HANDLING DATA IN AUTOMATED ANALYSIS PROCESSES

Stuart Moffatt **December 2019**

Automating analysis processes can become arduous if data dependencies and data structures are not planned carefully in advance. Commonly, software requirements evolve over time and changes to the software become more difficult to implement as packages grow in complexity. In this paper, a methodology is presented for handling data dependencies that uses a new central data management concept for handling varied data ontologies and data dependencies. This innovation in data management provides a scalable solution for software that adapts to changes in data structures as integrated analysis tools evolve. The data handling methodology is presented and its application within Axsym's software integration environment Teclab is demonstrated by a case study to automate a road network management analysis process.

1 INTRODUCTION

Simulation and analysis often involve repetitive manual processes. For example, data must be acquired and transformed into analysis inputs; then data files must be manipulated, and software packages run; finally, the results must be interrogated, displayed, checked and stored. Automating analysis processes can improve the quality of simulation results through reduced human error, improved repeatability and enabling a greater number of simulation cases to be processed. Faster analysis turnaround times can reduce project lead times and allow cases to be investigated in more detail.

Automating data handling between different elements of a simulation can be problematic if software architectures have not been carefully planned and data structures within the code become fragmented. Variables are handled according to their data type, some of which may not be known upfront to the developer, adding complexity to the development of software architecture. In addition, numerical algorithms may require spatial discretisation of analysis variables on a computational grid, involving interpolation and transfer between coordinate systems.

Axsym have extensive experience integrating engineering software for analytical process automation and for multi-physics simulation. To facilitate the integration of algorithms and datasets Axsym have developed innovate software methodologies for variables handling. The *Teclab* software package has been developed by Axsym to provide a framework for software integration and overcome data handling challenges when creating automation utilities.

2 DATA MANAGEMENT

Teclab uses a central datastore for managing the analysis variables that are accessed by different elements of an integrated analysis process. Each element in a process workflow is referred to as a *domain*, and includes analysis algorithms, data files and external software packages. The datastore provides a central management system for all variables within each of the domains. The datastore is structured to segregate the variables between domains, which allows individual domains to be copied to create a new instance of the domain without causing ambiguity across variables with the same

name, as each variable belongs to a specific instance of the domain. Access to variables between domains is controlled centrally by the governance process (Moffatt, 2020) which resolves data dependencies between domains and determines how each analysis variable is accessed by different elements of the simulation. The datastore manages variables by assigning them to a domain (or more specifically to an instance of a domain) within the analytical process. For each domain, the output variables, input variables and computational grids are stored in the centralised data management system known as the *datastore*.

2.1 Domain Variable Processing

The distinction between input and output variables is as follows:

- Input variables into a domain are set from the output variables from other domains. They include input variables into algorithms, data inputs into external software packages and data written to file. Input and output variables are handled differently, as the input of one domain is dependent on the output of another; input variables are only stored as *references* to other output variables.

- Output variables have the values set by the domain, such as the results from a numerical calculation, output from an external software package or data read in from file. The datastore groups all output variables from each domain and controls how other domains can access these as input variables.

In order to process complex model ontologies, the outputs of several modelling domains may require aggregating to provide a single input value. An example is a structural beam model that experiences loads from multiple sources, such as wind loading acting on different regions of a tower or aircraft wing that is attached at one end to a supporting structure. Figure 1 shows three wind load models providing forces at the structure's built-in end, mid-section and free end. The beam model will have an input variable, force, and the three wind load models each produce force as an output variable.

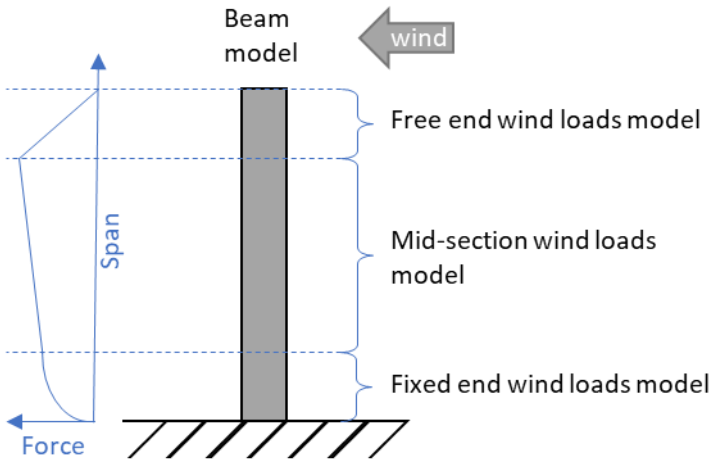

Figure 1: Example of superposition of variables from multiple algorithms, where the wind loads from three individual models are combined to provide overall loading on the beam structure.

The beam model therefore has a dependency on the wind models and requires the summation of the resulting force distributions from the three wind load models, thus providing the force as an input variable into the beam model. To aggregate output variables into a single input variable, the input variable is stored as a series of references to the corresponding output variables where there is a dependency. When the datastore is queried for the values of the input variable, the output variable values of the referenced domains are retrieved and then summed.

2.2 Declaration of Analysis Variables

The centralised datastore enables analysis variables to be declared at runtime, giving great flexibility in building integrated analytical systems. Software developers have the option to hard code analysis variables definitions into the source code when building in new analysis methods. Alternatively, variables can be defined in an external data file, enabling external software and datasets to be linked without needing to modify the source code. The ability to handle variables arbitrarily becomes especially important when linking with dynamic datasets such as spreadsheets, as discussed in the data adapter in Section 3.

2.3 Variable Types

Variables can be real numbers, integers or character strings and can be represented as scalars, vector fields or matrices.

2.3.1 Scalar

A scalar variable $\boldsymbol{v}$ distributed over a grid of size m by n will take the form

$$\boldsymbol{v} = \begin{bmatrix} v_{11} & \cdots & v_{1n} \\ \vdots & \ddots & \vdots \\ v_{m1} & \cdots & v_{mn} \end{bmatrix}$$

2.3.2 Vector Field

A vector field $\overline{\boldsymbol{v}}$ is a real variable that is spatially distributed over a computational grid $\overline{\boldsymbol{g}}$, where the value stored at each grid node $\bar{g}$ in the form of a vector $\bar{v}$. A vector field can represent the continuous distribution of a vector over any computational grid of up to three dimensions. The wind loads example shown in Figure 1 is a vector field of force $\overline{\boldsymbol{f}}$ distributed over a 1D grid $\overline{\boldsymbol{g}}$ that represents the beam geometry as a 1D grid of size m. The wind loads will be applied as 3D force vectors at each grid node.

A 3D force vector $\bar{f} = \begin{Bmatrix} f_i \\ f_j \\ f_k \end{Bmatrix}$ distributed over a 1D grid of m nodes would take the form

$$\overline{\boldsymbol{f}} = \begin{bmatrix} \begin{Bmatrix} f_i \\ f_j \\ f_k \end{Bmatrix}_1 \\ \vdots \\ \begin{Bmatrix} f_i \\ f_j \\ f_k \end{Bmatrix}_m \end{bmatrix}$$

where a force $\bar{f}$ is applied to each node $\bar{g} = \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}$ on grid $\overline{\boldsymbol{g}}$ of size m

$$\overline{\boldsymbol{g}} = \begin{bmatrix} \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}_1 \\ \vdots \\ \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}_m \end{bmatrix}$$

A 3D vector $\bar{v} = \begin{Bmatrix} v_i \\ v_j \\ v_k \end{Bmatrix}$ distributed over a 2D grid m by n would take the form

$$\overline{\boldsymbol{v}} = \begin{bmatrix} \begin{Bmatrix} v_i \\ v_j \\ v_k \end{Bmatrix}_{11} & \cdots & \begin{Bmatrix} v_i \\ v_j \\ v_k \end{Bmatrix}_{1n} \\ \vdots & \ddots & \vdots \\ \begin{Bmatrix} v_i \\ v_j \\ v_k \end{Bmatrix}_{m1} & \cdots & \begin{Bmatrix} v_i \\ v_j \\ v_k \end{Bmatrix}_{mn} \end{bmatrix}$$

where each vector field element $\bar{v}$ is applied to a node $\bar{g} = \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}$ on grid $\overline{\boldsymbol{g}}$ of size m by n

$$\overline{\boldsymbol{g}} = \begin{bmatrix} \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}_{11} & \cdots & \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}_{1n} \\ \vdots & \ddots & \vdots \\ \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}_{m1} & \cdots & \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}_{mn} \end{bmatrix}$$

Each vector field variable is assigned a *master grid*, providing the geometry onto which the values are defined. The values of a vector field variable can be requested on another grid, where the values will be interpolated by the datastore from the nodes on the master grid to the destination grid.

2.3.3 MATRIX

Matrix variables are structured in a similar manner to vector fields with the exception that they are not associated with a computational grid. Each element in a matrix variable represents a cell in the matrix and the contents of each cell can be real, integer or character strings. Since an analysis variable is specified to have a certain type, all cells in a matrix variable must have the same data type.

3 DATA ADAPTER

The data adapter is a user-configurable domain (or element) in the workflow process that provides an interface to an external software utility or dataset. It can read or write data to file and execute external software applications or scripts. Figure 2 shows how the data adapter connects external systems with the internal data structure. Information is passed through the data adapter as analysis variables which are made available to other domains (including other data adapters) through the central datastore.

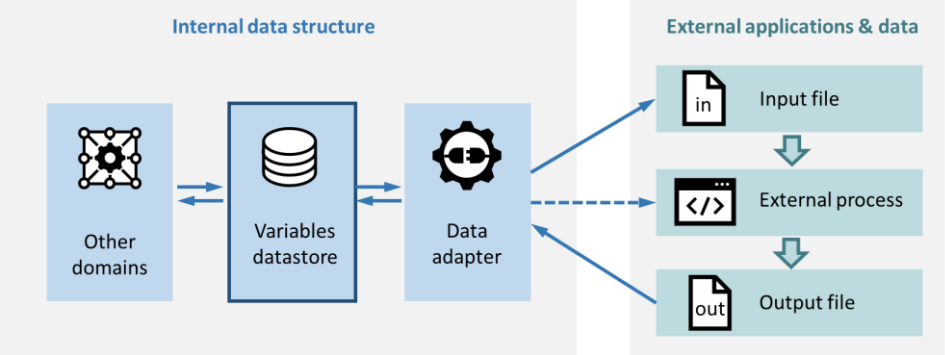

Figure 2. Data adapter connection to external applications and data

The data adapter is configured using a set of user-specified instructions contained within an XML file that is easily edited using an Excel spreadsheet utility. Specific information provided by the data adapter identifies the host operating system, parsing directives, data ontologies, scripts for invoking external software and network protocols. Input and output variables to be transferred between the datastore and the external software are specified in the data adapter configuration file, together with the necessary file parsing information that provides the semantics and data formats for reading and writing variables to file. External software applications are invoked through a system call, giving scope to run programs directly or through batch scripts to provide additional functionality such as networking protocols or filesystem manipulation.

4 PROCESS AUTOMATION CASE STUDY

The ability to integrate computational methods and datasets using varied data types distributed across the nodes of different grids is demonstrated for road network management. This case study provides a practical automation solution for the labour-intensive analysis process of traffic modelling on road networks, which is also applicable to other network simulations, including utilities and other transport modes.

4.1 Requirement for Automation

When planning changes to the road network such as upgrades or road closures, transport authorities combine microsimulation models with strategic models to assess traffic flow behaviour on the local and surrounding road systems. The need for two distinct modelling approaches requires a substantial amount of manual data processing of traffic volumes (network loads). This process is complicated by differences in model granularity, where smaller roads may not be present in both models (requiring data handling between different models) and vehicle categories may be aggregated (requiring summation of output data from one model as input for another).

Handling network models of different geographical extent involves a cordoning process to curtail the strategic model at a localised boundary, corresponding to the boundary of the microsimulation. The cordoning process is carried out using a series of utilities, producing numerous data files to be processed, catalogued and stored for future use. Due to differences in the granularity within the cordon zone, network loads cannot be directly transferred between models. Instead, the *change* in loading against a baseline case on one model is evaluated then applied to the baseline load on the other model. Since loading values can vary between very high and very low values, calculating load

changes as an increment or factors could introduce errors, therefore a more resilient numerical process is needed to weight loading values being transferred between domains.

4.2 SOLUTION

Axsym provided Transport for Greater Manchester with a software tool to automate the transfer of traffic flow behaviour between the Greater Manchester strategic road network model and a microsimulation package. The tool was delivered as a desktop utility to automate data processing, file management and the running of software utilities. An outline of the process is shown in Figure 3.

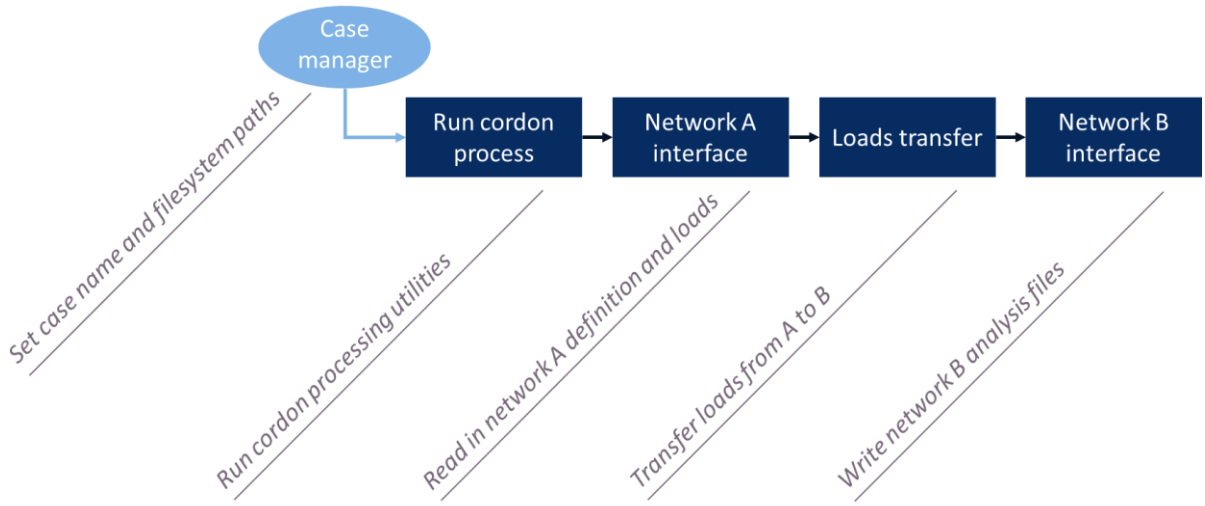

Figure 3. Network load transfer process

The interactions of variables with the datastore and with data files and applications are shown in Figure 4. The workflow has five domains: Case Manager, Data Adapter, Network Model A Interface, Network Loads Transfer and Network Model B Interface. These are discussed below:

Case Manager	Information about the case entered by the analyst is processed by the Case Manager domain. The Case Manager provides access to case definition variables to all relevant simulation domains.
Data Adapter	The data adapter runs a series of external software utilities for carrying out the network cordoning process, where file management and the execution of external codes is done through the system calls to the operating system.
Model A Interface	The interfacing domain for the strategic model (Network model A) reads network definition and loads data from files in a proprietary format and assembles this data into variable arrays that are uploaded to the datastore.
Loads Transfer	The network loads transfer domain compares network loading with baseline values on Network A and incorporates a weighting function to transfer the change in loading to the microsimulation (Network B). Aggregation of loading classes (in this case vehicle classes) and differences in node density around the cordon perimeter are handled according to the definitions

provided by the user in the Loads Classifications and Zone Correspondence files.

Model B Interface

The resulting loads for Network model B are obtained from the datastore and written in the required output file format to run the microsimulation of the Network B model. In the case study, a commercial traffic microsimulation package was selected, which required an analyst to interact with the windows menu driven operations at the end of the automated process.

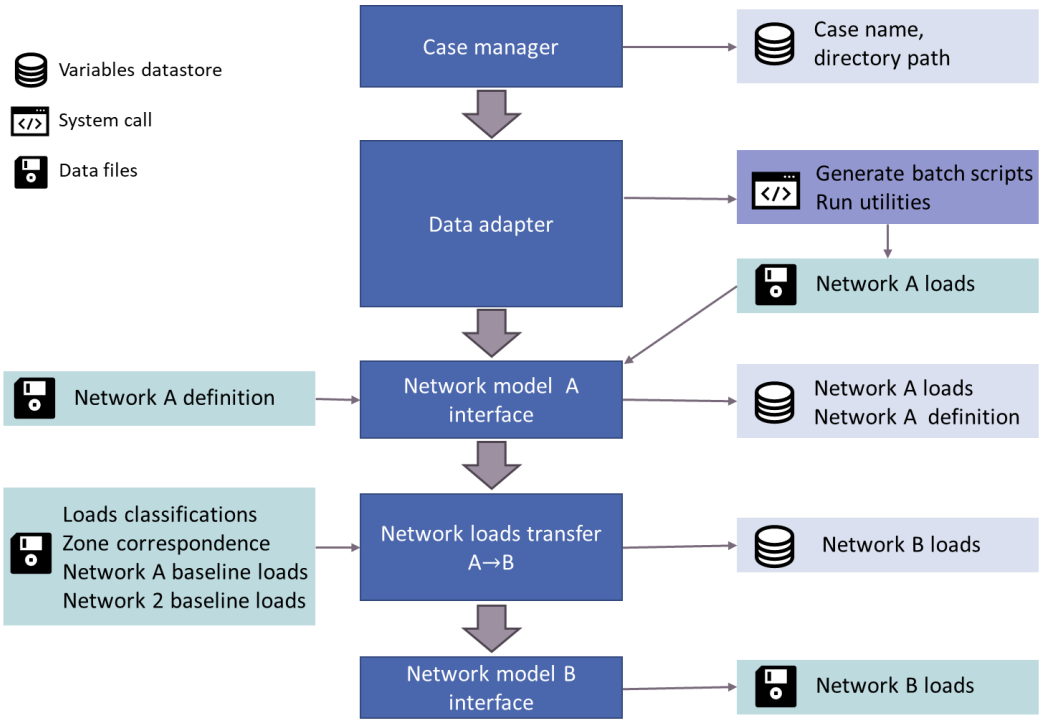

Figure 4. Interactions between variables datastore, external applications and data files.

5 CONCLUSION

A methodology for the centralised handling of data dependencies has been presented that enables integrated software tools to be developed organically without upfront knowledge of data structures. The datastore is a core part of the Teclab software package, created to integrate a variety of numerical algorithms, data files and external software packages. A variety of data types can be incorporated, including real numbers, integers and character strings that can be represented as matrices or spatially distributed variables. Real variables can be defined as vectors, enabling a single variable name to represent a multi-dimensional vector field discretised over a computational grid of up to three dimensions. A distinction is made between input and output variables, ensuring a cohesive set of data dependencies between all domains within the system. Integrated systems can be structured to aggregate the results of multiple domains, giving developers the ability to combine the outputs of multiple analysis methods into a single input variable.

The road network analysis case study demonstrates the use of the centralised datastore for handling a variety of data types stored as matrix variables that are linked through the data adapter to a series of external data files and software packages.

6 REFERENCES

Moffatt, S., 2020. Framework for complex modelling ontologies. Axsym Engineering Publications, Teclab Paper 02: http://www.axsym-engineering.com/publications/Teclab02.pdf

7 DOCUMENT INFORMATION

Author: Stuart Moffatt, *BEng PhD*
Title: Handling data in automated analysis processes
Report Type: Axsym Engineering Publications
Series: Teclab
Paper: 01
Issue: 1
Date: 30-12-2019

DOCUMENT REFERENCE

Moffatt, S., 2019. Handling data in automated analysis processes. Axsym Engineering Publications, Teclab Paper 01: http://www.axsym-engineering.com/publications/Teclab01.pdf

CORRESPONDENCE

email: stuart.moffatt@axsym.co.uk

KEYWORDS

Data Management, Data Adapter, Analytical Process, Automation

FUNDING

This research was co-funded by Axsym Limited and Innovate UK (project 710722, title TecLab: intuitive and adaptable visual drag and drop analytical simulation environment, end date 31-10-2016) to prove the concept of an innovative computational framework for automating data intensive simulation processes.