Verifying Influence Diagrams using Dimensional Analysis

G W Komanapalli
London South Bank University
103 Borough Road
London – SE1 0AA
Phone: ++4420 7815 8239
Email: komanaga@lsbu.ac.uk

Abstract

Developing a valid model is of primary importance. Various verification and validation procedures are used to establish confidence in the model output. To establish confidence that a model produces right behaviour for right reasons it is essential to ensure that the structure of model represents real world system. Amongst the verification procedures employed, dimensional analysis is used to verify the syntactical correctness of the equation. Despite of its significance, dimensional analysis is one of the less prioritised procedures used during model building process. Therefore lack of dimensional consistency raises serious doubts about the validity of model behaviour.

The aim of this paper is to summarise various problems and difficulties identified in System Dynamics (SD) modelling process and to suggest an alternative approach. Firstly, this paper discusses various problems faced by beginners, which can lead to errors in SD models. Secondly, discusses alternative approaches suggested by researchers in SD. Thirdly this papers presents an approach to generate mathematical model from Influence Diagrams. There are two principle benefits this approach can offer: A software tool based on this approach, and improve SD modelling experience, especially of those modellers who have limited mathematical experience so as to gain benefits of quantitative SD modelling.

Key Words: System Dynamics, Model Building process, Verification and Validation, Dimensional Analysis, Influence Diagrams.

1. Introduction

The philosophy and theory behind SD modelling is intellectually inspiring and draws people from various backgrounds who do not have prior experience in simulation modelling. As the modelling process advances modellers are likely to encounter difficulties in translating a qualitative model into a mathematical model of the system (Homer, 2007). While popular debates on SD methodology surrounds conceptualisation, model verification and validation procedures, beginners often have difficulties with moving from a qualitative formulation to mathematical implementation of the model. Due to a lack of expertise and experience, modellers tend to make mistakes that can go unnoticed and later produce erroneous behaviour.

Ballico-Lay and Coyle (1984) discuss a variety of such issues that lead to hidden errors in models. Simple errors like the omission of dimensions for physical quantities (variables) in the model can lead to incorrect behaviour of the model. In fact if variable dimensions are defined consistently this should lead to a verifiable representation of the system. However, a range of software tools are currently used to develop SD models and some allow the modeller to bypass dimensional analysis completely. This can lead to an incoherent representation of the system if dimensional errors exist which would result in the development of inappropriate policies for the system in study. A valid SD model gives behaviour which will match with the real world behaviour of the system (Barlas, 1989 & 1994), and for the behaviour of the model to be valid, the structure of the model should be verified. The broad context of this paper is to address the need for an approach which automatically generates a structurally verified quantitative SD model and provide support for the modeller who does not have mathematical expertise and experience in SD modelling.

This paper presents a novel approach to developing SD models that would effectively guide the modeller and improve the modelling process for those who do not have the required mathematical experience to develop a quantitative model. This will be accomplished by a fundamental revision of the sequence of events that occur in current practice and by automating the creation of the required equations directly from the influence diagrams and from the rigorous definitions of the variables. This is a significant new development in SD methodology and has the potential to provide a better modelling experience, and to save considerable time and cost in model verification, reworking and fault finding.

2. Critical review of relevant literature

This section highlights some of the problems in SD modelling practice. SD practitioners agree that a completely correct model has never been produced. It is more important to develop a valid model that justifies the purpose of the model (Sterman, 2000). Therefore the following sections present discussion on the verification and validation procedures that are used in the SD modelling process to highlight the weaknesses in carrying out these procedures.

2.1Confidence in the Model

SD community statistics according to Scholl (1993 & 1995) show that less than 30% of SD modellers have a general understanding of mathematical methods. From this it can be inferred that a majority of the modellers prefer to use SD as a tool for developing qualitative models using system thinking concepts. There are justifiable reasons why quantitative models are not developed, for example Coyle (2000b) and Wolstenholme (1998) state that in some scenarios a qualitative model is sufficient to address the objective of modelling project.

However, SD practitioners constantly emphasise that the strength of SD lies in its simulation experiments (Homer et al, 2001). Therefore in scenarios involving the need to quantify a model, modellers need to use mathematical methods. This is the fundamental shift in SD modelling that requires substantial help for those who do not have training or experience in mathematics. Lack of support at this stage can mean that modellers gain limited benefits from SD modelling. There is a need to support these modellers by enhancing the modelling process with an intelligent tool which will guide the modeller towards a quantitative model. A powerful software tool might not increase knowledge of SD, but it will improve the SD modelling experience for those who have a non-mathematical or non-engineering background. An important issue in modelling a system is to build a valid model which implements a correct structure and gives correct behaviour. It is very critical to ensure that such an intelligent tool provides support in producing a valid model.

Forrester and Senge (1980) relate confidence in the model to the purpose of the model. Many abstractions of a system can be developed and each serves a different purpose (Sterman, 2000). An output graph of selected variables is no guarantee of validity, as an endless variety of invalid components (equation forms) can exist to give the same apparent system behaviour. Forrester, (1961) and Coyle (1977,1996& 2000a) place emphasis on the model equations being dimensionally consistent and that all the constants in a model must be clearly defined and their dimensions must be stated. When the dimensional consistency is not maintained, the behaviour of the model is, therefore, incorrect and conclusions drawn from its outputs may be misleading. Thus, policy advice based on the model may be highly erroneous. Barlas(1984) reinforces this by saying that an equation in the model represents a part of the structure of the system and therefore a set of dimensionally consistent equations can ensure a structurally verified model. Parameter verification tests also contribute to ensuring that a variable represents part of the system. The literature on SD validation and verification agrees that "the defence of a model must rest on the defence of its details" (Forrester & Senge 1980). Dimensional errors are, therefore, a serious matter which led to the development of the dimensional analyser, within the COSMIC package, to detect and warn of dimensional errors (Ballico-Lay & Coyle, 1984). Some other software environments now claim to support dimensional analysis (DA) but doubts exist about the correctness of their algorithms (Coyle, 1996).

Balci (1995) defines model verification as the process of translating the model from one form to another while maintaining the original purpose of the model. Therefore when developing a quantified model, it is important to ensure that the two representations, both qualitative SD model and quantitative SD model are compatible.

This emphasises that the relationships between variables have to be dimensionally compatible. However, in current SD practice, DA is rarely applied (Scholl, 1993). This raises questions over the validity of the models developed. This paper also discusses appropriateness of using DA in ensuring that the correct models are developed.

2.2 Errors in models and correction through Dimensional Analysis

Dimensional analysis is a procedure often applied in physics, chemistry, and engineering to understand physical situations involving a mix of different kinds of quantities. In physics, dimensional analysis is used to validate the equations, and in some instances, develop equations (Barragan et al, 2005). In principle, quantities with the same dimensions can be added, subtracted or equated. An equation is said to be dimensionally consistent if the dimensions of the variable on the left side of the equation match the cumulative expressions on the right side of the equation.

Some of the errors occur due to poor modelling practice or a lack of modelling experience. Errors that are commonly found in SD range from simple typing errors to incomplete representations of the model which result in inconsistent dimensions, scaling errors, missing time constraints, incorrect use of dimensionless quantities, incorrect equation forms, incorrect use of numerical values etc.

Choudhari et al (1995) and Coyle (1996) discuss how conversion factors can contribute to dimensional problems. For example, when modelling temperature control in a room, in a conceptual description of the system, temperature of the room is normally described in terms of heat in the room. This would mislead a novice modeller to relate room temperature directly to the loss of heat or gain of heat. In mathematical terms, heat in measured in Calories while temperature in measured in Celsius or Fahrenheit. These types of inconsistencies can be trapped using DA.

Getmansky (1997) discusses how time constants are used to bring an equation to be dimensionally valid but represent an incorrect representation of a system. A variable is introduced into the equation which is not relevant in the system but is used to bring dimensional consistency to the equation. This is also called a parameter verification test. These types of mistakes can be avoided by applying DA. In this instance DA can verify whether an equation is conceptually correct or not.

Martin (2001) & Stange(1998) identify various other types of errors with dimension less quantities. In addition to these, the types of errors that could go unnoticed can be found out with DA (Coyle & Sharp, 1980). DYSMAP is one of the earliest software for developing SD models. A dimensional analyser unit in DYSMAP was effectively developed. It proved to be effective in the model validation procedure (Coyle, 1983).

Cost benefit analysis: A cost/benefit analysis of applying verification and validation procedures shows that verification procedures are of principle importance and next only to structure oriented behavioural tests and structure assessment. The study also shows that if DA is applied at the beginning of the modelling process, its benefits would have been more evident that if its applied at the end of the modelling process(Hoarfrost, Wakeland , 2005).

.

In the light of difficulties and complexities involved in SD modelling, researchers in SD modelling have proposed alternative model building process and methodologies, which are explored in the following sections.

2.3Alternative approaches

This section presents a review of two alternative approaches to SD modelling aiming to increase the versatility of the model building process and SD methodology. Firstly a matrix analysis of causal loop diagrams is presented and secondly a proposal to extend System Dynamics methodology by using OO concepts is offered.

Analysing Influence Diagrams as Directed Graphs: Influence Diagrams can be studied as graphs. Graphs contain vertices and edges that connect these vertices. Influence Diagrams consists of variables and their influences. In order to apply graph analysis principles to Influence Diagrams, the variables can be considered as vertices of a graph and the influences as directed edges of a graph. One of the methods employed to represent a directed graph is to develop an adjacency matrix of the graph. For a graph with n nodes, an adjacency matrix is of size n x n. For directed graphs the matrix element (a, b) is 1 if there is an edge from 'a' to 'b' or 0 if there is no edge going from 'a' to 'b.' This is illustrated in figure1.

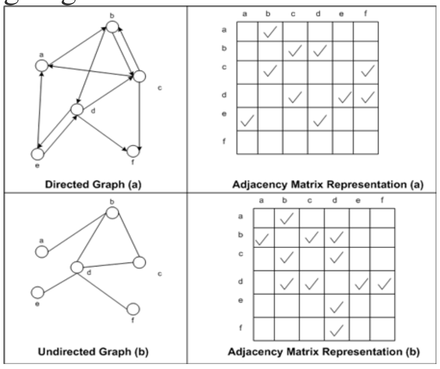

Directed Graph (a)
Adjacency Matrix Representation (a)
Undirected Graph (b)
Adjacency Matrix Representation (b)
a
b
c
d
e
f

Figure 1: Matrix analysis of a graph

Burns (2001 & 2002) presented an enhanced version of this approach by applying Set Theory principles and Matrix multiplication to structurally validate Causal Loop and Stock Flow Diagrams.

This approach used dimensional analysis to validate the derived equation. Although this is an interesting approach the use of matrix algebra is problematic both in terms of computational efficiency and in the size of matrix generated. SD models may have hundred of variables which can generate very large matrices. Therefore this approach using matrix multiplication will be efficient if applied to small models. In this paper, an alternative approach which uses these principles but that implements a different type of data structure is described, which removes the constraints on the number of variables that can be used.

Object Oriented Paradigm to SD: Myrtveit & Tignor (2000a & 200b) presents a thought-provoking discussion on adding OO features to SD methodology. This discussion proposes a conceptual leap in SD methodology and requires the modeller to use OO tools to develop models and identify the patterns of behaviour of the system under study. The unique advantage of using this approach is that models can be built by joining building blocks. It primarily requires the user to select the patterns of the dynamic behaviour from the existing software library and link the building blocks appropriately. This requires new measures for validating the links and verifying that correct behaviour is produced through joining these various building blocks of the system.

However, in building these generic behavioural structures of the system, the modeller needs to think of the components of the system as objects and links. The primary focus is on representing the structure of the system correctly. Structures and behaviour are interdependent. Therefore, if structure is developed correctly then correct behaviour can be obtained. When a modeller develops a model the focus is shifted from "movement" or flow in the system to the "appearance" or state of the system. There are benefits in this approach in that it starts from a high level conceptual representation of the system and grows to add the smallest detail of the model. Therefore it provides a rather less complicated process to develop the model, although a modeller may find it difficult to adapt to the OO "programming" concepts. However, the principle is to be able to obtain the patterns of the behaviour of various systems and guide the modeller to join the building blocks to build a full model.

Thus this approach can be adopted to enhance and extend SD modelling methodology. If the OO approach is implemented in a rather ingenious way in the software tool, it can be used to generate and help verify an SD model and thereby enhance the SD modelling process.

This paper therefore aims to present a revised approach to the SD modelling process that adapts OO concepts at the implementation level while allowing the modeller to formulate an SD model based on system thinking tools and qualitative models.

3. Analysing Influence Diagrams

One of the principle objectives of this paper is to analyse ID. Analysing an ID involves walking through the diagram and recognising the connections between variables. In order to implement this process as a programming module, an algorithm is initially developed providing a guideline to analyse ID.

Analysing an ID involves walking through the diagram and recognising the connections between variables. In order to implement this process as a programming module, an algorithm is initially developed providing a guideline to analyse ID. Some of the concepts that are used in designing the following algorithm to analyse ID are discussed in this section. First, ID is compared with Tree structures and thereby terminology of trees is applied to ID. Secondly tree traversal techniques are defined. Thirdly a brief review on the notation used to develop ID as an SD model is presented. This forms foundations to the design of the algorithm to traverse ID.

3.1Comparing Trees and Influence Diagrams

Traversing is a process by which every node or edge in the network is traced for a purpose. Traversal is commonly applied to graphs and trees. Graphs are diagrammatic notations used to represent a network while trees are data structures used in computing to store data in a meaningful way (Carrano, 2007). Graphs needs to be stored using an appropriate data structure. Feedback loops are the candidate features of ID, which are not present in trees. Therefore tree traversal concepts are thus applied to individual variables in ID rather than the whole of ID. Therefore traversing the tree of a variable would enable to find an equation for that variable and DA can be applied to determine the validity of the equation form. This is the fundamental idea behind applying tree traversal techniques.

While the ID cannot be compared to the tree due to the absence of loops in the later, the following table contrasts trees with IDs

Trees	**Influence Diagrams**
1. Root node is at the "top" of a tree – with 0, 1 or 2 children in a binary tree. 2. Child nodes are the branches of trees 3. Trees have arrows outbound arrows 4. In a tree, child nodes branch out from the parent node 5. In trees, arrows are unsigned and non-directional 6. Loops do not exist	1. Root node is a variable being influenced by other variables 2. Child nodes are the influencing variables that are connected to the variable of interest 3. A variable in an influence diagram has inbound arrows 4. Child nodes flow into the parent node 5. Arrows are signed and carry information about the child and the parent nodes 6. Loops are present

Table 1: Comparison of Trees with Influence Diagrams

Traversals of a Binary tree: There are different types of trees such as general tree and binary tree. In this current algorithm design binary tree traversals are considered. If the root node in the tree has less than or just two child nodes then the tree is a binary tree. Moving along the tree involves following a prescribed movement. This is formally called as tree traversal. For example to move along the binary tree, the traversal starts at root node and visits all the nodes on the left sub-tree and visits all the nodes on the right sub-tree. Three types of tree traversals used to move along the tree: Pre-Order Traversal, In-order traversal, Post Order Traversal.

Pre-Order Traversal: In this type of traversal, root node is first visited and branches out to visit left sub-tree and then completes with visiting right sub-tree.

In-order Traversal: Visits all the nodes on the left sub-tree, then visits root node and then finishes with visiting right sub-tree.

Post-Order Traversal: Left sub-tree is visited first in this traversal. Following this, all the nodes on the right sub-tree and finally visits the root node.

There are specific uses of these individual traversals. A much broader discussion on the traversals is beyond the scope of this paper. They are mentioned briefly here so as to state meaningfully that in-order traversal is chosen to use in the algorithm.

Notation used for drawing Influence Diagrams: When developing an ID it is necessary to follow the framework proposed by Coyle (1996). IDs use two types of arrows: solid lines and dotted lines. Solid lines represent physical flows while dotted lines represent information flows. IDs use three types of variables: level, rates or auxiliary and parameters or constants. A node with no inbound arrows is a parameter. If node has incoming arrows then the node represents a level/ rate or auxiliary variable.

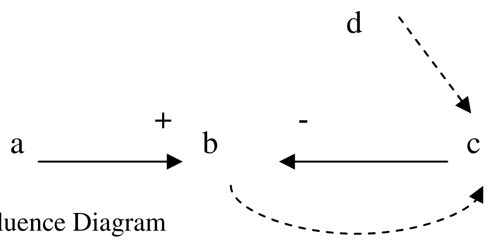


Figure 2: Sample Influence Diagram

In the above diagram, variable ‘b’ has two incoming arrows. Since they are solid lines they represent physical flows. The arrow from starting from ‘c’ has a negative sign and the arrow from ‘a’ has a positive sign. An important point to note is here, is to define the type of variable. Level variable has rates as input and outputs. The difference of these input and output rates determine the value of the level variable, where the variable is purely a physical quantity. There is a second type of level variable that is a computed or an average of rates, and represents information accumulation in the system.

In SD for each of these types of variables has a standard form of equation. Therefore, tree templates for these types of variables are designed. These templates can be used in context of teaching or applying the algorithm to an ID.

3.2Design of Algorithm

A number of models (influence diagrams) from (Coyle, 1996) are analysed and the following inferences are made.

Polarity of the incoming arrow signifies the direction of its influence on the other variable it is connected to.

In addition to DA, polarities of incoming variables help determine the type of algebraic operator to be used in the expression.

- If the polarities of the incoming variables are the same then the equation will consists of either addition or multiplication of these variables.
- If the polarities are opposite then the equation will consist of division or subtraction of the variables depending on the dimensions of the variables.

Steps involved in the “Tree Analysis of ID Algorithm” - TAIDA

Step 1: Consider each variable as a parent node.
Step 2: Identify Child nodes which are linked to the incoming arrows

Step 3: Save the variable as Tree using the templates provided depending on the type of the variable.
Step 4: Carryout In-Order tree traversal to generate the equation
Step 5: Apply DA to validate the equation

3.3 Applying the Algorithm to Influence Diagram

In order to apply the algorithm described below it is important to define parent node and child nodes of a variable. ID has three important types of variables: Level, Rate or Auxiliary and Parameters. As the diagram is drawn, modeller is required to specify, name, units and type of variable.

This algorithm is applied to the following diagram for illustration purposes. This ID represents a qualitative SD model taken from Coyle (1996).

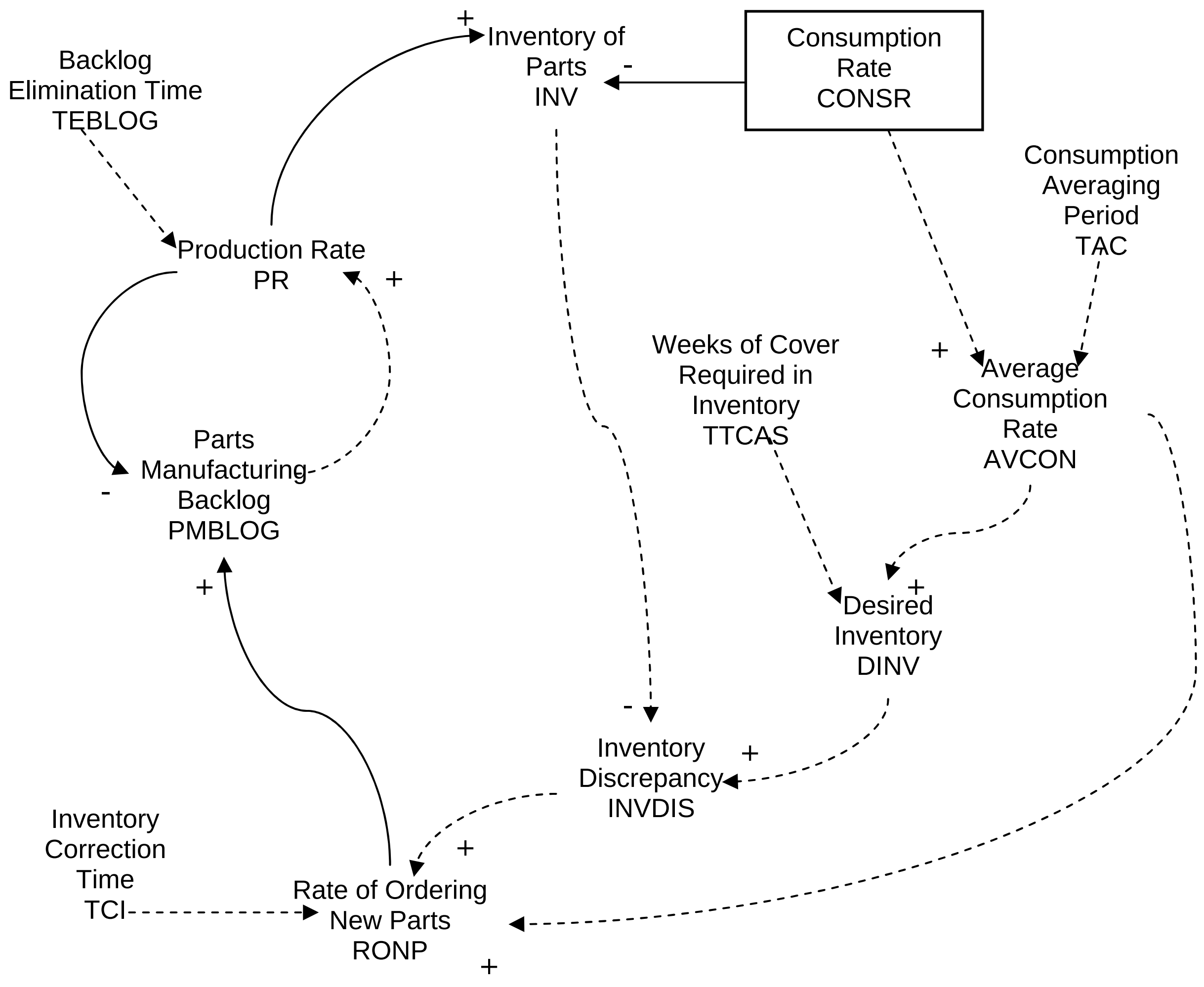

Inventory of Parts INV
Consumption Rate CONSR
Backlog Elimination Time TEBLOG
Consumption Averaging Period TAC
Production Rate PR
Weeks of Cover Required in Inventory TTCAS
Average Consumption Rate AVCON
Parts Manufacturing Backlog PMBLOG
Desired Inventory DINV
Inventory Discrepancy INVDIS
Inventory Correction Time TCI
Rate of Ordering New Parts RONP
+
-

Figure 3: Model taken from (Coyle,1996)

Before applying the algorithm it is important to define the dimensions for each variable. Table 2 outlines the definitions of variables.

Variable	Type	Dimensions
Average Consumption Rate (AVCON)	Level	[Parts]/[Time]
Consumption Rate: (CONSR)	Rate	[Parts]/[Time]
Desired Inventory: (DINV)	Auxiliary	[Parts]
Inventory of Parts: (INV)	Level	[Parts]
Inventory Discrepancy: (INVDIS)	Auxiliary	[Parts]
Production Rate: (PR)	Rate	[Parts]/[Time]
Parts Manufacturing Backlog: (PMBLOG)	Level	[Parts]
Rate of Ordering New Parts: (RONP)	Rate	[Parts]/[Time]
Consumption Averaging Period: (TAC)	Parameter	[Time]
Inventory Correction Time: (TCI)	Parameter	[Time]
Backlog Elimination Time [TEBLOG]	Parameter	[Time]
Weeks of Cover Required in Inventory [TTCAS]	Parameter	[Time]

Table 2: Variable Definitions

The following is an illustration for one Variable RNOP. A segment of the ID is shown separately (in Figure 6) to show the how the algorithm is applied.

Step1: RNOP is rate variable
Step2: Child nodes for RNOP are linked to the incoming arrows which are, AVCON, INVDIS and TCI
Step 3: Rate Tree Template is used to produce tree for RNOP (Figure 5)
Step 4: By applying In-order traversal an equation can be obtained.

In-order traversal is carried by visiting left sub-tree, root node, and right sub-tree. This can be applied to the variable tree developed for RNOP. This tree is shown in the "Diagram 3". When In-order traversal is applied the following equation is obtained:

RNOP = AVCON (+, -, *,/) INVDIS (+, -, *,/) TCI

Step 5: DA is applied to decide which arithmetic operator occupies parent node at level 1 and 2 of the tree.

When DA is applied to the above tree, the following equation would be derived.
RNOP = AVCON + INVDIS /TCI

Above sequence of steps can be applied until all the level, rate and auxiliary variables are covered.

Figure 4: Standard Tree Templates for Rate Equations

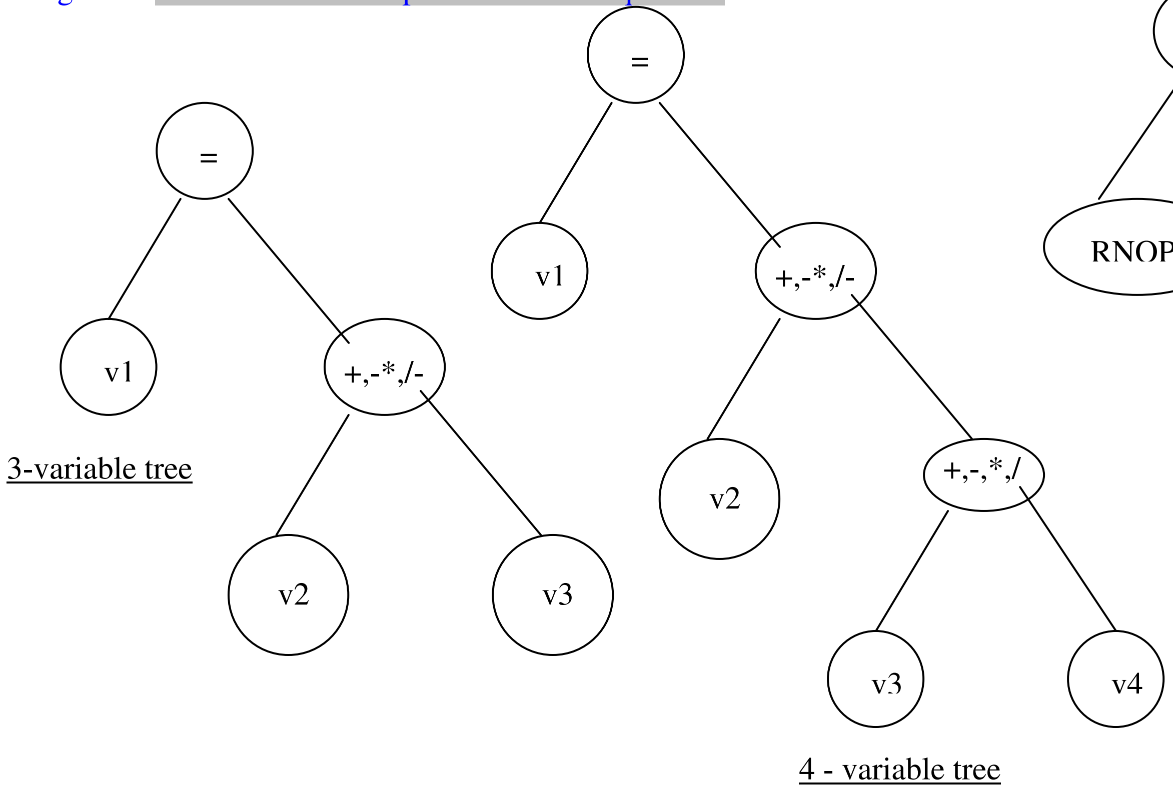


Figure 5:
Rate Equation Tree for RNOP

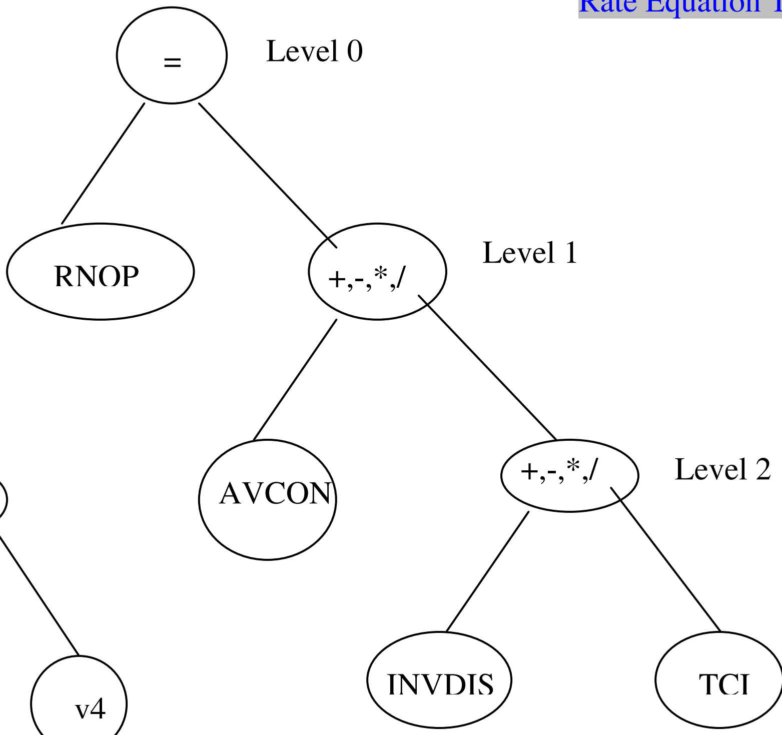
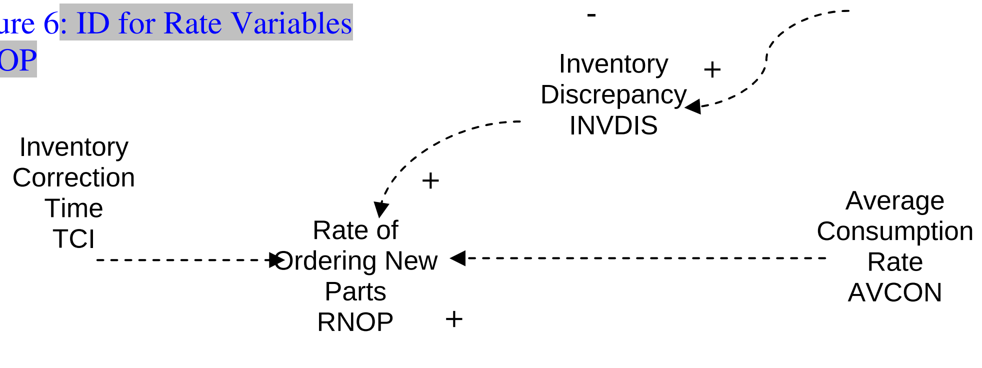


Figure 6: ID for Rate Variables RNOP



-
Inventory Discrepancy INVDIS
+
Inventory Correction Time TCI
+
Rate of Ordering New Parts RNOP
+
Average Consumption Rate AVCON

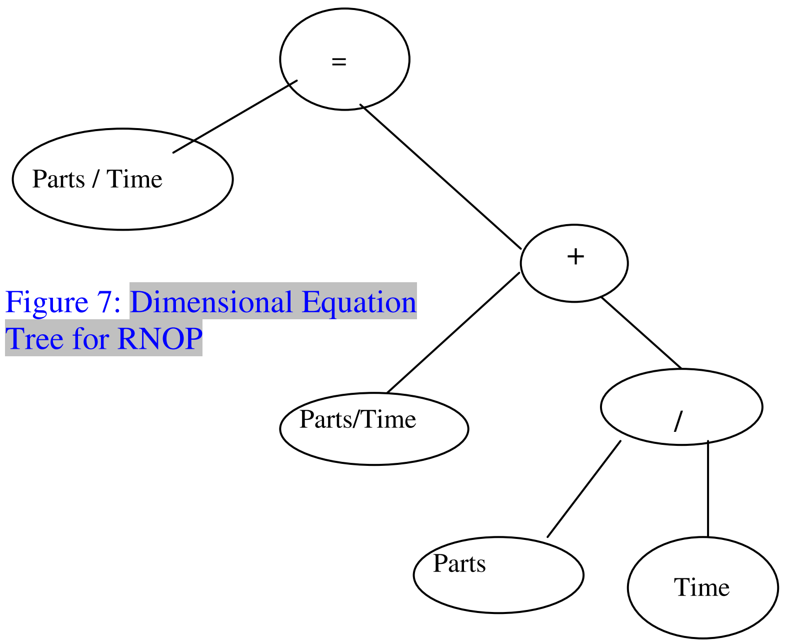


Figure 7: Dimensional Equation Tree for RNOP

Complete Set of equations obtained by applying the TAIDA

$$INV = INV + (PR - CONSR) * DT$$
$$PR = PMBLOG / TEBLOG$$
$$PMBLOG = PMBLOG + (RNOP - PR) * DT$$
$$RNOP = INVDIS / TCI + AVCON$$
$$INVDIS = DINV - INV$$
$$DINV = AVCON * TTCAS$$
$$AVCON = AVCON + (CONSR - RONP) * DT / TAC$$

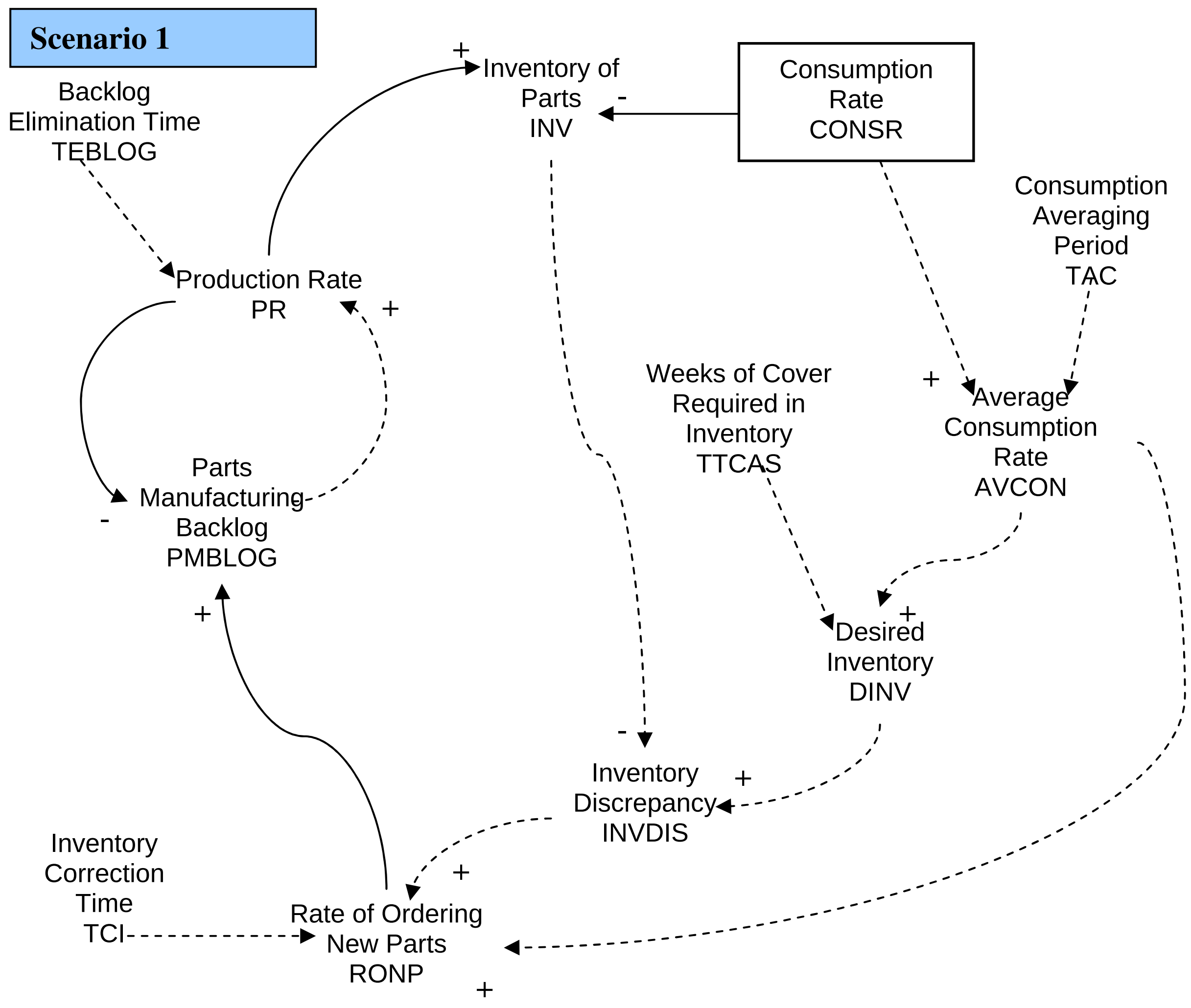
4 Analysis of Algorithm and Insights

The algorithm is applied to the above diagram with three scenarios as stated below.

Scenario 1: All variables and influences defined correctly.

Scenario 2: Incorrect types of influences and correct dimensions for the variable INVDIS to RNOP;

Scenario 3: Missing variable for RNOP.


Scenario 1
Backlog Elimination Time TEBLOG
Production Rate PR
Inventory of Parts INV
Consumption Rate CONSR
Consumption Averaging Period TAC
Weeks of Cover Required in Inventory TTCAS
Average Consumption Rate AVCON
Parts Manufacturing Backlog PMBLOG
Desired Inventory DINV
Inventory Discrepancy INVDIS
Inventory Correction Time TCI
Rate of Ordering New Parts RONP
+
-
+
+
-
+
+
+
-
+
+
+
+

Complete Set of equations obtained for Scenario 1 by applying the TAIDA

$INV = INV + (PR - CONSR) * DT$

$PR = PMBLOG / TEBLOG$

$PMBLOG = PMBLOG + (RNOP - PR) * DT$

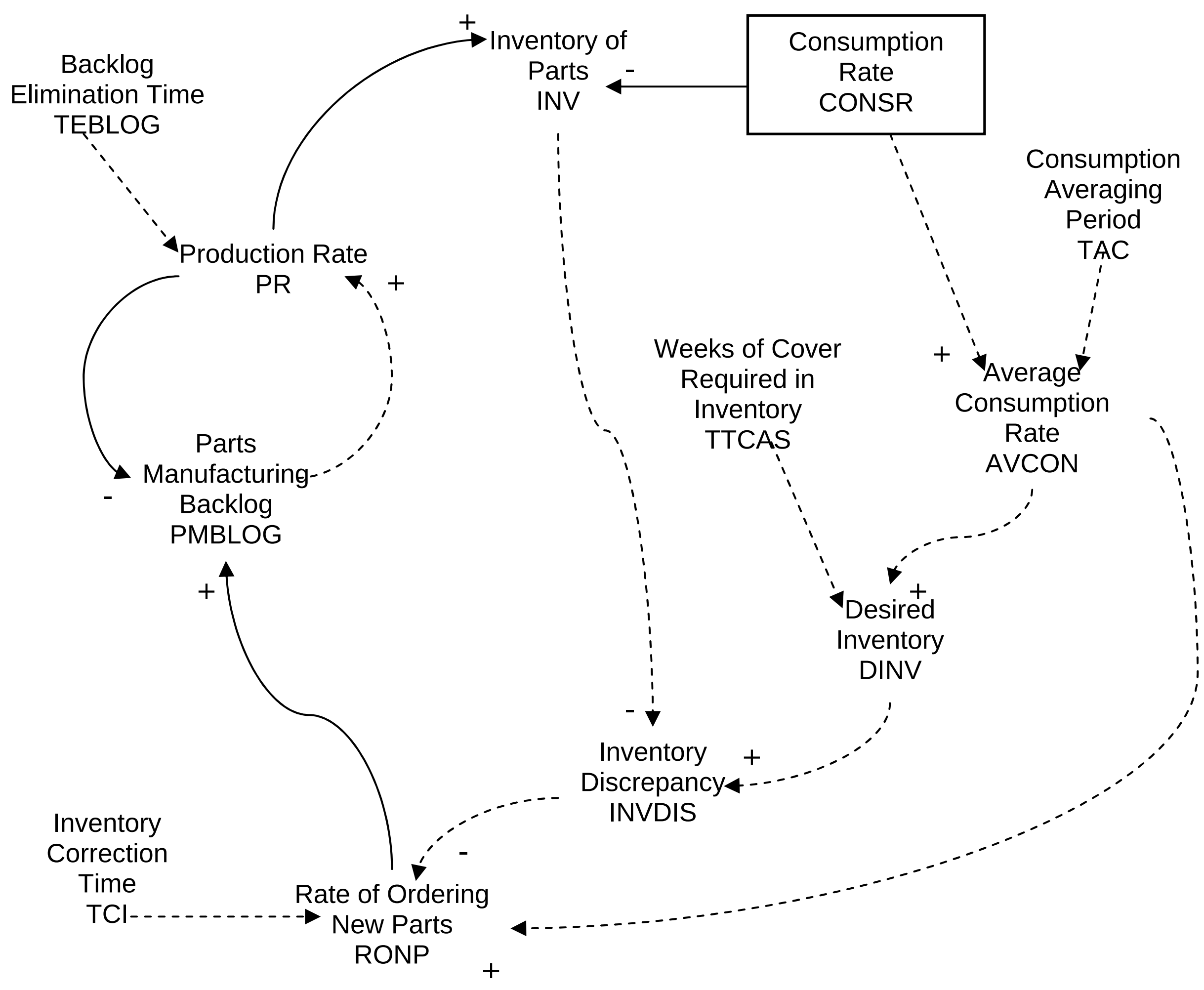
$RNOP = INVDIS / TCI + AVCON$

$INVDIS = DINV - INV$

$DINV = AVCON * TTCAS$

$AVCON = AVCON + (CONSR - RONP) * DT / TAC$

Scenario 2


Backlog Elimination Time TEBLOG
Inventory of Parts INV
Consumption Rate CONSR
Consumption Averaging Period TAC
Production Rate PR
Weeks of Cover Required in Inventory TTCAS
Average Consumption Rate AVCON
Parts Manufacturing Backlog PMBLOG
Desired Inventory DINV
Inventory Discrepancy INVDIS
Inventory Correction Time TCI
Rate of Ordering New Parts RONP
+
-

Complete Set of equations obtained for Scenario 2 by applying the TAIDA

$INV = INV + (PR - CONSR) * DT$

$PR = PMBLOG / TEBLOG$

$PMBLOG = PMBLOG + (RNOP - PR) * DT$

$RNOP = AVCON - INVDIS / TCI$

$INVDIS = DINV - INV$

$DINV = AVCON * TTCAS$

$AVCON = AVCON + (CONSR - RONP) * DT / TAC$

Insights obtained Scenario 2: If both the polarities are positive then the arithmetic operator on the Right Hand Side(RHS) of equation would be either '+' or '*'. On the other hand if the polarities of both the incoming arrows are negative, then the arithmetic operator of RHS of equation will have either "-"or "/". This principle works for the rate variables: PR and RNOP.

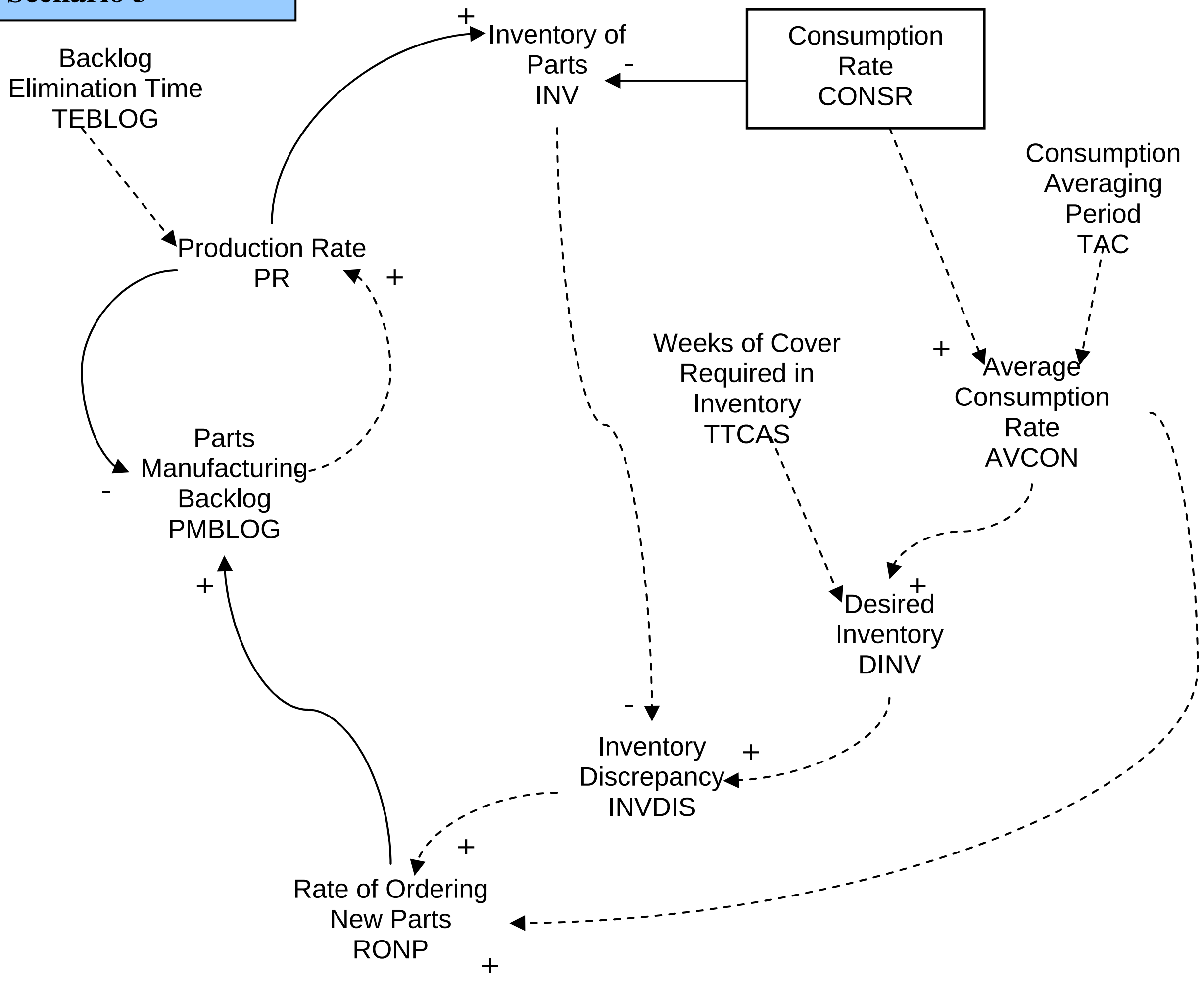
Consider RNOP rate variable. It has three incoming arrows from TCI, INVDIS, and AVCON. Therefore, RNOP is the root node and it has three sub nodes. Now by applying DA and using polarities of the variables the equation can be formulated as follows.

RNOP = (INVDIS/TCI) + AVCON

However, if the user the makes a mistake and indicates that AVCON variable has a negative influence on RNOP then even though DA gives a positive results, the equation takes a different form as follows

RNOP = (INVDIS / TCI) – AVCON

Scenario 3


Backlog Elimination Time TEBLOG
Inventory of Parts INV
Consumption Rate CONSR
Consumption Averaging Period TAC
Production Rate PR
Weeks of Cover Required in Inventory TTCAS
Average Consumption Rate AVCON
Parts Manufacturing Backlog PMBLOG
Desired Inventory DINV
Inventory Discrepancy INVDIS
Rate of Ordering New Parts RONP
+
-
+
-
+
+
+
-
+
+
+

Complete Set of equations obtained for scenario 3 by applying TAIDA

$$INV = INV + (PR - CONSR) * DT$$
$$PR = PMBLOG / TEBLOG$$
$$PMBLOG = PMBLOG + (RNOP - PR) * DT$$
$$RNOP = error$$
$$INVDIS = DINV - INV$$
$$DINV = AVCON * TTCAS$$
$$AVCON = AVCON + (CONSR - RONP) * DT / TAC$$

Insights obtained Scenario 3: This scenario demonstrates the need to develop a software tool that works according to the algorithm and supplies possible forms of valid equations. The algorithm is applied to the diagram free hand and therefore, the researcher has not entered equation for RNOP. In ideal situation, DA part of the algorithm would suggest a valid variable type based on the dimensions.

Limitations of the algorithm: This scenario 2 identifies that dimensional analysis cannot help or prevent the user from using incorrect use of influences on the variables. Dimensional analysis can help formulate equations however in order to ensure complete accuracy there must be another verification procedure to make sure that the type of influences are correctly identified.

5 Conclusions and Recommendations

In SD modelling, conceptualising is generally the most difficult phase of the process. Complexity involved at this stage of modelling is purely psychological in that the modeller and the people involved in the model building process have to define the boundaries of the system, identify leverage points, and identify dynamic feedback structures. This generates complex behaviour that can be difficult to comprehend giving rise to psychological complexity (Kalokota et al, 1993). In order to achieve success in this area, multi-paradigm methodologies are proposed (see Sec 2.3) such as adding Object Oriented features to SD modelling methodology. This study provides guidelines for conceptualising a system and developing OO modules for SD models, thereby contributing to the reduction of psychological complexity involved in modelling process.

The paper presents a critical summary of verification and validation procedures used in SD. This summary identifies how DA is sidelined in the normal practice of SD model building. As mentioned in the previous sections, model behaviour based on incorrect forms of equations give rise to incorrect policy suggestions. This paper reinforces the importance of using DA and shows how DA can generate correct forms of equations and help develop a structurally valid model.

Software implementation of the algorithm illustrates how this approach has by-passed some of the fundamental stages (i.e. writing equations) in the current model building process, representing a breakthrough in SD modelling practice. Future enhancements to the software can help engender error proof modelling.

So far there is no algorithm available to derive equations from diagrams. Significance of the approach discussed comes from the analysis of influence diagrams which have more precision than causal loop diagrams. The scope of this paper is limited in that the algorithm is not applied to every other form of diagram that is used in SD modelling. However the author believes the same generic algorithm can be extended to various other forms of diagrams such as Stock Flow Diagrams. Therefore, the originality of this approach lies in the algorithm developed to derive equations for variables in the influence diagrams. The anticipated change in modelling process will be evident from the software tool which implements this algorithm to generate equations from ID.

References

Balci O., 1995. Principles and techniques of simulation validation, verification, and testing. Proceedings of the 27th conference on Winter simulation, p147 – 154.

Ballico_Lay B and Coyle R G., 1984. Concepts and software for dimensional analysis in modelling. IEEE Transactions on Systems, Man and Cybernetics 14(3).

Barlas Y., 1989. Multiple Tests for Validation of System Dynamics Type of Simulation Models. European Journal of Operations Research 42(1), pp. 59-87.

Barlas, Y., 1996. Formal Aspects of model validity and validation in system dynamics. System Dynamics Review 12 (3), 183–210.

Barragan J, Puig-Bargues J., Ramirez de Cartagena F., 2005. Development of Equations for calculating the Head Loss in effluent Filtration in Micro irrigation Systems using Dimensional Analysis, Biosystems Engineering 92(3), 383-390.

Bell G., Warwick J., 2006. Towards establishing the use of holons as an enquiry method. International Transactions in Operational Research 14(1), p 55 – 73.

Burns J.R., 2001. Structural Validation of Causal Loop Diagrams. Proceedings of the System Dynamics Society 19th Annual Conference.

Burns J.R., Ulgen.,2002. A Matrix Architecture for Development of System Dynamics Models. Proceedings of the System Dynamics Society 20th Conference.

Buckingham E., 1914. On physical systems. Illustration of the use of dimensional equations. Physics Review, 4(4).

Choudhari M., Gary M S., Glick M., Oh A., 1995. Mistakes and Misunderstandings: Examining Dimensional Inconsistency, D-4452-1

Carrano F M., 2007. Data Structures and Abstractions with Java. Pearson Prentice Education.

Coyle R.G., 1977. Management System Dynamics. John Wiley & Sons.

Coyle R.G., Sharp J.A., 1980. System Dynamics Problems. University of Bradford Printer.

Coyle R.G., 1983. The technical elements of the system dynamics approach. European Journal of Operational Research Vol. 14,

Coyle R G., 1996. System Dynamics Modelling: A Practical Approach. CRC Press.

Coyle R.G., 1998., The practice of system dynamics: milestones, lessons and ideas from 30 years experience. System Dynamics Review 14(4).

Coyle R.G., Exelby D.,2000a. The validation of commercial system dynamics models. System Dynamics Review 16(1).

Coyle, R. G., 2000b. Qualitative and quantitative modelling in system dynamics: some research questions. System Dynamics Review 16(3).

Easterbrook S., Damian D., Singer J., Storey M., 2008. Guide to Advanced Empirical Software Engineering. Springer

Forrester J.W., 1961. Industrial Dynamics. Cambridge, MA: MIT Press.

Getmansky M., 1997. Mistakes and Misunderstandings: Time Constants and Decay Fractions. D-4679

Hoarfrost M., Wakeland W., 2005. The case for thoroughly testing complex system dynamics models. The 23rd International Conference of The System Dynamics Society, Boston, USA.

Homer J., Oliva R., 2001. Maps and models in system dynamics: a response to Coyle, System Dynamics Review 17(4).

Homer J., 2007. Reply to Jay Forrester's System dynamics - the next fifty years. System Dynamics Review 23(4).

Kalakota R., Rathnam S., Whinston A.B., 1993. The role of complexity in object-oriented systems development. Proceeding of the Twenty-Sixth Hawaii International Conference on System Sciences 4, p759-768

Martin M A., 2001. Mistakes and Misunderstandings: Table Functions. D-4679.

Myrtveit M., 2000a. Object Oriented Extensions to System Dynamics. Conference Proceedings of the 18th International Conference of the System Dynamics Society, Bergen, Norway.

Myrtveit M., Tignor W., 2000b. Object Oriented Design Patterns and System Dynamics Components. Conference Proceedings of the 18th International Conference of the System Dinamics Society, Bergen, Norway.

Scholl, G J., 1995. Benchmarking the System Dynamics Community: Research Results. System Dynamics Review 11(2),p139-155.

Panneerselvam, R., 2004. Research Methodology. Prentice-Hall of India

Senge, P. M., 1990. The Fifth Discipline: The Art and Practice of the Learning Organization. New York: Currency Doubleday

Sargent R G., 1998. Verification and validation of simulation models. Proceedings of the Winter Simulation Conference

Stange K A., 1998. Mistakes and Misunderstandings: Hidden Time Constants and Growth Fractions. D-4768

Wolstenholme E. F., 1999. Qualitative vs. Quantitative Modelling: The Evolving Balance. Journal of the Operational Research Society 50, 422–428.