generic structures help system dynamists to generalize their learning results from specific social problems by storing and applying their insights in an integrative form (Lane 1998). Forrester has asserted that 'probably twenty basic structures would span 90% of the policy issues that most managers encounter' (Forrester, 1980b), underscoring the importance of generic structures.

Up to now, generic structures are used to help leaning and model building by modelers. This work, however, demonstrates a way to use them for automatic model conceptualization, which is a further operationalization of them. Moreover, Hines (1996) studies structure blocks that are often used in system dynamics modeling and proposed a repository of structure molecules. They range from 'bathtub' to 'aging chain', each associated with a situation and able to simulate. Arguably, these molecules of structure could be operationalized in the same way as in this work, reinforcing the capacity of the knowledge base.

Feedback loops characterized by a chain of functions

Feedback loops are of great importance in a generic structure. As demonstrated in 4.1.2, although a feedback loop is usually called by its name (e.g. 'first order negative feedback loop') or the generic structures name (e.g. 'goal-gap model'), they are in essence neither a name nor a set of variables' names, but a chain of functions. Taking the example in 4.1.2, a goal-gap model consists of the following variables:

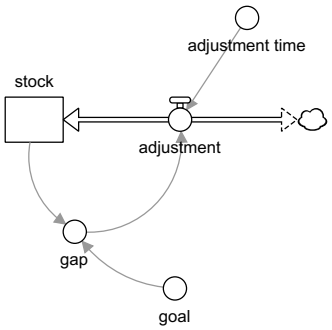


Figure 48 A goal-gap model

However, what characterizes its essence would be a chain of functions:

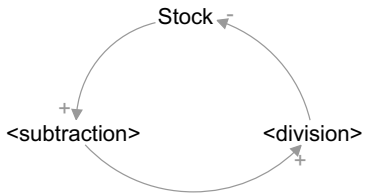


Figure 49 Function chain elicited from the goal-gap model

In this way, we can distill a chain of functions from every feedback loop and two loops that have the same chain of functions will be categorized as the same type. This provides three new opportunities:

First, an opportunity to automatically categorize a large number of models. Models with the same chain of functions would probably show similar behavior.

Second, an opportunity to automate the simplification of a complicated model. Saysel, Kerem, and Barlas (2006) proposed a workflow to simplify a complicated system dynamics model while keeping its key structures and behaviors unchanged. This workflow is to iteratively go through feedback structures, parameters, and flow equations to test if some of them could be simplified by aggregation. Dismantling model structure into functions and parameters is a key step in this process. As automatic elicitation of function chains is already realized in this work, it gives the opportunity to automate model simplification.

Third, an opportunity to automate loops analysis, because it enables models to be seen as 'chains of functions and parameters', instead of 'parameters and the model'. This new perspective allows the model to be tested loop by loop, and sensitivity analysis can also be performed loop by loop, instead of parameter by parameter.

Enhancement of model analysis and policy making

Because of the graph network-based representation of SD models, modification of model structure could be done in an automated way more easily. Previously, automated model analysis is mainly about tuning parameters. With the help of a better understanding of loop as discussed above, automated model analysis and policy generation could also make structural change by itself. Such a structural change could be building an additional flow to a stock or closing a feedback loop by adding a causal relationship. Changes that alleviate the problematic behavior can be further considered for policy suggestions.

5.3 Limitations and future research directions

Although achieved its objectives, this work is still subject to the following limitations, or at least could be improved by working on them. This last section will discuss the limitations and point out possible future directions for research.

Experiments on more complicated situations

Although designed to test a prototype, the experiments in Chapter 4 are simple. A more complicated case will at least challenge the system from the following three aspects:

First, there will be more variables and more reference modes. An increase in the scale of a model will raise the uncertainty of the target structure. As shown in Chapter 2, numerical-data based methods have difficulty when multiple time series are provided for them in the same time. The same uncertainty is also faced by the method proposed in this work, and it is still unknown how will this system behave under large amount of data. From experiment results, it is clear that structural information and a base of knowledge *a priori* would help a lot in reducing uncertainties, but to what extent can it help in complicated conditions still needs to be tested.

Second, dynamic behavior of a reference mode becomes more complicated. The experiments carried out so far use reference modes that can be easily categorized into generic behavior patterns in Barlas and Kanar (2000). However, in cases from the real world, reference modes may be influenced by exogenous inputs or synthesized with behavior of other variables. For example, a behavior going upward may be synthesized with an oscillatory curve and yield an upward oscillation. This situation may be hard to model even for an experienced modeler. Theoretically, if the system has a model in its knowledge base which could produce a similar behavior, there is still a chance for it to find a solution. However, one could neither rely on the system to have every possible behavior pre-stored. Then it is still not clear if the best strategy would be to decompose this synthesized dynamic behavior, or to ask the modeler for more structural information.

Third, some information provided for the system might be wrong or self-conflicting. Time-series data could contain noise, sometimes the noise is so distracting that the data become misleading. Structural information could also be wrong, or sometimes they are just guessed. Further improvements are therefore needed in response to this uncertainty of information reliability.

More strategies to compare situations

One feature of the modeling platform is a knowledge base consisting of previous models that could be

entirely or partly imported to generate a candidate structure. The selection of model is based on behavioral similarity. The method used for comparison is simple and can only compare time series. However, in the real world, similarity between situations is more complex and does not only have to do with numerical data, but also structural information. For example, a water sink and a bathtub are similar to each other because they are both connected with an inflow and an outflow. Such structural comparison, if made possible, would greatly help the algorithm to find a proper model to import by better identifying similarity between situations.

This point also has to do with case-based reasoning, a popular method in strategic decision making. In cased-based reasoning, a previously solved case is retrieved to inform the policy making for a new case provided the two cases are similar enough. A process of four steps is formulated as "Retrieve, Reuse, Revise, and Retain" (Aamodt and Plaza, 1994), in which 'Retrieve' is based on relevance between cases. A study of case-based reasoning may be helpful to improve situation comparison in this research.

More concepts to be understood by computer

Concepts can help people remember, categorize, communicate, and create new concepts. Understanding of a concept has many levels: to have seen it is higher than to know its existence, and to have used it is higher than to have seen it. One may have seen a harmer but never used it, but once he/she has used a harmer, he/she will definitely have a better understanding of the concept 'harmer'.

In system dynamics modeling, concepts such as 'stock', 'flow', 'function', and 'feedback loop' help modelers to read and build models. Once thinking of a concept 'negative feedback loop', a modeler could see beyond the concept itself and recall things such as requirements for a feedback loop to be negative, typical negative feedback loops, behavior of a negative feedback loop, and that a negative feedback loop could be used to balance a positive feedback loop. Those that come to a modeler's mind will altogether form his/her understanding of the concept 'negative feedback loop'. In contrast, if a model that contains a negative feedback loop is stored in a computer, the computer will neither have the above thinking nor understand the concept of that loop, even though it physically stores it.

"In 1968, the mathematician and philosopher Gian-Carlo Rota wrote, 'I wonder whether or when artificial intelligence will ever crash the barrier of meaning.' Here, the phrase 'barrier of meaning' refers to a belief about humans versus machines: humans are able to 'actually understand' the situations they encounter, whereas AI systems (at least current ones) do not possess such understanding." (Santa Fe Institute, 2018) A model stored on hard disk has a barrier of meaning for a computer, but the barrier would supposedly become lower if the computer knows how to use the model.

In this work, the proposed the algorithm is able to recognize a negative feedback loop by eliciting a chain of functions from it, and to use this chain of functions to create a new feedback loop which would generate a desired behavior. More importantly, these actions are guided by a purpose of getting better performance (measured by candidate structure's likelihood) rather than stochastically – even a strong cat can wield a harmer but only a human can use it as a tool, because human has a purpose. In this case, the 'barrier of meaning' of the concept of negative feedback loop is lowered for the computer because it can use the concept to achieve a goal, though to a very limited extent. It is interesting to imagine how much better a computer will become at modeling if more concepts in system dynamics are operationalized and understood by it. And this is probably another direction to go down.

References

1. Aamodt, A., & Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI communications, 7(1), 39-59.
2. Abdelbari, H., Elsawah, S., & Shafi, K. (2015). Model learning using genetic programming under full and partial system information conditions. In Proceedings of the 2015 International System Dynamics Conference. Cambridge, Massachusetts, USA, System Dynamics Society.
3. Abdelbari, H., & Shafi, K. (2016, July). Optimising a constrained echo state network using evolutionary algorithms for learning mental models of complex dynamical systems. In 2016 International Joint Conference on Neural Networks (IJCNN) (pp. 4735-4742). IEEE.
4. Abdelbari, H., & Shafi, K. (2017). A computational Intelligence-based Method to "Learn" Causal Loop Diagram-like Structures from Observed Data. System Dynamics Review, 33(1), 3–33.
5. Abdelbari, H., & Shafi, K. (2018). Learning structures of conceptual models from observed dynamics using evolutionary echo state networks. Journal of Artificial Intelligence and Soft Computing Research, 8(2), 133-154.
6. Barlas, Y. (1996). Formal aspects of model validity and validation in system dynamics. System Dynamics Review: The Journal of the System Dynamics Society, 12(3), 183-210.
7. Barlas, Y., & Kanar, K. (2000, August). Structure-oriented behavior tests in model validation. In 18th international conference of the system dynamics society, Bergen, Norway (pp. 33-34).
8. Berndt, D. J., & Clifford, J. (1994, July). Using dynamic time warping to find patterns in time series. In KDD workshop (Vol. 10, No. 16, pp. 359-370).
9. Chen, Y. T., & Jeng, B. (2002, July). Yet another representation for system dynamics models, and its advantages. In Proceeding of the 20th International Conference of the System Dynamics Society.
10. Chen, Y. T., & Jeng, B. (2004). Policy design by fitting desired behavior pattern for system dynamic models. In Proceedings of the 2004 International System Dynamics Conference.
11. Chen, Y. T., Tu, Y. M., & Jeng, B. (2011). A machine learning approach to policy optimization in system dynamics models. Systems Research and Behavioral Science, 28(4), 369-390.
12. Drobek, M., Gilani, W., & Soban, D. (2014). A data driven and tool supported CLD creation approach. In The 32nd International Conference of the System Dynamics Society, Delft (pp. 1-20).
13. Drobek, M., Gilani, W., Molka, T., & Soban, D. (2015). Automated Equation Formulation for Causal Loop Diagrams. In Business Information Systems (Vol. 208, pp. 38-49). Cham: Springer, Cham.
14. Forrester, J. W. (1980). Information sources for modeling the national economy. Journal of the American Statistical Association, 75(371), 555-566.
15. Forrester, J.W. (1980). System dynamics – future opportunities. In: Legasto A.A., Forrester J.W. and Lyneis J.M. (Eds.), System Dynamics. TIMS Studies in the Management Sciences Vol. 14. North-Holland: Oxford, pp 7-21.

16. Forrester, J. W. (1994). System dynamics, systems thinking, and soft OR. System dynamics review, 10(2-3), 245-256.
17. Hines, J. H., (1996). Molecules of Structure. System Dynamics Group, Sloan School of Management, MIT.
18. Hofstadter, D. R., & Mitchell, M. (1994). The Copycat project: A model of mental fluidity and analogy-making.
19. Houghton, J., & Siegel, M. (2015). Advanced data analytics for system dynamics models using PySD. revolution, 3(4).
20. Jeng, B., Chen, J. X., & Liang, T. P. (2006). Applying data mining to learn system dynamics in a biological model. Expert Systems with Applications, 30(1), 50-58.
21. Lane, D. C. (1998). Can we have confidence in generic structures?. Journal of the Operational Research Society, 49(9), 936-947.
22. Moxnes, E. (2017). Teachings in lectures.
23. Paich, M. (1985). Generic structures. System Dynamics Review, 1(1), 126-132.
24. Richardson, G. P. (1991). System dynamics: Simulation for policy analysis from a feedback perspective. In Qualitative simulation modeling and analysis (pp. 144-169). Springer, New York, NY.
25. Richardson, G. P., & Pugh, A. L. (1981). Introduction to system dynamics modeling with DYNAMO (Vol. 48). Cambridge, MA: MIT press.
26. Santa Fe Institute. (2018, October 9). Artificial Intelligence and the “Barrier” of Meaning. Retrieved from https://www.santafe.edu/events/artificial-intelligence-and-barrier-meaning
27. Saysel, A. K., & Barlas, Y. (2006). Model simplification and validation with indirect structure validity tests. System Dynamics Review, 22(3), 241-262.
28. Sterman, J. (2000). Business dynamics: systems thinking and modeling for a complex world (No. HD30.2 S7835 2000).
29. Wagon, S., & Portmann, R. (2005). How quickly does water cool. Mathematica in Education and Research, 10(3).
30. Young, T., Hazarika, D., Poria, S., & Cambria, E. (2018). Recent trends in deep learning based natural language processing. IEEE Computational intelligence magazine, 13(3), 55-75.
31. Yücel, G., & Barlas, Y. (2007). Pattern-based system design/optimization. In Proceedings of the 25th international conference of the system dynamics society, Boston.
32. Yücel, G., & Barlas, Y. (2011). Automated parameter specification in dynamic feedback models based on behavior pattern features. System Dynamics Review, 27(2), 195–215.
33. Yücel, G., & Barlas, Y. (2015). Pattern recognition for model testing, calibration, and behavior analysis. Analytical methods for dynamic modelers, 173-206.

Appendix 1

The software prototype developed for this project is an interactive modeling platform.

A version pre-released for this submission could be found at:
https://github.com/Rutherford1895/Stock-and-Flow-in-Python/releases/tag/0.9

The latest repository of source code could be found at:
https://github.com/Rutherford1895/Stock-and-Flow-in-Python