# Need 1

Generate a table like this:

| **Name** | **Summary** | **Port** | **Direction** | **Physical Link** | **Property Value** |
| --- | --- | --- | --- | --- | --- |
| Name of Node Physical Component | Summary of the Node Physical Component | Name of the NPC port | Direction of the BPC port allocated to the NPC port | Name of the Physical Link | Property Value attached to the NPC port |

(PC and ports should be displayed even if they have no PhysicalLink or BPC port or Property value defined)

# Need 2

 Import Node PC with owned Physical Link into Physical Architecture. For example:

|  |  |  |  |
| --- | --- | --- | --- |
| Level 1 | Level 2 | Level 3 | Description |
| NP 2 |  |  |  |
| NP 1 |  |  |  |
|  | NP 11 |  |  |
|  | NP 12 |  |  |
|  |  | NP 121 |  |
|  | NP 21 |  |  |

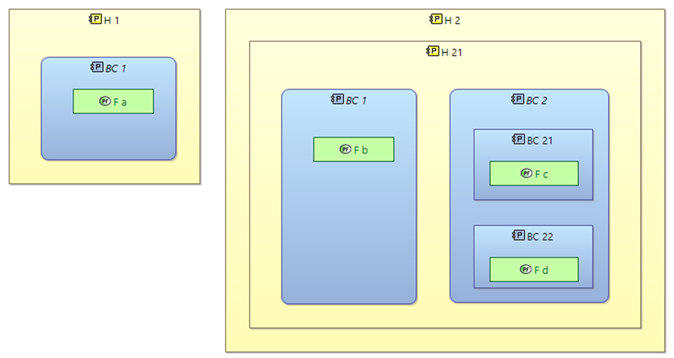
and

| **NP Name** | **Port** | **Physical Name** | **Port** | **NP Name** |
| --- | --- | --- | --- | --- |
| NP1 | NP1\_Port | Physical Link | NP2\_Port | NP 2 |

# Need 3

* Interfaces (all FE) between 2 components (logical for instance)
* Elements (example function) with associated Property Values

# Need 4



* For each function, get the hosting component
* Create a table hosting component / behavior component / functions including the hierarchy

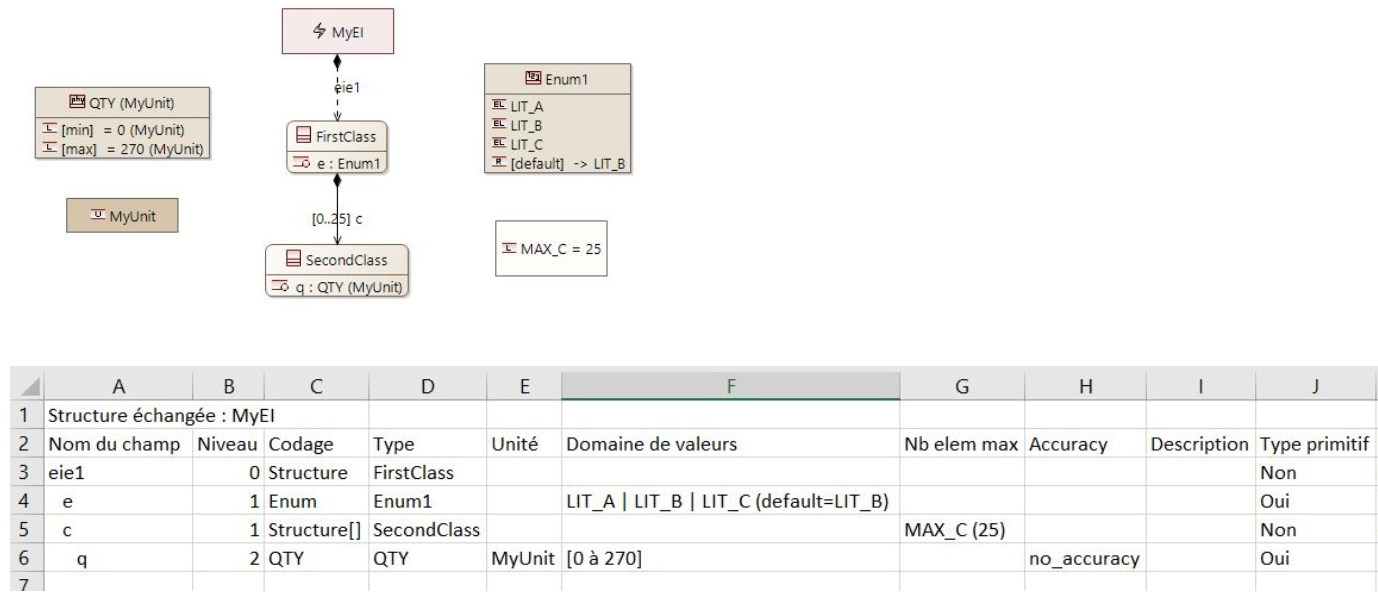
# Need 5

Different kind of imports:

* Import property values
* Import realisation links between Exchange items
* Import Actors (not that important because not so many objects)
* Import interfaces provided/required by ports

# Need 6

Export the complete definition of an ExchangeItem and its used types. With representation of imbricated types (through properties and associations). Example:



# Need 7

* Export list of requirements linked to model elements of a subsystem: to components, functions, ... along with the IDs
* Import list of requirements in a subsystem model and rebuild the links to components, functions, …

# Need 8

extract description of elements in both:

* HTML format
* in plain text (after removing HMTL tags <...\>

# Need 9

 perform parametric calculations on Python, based on properties defines through PVMT

* Extract PV from model elements
* Process them and perform calculations
* Assign calculated values to properties of other model elements
* Variant: create new PVs in those model elements and assign them these values