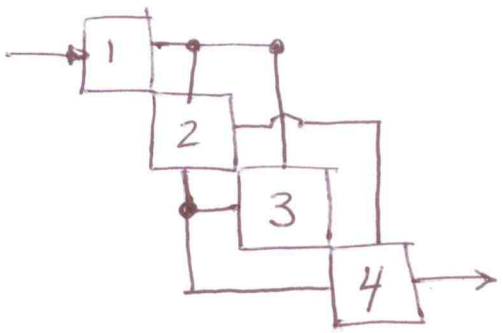
Thoughts on a Directed Graph Representation for Code Connectivity in MDO

- B. German, June 2012

→ Let's begin by thinking about a design structure matrix (DSM) or N^2 diagram:



- Each code is assigned <u>one</u> block in the diagram.
- Connectivity denoted with lines
- Feed-forwards in the upper triangular portion.
- Feed-backs in the lower triangular portion.

→ We can represent a DSM as a <u>directed graph</u> (typically, a <u>directed cyclic graph</u> if feedbacks are involved.).

Here's the <u>adjacency matrix</u> for the graph above:

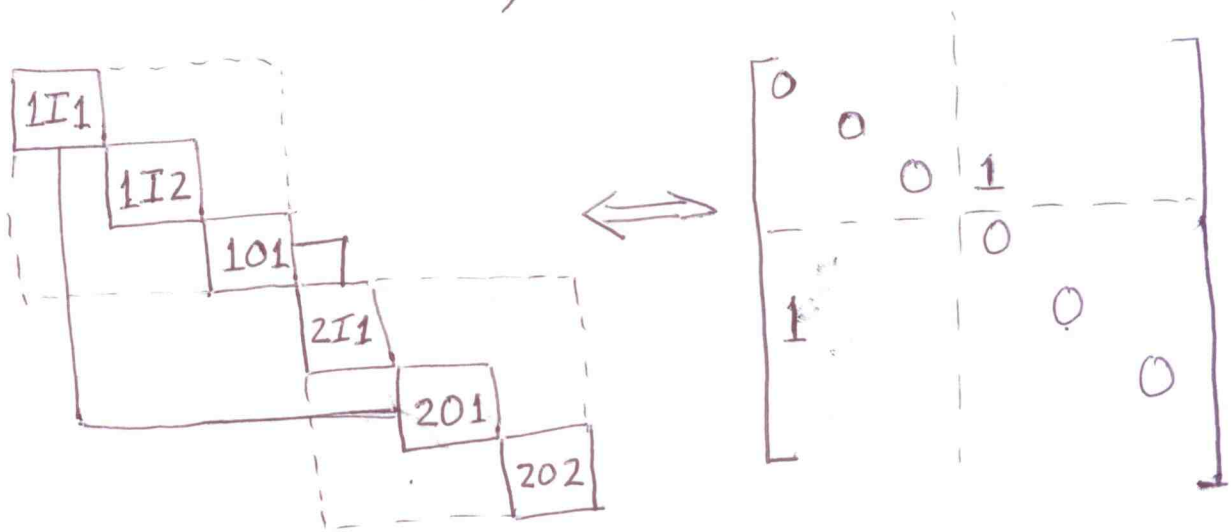
$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

← Each element a_{ij} represents linkage from "i to j".
Feed-forwards are upper triangular.
Feed-backs are lower triangular.

→ Problems w/ the DSM representation:

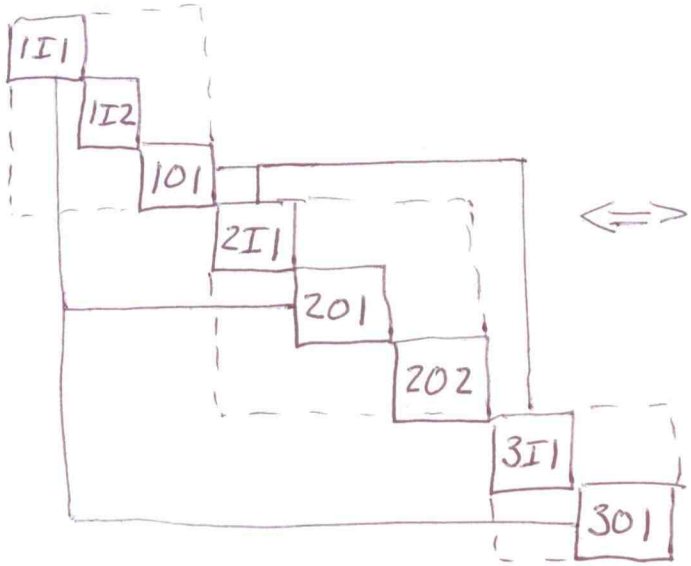
- Typically represents a "working" MDO interface; not good for exploring "potential" interfaces
- Does not allow us to detect "conflicts" such as multiple codes producing the same output variable that is fed-back as an input to another code.
- Obscures individual inputs and outputs

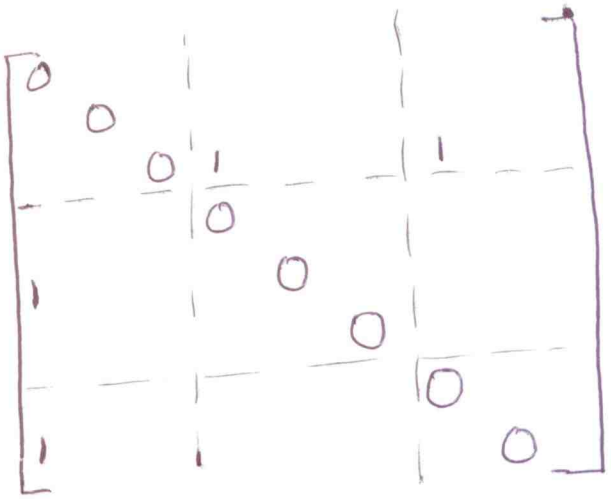
→ An alternate approach would be to express all inputs and outputs explicitly:



- Edges between inputs and outputs not drawn within a code
- The relationship between any two codes is therefore a bipartite graph

→ Now let's imagine that we add a third code which produces an output that can provide input to 1I1:





A column sum > 1 indicates a conflict

⇒ Need to choose which code's output to use in the feedback

- A row sum > 1 indicates that one code's output is used as an input to multiple other codes.

→ This approach implies a "block" structured adjacency matrix.

- Add additional codes to consideration list by extending matrix with additional zero blocks on diagonal
- Indicate connections in off-diagonal elements