Project-Graph-Design

Reducing the human burden of linking essential programme components together, to create a real-time decision support capability

Russell Plummer

2023-04-15

Table of contents

List of Figures

1 Project Graph

1.1 Intent / Purpose

1.1.1 Background - Programme Parts

A large and complex programme can have many moving parts which are essential to delivery of the whole, but these parts are represented by differing sets of artefacts and internal controls, each of which can be changing. A relatively simple example, based on some of the components of DEFRA's Farming & Countryside Programme is shown below in @ProgComp

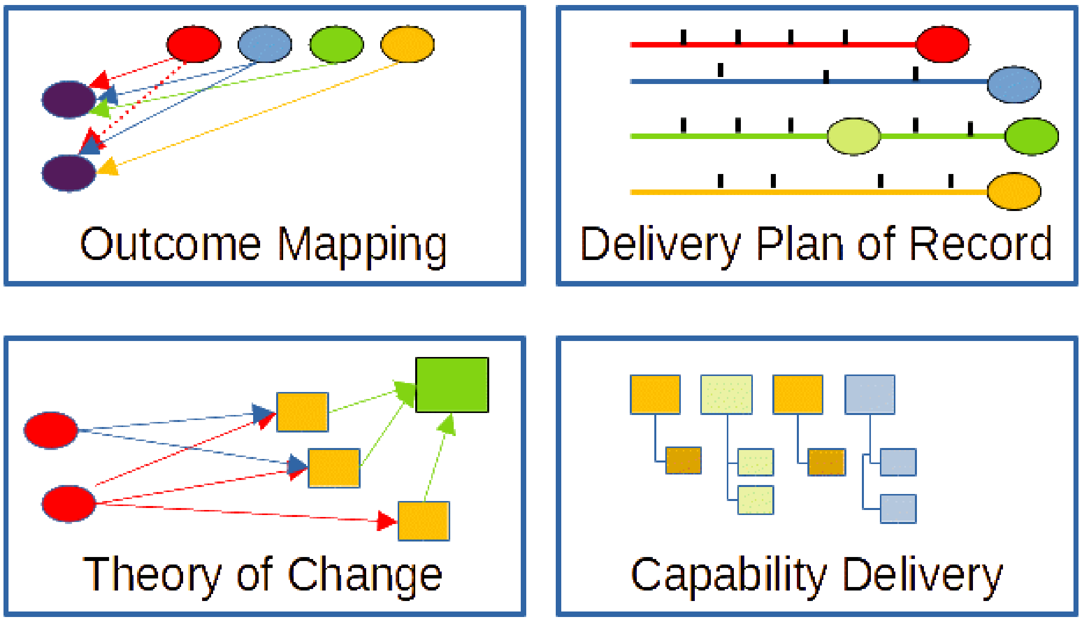


Figure 1: ProgComp

Taken in isolation, each part has it's own set of rules, and ways that progress can be evaluated or measured

Understand impacts - get an end-end view, but cut out all of the noise. Today it is done by people knowing

Be able to model impacts

1.1.2 Co-ordination across the Programme

However, these different parts are actually intrinsically linked, and the co-ordination

1.1.3 Software Aims - Design Principles

- To be as configurable as possible
- To be usable by anyone without needing to have software developers involved at every step

Built using smaller modules

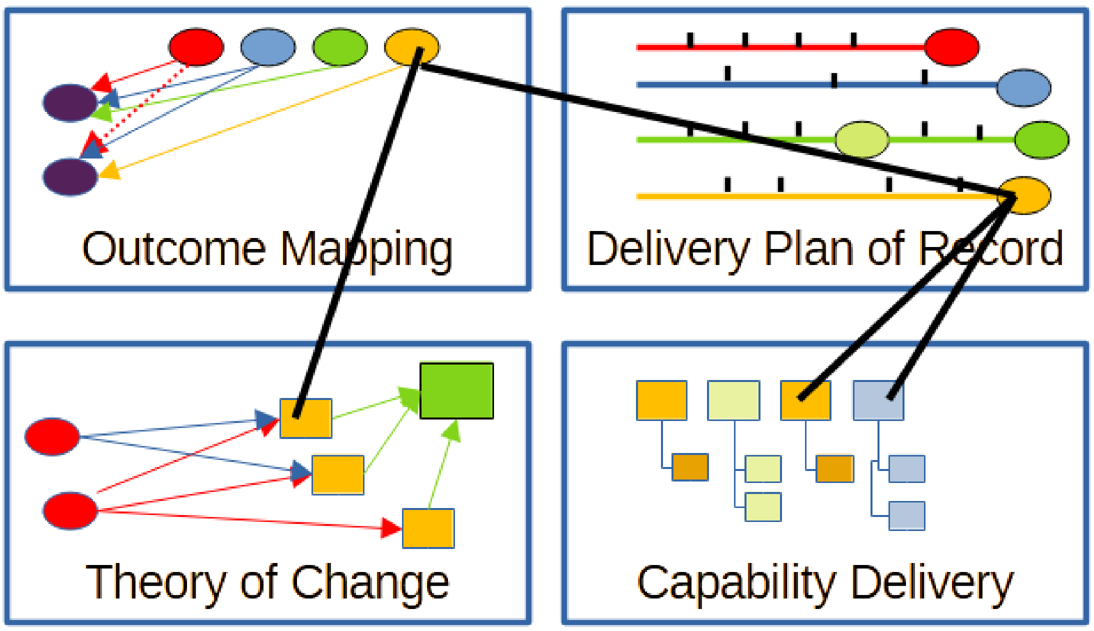
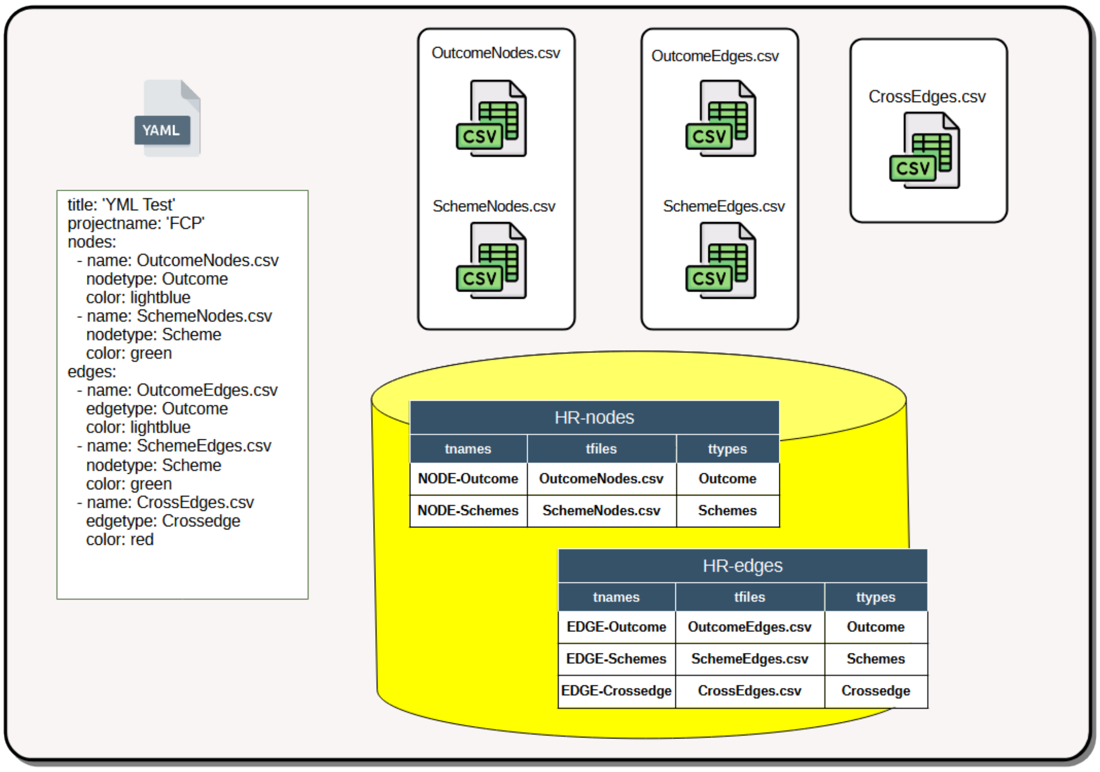


Figure 2: Project link data overlay

1.2 Data Sets

1.2.1 Table names for the master data.

HR = Human readable, G = Graph



```
title: 'YML Test'
projectname: 'FCP'
nodes:
  - name: OutcomeNodes.csv
    nodetype: Outcome
    color: lightblue
  - name: SchemeNodes.csv
    nodetype: Scheme
    color: green
edges:
  - name: OutcomeEdges.csv
    edgetype: Outcome
    color: lightblue
  - name: SchemeEdges.csv
    nodetype: Scheme
    color: green
  - name: CrossEdges.csv
    edgetype: Crossedge
    color: red
```

HR-nodes

tnames	tfiles	ttypes
NODE-Outcome	OutcomeNodes.csv	Outcome
NODE-Schemes	SchemeNodes.csv	Schemes

HR-edges

tnames	tfiles	ttypes
EDGE-Outcome	OutcomeEdges.csv	Outcome
EDGE-Schemes	SchemeEdges.csv	Schemes
EDGE-Crossedge	CrossEdges.csv	Crossedge

1.2.1.1 HR-nodes Table Format

Column	Type	Description
tnames	string	Name of a table in the database containing Human Readable data that has been loaded from a CSV format file. To assist readability, the table is named “NODE-*xxxxx*” where xxxxx is taken from *nodetype* in the yaml configuration file
tfiles	string	The filename of the CSV file uploaded to create the table in *tnames* This file is taken from the *name* field in the YAML file
ttypes	string	The type of the data in node, taken from *nodetype* in the yaml configuration file This column is used to define Groups in each set of Edges in the network

1.2.1.2 HR-edges Table Format

ColumnType		Description
tnames	string	Name of a table in the database containing Human Readable data that has been loaded from a CSV format file. To assist readability, the table is named "EDGE-*xxxxx*" where xxxxx is taken from *edgetype* in the yaml configuration file
tfiles	string	The filename of the CSV file uploaded to create the table in *tnames* This file is taken from the *name* field in the YAML file
ttypes	string	The type of the data in node, taken from *edgetype* in the yaml configuration file This column is used to define Groups in each set of Edges in the network

# graph_nodes: id, label, title, shape, group

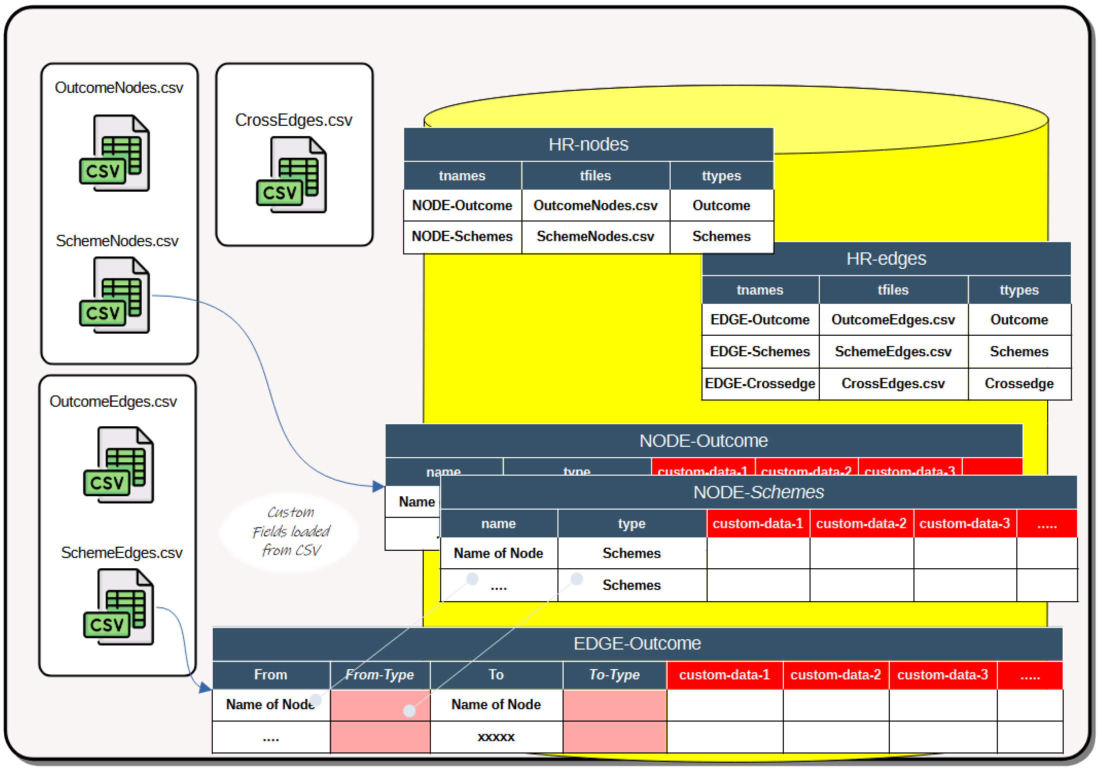
1.2.1.3 G-nodes Table Format

Column	Type	Description
id	integer	Node ID - passed to the graph
label	string	Label for the node - shown by visNetwork when rendering the graph
title	string	Title for the node - shown by visNetwork when doing a mouseover
shape	string	Shape that the node will take
group	string	Group that the node belows to - used to set group attributes by visNetwork

1.2.1.4 G-edges Table Format

Column	Type	Description
id	integer	Edge id - passed to the graph
from	integer	Node id of start node
to	integer	Node id of end node
label	string	Label shown by visNetwork

1.2.2 HR Data Files



HR-nodes

tnames	tfiles	ttypes
NODE-Outcome	OutcomeNodes.csv	Outcome
NODE-Schemes	SchemeNodes.csv	Schemes

HR-edges

tnames	tfiles	ttypes
EDGE-Outcome	OutcomeEdges.csv	Outcome
EDGE-Schemes	SchemeEdges.csv	Schemes
EDGE-Crossedge	CrossEdges.csv	Crossedge

NODE-Outcome

name	type	custom-data-1	custom-data-2	custom-data-3
Name				

NODE-*Schemes*

name	type	custom-data-1	custom-data-2	custom-data-3	.....
Name of Node	Schemes				
....	Schemes				

EDGE-Outcome

From	*From-Type*	To	*To-Type*	custom-data-1	custom-data-2	custom-data-3	.....
Name of Node		Name of Node					
....		xxxxx					

node table = 'NODE-'{goup}

# graph_edges: id, from {node id from grpah nodes}, to

# edge table = 'EDGE-'{group}

# cross_group_edges: from {node_lable}, from_node_group, to {node_lable}, to_node_group