UNICOM® Systems, Inc.
A Division of UNICOM Global

Practical Application of System Architect OV-3 and SV-6 Import Methodology

By Larry McCaskill
Enterprise Architecture SME and CEO,
Sorcerer Staffing

Publication information
March 2018
Information in this publication is subject to change. Changes will be published in new editions.

Contents

About the Author

Lawrence P. McCaskill

Enterprise Architecture SME and CEO
Sorcerer Staffing
lawrence.mccaskill@gmail.com

Lawrence McCaskill independent consultant and owner of a certified Service Disabled Veteran Owned Small Business (SDVOSB). He has been working with UNICOM® System Architect for over 15 years, and is a recognized expert in Enterprise Architecture theory and implementation using the DoD Architecture Framework (DoDAF). He has overseen creation of over 100 architecture efforts supporting the Joint Capabilities Integration and Development System (JCIDS) and Information Support Plan (ISP) processes, including: F-22A, B-2, MV/CV-22, E-6, E-2C/D, Joint Light Tactical Vehicle, Predator/Reaper UAV, and many more. In 2011 he was awarded the DoD Enterprise Architecture Achievement Award in the Industry Individual category, and was hand-picked by the Joint Staff/J-6 to represent the architecture practitioner's perspective in the Net-Ready Key Performance Parameter working group, leading to the rewrite of Chairman of the Joint Chiefs of Staff Instruction 6212. He was an active participant developing the Activity Based Methodology (ABM) for DoDAF, and has developed and provided instruction in tool-neutral DoDAF and DoDAF2 courses.

Abstract

When I read and tried to implement the methodology explained in the whitepaper *How to Share DoDAF2 Data with System Architect*, I found that with real-world data, it didn't work. The methodology was sound, but didn't take into account that imported names of artifacts could only be 80 characters long; artifact names were built by concatenating artifact names from the OV-3 – with up to 5 items being concatenated, import artifact names "collided" when uploading spreadsheets as the article instructed. This paper discusses a methodology to prevent that in 99% of the OV-3 entries. Additionally, it discusses the DoDAF 2.0 theory behind "how things fit together," so the reader develops an understanding of what is being built via the file imports.

Introduction

This article describes my discoveries as a result of operationally exercising the methodology demonstrated in the article *How to Share DoDAF 2 Data with System Architect* written by Chuck Faris.

There's definitely a lot of good information provided in that article, but it assumes the reader knows a lot about UNICOM® System Architect® DoDAF2 implementation internals. My assertion is most people using the tool don't. Most people need some sort of tutorial to get them to the point where they're ready to actually use the spreadsheets and methodology Mr. Faris provided. So… in effect, this is the prequel, and retelling of the story from the Operational View perspective (where the article written by Chuck Faris demonstrated the Systems Views).

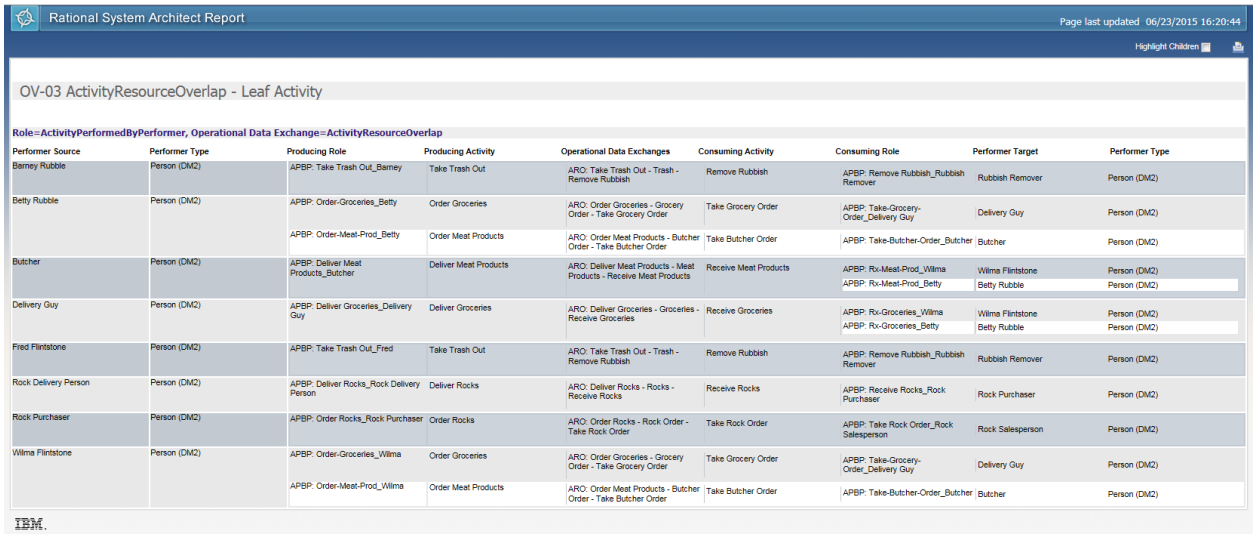
Understanding OV-3 Architecture Primitives

Chuck Faris's paper assumes one is intimately familiar with the format of the OV-3 that System Architect produces after all the architecture primitives are populated in the tool (with the OV-3 being a report on what has been modeled in the tool, detailing what Organizations communicate as a result of the assignment of Activities to the Organization [or Person, who, by extension, is assigned to an Organization]). This OV-3 format generated in the report that one is taught to use during the UNICOM® DoDAF2 user training is a bit different than many are used to seeing. Assuming you have an encyclopedia that has all the Operational Viewpoint primitives populated (i.e., have followed all the steps for populating the OV-5a/b, OV-4, and OV-2 in the System Architect DoDAF2 training material), getting to the report isn't as intuitive as they've made it for the SV's (which are under Reports > DoDAF2 Reports).

To get to the report and run it from the System Architect menu toolbar:

1. Click **Reports** > **Report Generator**
2. Click **File** > **Open Report File**
3. Select **DAF2.rpt**
4. Click **OK**

Within this set of reports, there is a report name **OV-03 ActivityResourceOverlap – Leaf Activity**. The output of this report is similar to what is shown in "*Figure 1. OV-03 ActivityResourceOverlap – Leaf Activity*".



Rational System Architect Report

Page last updated 06/23/2015 16:20:44

Highlight Children

OV-03 ActivityResourceOverlap - Leaf Activity

Role=ActivityPerformedByPerformer, Operational Data Exchange=ActivityResourceOverlap

Performer Source	Performer Type	Producing Role	Producing Activity	Operational Data Exchanges	Consuming Activity	Consuming Role	Performer Target	Performer Type
Barney Rubble	Person (DM2)	APBP: Take Trash Out_Barney	Take Trash Out	ARO: Take Trash Out - Trash - Remove Rubbish	Remove Rubbish	APBP: Remove Rubbish_Rubbish Remover	Rubbish Remover	Person (DM2)
Betty Rubble	Person (DM2)	APBP: Order-Groceries_Betty	Order Groceries	ARO: Order Groceries - Grocery Order - Take Grocery Order	Take Grocery Order	APBP: Take-Grocery-Order_Delivery Guy	Delivery Guy	Person (DM2)
		APBP: Order-Meat-Prod_Betty	Order Meat Products	ARO: Order Meat Products - Butcher Order - Take Butcher Order	Take Butcher Order	APBP: Take-Butcher-Order_Butcher	Butcher	Person (DM2)
Butcher	Person (DM2)	APBP: Deliver Meat Products_Butcher	Deliver Meat Products	ARO: Deliver Meat Products - Meat Products - Receive Meat Products	Receive Meat Products	APBP: Rx-Meat-Prod_Wilma	Wilma Flintstone	Person (DM2)
						APBP: Rx-Meat-Prod_Betty	Betty Rubble	Person (DM2)
Delivery Guy	Person (DM2)	APBP: Deliver Groceries_Delivery Guy	Deliver Groceries	ARO: Deliver Groceries - Groceries - Receive Groceries	Receive Groceries	APBP: Rx-Groceries_Wilma	Wilma Flintstone	Person (DM2)
						APBP: Rx-Groceries_Betty	Betty Rubble	Person (DM2)
Fred Flintstone	Person (DM2)	APBP: Take Trash Out_Fred	Take Trash Out	ARO: Take Trash Out - Trash - Remove Rubbish	Remove Rubbish	APBP: Remove Rubbish_Rubbish Remover	Rubbish Remover	Person (DM2)
Rock Delivery Person	Person (DM2)	APBP: Deliver Rocks_Rock Delivery Person	Deliver Rocks	ARO: Deliver Rocks - Rocks - Receive Rocks	Receive Rocks	APBP: Receive Rocks_Rock Purchaser	Rock Purchaser	Person (DM2)
Rock Purchaser	Person (DM2)	APBP: Order Rocks_Rock Purchaser	Order Rocks	ARO: Order Rocks - Rock Order - Take Rock Order	Take Rock Order	APBP: Take Rock Order_Rock Salesperson	Rock Salesperson	Person (DM2)
Wilma Flintstone	Person (DM2)	APBP: Order-Groceries_Wilma	Order Groceries	ARO: Order Groceries - Grocery Order - Take Grocery Order	Take Grocery Order	APBP: Take-Grocery-Order_Delivery Guy	Delivery Guy	Person (DM2)
		APBP: Order-Meat-Prod_Wilma	Order Meat Products	ARO: Order Meat Products - Butcher Order - Take Butcher Order	Take Butcher Order	APBP: Take-Butcher-Order_Butcher	Butcher	Person (DM2)

IBM

Figure 1. OV-03 ActivityResourceOverlap – Leaf Activity

From left-to-right, here's what's on the report: Performer Name, Performer Type, Producing Role, Producing Activity, Operational Data Exchange, Consuming Activity, Consuming Role, Performer Target, and Performer Type. We'll examine each of these (**Note**: the encyclopedia example depicted within this paper is called "*Meet the Flintstones*" – it's unclassified, its workings are not proprietary, and no one but perhaps Hanna-Barbera can complain that I'm appropriating their data):

- **Performer Source:** The Performer performing the Activity. It can be any of the allowable Performer Types in DoDAF2, but is usually:
    - **Organization (DM2)**
    - **Person (DM2)**
    - **System (DM2)**
- **Performer Type:** Tells which definition type the Performer is. These are derived from the definition types within System Architect that are used for Performers. When you develop your OV-3 for the import files later (you will need to manually populate this field in your OV-3), expand and look under of **Definitions** within the **Explorer** pane, and observe the artifacts available within a populated encyclopedia (see "Figure 2. System Architect explorer window"):

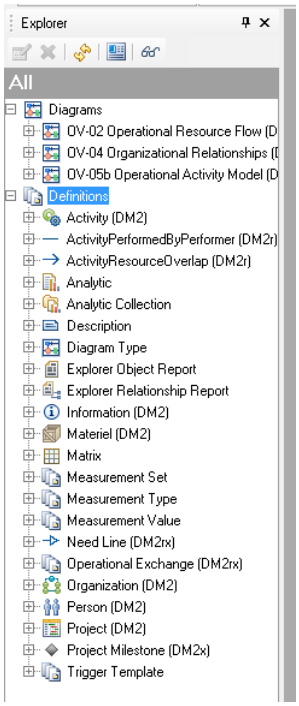


Figure 2. System Architect explorer window

In this case, the available Performers are **Person (DM2)** and **Organization (DM2)**. This example OV-3 in "*Figure 1. OV-03 ActivityResourceOverlap – Leaf Activity*" uses **Person (DM2)**. As previously stated, the other primitive that can be used in this case is **System (DM2)**, which would represent the case where an Activity is completely automated (i.e., not assigned to any human). This isn't "intuitively obvious" – why I'm showing you this is to enable you to figure out the text string defining the performer type that we'll use later to populate the spreadsheets for import.

- **Producing Role:** A relationship between Activity and Performer captured in the **ActivityPerformedByPerformer (DM2r)** definition. From a dogmatic perspective, I personally don't agree that the relationship "Activity performed by Performer" is a *Role* (that would be what the Role does, not what the name of the Role is... but I digress…), but this is the label the implementers of UNICOM® System Architect® have assigned to this field, so we're "stuck with it."
- **Producing Activity**: The **Activity (DM2)** producing the exchange.
- **Operational Data Exchanges:** Contain items of type **ActivityResourceOverlap (DM2)**, which is a compound definition. This definition captures the assertion that two Activities require a means to communicate, and has one or more Resources associated with the **pipe** between Activities, and is captured by the **ActivityResourceOverlap (DM2r)** definition in System Architect. Salient sidebar about Resources and the ActivityResourceOverlap (ARO):
    - The "…one or more Resources associated with the **pipe** between Activities…" statement is key: the UNICOM® System Architect® implementation of DoDAF2 treats the line between Activities as a *pipe*, in that one or more Resources can be exchanged over the *pipe* (i.e., the ARO). In its most elegant implementation, there would only be one unidirectional pipe between any two Activities, with one-to-many resources traveling over the *pipe* – this paradigm drives the capture of the need to communicate to communicate down a level from that in

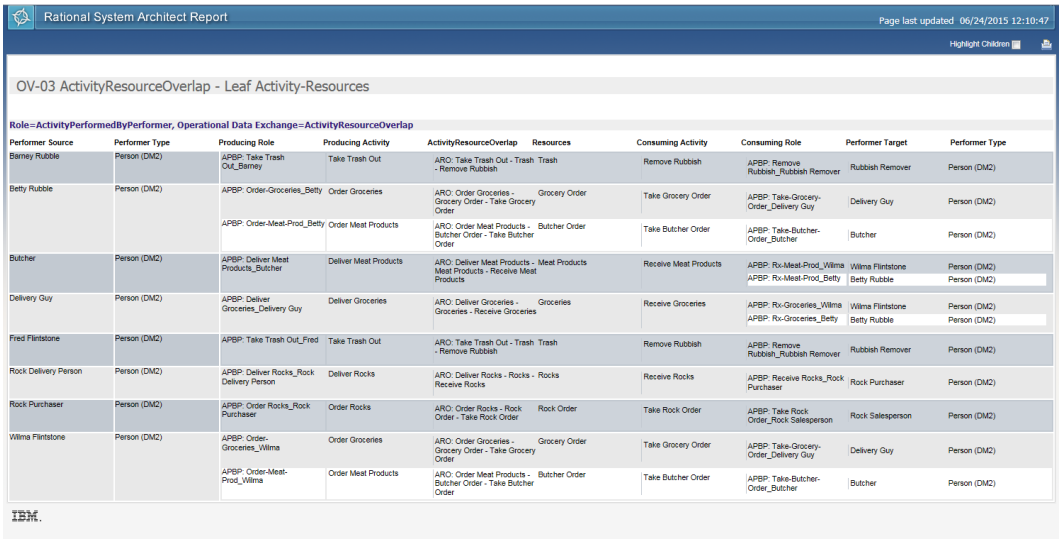
DoDAF 1.5, where the requirement to communicate was left at the level of the OV-2. It expressed a need for Roles or Organizations to communicate with one another, based on the information being exchanged by the Activities assigned to the Role or Organization (hence the term *Needline* – it depicts a communications need between 2 entities). Here, the ARO captures the need to communicate between specific Activities.

- However, the implementation allows one to have <u>many</u> of these unidirectional pipes going between Activities. This is how UNICOM® training documentation instructs users to implement Resources: one Resource per pipe. This enables the resultant diagrams to appear to be similar to IDEF0 (the modeling notation most people used to create OV-5's in DoDAF 1.5). However, because the ARO is just a pipe, it isn't *the <u>thing being communicated</u>*. What is being communicated is a Resource. The Resource is the focus of *<u>who</u> needs to communicate <u>what</u> to <u>whom</u>*.
- To figure out what was being produced in the OV-3 report, when I created the underlying definitions for **ActivityPerformedByPerformer (DM2r)** and **ActivityResourceOverlap (DM2r)**, I used abbreviations within the definitions to tell me what was being produced in the reports and embedded these in the names of the artifacts being captured/produced (e.g., the definitions were preceded by "`APBP:` " for "ActivityPerformedByPerformer" or "`ARO:` " for "ActivityResourceOverlap." Otherwise, I would have assumed Operational Data Exchanges in the report is the same thing as the definition **Operational Exchange (DM2)** shown in the explorer pane – unfortunately, it's not the same thing. Here, what's labeled **Operational Data Exchanges** is actually **ActivityResourceOverlap (DM2r)**, as indicated by the report generating "ARO: …" for every line.
- Enhancement Request: I've asked that UNICOM® provide a different OV-3, because this report is insufficient for most DoDAFians' purposes in that it doesn't provide Resource-level information, which should be the entire focus of the OV-3. Stated differently: typical OV-3's are one Resource per line. What's in this report is one ARO per line (which tells one that there is a need to communicate, but not *<u>what</u> is being communicated*). I've provided UNICOM® the Report Generator code to get Resource into the report (vs. **ActivityResourceOverlap (DM2r)** which is in the report). That said, getting the Resource Type into the report (e.g., **Information (DM2)** or **Materiel (DM2)**) and get the other fields (Consuming Activity, Consuming Role, Performer Target, Performer Type) to print as well has been a bridge too far for me. We'll need to manually populate these fields in our "for import" OV-3, which we will use later to build the artifacts to import and create models.
- In the meantime, "Figure 3. UNICOM® System Architect® Report Generator code to *output Resources*" is the Report Generator code that gets out Resource instead of ARO. See your SA guru (or… for a small price… I can help you do it… ;-)) to get this into your Report Generator as a new report.

```
REPORT "OV-03 ActivityResourceOverlap - Leaf Activity-Resources"
DESCRIPT "Role=ActivityPerformedByPerformer, Operational Data
Exchange=ActivityResourceOverlap"
ID 52510
{
FONT "Font4" { NAME "Courier" HEIGHT 12 }
SETTING { DECIMALSEPARATOR "." }
SETTING { LISTSEPARATOR "," }
SETTING { MEASUREMENT ENGLISH }
SETTING { PAGESIZE -1", 0.00 }
SETTING { HEADER 1 "OV-03 ActivityResourceOverlap" }
SETTING { REPORTFORMAT 4 }
SETTING { OUTPUTFILE "OV-03.htm" }
SETTING { STYLESHEETFILE "Reports\Stylesht\HTML Tables.xsl" }
               FIELD "Parent of Activities <-- activityParentOfActivity" { SOURCE
PROPERTY "activityParentOfActivity" LENGTH 1200 TYPE MEMO LEGEND "Parent of
Activities" }
               FIELD "Resources <-- Resources" { SOURCE PROPERTY "Resources" LENGTH
1200 TYPE MEMO  }
TABULAR 1
 {
       SELECT "Name" LEGEND "Performer Source" LEGENDFONT Font4
       WHERE Class = Definition
       WHERE "Type Number" = 1372, 1373, 1374, 1367, 1375, 1376, 1377   REM "Performer
Source"
       ORDERBY "Name"
       JOIN
       WHERE REFERENCEDBY = "Performer"
       JOIN
       SELECT "Name" LEGEND "Producing Role"
       WHERE Class = Definition
       WHERE "Type Number" = 1380              REM "ActivityPerformedByPerformer"
       JOIN
       WHERE REFERENCES = "Activity"
       JOIN
       SELECT "Name" LEGEND "Producing Activity"
       WHERE Class = Definition
       WHERE "Type Number" = 1326          REM "Producing Activity"
       WHERE "Parent of Activities <-- activityParentOfActivity" = ""
       JOIN
       WHERE REFERENCEDBY = "producingActivity"
       JOIN
       SELECT "Name" LEGEND "ActivityResourceOverlap", "Resources <-- Resources"LEGEND
"Resources"
       WHERE Class = Definition
       WHERE "Type Number" = 1383          REM "ActivityResourceOverlap"
       JOIN
       WHERE REFERENCES = "consumingActivity"
       JOIN
       SELECT "Name" LEGEND "Consuming Activity"
       WHERE Class = Definition
       WHERE "Type Number" = 1326          REM "Consuming Activity"
       WHERE "Parent of Activities <-- activityParentOfActivity" = "" REM
"activityParentofActivity blank = Leaf Activity"
       JOIN
       WHERE REFERENCEDBY = "Activity"
       JOIN
       SELECT "Name" LEGEND "Consuming Role"
       WHERE Class = Definition
       WHERE "Type Number" = 1380              REM "ActivityPerformedByPerformer"
       JOIN
ES = "Performer"
       JOIN
       SELECT "Name" LEGEND "Performer Target" LEGENDFONT Font4, "Type" LEGEND
"Performer Type"
       WHERE Class = Definition
       WHERE "Type Number" = 1372, 1373, 1374, 1367, 1375, 1376, 1377             REM
"Performer Target"
  }
}
```

Figure 3. UNICOM® System Architect® Report Generator code to output Resources

The output of this report is shown in Figure 4. OV-3 with Resources.



Rational System Architect Report

Page last updated 06/24/2015 12:10:47

Highlight Children

OV-03 ActivityResourceOverlap - Leaf Activity-Resources

Role=ActivityPerformedByPerformer, Operational Data Exchange=ActivityResourceOverlap

Performer Source	Performer Type	Producing Role	Producing Activity	ActivityResourceOverlap	Resources	Consuming Activity	Consuming Role	Performer Target	Performer Type
Barney Rubble	Person (DM2)	APBP: Take Trash Out_Barney	Take Trash Out	ARO: Take Trash Out - Trash - Remove Rubbish	Trash	Remove Rubbish	APBP: Remove Rubbish_Rubbish Remover	Rubbish Remover	Person (DM2)
Betty Rubble	Person (DM2)	APBP: Order-Groceries_Betty	Order Groceries	ARO: Order Groceries - Grocery Order - Take Grocery Order	Grocery Order	Take Grocery Order	APBP: Take-Grocery-Order_Delivery Guy	Delivery Guy	Person (DM2)
		APBP: Order-Meat-Prod_Betty	Order Meat Products	ARO: Order Meat Products - Butcher Order - Take Butcher Order	Butcher Order	Take Butcher Order	APBP: Take-Butcher-Order_Butcher	Butcher	Person (DM2)
Butcher	Person (DM2)	APBP: Deliver Meat Products_Butcher	Deliver Meat Products	ARO: Deliver Meat Products - Meat Products - Receive Meat Products	Meat Products	Receive Meat Products	APBP: Rx-Meat-Prod_Wilma	Wilma Flintstone	Person (DM2)
							APBP: Rx-Meat-Prod_Betty	Betty Rubble	Person (DM2)
Delivery Guy	Person (DM2)	APBP: Deliver Groceries_Delivery Guy	Deliver Groceries	ARO: Deliver Groceries - Groceries - Receive Groceries	Groceries	Receive Groceries	APBP: Rx-Groceries_Wilma	Wilma Flintstone	Person (DM2)
							APBP: Rx-Groceries_Betty	Betty Rubble	Person (DM2)
Fred Flintstone	Person (DM2)	APBP: Take Trash Out_Fred	Take Trash Out	ARO: Take Trash Out - Trash - Remove Rubbish	Trash	Remove Rubbish	APBP: Remove Rubbish_Rubbish Remover	Rubbish Remover	Person (DM2)
Rock Delivery Person	Person (DM2)	APBP: Deliver Rocks_Rock Delivery Person	Deliver Rocks	ARO: Deliver Rocks - Rocks - Receive Rocks	Rocks	Receive Rocks	APBP: Receive Rocks_Rock Purchaser	Rock Purchaser	Person (DM2)
Rock Purchaser	Person (DM2)	APBP: Order Rocks_Rock Purchaser	Order Rocks	ARO: Order Rocks - Rock Order - Take Rock Order	Rock Order	Take Rock Order	APBP: Take Rock Order_Rock Salesperson	Rock Salesperson	Person (DM2)
Wilma Flintstone	Person (DM2)	APBP: Order-Groceries_Wilma	Order Groceries	ARO: Order Groceries - Grocery Order - Take Grocery Order	Grocery Order	Take Grocery Order	APBP: Take-Grocery-Order_Delivery Guy	Delivery Guy	Person (DM2)
		APBP: Order-Meat-Prod_Wilma	Order Meat Products	ARO: Order Meat Products - Butcher Order - Take Butcher Order	Butcher Order	Take Butcher Order	APBP: Take-Butcher-Order_Butcher	Butcher	Person (DM2)

IBM.

Figure 4. OV-3 with Resources

- While on the topic of Resources… even though you can import **Resource (DM2)** into System Architect, it is not usable within any of the diagrams. This is because it is a supertype and is not included on any diagram. So, you'll need to import the Resource Type appropriate to the Resource exchanged. This is usually of type **Information (DM2)** or **Materiel (DM2).** This is a field in the import spreadsheets, so it's a good thing to remember.

- **Consuming Activity:** The Activity (DM2) consuming the exchange.
- **Consuming Role:** Consuming ActivityPerformedByPerfromer (DM2r)
- **Performer Target:** Consuming Performer (see Performer Source above for allowable types)
- **Performer Type:** Type of Performer (see Performer Type above).

Knowing how UNICOM® System Architect® views the primitives of the OV-3, there are some key relationships between artifacts that you need to understand before proceeding.

For the Operational Viewpoint, UNICOM® System Architect® implements the primitives via relationship definitions (these are denoted with **bolded text** below) that allow you to build an OV-3. In this exercise, you'll import the artifacts piecewise to reverse-engineer the primitives of the OV-3 to build the artifacts within this model:

- **Need Line (DM2rx)**
  - Has Source/Target that are Performers usually of type:
    - **Person (DM2)**
    - **Organization (DM2)**
    - **System (DM2)**
  - Contains: **Operational Exchange (DM2rx)**
    - Has Source/Target: **ActivityPerformedByPerformer (DM2r)**
    - Contains: **ActivityResourceOverlap (DM2r),** and multiples between same activities, although redundant, are allowed

- Has Producing Activity/Consuming Activity = **Activity (DM2)**
- Contains Resources – typical types are:
  - **Information (DM2)**
  - **Materiel (DM2)**

For the Systems Viewpoint, the primitives and relationship definitions mirror the Operational Viewpoint's, with some name changes as follows:

- **System Resource Flow (DM2rx)**
  - Has Source/Target that are Performers:
    - **Person (DM2)**
    - **Organization (DM2)**
    - **System (DM2)**
  - Contains: **System Exchange (DM2rx)**
    - Has Source/Target that are **ActivityPerformedByPerformer (DM2r)** (where **Activity (DM2)** is replaced by **System Function (DM2x)** – System Function is a subtype of Activity – System Function is "that which one wishes to automate")
    - Contains: **System Data Flow (DM2rx),** the Systems Viewpoint analog to ActivityResourceOverlap
      - Has Source/Target that are System Function (DM2x)
      - Contains Resources – allowable types:
        - **Data (DM2)** primarily…, but also allowable are
        - **Information (DM2)**
        - **Materiel (DM2)**

Assumptions / Constraints:

I would be remiss as an "astute archetype" if I didn't also mention assumptions and constraints (these are my AV-2ish declarations) I had regarding this exercise:

- Regarding OV-3:
  - In a *normal* OV-3, the report has one line for each of the following "sideways V" relationships:
    - **Organization** *performing*
      - **Activity** *produces*
        - **Resource** *consumed by*
    - **Activity** *performed by*
    - **Organization**

  - ***IF*** the OV-3 was produced using the Activity Based Methodology (ABM - which I am a huge proponent of), the OV-3 has:
    - **Organization** *contains*
      - **Role** (one or more of which can be assigned to a Person) *performing*
        - **Activity** *produces*
          - **Resource** *consumed by*
        - **Activity** *performed by*
      - **Role** *assigned to*
    - **Organization**. The only reason you might not know what the Role is within the Organization producing/consuming the Resource is in the

case of an External Organization or Service providing/consuming whatever the Resource is, where you might not have that detail. Within this exercise, you have the detail of the Role (Person) in both the internal and external parts of the model, and thus, can utilize an **ABM-ish** paradigm for creating the resulting models.

- For Solution Architectures, the OV-3 is only required to report on the exchanges that occur between internal Activities and external Organizations' Activities associated with the solution. Therefore, most OV-3's ***do not*** include internal crosstalk; stated differently, Resources that flow between Activities internal to the model are generally not captured in the OV-3 or SV-6. They'll have to be created afterwards when you build the Activity model. Therefore, you'll need to have some a priori knowledge of what the activity tree looks like. If you don't all you'll be able to create is "everything on one page" for OV-5b Activity Model, and your OV-5a will be a "flat tree."
- Key primitives that need to be created (and be defined, unless it's of type "relationship"), and the general order in which they are created using this methodology:
  - **Information (DM2)** (primitive – definition needed)
  - **Materiel (DM2)** (primitive)
  - **Activity (DM2)** (primitive)
  - **ActivityResourceOverlap (DM2r)** (relationship – no definition needed)
  - **Person (DM2)** (primitive)
  - **Organization (DM2)** (primitive)
  - **ActivityPerformedByPerformer (DM2r)** (relationship)
  - **Operational Exchange (DM2rx)** (relationship)
  - **Need Line (DM2rx)** (relationship)

Meet the Flinstones

The encyclopedia I'm using as an example to demonstrate the methodology is called *Meet the Flintstones*, it is a Fred-and-Barney centric model, with these Organizations:

- Internal:
  - Casa de Flintstone
  - Casa de Rubble
  - Slate Rock and Gravel Company
  - Loyal Order of the Water Buffaloes
- External:
  - Bedrock Butcher
  - Bedrock Trash Company
  - Safestone's Grocery
  - Rock Vendor

Meet the Flintstones has some oddities:

- It has multiple Performers doing the same thing (for example, Betty and Wilma both order groceries, and Barney and Fred each take out the trash)
- It has the same Performers in multiple Organizations (Barney and Fred are part of: Slate Rock and Gravel Company; Casa de Flintstone and Casa de Rubble, respectively; and the Loyal Order of the Water Buffaloes).

- *Usually* for Enterprise Architectures, one models a person's Role as **Person (DM2)** instead of the instance (person's name…). However, after creating the model, I realized I did this for most of the internal "people" in the architecture (Fred, Wilma, Barney, Betty, etc.). Elaborating on this using Betty Rubble as an example: I modeled Betty Rubble (a Person *instance*) as **Person (DM2)**, but I did not model her implied Role *type* of Grocery Purchaser. However, since I had no *instance* information for Butcher or Delivery Guy (as in: their names), I modeled these Role *types* as **Person (DM2)**. When this is reported on in the OV-3, Betty Rubble (*instance* of **Person (DM2)**) has **Operational Resource Exchange (DM2rx)** of "___ Delivery Order" with Delivery Guy/Butcher (which are Role *types* modeled as **Person (DM2)**. It's a small nuance, but endgame lesson learned: be careful AND consistent how you model.
- Multiple types of Performers can communicate with non-same-type performers:
    - Organization communicates with Organization
    - Organization communicates with Person
    - Etc.

The point of Meet the Flintstones is to have the encyclopedia put the methodology through its paces – oddball things like this ARE REAL, and exist as models in many places.

- An OV-4 is pre-created in the tool before importing the spreadsheets we will build using the OV-3 as a starting point. Thus, Organizations and Performers are pre-known. "Figure 5. OV-4: Meet the Flintstones" shows the OV-4 for Meet the Flintstones. I will provide XML for this OV-4 with the example spreadsheets that accompany this White Paper.

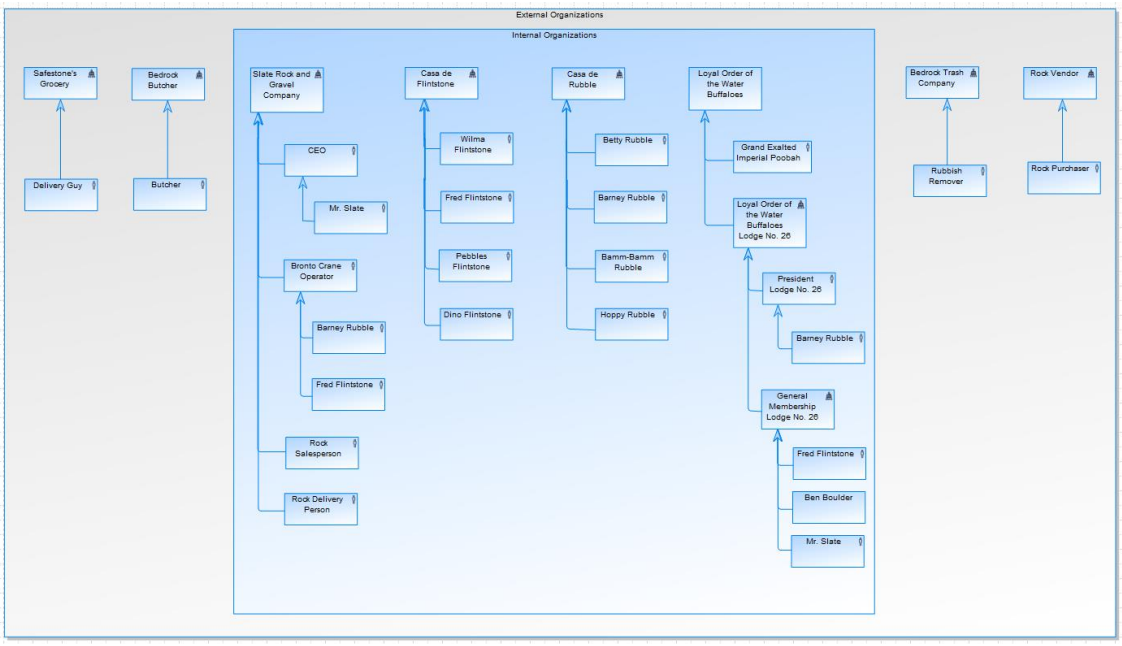


Figure 5. OV-4: Meet the Flintstones

Modifying the System Viewpoint Spreadsheets to Operational Viewpoint Spreadsheets

The spreadsheets provided with the *How to Share DoDAF2 Data with System Architect* paper were created for importing SV-6 artifacts. These need to modified for use with OV-3 artifacts. In the paper, the OV-3 is on the left side of each of the import spreadsheets, and the data provided with the OV-3 is used to build the individual primitives and relationship definitions used to import into System Architect (via csv import) on the right of each of the spreadsheets.

For this paper, the “common” Meet the Flintstones OV-3 is used across all 4 import spreadsheets. This is shown in “Figure 6. OV-3 used to build import files.”

	A	B	C	D	E	F	G	H	I	J	K
1	Performer Source	Performer Type	Producing Role	Producing Activity	Activity Resource Overlap	Resource	Resource Type	Consuming Activity	Consuming Role	Performer Target	Performer Type
2	Wilma Flintstone	Person (DM2)	APBP: Order Groceries_Wilma Flintstone	Order Groceries	ARO: Order Groceries_Grocery Order_Take Grocery Order	Grocery Order	Information (DM2)	Take Grocery Order	APBP: Take Grocery Order_Delivery Guy	Delivery Guy	Person (DM2)
3	Betty Rubble	Person (DM2)	APBP: Order Groceries_Betty Rubble	Order Groceries	ARO: Order Groceries_Grocery Order_Take Grocery Order	Grocery Order	Information (DM2)	Take Grocery Order	APBP: Take Grocery Order_Delivery Guy	Delivery Guy	Person (DM2)
4	Wilma Flintstone	Person (DM2)	APBP: Order Meat Products_Wilma Flintstone	Order Meat Products	ARO: Order Meat Products_Butcher Order_Take Butcher Order	Butcher Order	Information (DM2)	Take Butcher Order	APBP: Take Butcher Order_Butcher	Butcher	Person (DM2)
5	Betty Rubble	Person (DM2)	APBP: Order Meat Products_Betty Rubble	Order Meat Products	ARO: Order Meat Products_Butcher Order_Take Butcher Order	Butcher Order	Information (DM2)	Take Butcher Order	APBP: Take Butcher Order_Butcher	Butcher	Person (DM2)
6	Delivery Guy	Person (DM2)	APBP: Deliver Groceries_Delivery Guy	Deliver Groceries	ARO: Deliver Groceries_Groceries_Receive Groceries	Groceries	Materiel (DM2)	Receive Groceries	APBP: Receive Groceries_Wilma Flintstone	Wilma Flintstone	Person (DM2)
7	Delivery Guy	Person (DM2)	APBP: Deliver Groceries_Delivery Guy	Deliver Groceries	ARO: Deliver Groceries_Groceries_Receive Groceries	Groceries	Materiel (DM2)	Receive Groceries	APBP: Receive Groceries_Betty Rubble	Betty Rubble	Person (DM2)
8	Butcher	Person (DM2)	APBP: Deliver Meat Products_Butcher	Deliver Meat Products	ARO: Deliver Meat Products_Meat Products_Receive Meat Products	Meat Products	Materiel (DM2)	Receive Meat Products	APBP: Receive Meat Products_Wilma Flintstone	Wilma Flintstone	Person (DM2)
9	Butcher	Person (DM2)	APBP: Deliver Meat Products_Butcher	Deliver Meat Products	ARO: Deliver Meat Products_Meat Products_Receive Meat Products	Meat Products	Materiel (DM2)	Receive Meat Products	APBP: Receive Meat Products_Betty Rubble	Betty Rubble	Person (DM2)
10	Fred Flintstone	Person (DM2)	APBP: Take Trash Out_Fred Flintstone	Take Trash Out	ARO: Take Trash Out_Trash_Remove Rubbish	Trash	Materiel (DM2)	Remove Rubbish	APBP: Remove Rubbish_Rubbish Remover	Rubbish Remover	Person (DM2)
11	Barney Rubble	Person (DM2)	APBP: Take Trash Out_Barney Rubble	Take Trash Out	ARO: Take Trash Out_Trash_Remove Rubbish	Trash	Materiel (DM2)	Remove Rubbish	APBP: Remove Rubbish_Rubbish Remover	Rubbish Remover	Person (DM2)
12	Rock Purchaser	Person (DM2)	APBP: Order Rocks_Rock Purchaser	Order Rocks	ARO: Order Rocks_Rock Order_Take Rock Order	Rock Order	Information (DM2)	Take Rock Order	APBP: Take Rock Order_Rock Salesperson	Rock Salesperson	Person (DM2)
13	Rock Delivery Person	Person (DM2)	APBP: Deliver Rocks_Rock Delivery Person	Deliver Rocks	ARO: Deliver Rocks_Rocks_Receive Rocks	Rocks	Materiel (DM2)	Receive Rocks	APBP: Receive Rocks_Rock Purchaser	Rock Purchaser	Person (DM2)
14	Grand Exalted Imperial Poobah	Person (DM2)	APBP: Provide Imperious and Insightful Guid_Grand Exalted Imperial Poobah	Provide Imperious and Insightful Guidance	ARO: Provide Imperious and In_Loyal Order of the Water _Receive Loyal Order of t	Loyal Order of the Water Buffaloes Laws, Policies, and Regulations	Information (DM2)	Receive Loyal Order of the Water Buffaloes Laws Policies and Regulations	APBP: Receive Loyal Order of the Water Buff_General Membership Lodge No. 26	General Membership Lodge No. 26	Organization (DM2)
15	President Lodge No. 26	Person (DM2)	APBP: Send Water Buffaloes Lodge Dues Overd_President Lodge No. 26	Send Water Buffaloes Lodge Dues Overdue Notice	ARO: Send Water Buffaloes Lod_Water Buffaloes Lodge Due_Receive Mail	Water Buffaloes Lodge Dues Overdue Notice	Information (DM2)	Receive Mail	APBP: Receive Mail_Fred Flintstone	Fred Flintstone	Person (DM2)
16	Fred Flintstone	Person (DM2)	APBP: Pay Water Buffaloes Lodge Dues_Fred Flintstone	Pay Water Buffaloes Lodge Dues	ARO: Pay Water Buffaloes Lodg_Water Buffaloes Lodge Due_Receive Mail	Water Buffaloes Lodge Dues	Materiel (DM2)	Receive Mail	APBP: Receive Mail_President Lodge No. 26	President Lodge No. 26	Person (DM2)

Figure 6. OV-3 used to build import files

This is definitely not what we’re used to seeing in a DoDAF 1.5 OV-3. However, this mirrors the OV-3 that would be produced by UNICOM® System Architect® in the DoDAF2 implementation, and includes all the artifacts you need to build the primitives and relationships in the subsequent 4 import spreadsheets discussed below. Caveat: *Some assembly required…* Details:

- **Primitives**: Assuming the OV-3 was created to match the DoDAF 1.5 specification for OV-3, the following primitives should be created in the model using the primitives from the OV-3 in “Figure 6. OV-3 used to build import files”: Performer Source, Producing Activity, Resource, Receiving Activity, and Performer Target.
- **Types**: Performer Type and Resource Type are definition types where System Architect stores the information related to the definitions. Each subsequent spreadsheet import provides more of the picture that System Architect uses to build primitives and relationships. For this example:
  - Performer Type is either **Person (DM2)** or **Organization (DM2)**
  - Resource Type is either **Information (DM2)** or **Materiel (DM2)**
- **Relationships**: Producing/Consuming Role (**ActivityPerformedByPerformer (DM2r)**) and the relationship that states “Activity produces Resource consumed by Activity” (**ActivityResourceOverlap (DM2r)**) are compound definitions built using Microsoft XL functions, including CONCATENATE, VLOOKUP (which were discussed in detail in *How to Share DoDAF2 Data with System Architect*), and LEFT. The LEFT function is new. It’s a key lesson learned from using this methodology in practice. The spreadsheets in the developerWorks article used the Microsoft Excel CONCATENATE function. This is perfectly fine if the Performer, Activity, and Resource Names are short. But, this was proved untenable with concantenated longer names. The UNICOM® System Architect® Definitions for **ActivityResourceOverlap (DM2r)**, **ActivityPerformedByPerformer (DM2r)**, **Operational Exchange (DM2rx)**, and **Need Line (DM2rx)** names are limited to 80 characters. As such, when I tried this with *real* data, the concatenated names were longer than 80 characters, and were chopped off to the first 80 characters on import. In

many cases, this caused the creation of duplicate names on attempt to import the data (especially in the case of **Operational Exchange (DM2rx)**, which concatenates 5 primitives). This caused the duplicate items not to import, regardless of the instance information, thereby losing data on import. To fix this problem, Microsoft Excel has several functions that let you manipulate text strings. The one I chose for this was LEFT, with the format of LEFT(Cell#, #characters-you-want-of-it). So, the format for the Activity Resource Overlap was:

```
=CONCATENATE("ARO: ",
LEFT(D2,24),"_",LEFT(F2,25),"_",LEFT(H2,24))
```

Which, in this case, resulted in concatenating:

- "ARO: "
- The left 24 characters of contents of the Producing Activity in cell D2 (Order Groceries)
- The left 25 characters of contents of the Resource in cell F2 (Grocery Order)
- The left 24 characters of contents of the Consuming Activity in cell H2 (Take Grocery Order)
- Resulting text (80 or less characters): ARO: Order Groceries_Grocery Order_Take Grocery Order

In this case, none of the LEFT commands affected the text, because they were less than the character count. However, if one looks at lines 14-16, the LEFT command definitely had an effect. This fixed about 99% of the import problems in my real-world data. The only duplications were in the case of two of my receiving Performers having similar names. In that case, the first line imported **won**, and I had to recreate the exchange for the 2nd line. So, on import of the data, trust, but verify.

- Specific column name changes within paper needed to make them work for OV-3 instead of
  SV-6 (depicted in the *How to Share DoDAF2 Data with System Architect* paper):
  - Producing Function becomes Producing Activity
  - Consuming Function becomes Consuming Activity
  - System Data Flow becomes Activity Resource Overlap
    Of note… I had a glitch with this one. I initially named it `ActivityResourceOverlap`, however, in the spreadsheet that creates Operational Exchanges, there is a right side field named `ActivityResourceOverlap` that System Architect uses to build the **Operational Exchange (DM2rx)**. On import, System Architect tried to use both ActivityResourceOverlap fields to build **ActivityResourceOverlap (DM2r)**'s, and the results weren't as expected – lesson learned: be careful with your naming convention.
- Columns added: The SV spreadsheets only used the System Data Flow, which is analogous to ARO. Because most OV-3's list one line per Resource, I added 2 columns: Resource, and Resource Type to the OV-3.

The four SV spreadsheets provided in conjunction with the original article also need tailoring to "Operationalize" them, and make them ready for use with OV-3. As part of this article, I will be providing the 4 import spreadsheets "already built" – however, I'm going to describe how I built them below. For each spreadsheet, I start with a short description of what the original SV

spreadsheet does, and then how to build the OV analog for each. Each spreadsheet needs to have the OV-3 data copied into the columns to the left of the **Name** column.

- The spreadsheet **From SV-06 - System Data Flow (SV-04).xlsx** is used to build **System Data Flow (DM2rx)**. It provides Source and Target System Functions, and a name for the System Data Flow.

  For OV-3, the analogous definition type to **System Data Flow (DM2rx)** is **ActivityResourceOverlap (DM2r)**. – Other changes are:
  - Sheet name changes from:
    - "From SV-06 - System Data Flow (SV-04).xlsx" to "From OV-03 – ActivityResourceOverlap (OV-05).xlsx"
    - Parenthesis tells one in which diagram type the artifacts being imported are used
  - Because ActivityResourceOverlap already exists in the OV-3, the Name field merely copies the name, where the SV version builds the name. Remember, the ARO: naming convention is used to tell what the items are in reports built from the data.
  - Column name changes:
    - Change "Source" to "Producing Activity"
    - Change "Target" to "Consuming Activity"
    - Change "**System Function (DM2x)"** to "**Activity (DM2)"** in the lookup fields for each. For example:
      `=CONCATENATE(VLOOKUP("Activity (DM2)",Lookup!$A$2:$B$32,2),"""",D2,"""")`
  - Resource Types: Add **Information (DM2)** and **Materiel (DM2)** to the lookup tab for use in the newly created Resources column for import. The function to build its contents is:
    `=CONCATENATE(VLOOKUP(G2,Lookup!$A$2:$B$32,2),"""",F2,"""")`

- From SV-06 - ActivityPerformedByPerformer (SV-04).xlsx: provides System Function performed by System assigned System or Organization, and names the ActivityPerformedByPerformer (APBP).

  For OV-3, the relationship is stored in the same definition (ActivityPerformedByPerformer (DM2r)).
  - Change the sheet name
    - "From SV-06 - ActivityPerformedByPerformer (SV-04).xlsx" to "From OV-03 - ActivityPerformedByPerformer (OV-05).xlsx"
  - In the **Activity** column, replace **System Function (DM2x)** with **Activity (DM2)** in all columns. **Activity (DM2)** is already in the Lookup tab so you don't need to create it.
  - This spreadsheet needs to account for both the Producing and Consuming Role. Therefore, it requires a set of rows for the Producing APBP, and a second set of rows for the Consuming APBP.
    Note: The column names for the Consuming set of Roles changes because of the addition of columns for Resource and Resource Type in OV-3.
    - **Example snapshot**: 1 header row and 15 rows of data produces 31 total rows of data after both the Producing APBP and Consuming APBP are accounted for. This is shown in "Figure 7. Activity performed by Performer Spreadsheet."

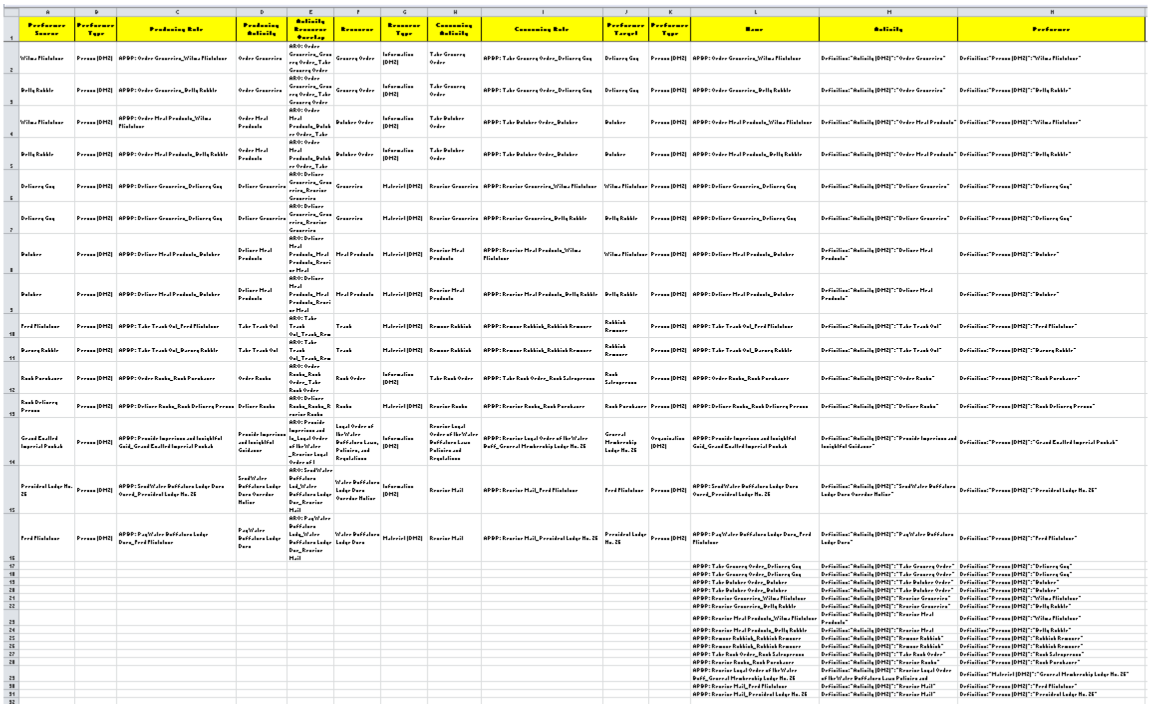


Figure 7. Activity performed by Performer Spreadsheet

- System Resource Flow (SV-01).xlsx: provides Source/Target System, and names the System Interface line going between them.

  For OV-3: The analogous definition for **System Resource Flow (DM2rx)** is **Need Line (DM2rx)**.
  - Change the sheet name from:
    "From SV-06 - System Exchange (SV-01z).xlsx" to
    "From OV-03 – Operational Exchange (OV-02z).xlsx"
  - The name column changes to accommodate the new Resource and Resource Type columns created in OV-3:
    `=CONCATENATE(LEFT(A2,36)," TO ",LEFT(J2,36))`
  - The Perfomer Target Lookup changes to accommodate new Resource and Resource Type columns:
    `=CONCATENATE(VLOOKUP(K2,Lookup!$A$2:$B$31,2),"""",J2,"""")`

- **System Exchange (SV-01z).xlsx**: Provides the Sending System/System Function-Resource-Receiving System/System Function relationship. It is an intermediate artifact created to automate the creation of SV-6 in SA.

  For OV-3: The analogous definition to System Exchange (DM2rx) is Operational Exchange (DM2rx).
  - The sheet name changes from:
    "From SV-06 - System Exchange (SV-01z).xlsx" to
    "From OV-03 – Operational Exchange (OV-02z).xlsx"
  - Name: Concatenates parts of the Performer Source, Producing Activity, Resource, Consuming Activity, and Performer Target:

```
=CONCATENATE("OPEX: ",
LEFT(A2,14),"_",LEFT(D2,14),"_",LEFT(F2,14),"_",LEF
T(H2,14),"_",LEFT(J2,14))
```

I prefer underscores to hyphens as a separator, as I use a lot of hyphens in artifact names.

- Source/Target: Ensure that the Source/Target point to the correct columns (Producing Role, Consuming Role)
- "Data Flow" column becomes "ActivityResourceOverlap" – build string becomes:

```
=CONCATENATE(VLOOKUP("ActivityResourceOverlap
(DM2r)",Lookup!$A$2:$B$32,2),""""",E2,""""")
```

Need to add "ActivityResourceOverlap (DM2r)" to lookup.

- "System Resource Flow" column becomes "NeedLine" - need to add "Need Line (DM2rx)" to Lookup, and change VLOOKUP to find the correct definition:

```
=CONCATENATE(VLOOKUP("Need Line
(DM2rx)",Lookup!$A$2:$B$30,2),""""",LEFT(A2,36)," TO
",LEFT(J2,36),""""")
```

With these 4 spreadsheets built, you'll need to export the first worksheet within each spreadsheet as a comma separated values (.csv) file type so that you can import it into UNICOM® System Architect®.

To do this within Microsoft Excel:

1. Click File > Save As >
2. Click the down-arrow on **Save as type**
3. Select **CSV (Comma delimited) (*.csv)**

It's time to start importing artifacts. Remember the notional order:

- **Information (DM2)** (primitive – definition needed for all primitives)
- **Materiel (DM2)** (primitive)
- **Activity (DM2)** (primitive)
- **ActivityResourceOverlap (DM2r)** (relationship – no definition needed)
- **Person (DM2)** (primitive)
- **Organization (DM2)** (primitive)
- **ActivityPerformedByPerformer (DM2r)** (relationship)
- **Need Line (DM2rx)** (relationship): Note: I changed the order of import of Operational Exchange and Need Line in practice - I found that only 1/3 of the Need Lines came through correctly if I didn't change the order. I have no idea why.
- **Operational Exchange (DM2rx)** (relationship)

For the items marked **primitive** above, you'll need to have separate csv sheets with a Name and a Definition for each item accounted for in the OV-3. Import these into UNICOM® System Architect® one at a time (Dictionary > Import Definitions, choose the file, choose the definition type, and under "Collision option:" choose "Update single fields when data supplied" – this lets you update individual definitions with relationship data stepwise).

After the imports, and creating the OV-4 as shown, you can go in and create an OV-2. There will be strange behaviors here, but nothing we can't overcome with "Stupid Human Tricks" in SA. SA tries very hard to keep what they call representational consistency intact. If we expand the Definitions in the Explorer pane, and drag-and-drop all entities in the **Organization (DM2)** and

Person (DM2) onto the screen, their relationships show up. However, if you recall, Barney and Fred are in 3 Organizations; you'll need to copy/paste multiples where applicable – the "contains" relationship in SA is equivalent to the lines drawn in the OV-4, so SA will gripe if it thinks you've got that relationship wrong. To see what SA thinks is wrong, Reports > Show Diagram Inconsistencies Report. It will tell you what it thinks is wrong on the right side.

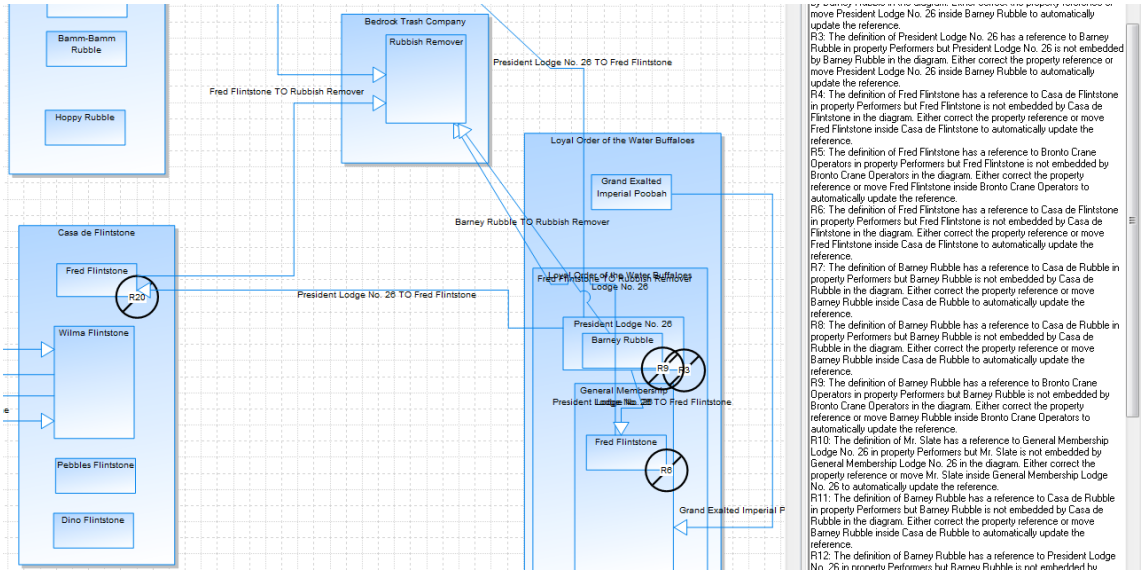


Figure 8 - UNICOM SA Inconsistency Report Output

In most cases, this can be corrected by moving the box out, and back into the container object (stupid human trick…). R3 is backwards – that's a bug in the Inconsistencies report – submitted to UNICOM® already. Nonetheless, after some Feng Shui'ing (Note: there's no Feng Shui button in SA…), you'll note there's duplicate lines from the 2 instances of Fred Flintstone and Barney Rubble to the Rubbish remover.

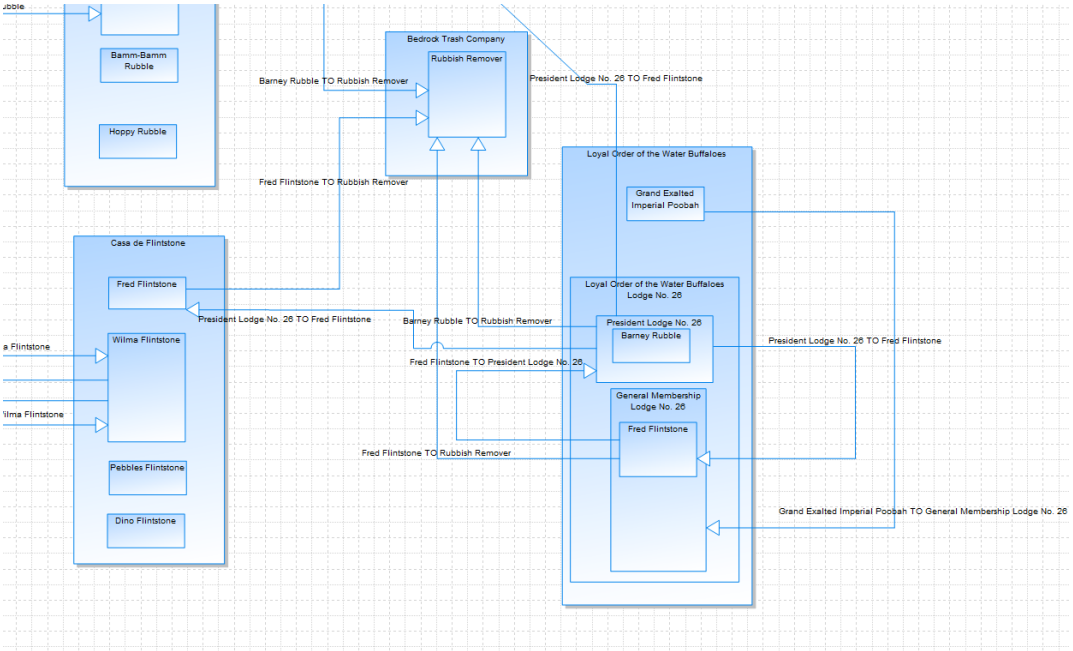


Figure 9 - "Clean" Diagram

These are only applicable to Fred and Barney when they're home (at Casa de Flintstone and Casa de Rubble, respectively). To "get rid of" the lines you don't want to show, select the lines, right-click, then select "Hide selected Need Line (DM2rx) Relationship Lines…"

In some cases, I've also run across instances where the Needline exists, all the sub-definitions are correct (see picture for these), but it doesn't render. In this case, draw the Needline, name it the same as the existing Needline in the definitions, and it will tell you it already exists, and ask whether you wish to use that definition – say Yes.

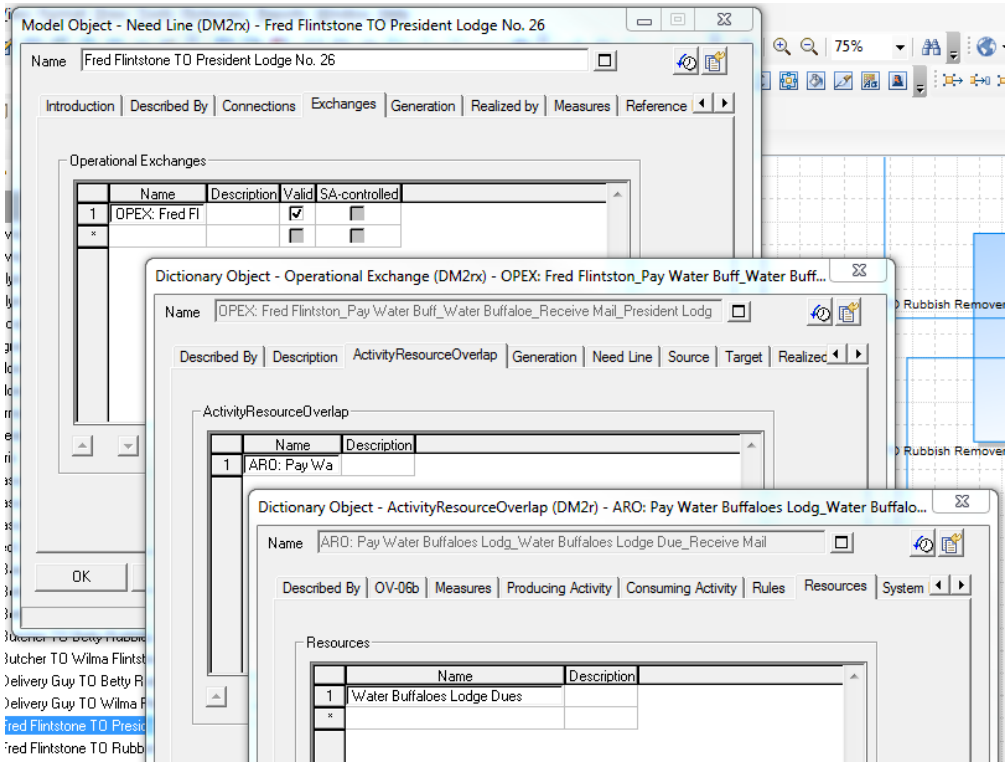


Figure 10 - Needline Sub-Definitions Needed to build OV-3 Line Item

Every time you breathe on the diagram, the diagram inconsistency indicators come up. Again, this is because Fred Flinstone and Barney Rubble are represented twice on the diagram – each symbol instance is producing a warning that it is not in all the boxes it's supposed to be in (the Fred Flinstone in Casa de Flinstone is not in the Loyal Orders of the Buffaloes Lodge; and the Fred there isn't in Casa de Flinstone, etc) Reports > Hide Diagram Inconsistency Indicators "fixes" that problem.

After a fair amount of Feng Shui, all the Needlines called for by the spreadsheets show.

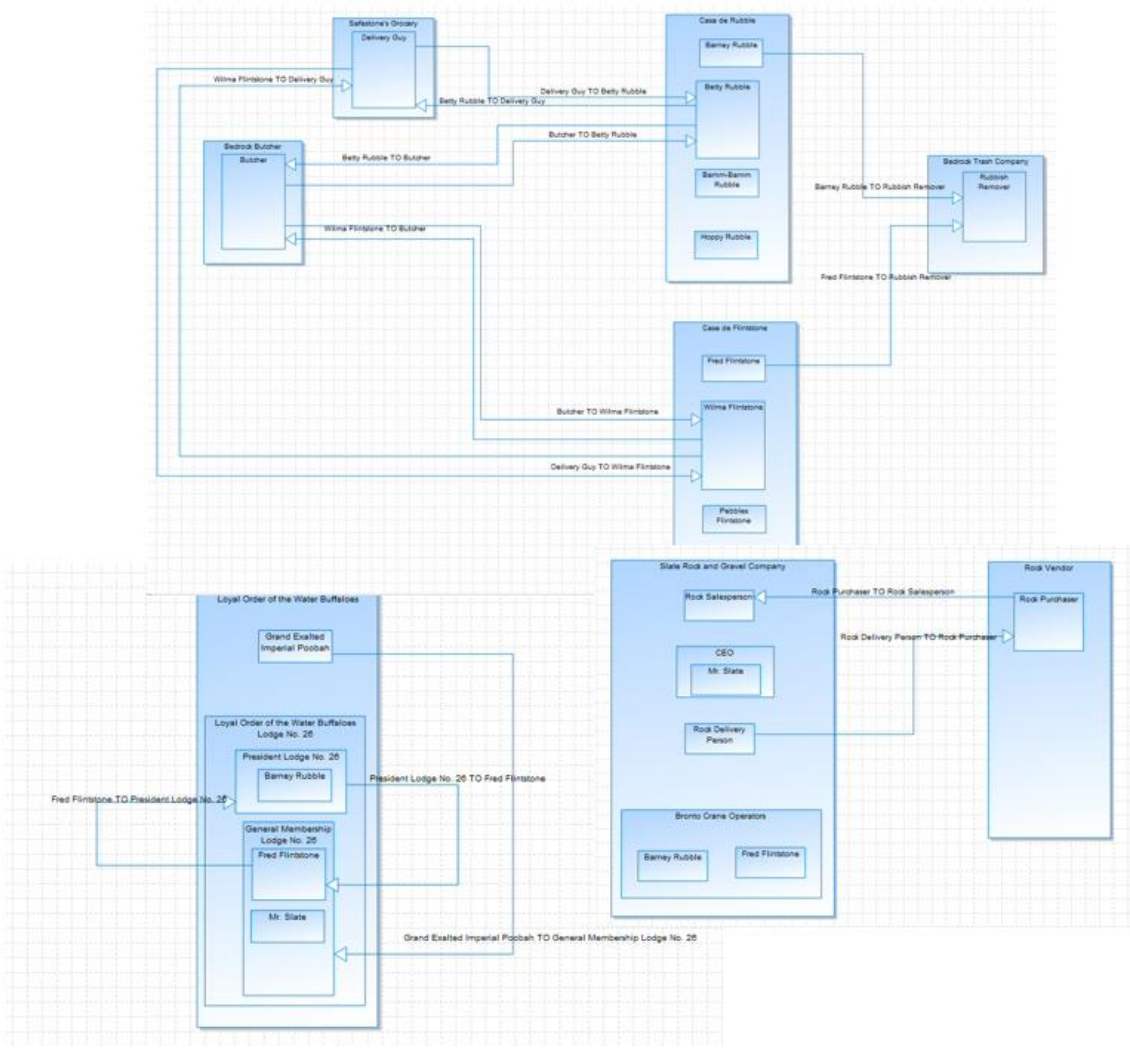


Figure 11. OV-2 Meet The Flintstones

I did not create a fully populated OV-5a and OV-5b model. What I am able to do with the imported items is to create a "flat" OV-5b (i.e., one diagram with everything on it, vs. a hierarchical set of diagrams a la IDEF0) that has all the Organizations, associated Performers, Activities, and ARO lines. It's an OV-2ish looking diagram, with the ARO's depicted instead of the Needlines (analogous to System-to-System relationships allowable in SV-1 diagrams that always ended up looking like a rat's nest for anything more complicated than depiction of the operation of a coffee machine… but… I digress…).

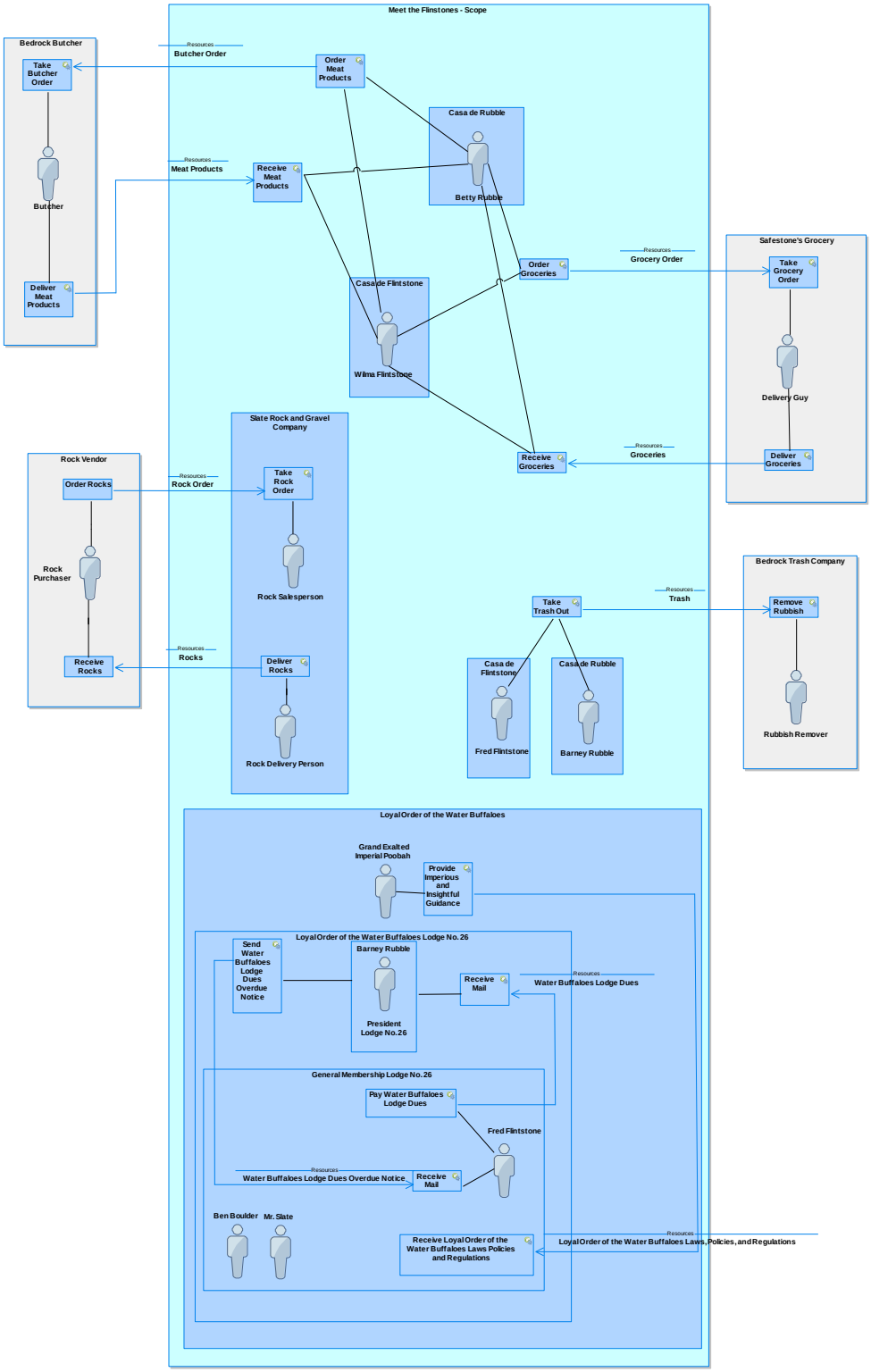


Figure 12 OV-5b-"ish" Meet the Flintstones

Some key settings/changes to default settings to make this "somewhat pretty:"

- **Organization (DM2)**: drag-drop Organizations on the page, and Feng Shui until you're satisfied.

    - Right click > Display Selected Items Conventionally. This allowed me to embed Performers within Performers
    - Right click > Display Mode > Symbol Type On This Diagram > Enabled > leave everything unselected. Then Save All > Close. This gets rid of the Members and Performer Members text in the box.
- **Person (DM2)**: drag-drop Performers on the page…
    - Right click > Symbol Format > Text Position > Place name outside. Allows one to move the name as necessary to read it.
- **Activity (DM2)**: drag-drop Activities on the page…
    - Right-click > Display Selected Symbols Conventionally. Cleaner looking.
    - *maybe* Right-click > Display Selected Symbols With Adornment Without Resizing. Tells you what type of box it is…
- **ActivityPerformedByPerformer (DM2r)** lines: are autodrawn for you. Rejoice freely.
    - Color them Black to differentiate them from ARO lines.
- **ActivityResourceOverlap (DM2r)** lines: are autodrawn for you. Rejoice freely.
    1. Right-click > Display Mode > Symbol Type On This Diagram > Enabled
    2. Choose Resources
    3. *may* want to choose "Hide Symbol Name" – this looks goofy, but it tells you what Resources are traversing the ARO, without showing the ARO name (which will just confuse the diagram reader anyway…).
    4. Save All > Close
    - Note: Sometimes this doesn't work and the only way to make Resources appear on pre-created lines is to delete the line, recreate the line, recreate all the references (ARO to Resource, OPEX to ARO), then it works. Service Request generated with UNICOM®.
- Rectangle: serves as a border around what I considered internal vs. external in this architecture.

One last note: If you do anything new after these imports (e.g. assert Fred Flintstone orders Groceries as an APBP), UNICOM® System Architect® (as of: version 11.4.3.4) has a bug with the Operational Exchange generator (Tools > DoDAF2 Utilities > Generate Operational Exchanges). If exchanges exist, SA does not generate new ones; you have to either: delete all exchanges and regenerate them, or manually create the Operational Exchange (DM2rx) and Needline. A Service Request has been submitted to UNICOM® about this issue.

Conclusion

This paper was intended to be a "prequel" to the UNICOM® whitepaper *How to Share DoDAF2 Data with System Architect*. The paper provided the reader with a solid knowledge foundation of the DoDAF 2.0 theory behind "how things fit together," enabling the reader to understand what is being built via the file imports, prior to implementing the methodology discussed in both papers. In doing so, it provided the reader with a new OV-3 report for use in the tool. The paper also provided updates to the methodology, enabling the import of real-world data without errors. This enables the reader to reuse previously created OV-3 and SV-6 data in UNICOM® System Architect®, without encountering errors.

Attachments

The following attachments are provided with this paper, in the zip file Attachments_Practical_Application_of_SA_OV-3_and_SV-6_Import_Methodology.zip

- From OV-03 - ActivityPerformedByPerformer (OV-05) - w left.xlsx
- From OV-03 - ActivityResourceOverlap (OV-05) - w left.xlsx
- From OV-03 - Need Line (OV-02) - w left.xlsx
- From OV-03 - Operational Exchange (OV-02z) w left.xlsx
- OV-4 Meet the Flintstones.xml