Structural Modeling Project

Joseph J. Simpson
Mary J. Simpson
August 24, 2015

Revised Project Schedule

Version 0.16

Table of Contents

I. Introduction

The structural modeling project is based on the material developed in the previous Basic Structural Modeling Project (BSMP). The BSMP materials are located in the following places:

http://www.systemsconcept.org/
https://github.com/jjs0sbw/bsmp
https://www.researchgate.net/profile/Joseph_Simpson3

This project embraces system structural modeling in its most expansive sense. The mathematical models associated with Basic Structural Modeling are not restricted to matrix forms, as is the case with most published material associated with Structural Modeling. The Interpretive Structural Models in this project allow N-Squared Charts, Fuzzy Logic, probabilistic models and other types of system models that include empirical data and/or information in the model**.**

Given this wide range of model types, a clear model naming convention is required. The Abstract Relation Type (ART) form will be used to clearly identify each model type.

II. Project Definition

The SMP is defined in the SMP Project Description Document. The project goals and objectives are described along with the project context. The primary software product of the SMP is an adaptable software system designed to support a wide range of structural modeling activities.

The initial software will support the binary matrix mathematics developed by Warfield as it is implemented in the GMU ISM software shown in *A Handbook of Interactive Management, Second Edition (*Warfield and Cardenas, 1994*)*. Based on this resource, there are three software commands that may be used to structure a system. Only one of these software commands may be used on a single activity. These three software structuring commands are:

1. DOMODEL- This is the most general command, and is applied to any ISM structuring application.

2. DOCLUS- This command is restricted to systems with cycles (one or more).

3. DOPRIOR- This command is used on a priority structure with a single path.

The current System Concepts open source software is similar to the DOCLUS and DOPRIOR software commands; it is restricted to a specific type of natural language relationship. The current open-source web application is ‘Orders Cities’ based on the 'north-of' natural language relationship. The ‘north-of’ natural language relationship is transitive, irreflexive and asymmetrical. Further, the current BSMP Javascript web application allows only one city at each level.

Each software application must be identified by a relationship name and properties as well as other information about proper model application. In the case of the System Concepts BSMP Javascript web application, the information is:

- Relationship Name => north-of (and south-of)
- Relationship Properties => transitive, irreflexive, asymmetrical
- Relationship Conditions => only one city at each level

The natural language relationship, ‘north-of,’ is paired intuitively with the ‘south-of’ natural language relationship. By restricting the ordering to allow only one city at each level, the case where one city is ‘not related’ to another is eliminated. A city is either north, or south, of another city unless they are at the same latitude.

A new Javascript web application is being designed and developed that has the following relationship information:

- Relationship Name => north-of (and south-of)
- Relationship Properties => transitive, irreflexive and asymmetrical
- Relationship Conditions => one or more cities at each level.

The Structural Integration Modeling area is designed to evaluate, organize and communicate the structuring concepts and methods necessary to properly encode a selected natural language relationship into an executable mathematical relation. The DOCLUS (stands for 'do clustering') command will be implemented after the ‘north-of’ example is complete.

III. Phase One, Two and Three Summary

Activity Summary

Ruby code for the partially ordered 19 cities problem was developed as a mathematical calculation engine. This Ruby code was demonstrated during the INCOSE 2015 Symposium Tutorial, “Foundational Aspects of System Complexity Reduction.”

IV. Phase Four

Adapt the Ruby mathematical calculation engine to Javascript and create a Javascript web application. The Javascript web application will have a web graphic interface that connects to the Javascript computational engine. The ability to identify alternative orderings or to create different system structures based on the available object relation data may also be added to this web application version if there is enough time. In any case, a “group abduction” capability will be outlined and developed using Python code.

V. Phase Five

Demonstrate the final Javascript web application version.

VI. Timeline

Phase Four: Will start on September 28th 2015

- Adapt Ruby classes and modules requirements into Javascript requirements
- Document required behavior and application architecture
- Write Behavior Driven Development (BDD) code
- Write Test Driven Development (TDD) code

Phase Five: Planned start December 5th 2014

- Present completed web application
- Verify and validate application function
- Verify and validate application behavior
- Follow up presentation on January 9th , 2016 to address any issues