Observations re. "ISM Algorithm," by Constantina Spanoude (2015)

February 28. 2016

Joseph J. Simpson Mary J. Simpson

Introduction

C. Spanoude's document details an Interpretive Structural Modeling (ISM) algorithm associated with the Structured Dialogic Design Process (SDDP). SDDP is one of a number of system analysis and evaluation techniques based on the ISM approach. ISM is one component of a more general analysis and evaluation technique called Structural Modeling (SM.) SM has three components: 1) basic structural modeling (BSM), 2) ISM, and 3) structural integration modeling (SIM). John N. Warfield developed and refined the BSM and ISM structural modeling components; Simpson and Simpson defined the SIM structural modeling component. Structural Integration Modeling is under active method and tool development. An important point associated with structural modeling is the goal of complexity reduction. A key element of complexity reduction is the reduction of uncertainty. This preliminary analysis highlights the *uncertainty reduction* aspects of this particular SDDP ISM approach.

This preliminary analysis also addresses another key concept related to uncertainty reduction: that is, the concept of an isomorphic arrangement between natural language relationships, structural graphs and mathematical relations. The referenced document would benefit from a clear delineation and description of which component of SM is associated with each method, process and/or technique. This clarity will also tend to reduce uncertainty and complexity.

The SDDP "Tree Of Influence"

The SDDP "tree of influence" is constructed from a set of objects to be structured. The system structure is incrementally created based on an empirical evaluation of the statement:

> *Suppose we were able to implement Statement A [object A]. Will this help SIGNIFICANTLY in implementing Statement B [object B] in the context of improvement planning?*

The first natural question that arises is: Are the words 'influence' and 'help significantly' interchangeable? More specifically and in this context, is the natural language relationship intended to be the same between 'influence' and 'help significantly'?

A positive answer to that question then leads to: "What are the logical properties of reflexivity, symmetry and transitivity that are associated with this *influence* type of natural language relationship?

1. Reflexivity

A natural language relationship associated with this situation seems to have an irreflexive property, because two different statements or objects are being addressed. In this context, it may make no sense to say:

> "Suppose we were able to implement Statement A. Will this help SIGNIFICANTLY in implementing Statement A in the context of improvement planning?"

Therefore, an **irreflexive** logical property is assigned to the *influence* natural language relationship.

2. *Symmetry*

Continuing with the logical property analysis, what type of logical property for symmetry is associated with the *influence* natural language relationship?

- An asymmetric logical property will allow object A to influence object B, but will not allow object B to influence object A.
- A symmetric logical property will allow object A to influence object B and object B to influence object A.
- A non-symmetric logical property will allow both asymmetric and symmetric logical properties to exist.

It appears that the **non-symmetric** logical property is the most appropriate logical property for the *influence* natural language relationship.

3. *Transitivity*

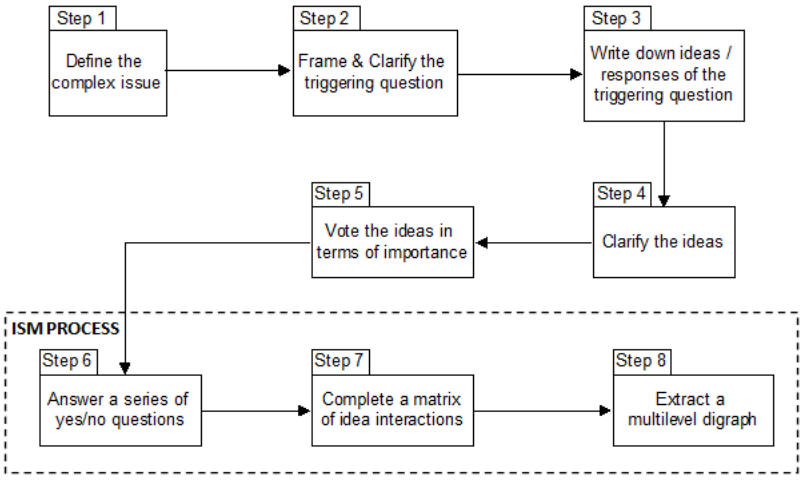
Now the transitive properties associated with the influence natural language relationship will be addressed. The property for transitivity involves three or more objects: A, B and C.

- If object A maintains an *influences* relationship with object B and object B maintains an *influences* relationship with object C, and object A maintains an *influences* relationship with object C, then the *influences* relationship is transitive.
- The *influences* natural language relationship may be intransitive because object A may not influence object C.
- Or it is possible that both transitive and non-transitive properties are associated with the *influences* natural language relationship.

In this case, the **non-transitive** logical property will be associated with the *influences* natural language relationship.

This quick analysis assigns the logical properties of irreflexive, non-symmetric and non-transitive to the *influences* natural language relationship.

C. Spanoude's Figure 1 from page 2 of the referenced document is shown below, and identifies the "Basic Steps of a typical SDDP."

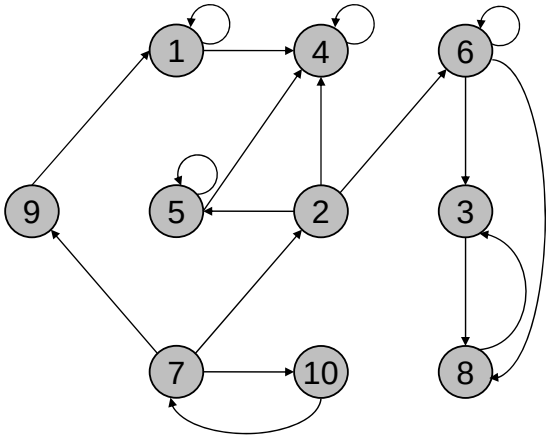


As shown, steps 6, 7 and 8 of the SDDP process are identified as the ISM process. These SDDP process steps are:

Step 6: Answer a series of yes/no questions
Step 7: Complete a matrix of idea interactions
Step 8: Extract a multilevel digraph.

These three steps are further aligned with binary matrix construction based on a contextual relationship. The natural language contextual relationship that is used in Spanude's document is *influences.* An example binary adjacency matrix A is provided (replicated below left) as well as the digraph associated with the adjacency matrix (replicated below right).

$$A = \begin{array}{c|cccccccccc} & e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 & e_9 & e_{10} \\ \hline e_1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ e_2 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ e_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ e_4 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ e_5 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ e_6 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ e_7 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ e_8 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ e_9 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ e_{10} & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{array}$$



The concept of an adjacency matrix, as it is used within the ISM process, raises a number of significant analytic, evaluation and semantic issues, creating a high degree of uncertainty and therefore complexity.

The binary matrix A indicates that the reflexive logical property associated with matrix A is non-reflexive. This can be determined by noting that the 10 by 10 matrix has four (4) ones (1) on the matrix diagonal as well as six (6) zeros (0) on the binary matrix diagonal. The digraph of the adjacency matrix A also indicates a non-reflexive logical property because objects 1, 4, 5, and 6 have self-loops indicating a reflexive relationship. Objects 2, 3, 7, 8, 9, and 10 do not have self-loops, and so indicate an irreflexive logical property. Accommodating both irreflexive and reflexive logical properties makes this a graph of a natural language relationship that has a non-reflexive logical property.

The term 'adjacency matrix', has a very specific meaning in mathematics. Warfield provided an example of a binary matrix based on the natural language relationship 'is-adjacent-to', which is different than the adjacency matrix from mathematics. In the mid-1970's Andy Sage wrote a book that attempted to present Warfield's structural modeling work. Warfield always claimed that Andy miscopied Warfield's Battelle Monograph titled "Structuring Complex Systems." One key area of conflict and confusion associated with the different approaches to ISM documented in these two books is the area associated with the application of adjacency matrices in the ISM process.

Simpson and Simpson developed the Augmented Model-Exchange Isomorphism (AMEI), in part, to provide a conceptual foundation upon which the differences in these approaches may be evaluated and discussed. A detailed explanation of which adjacency matrix type is used, why this specific approach was selected, and how it is used in this structuring process (based

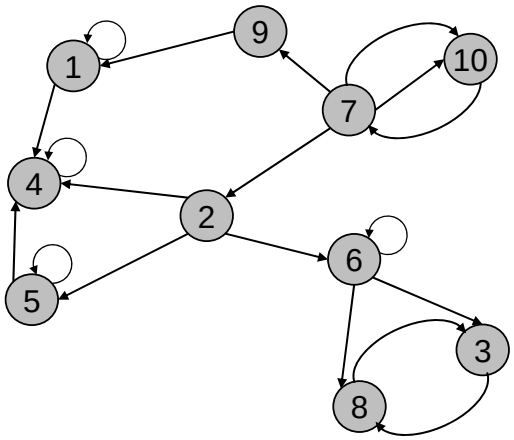
on the *influences* natural language relationship) will greatly reduce the uncertainty and complexity associated the SDDP approach. A key aspect in Warfield's use of the natural language relationship *is-adjacent-to* is the fact that he used it as an example of transitive closure. This fact alone creates uncertainty and complexity, because most individuals would not assign a transitive logical property to the natural language relationship *is-adjacent-to*. The plethora of ISM techniques developed over the past five (5) or six (6) decades creates substantial uncertainty. The more appropriate term of Relationship Mapping, originally proposed by Warfield, will be used instead of ISM as our more detailed analysis process progresses.

While both the binary adjacency matrix and the digraph of the binary adjacency matrix indicate a non-symmetric logical property associated with the natural language structuring relationship, it is not clear how the digraph was constructed from the presented matrix. There appears to be a number of issues with the matrix-digraph pair presented. A clear description of how the digraph was constructed from the given binary matrix is needed before any further analysis can be accomplished.

The transitive logical property cannot be read directly from the digraphs or the binary matrix. This is an important piece of contextual information that must be well documented and communicated to support the goal of uncertainty reduction and complexity reduction. Simpson and Simpson created the Abstract Relation Type (ART) and the AMEI as canonical forms of structural modeling information encoding. The information should be recorded and available in the AMEI and ART forms.

In an effort to further evaluate the Spanoude matrix A and its associated digraph, the SageMath open source software application was used to generate the actual matrix associated with the original Spanoude digraph. The resultant matrix - $\mathbf{A_1}$ - is shown below left. Using the $\mathbf{A_1}$ matrix, the standard SageMath **'plot'** function produced the digraph as shown below right.

$$A_1 = \begin{array}{c} \\ e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \\ e_6 \\ e_7 \\ e_8 \\ e_9 \\ e_{10} \end{array} \begin{array}{c} \begin{array}{cccccccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 & e_9 & e_{10} \end{array} \\ \left[\begin{array}{cccccccccc} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{array}\right] \end{array}$$



It becomes clear from this digraph, that there are two symmetric feedback connections in the digraph: one between elements 3 and 8, and a second between elements 7 and 10.

Following discussions with Kevin Dye and Yiannis Laouris, it was determined that the original cited example of the binary adjacency matrix A was in error. The natural language system structuring relationship has an irreflexive logical property, not a reflexive logical property. Given

this new information, matrix $\mathbf{A_1}$ was modified to put the matrix in 'proper' mathematical form. The mathematical form was accomplished by:

- Replacing the ones (1) on the matrix diagonal with zeros (0). When the logical property is irreflexive, there are no ones (a) on the diagonal of the matrix.
- The symmetric feedback connections were condensed into a single node on the graph. Elements 3 and 8 were condensed into element 3, and elements 7 and 10 were condensed into element 7.

Below left, matrix $\mathbf{A_1}$ (a 10x10 matrix), is shown with the ones present on the diagonal circled in blue. The matrix $\mathbf{A_2}$ (an 8 x 8 matrix), in proper mathematical form, is shown below on the right.

$A_1 =$

	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}
e_1	1	0	0	1	0	0	0	0	0	0
e_2	0	0	0	1	1	1	0	0	0	0
e_3	0	0	0	0	0	0	0	1	0	0
e_4	0	0	0	1	0	0	0	0	0	0
e_5	0	0	0	1	1	0	0	0	0	0
e_6	0	0	1	0	0	1	0	1	0	0
e_7	0	1	0	0	0	0	0	0	1	1
e_8	0	0	1	0	0	0	0	0	0	0
e_9	1	0	0	0	0	0	0	0	0	0
e_{10}	0	0	0	0	0	0	1	0	0	0

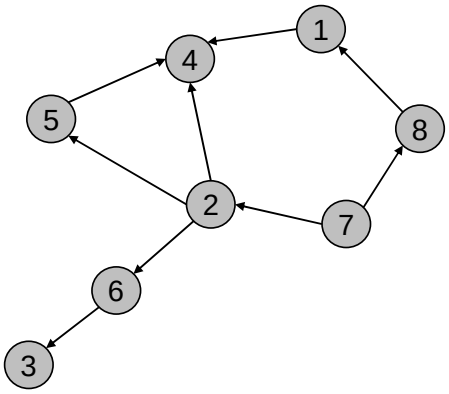
$\mathbf{A_2} =$

	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8
e_1	0	0	0	1	0	0	0	0
e_2	0	0	0	1	1	1	0	0
e_3	0	0	0	0	0	0	0	0
e_4	0	0	0	0	0	0	0	0
e_5	0	0	0	1	0	0	0	0
e_6	0	0	1	0	0	0	0	0
e_7	0	1	0	0	0	0	0	1
e_8	1	0	0	0	0	0	0	0

Once the matrix – now Matrix $\mathbf{A_2}$ – is put in 'proper mathematical form, a new digraph is generated. Using the $\mathbf{A_2}$ matrix (below left), the standard SageMath **'plot'** function produced the digraph as shown below right.

$\mathbf{A_2} =$

	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8
e_1	0	0	0	1	0	0	0	0
e_2	0	0	0	1	1	1	0	0
e_3	0	0	0	0	0	0	0	0
e_4	0	0	0	0	0	0	0	0
e_5	0	0	0	1	0	0	0	0
e_6	0	0	1	0	0	0	0	0
e_7	0	1	0	0	0	0	0	1
e_8	1	0	0	0	0	0	0	0



This newly generated digraph is an acyclic directed graph that can be further analyzed using SageMath functions to determine levels, display layouts and much more. Appendix A contains the SageMath commands that generate level set, topological sort, and four display layouts (acyclic, planar, spring, and circular).

Appendix A

Please note that all of the following graphics generated through the SageMath software assumes a base 0 – so that the numbering of each node starts with a zero (0) as opposed to a one (1). All of the elements are numbered accordingly. That is, the zero represents element number 1, the one represents element number 2, and so on.

Level Sets of the Digraph

```
SDDP_ISM_A_modified_g =
Matrix([[0,0,0,1,0,0,0,0],
        [0,0,0,1,1,1,0,0],
        [0,0,0,0,0,0,0,0],
        [0,0,0,0,0,0,0,0],
        [0,0,0,1,0,0,0,0],
        [0,0,1,0,0,0,0,0],
        [0,1,0,0,0,0,0,1],
        [1,0,0,0,0,0,0,0]])
gSDDP_ISM_A_modified_g = DiGraph(SDDP_ISM_A_modified_g)
gSDDP_ISM_A_modified_g.level_sets()

        [[6], [1, 7], [4, 5, 0], [2, 3]]
```

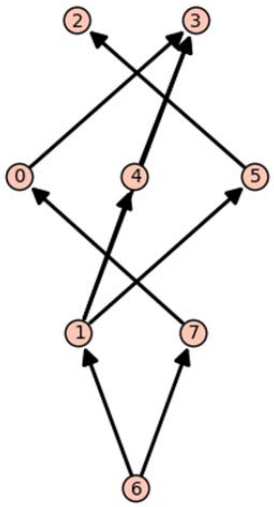
Topological Sort of the Digraph

```
SDDP_ISM_A_modified_g =
Matrix([[0,0,0,1,0,0,0,0],
        [0,0,0,1,1,1,0,0],
        [0,0,0,0,0,0,0,0],
        [0,0,0,0,0,0,0,0],
        [0,0,0,1,0,0,0,0],
        [0,0,1,0,0,0,0,0],
        [0,1,0,0,0,0,0,1],
        [1,0,0,0,0,0,0,0]])
gSDDP_ISM_A_modified_g = DiGraph(SDDP_ISM_A_modified_g)
gSDDP_ISM_A_modified_g.topological_sort()

      [6, 7, 1, 4, 5, 2, 0, 3]
```

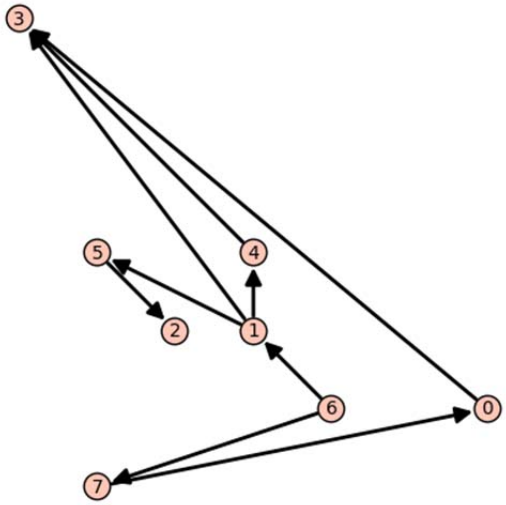
Acyclic Display Layout of the Digraph

```
SDDP_ISM_A_modified_g =
Matrix([[0,0,0,1,0,0,0,0],
        [0,0,0,1,1,1,0,0],
        [0,0,0,0,0,0,0,0],
        [0,0,0,0,0,0,0,0],
        [0,0,0,1,0,0,0,0],
        [0,0,1,0,0,0,0,0],
        [0,1,0,0,0,0,0,1],
        [1,0,0,0,0,0,0,0]])
gSDDP_ISM_A_modified_g = DiGraph(SDDP_ISM_A_modified_g)
gSDDP_ISM_A_modified_g.show(layout='acyclic')
```



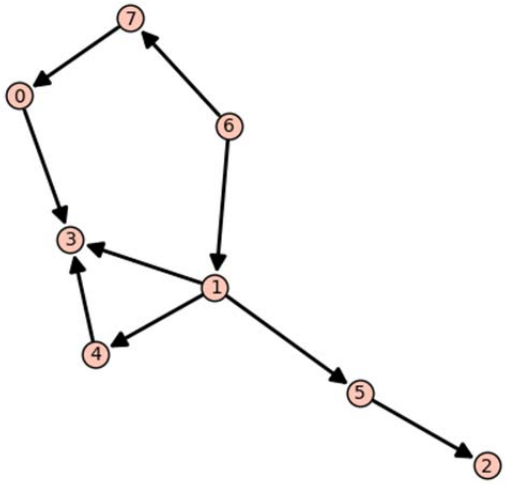
Planar Display Layout of the Digraph

```
SDDP_ISM_A_modified_g =
Matrix([[0,0,0,1,0,0,0,0],
        [0,0,0,1,1,1,0,0],
        [0,0,0,0,0,0,0,0],
        [0,0,0,0,0,0,0,0],
        [0,0,0,1,0,0,0,0],
        [0,0,1,0,0,0,0,0],
        [0,1,0,0,0,0,0,1],
        [1,0,0,0,0,0,0,0]])
gSDDP_ISM_A_modified_g = DiGraph(SDDP_ISM_A_modified_g)
gSDDP_ISM_A_modified_g.show(layout='planar')
```



Spring Display Layout of the Digraph

```
SDDP_ISM_A_modified_g =
Matrix([[0,0,0,1,0,0,0,0],
        [0,0,0,1,1,1,0,0],
        [0,0,0,0,0,0,0,0],
        [0,0,0,0,0,0,0,0],
        [0,0,0,1,0,0,0,0],
        [0,0,1,0,0,0,0,0],
        [0,1,0,0,0,0,0,1],
        [1,0,0,0,0,0,0,0]])
gSDDP_ISM_A_modified_g = DiGraph(SDDP_ISM_A_modified_g)
gSDDP_ISM_A_modified_g.show(layout='spring')
```



Circular Display Layout of the Digraph

```
SDDP_ISM_A_modified_g =
Matrix([[0,0,0,1,0,0,0,0],
        [0,0,0,1,1,1,0,0],
        [0,0,0,0,0,0,0,0],
        [0,0,0,0,0,0,0,0],
        [0,0,0,1,0,0,0,0],
        [0,0,1,0,0,0,0,0],
        [0,1,0,0,0,0,0,1],
        [1,0,0,0,0,0,0,0]])
gSDDP_ISM_A_modified_g = DiGraph(SDDP_ISM_A_modified_g)
gSDDP_ISM_A_modified_g.show(layout='circular')
```