Complex System Modeling, Testing and Re-Engineering with Complementary Methods for System Dynamics

Bharath Dantu and Eric D. Smith
IMSE: Industrial, Manufacturing & Systems Engineering Department
RIMES: Research Institute for Manufacturing and Engineering Systems
UTEP: University of Texas at El Paso
El Paso, TX 79968
ESmith2@UTEP.edu
(915)747-5205

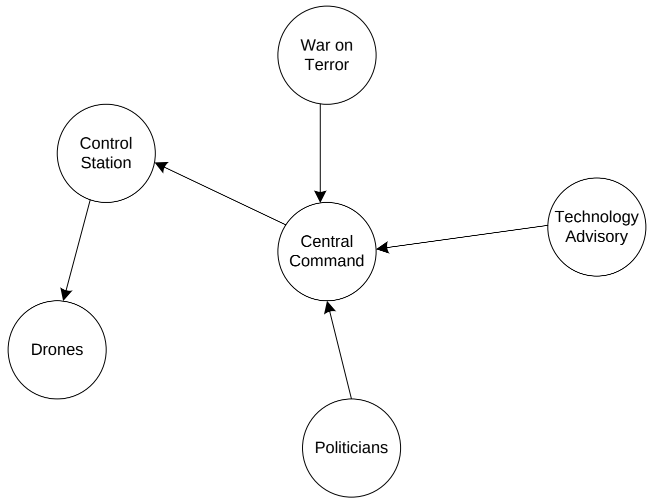
Synopsis:
Complex and amorphous systems modelling has traditionally been accomplished with soft system methodologies, of which System Dynamics (SD) is an example. While System Dynamics efficiently focuses on key parameters and relationships, often a system will not be well understood unless the boundary, background and context of the problem is explicitly presented. A further challenge to modelling effectively results from the dual nature of parameters and relationships. Parameters may be either quantitative or qualitative, while relationships may be either syntactic or semantic. A robust system dynamics methodology is presented that incorporates a Zachman framework as an example background, as well as incorporating a two-sided complementary parameters and relationships. With these improvements, system modelling is significantly richer in the diversity of available elements, and rendered more precise by the targeted use of modelling components. The augmented methodology is applied to the modelling, testing and re-engineering of an example system.

Key words:
System Dynamics, complementary, quantitative, qualitative, syntactic, semantic, context

Sessions:
Model-Based Test & Evaluation: "Modeling to Manage Complexity"
    Managing SoS Architectural Framework Models
Systems Engineering:
    The Systems Engineering of Test & Evaluation

System Dynamics

System Dynamics, as a modeling tool, begins with a diagram that shows entities and influences among the entities, as shown below.



Influence diagram from System Dynamics

System Dynamic influence diagrams can be augmented with the characterization of the influences with positive or negative correlations, as well as with mathematical equations. These types of augmentations will not be explored here.

Adding the Zachman Framework

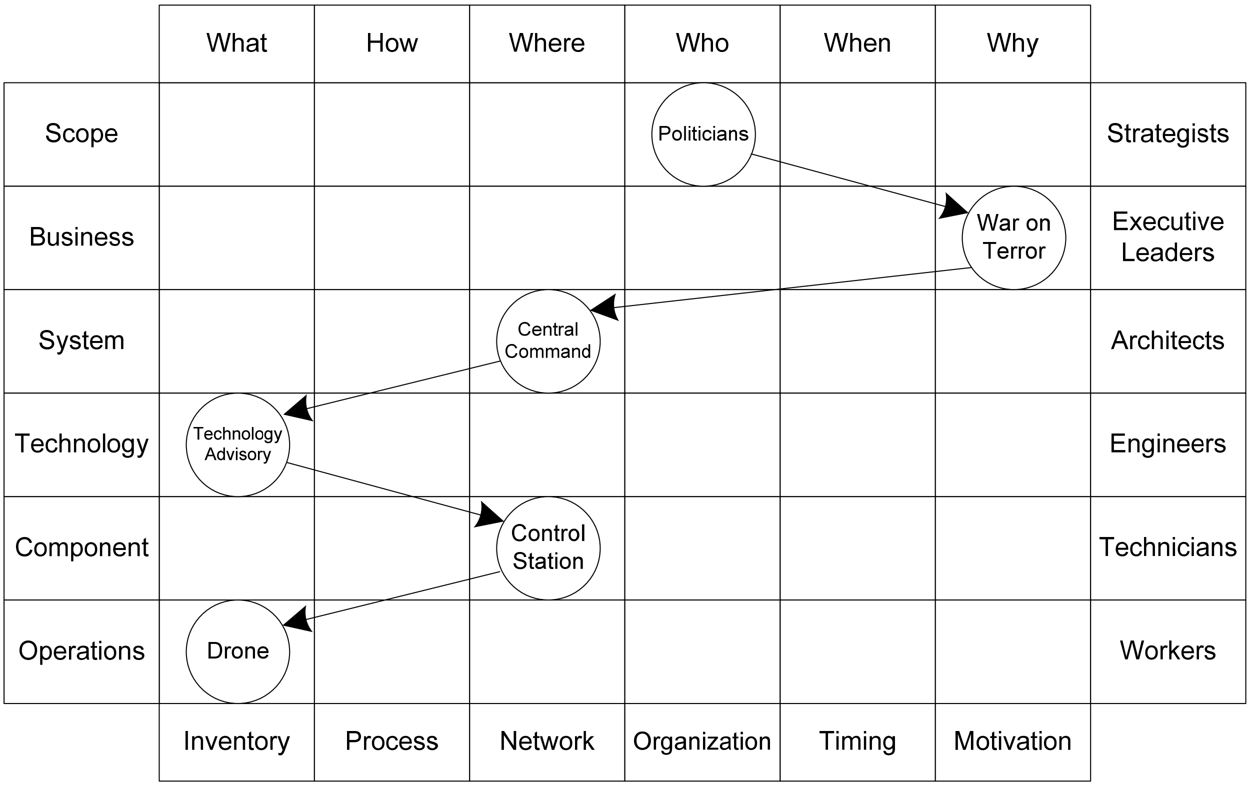
The Zachman Framework is shown below.

	What	How	Where	Who	When	Why	
Scope							Strategists
Business							Executive Leaders
System							Architects
Technology							Engineers
Component							Technicians
Operations							Workers
	Inventory	Process	Network	Organization	Timing	Motivation	

Zachman Framework

System Dynamics on a Zachman Framework

The entities of the influence diagram, when placed in the Zachman framework, must necessarily be categorized according to the Zachman's rows and columns. When entities are so arranged, their character can more immediately be comprehended by a person who is reviewing the model. The result of this overlay is as follows.



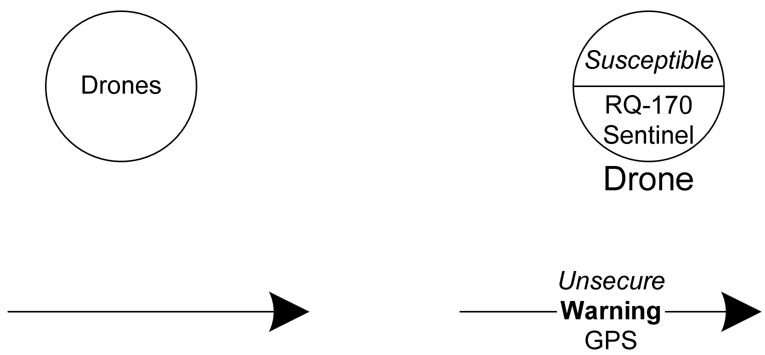
Influence Diagram in Zachman framework background

Note that placing an influence diagram in a Zachman framework background allows the easy re-consideration of the relative levels at which the entities properly exist. Previously, all the entities in the influence diagram were floating, and there were no points of reference as to their levels. Once the levels of the entities are properly determined, the influences among the entities can be re-considered. In the diagram, the influences now shown are the top-to-bottom influences only. The bottom-to-top influences are omitted for clarity. Note that that influences that occur across adjacent levels are more likely to be strong, sustained, robust and defined influences, and thus will probably be more important and influential than those that jump across levels.

Complementarity

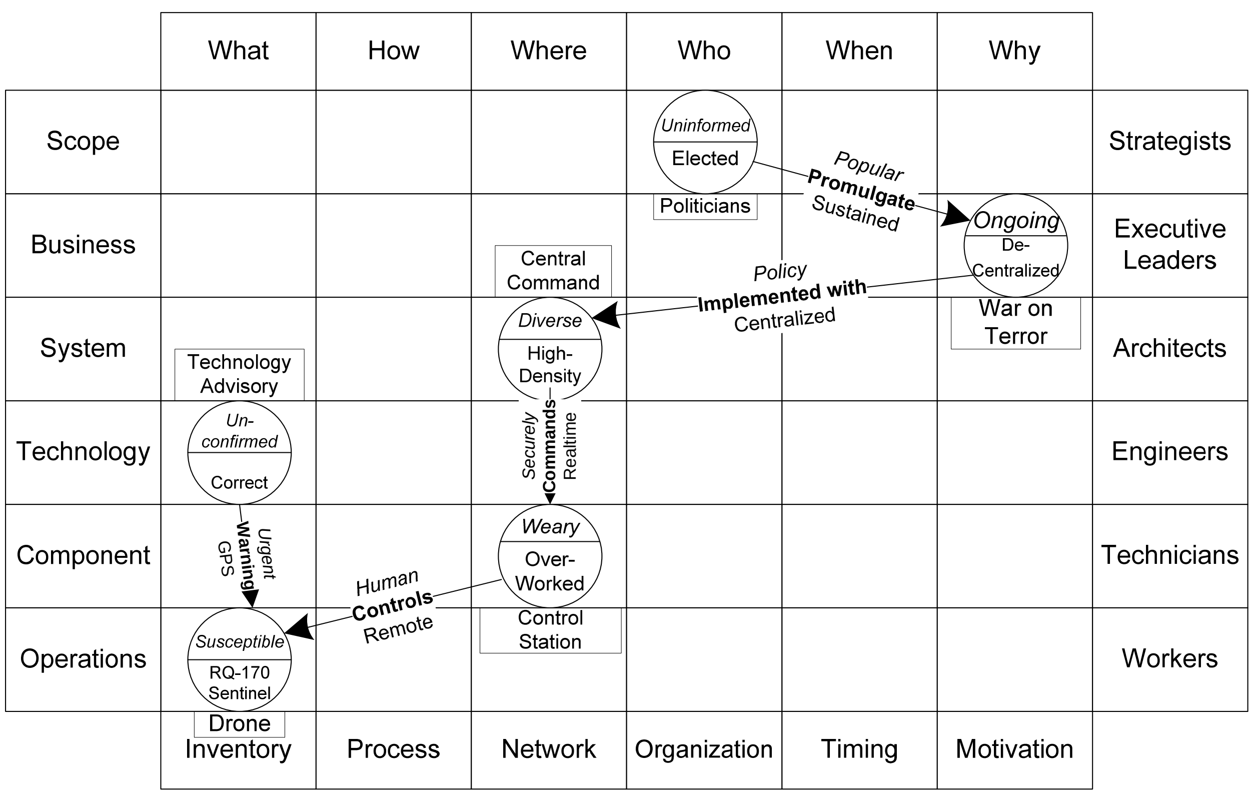
A deficiency of a traditional influence diagram is that there is no provision of a further characterization of the entities, or of the influences / relationships among the entities. The most important way to characterize entities and relationships is with a summary title, as well as with the most important feature and qualitative assessment.

For example, an entity and an influence can be augmented as follows.



Augmented entity and augmented influence

The previous influence diagram overlaid on a Zachman framework background is thus augmented as follows.



Complementary descriptions for entities and influences in a

System Dynamics diagram overlaid on a Zachman framework background

The result is a diagram that shows the most important entities, which each placed in the correct overall background and context. The final diagram contains a limited number of entities which can be comprehended as a single model upon quick inspection. Each entity is characterized with a name label, as well as with the most pertinent technical and qualitative description. All the entities are connected with influence relationships that are summarily described both in a *qualitative* and quantitative manner.

As the complexity of these diagrams grows, the complexity can be numerically characterized and controlled by the following index.

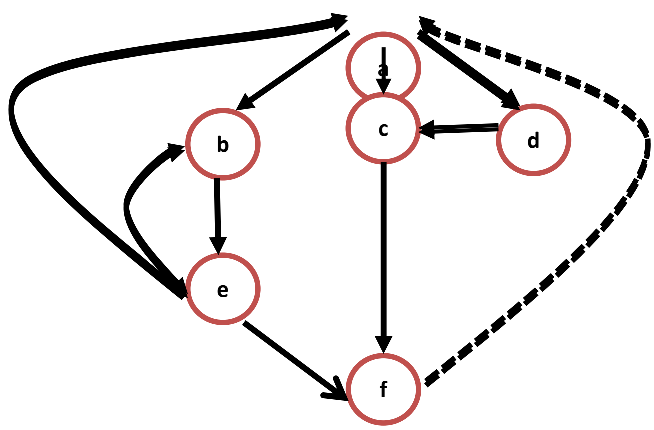
Cyclomatic Complexity

The modularization of software systems improves the ability to test and maintain a software system. Without modularization, a company can potentially spend half of its development time in testing and can spend most of its dollars in maintaining a system. It's essential to have a mathematical technique that provides a quantitative basis for modularization and allow us to identify software and system modules that either difficult or easy to test and maintain. The Cyclomatic Complexity is a tool that can measure the complexity in a System Dynamics model. The complexity measure approach allows to measure and control the number of paths through a program or system. The complexity measure developed is defined in terms of basic paths that can generate all the possible paths.

The Cyclomatic number V (G) of a graph G with n vertices, e edges and p connected components is

$$V(G) = e - n + p.$$

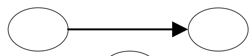
To utilize this mathematical tool, the program must associate with a directed graph that has unique entry and exit nodes. Each node in the graph represents a block of code in the program, flow represents the sequence among the blocks of code and the arcs represent the braches taken in the program. This graph is classically known as the program control graph and it is assumed that each node enters and exits from the succeeding and preceding nodes.

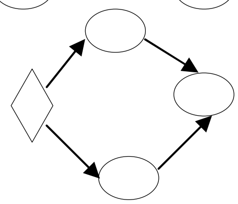


Example directed graph for characterization by the Cyclomatic Complexity

The strategy behind the whole study is to measure the complexity of a program by computing the number of linearly independent paths V (G). Below diagrams provide a better understanding about the control graphs.

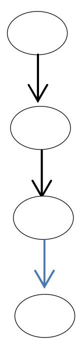
Control Structures for the Cyclomatic Complexity Index

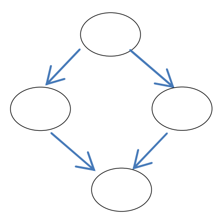
Sequence  $v = 1 - 2 + 2 = 1$

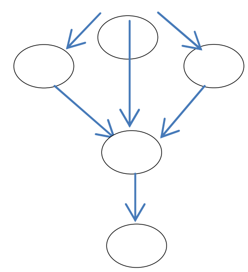
If Then Else  $v = 4 - 4 + 2 = 2$

From the above complexity equation the term "P" is the number of connected components. A defined program control graph which has a unique entry and exit nodes, all nodes reachable from the entry and the exit reachable from all nodes would result in all control graphs having only one connected component.

To emphasize more on the program control graphs, let's imagine a main program M and two subroutines A and B having a control structure.

M: 

A: 

B: 

Looking at the above diagrams,

$$v\,(M \cup A \cup B) = e - n + 2p = 13 - 13 + 2X3 = 6$$

This method with $p \neq 1$can be used to calculate the complexity of a collection of programs, particularly a hierarchical nest of subroutines. This expression can also be written as,

$$v\,(M \cup A \cup B) = v\,(M) + v\,(A) + v(B) = 6$$

In general, the complexity of the entire control graphs "C" with "k" connected components is equal to the summation of their complexities.

$$v\,(C) \quad = e - n + 2p = \Sigma\, e_i - \Sigma\, n_i + 2k$$

$$= \Sigma\,(e_i - n_i + 2) = \Sigma\, v\,(C_i).$$

Conclusion

Although system complexity poses significant challenges, complex situations can be better understood by determining the most significant entities and the relationships among the entities, with the methods of system dynamics. The problem of floating entities is fixed by providing a Zachman framework background, which fixes the entities within their respective domains. A further modeling innovation of showing both the quantitative and qualitative characteristics of both the entities and the relationships among the entities helps the reader differentiate between the concrete features of the model and the qualitative influences that exist. The complexity of any model can then be determined by accepted measures of complexity, such as the Cyclomatic Complexity index.

References

Eric D. Smith and Ricardo Pineda, *Clarity, Variety, Allocation, Synergy and Emergence in the Formulation of Tradeoff Studies*, Conference on Systems Engineering Research (CSER), Stevens Institute of Technology and the University of Southern California, Hoboken, NJ, 2010.

Eric D. Smith, *Complementarity in Systems Architecting*, Conference on Systems Engineering Research (CSER), Stevens Institute of Technology and the University of Southern California, Los Angeles, CA, 2010.

Bharath Dantu and Eric D. Smith, *Clinical Diagnosis Modeling Techniques with a Systems Dynamics Focus*, Society for Health Systems Conference, Orlando, FL, 2011.

Bharath Dantu and Eric D. Smith, *Mishap Analysis of Unmanned Aerial Vehicle (UAV) and Black Hawk Friendly Fire Incidents*, Office of Naval Research (ONR) Partnership Conference, Crystal City, VA, 2010.

McCabe, Thomas, *A Complexity Measure*, IEEE Transactions on Software Engineering, December, 1976, 308-320.