Release arKItect 5.1

arKItect 5.1 release notes

The new arKItect release includes new features that will allow two very important new capabilities with arKItect platform:

- You can now design your system from functions to implantation in arKItect: 2D views capability allows managing geometric properties of objects, determine dimensions of objects (e.g. wire&harness definition and length, concrete volume, rooms surface…). This capability proves useful for wiring and harnesses definition, multiple networks design in construction (ventilation, high and low currents, signals, fluids, waters…).
- You can now configure arKItect to support any diagramming language: SysML like tools, Matlab/Simulink, any modelica based language… Such capability was foreseen by Manfred Broy teams in TUM more than 10 years ago in their paper "Seamless Model-based Development: from Isolated Tools to Integrated Model Engineering Environments, Broy&al, see https://ieeexplore.ieee.org/abstract/document/5420030

Very happy to say that arKItect is the unique platform gathering all these possibilities.

In this paper we present in more details:

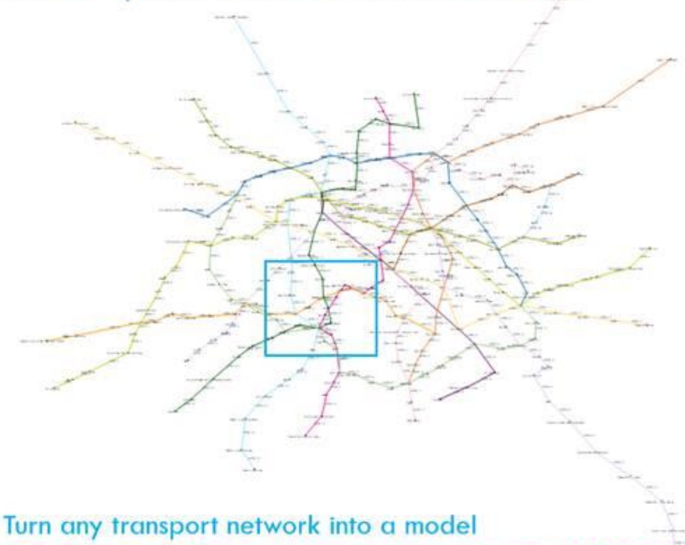
2D objects and views

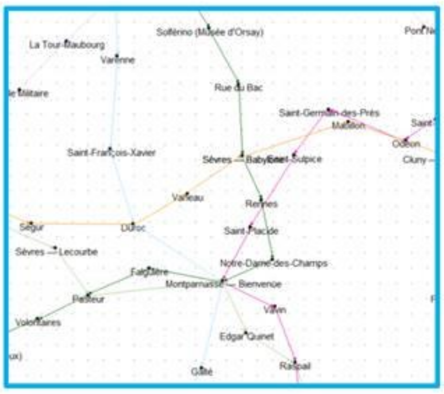
This is a major change in arKItect as 2D views get geometric capabilities. It's now possible to define objects with a 2D behavior.
Such objects will behave as usual in logical viewpoints but will behave as 2D maps in 2D viewpoints.

First see some examples of these new capabilities.

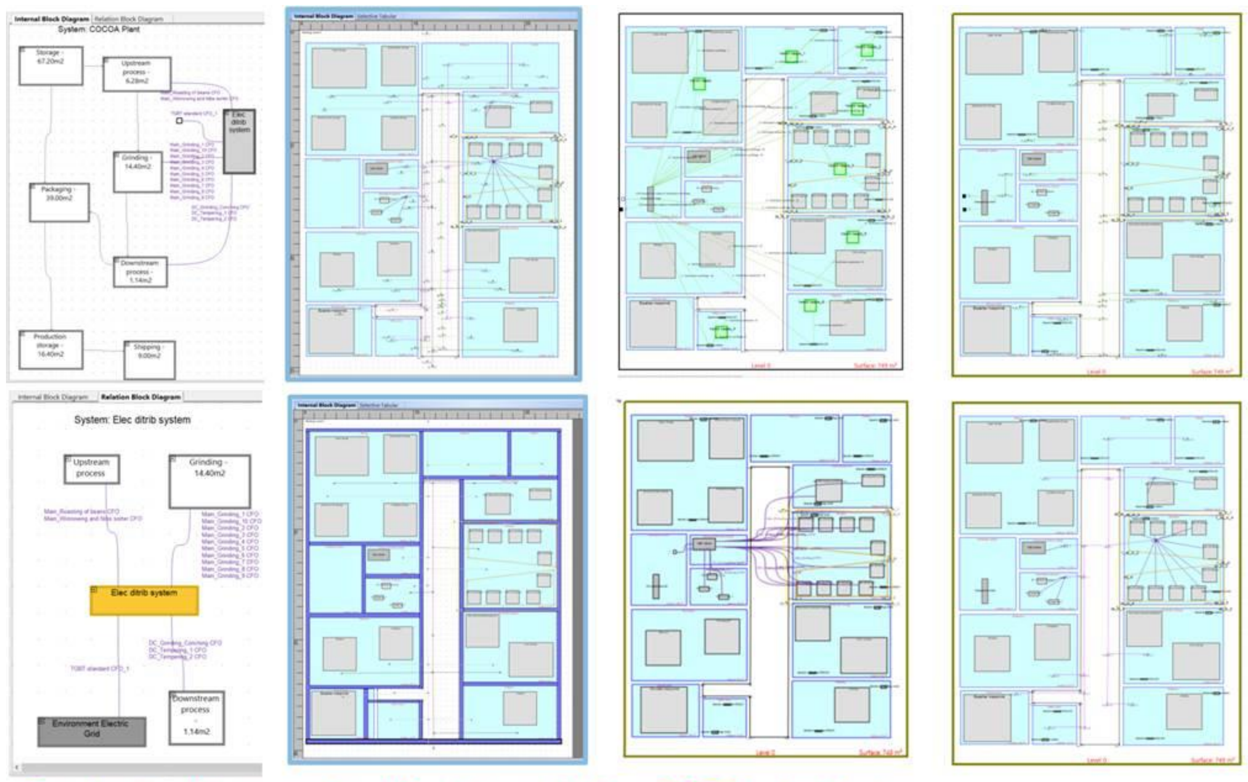
Knowledge Inside
arKItect your system

Public Transports in Paris area modeled in arKItect





Turn any transport network into a model
Benefit from arKItect phases, options and variants features to model formally how the network should change over time or during a project.



From functional to geometric arKItect viewpoints for a COCOA plant design

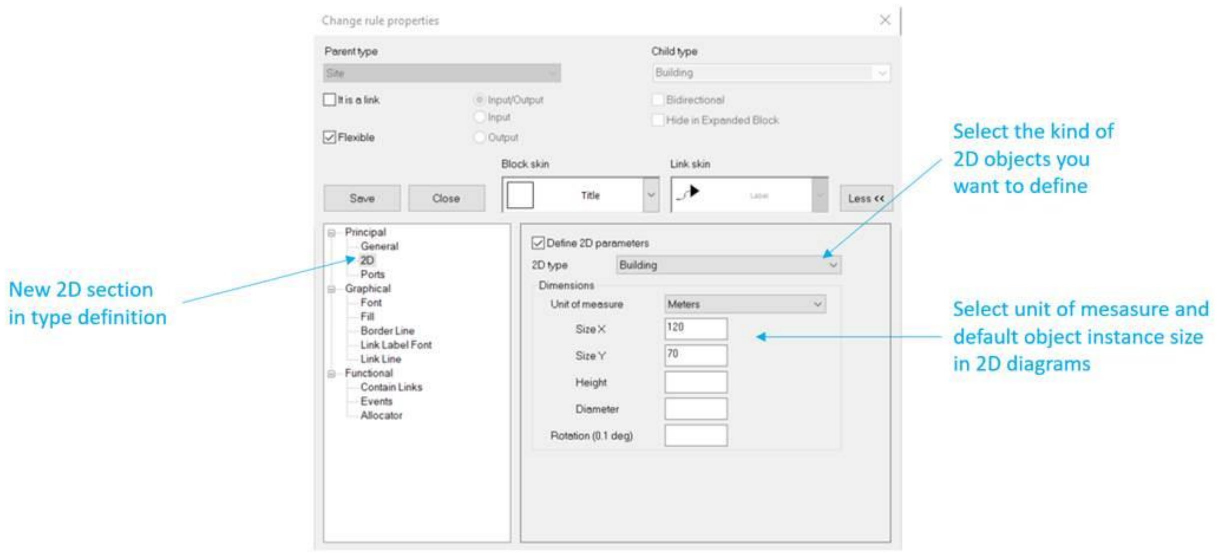
Wiring a car or a plane would not be much different from a building from a tooling point of view. The point is that we can manage as well requirements, functional and logical architecture seamlessly.

Let's look at the changes from a modeler point of view.

Changes in the data model editing:

<u>Types new attributes</u>
Types may now have a 2D attributes.
A new menu is available where you can declare that type will have a specific behavior in a 2D view. Note that types that are not 2D can exist in a 2D view but corresponding objects will not be displayed in Internal View. You will see them however in the TreeView. This is particularly convenient for meta informations that need to be allocated in the geometric tree but that should not impact your 2D drawings, e.g. requirements.

Type with 2D attributes will be displayed as usual in logical (non 2D) views.


Change rule properties
Parent type: Site
Child type: Building
☐ It is a link
☑ Flexible
Block skin: Title
☑ Define 2D parameters
2D type: Building
Dimensions
Unit of measure: Meters
Size X: 120
Size Y: 70
Height:
Diameter:
Rotation (0.1 deg):

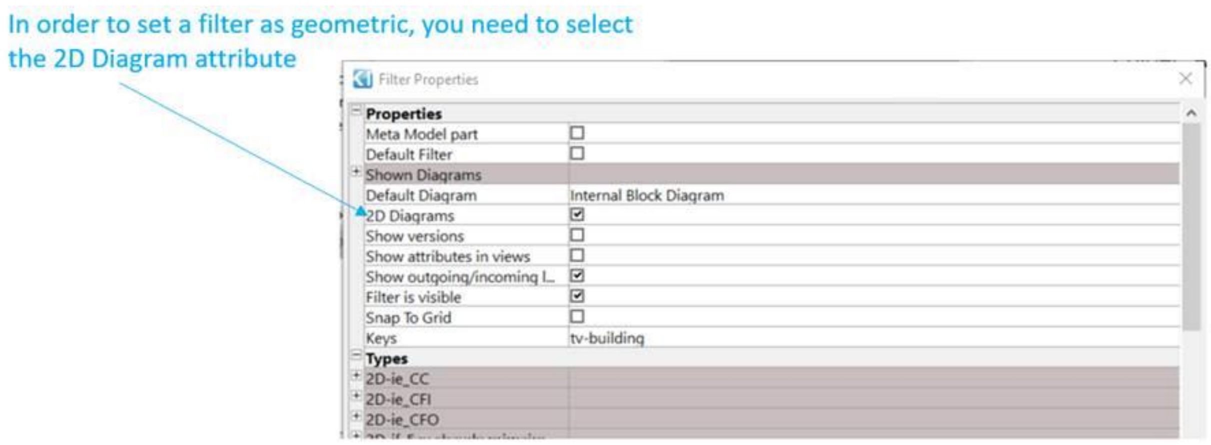
2D type categories correspond to different order of magnitude when browsing in a 2D view.
From the highest level to the lowest level of scale and abstraction, the arKItect type you define could be associated with one and only one of the following kinds:

- *Site* corresponds to the highest level of abstraction. It's size would be in the range of square kilometers. Site contains Buildings but may also contain Closed Networks or Networks. Site is displayed as an expanded object, you see everything inside.

- *Building*s correspond to a scope of a few thousands of square meters. It is really featuring a Building and may contain several floors. For this reason, a Building is folded and cannot be unfolded. Size of the building and floors are independant at first. Normally, the building will inherit from the surface of it's implantation in the ground, but user is free to do what he wants. Inside the Building only Levels are possible.
- *Level* corresponds to a floor in a building, it is displayed in same scale as the building. Levels are always expanded
- *Rooms* are surfaces of interest that normally will have both a functional and spatial role. They will typically be in a range of few tenths to few hundreds of square meters but you may use them for designing the wiring of a car, train or plane. Rooms correspond to the scale where people live. Rooms are always expanded when they are displayed which means you always see what's inside.
- *Process, Equipment and Wall* correspond to items that will be allocated into a room or into a level or into other generic sized objects. The only difference between them will be the default 2D parameter values that may change from one kind to another. Process, Equipements and Walls are always black box. You cannot see what's inside in a 2D view. They are supposed to be the leaf objects of a 2D view.
- *Network* and *Crossing Points* are used to design networks, e.g. electrical network, ventilation network, fluid network, communication buslenghtses network… Crossing points are the nodes of a network. These nodes are linked with some kind of straight-line flows. User never has to deal with links, the interface allows to draw a link by a succession of clicks corresponding to crossing points (first click corresponds to where the network starts). Everything is then generated automatically.
- *Closed network* is used to design a shape that is not simple. They can be used to draw geographical zones e.g. for fire analysis. Closed networks use crossing points to design their outlines exactly like Networks.
- *Zone* is a particular of Polyline shapes. It's actually more a beta version. It plays the same role as closed networks.

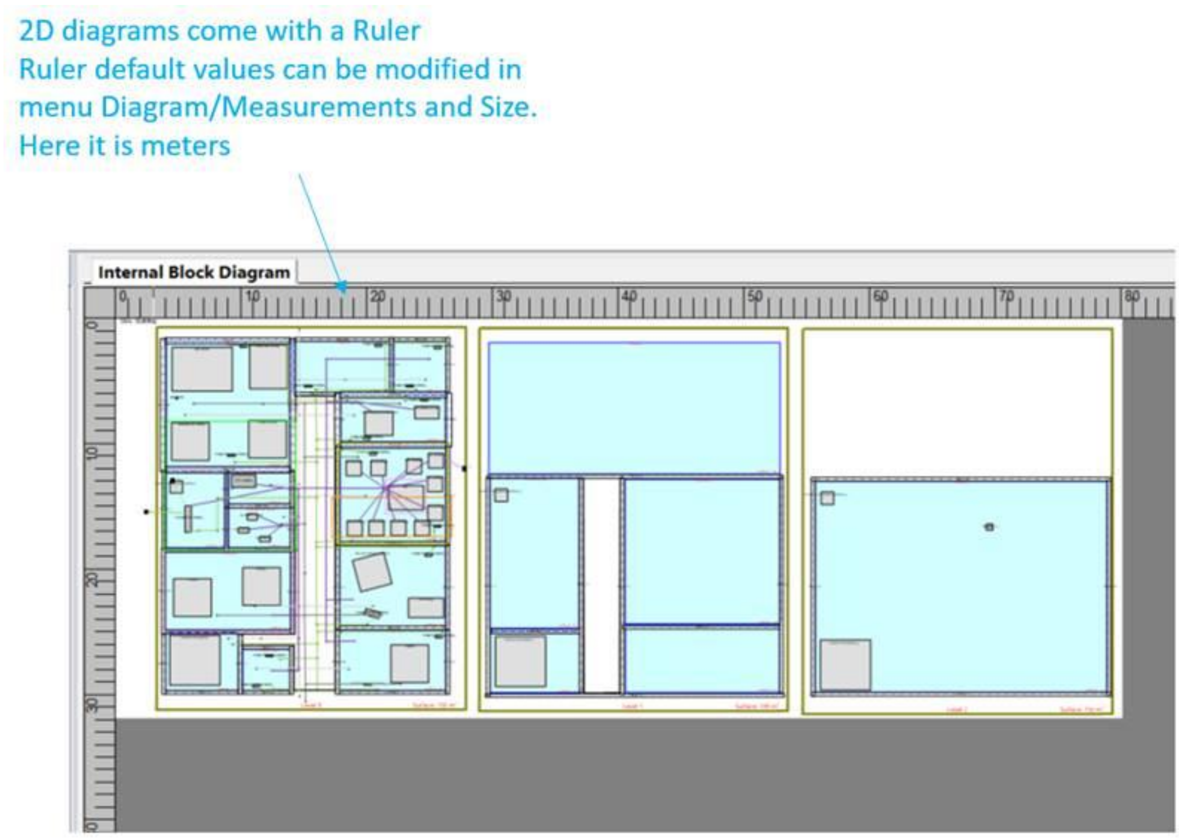
Filters/viewpoints new properties:

Once you have define arKItect types in your data model and associated these types with 2D types. You can then build 2D filters.

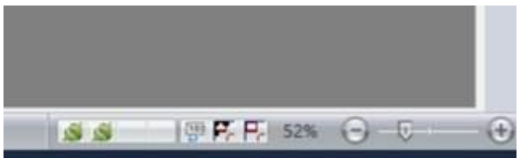

In order to set a filter as geometric, you need to select the 2D Diagram attribute
Filter Properties

Properties	
Meta Model part	☐
Default Filter	☐
Shown Diagrams	
Default Diagram	Internal Block Diagram
2D Diagrams	☑
Show versions	☐
Show attributes in views	☐
Show outgoing/incoming l...	☑
Filter is visible	☑
Snap To Grid	☐
Keys	tv-building
Types	
2D-ie_CC	
2D-ie_CFI	
2D-ie_CFO	

Once you work in a 2D diagram, you see immediately a ruler giving you the scale:

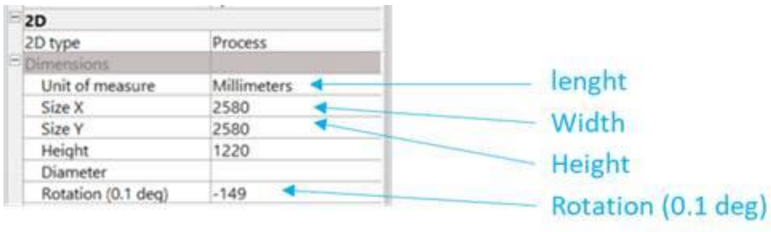

2D diagrams come with a Ruler
Ruler default values can be modified in menu Diagram/Measurements and Size.
Here it is meters

The scale button at the left right corner allows to browse with adapted scale from 1% to 1500% corresponding to a screen of 2 meters to 4000 meters for a building levels description.



<u>Changes in Objects manipulation in views</u>

When manipulating objects, the scale of the display adapts to the level of abstraction.
Also you get now 2D attributes for objects having a 2D type.



2D	
2D type	Process
Dimensions	
Unit of measure	Millimeters
Size X	2580
Size Y	2580
Height	1220
Diameter	
Rotation (0.1 deg)	-149

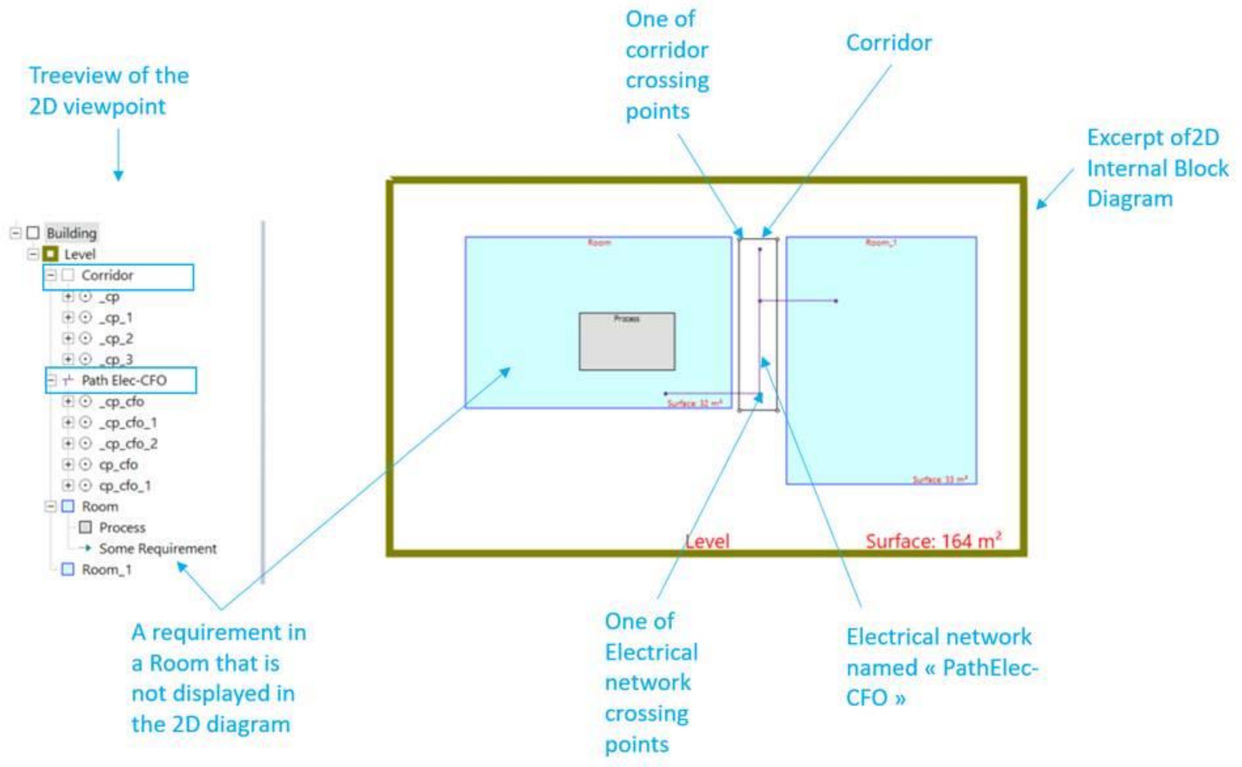
Attributes are name x, y, z and correspond to the dimensions according to the diagram. However it's good practice to use X as length, Y as width and use rotation in case the Y > X

Default values are defined according to the 2D type default values.

It’s possible to change labels positions depending on types in 2D views.

Special case of networks

Networks and closed networks have a specific behavior: they do not appear in a graphical 2D view but are displayed in the Treeview. There exist now a set.


Treeview of the 2D viewpoint
Building
Level
Corridor
_cp
_cp_1
_cp_2
_cp_3
Path Elec-CFO
_cp_cfo
_cp_cfo_1
_cp_cfo_2
cp_cfo
cp_cfo_1
Room
Process
Some Requirement
Room_1
A requirement in a Room that is not displayed in the 2D diagram
One of corridor crossing points
Corridor
Excerpt of2D Internal Block Diagram
Room
Process
Room_1
Level
Surface: 164 m²
One of Electrical network crossing points
Electrical network named « PathElec-CFO »

In this example, the corridor is defined by 4 crossing points. Although crossing points with links together are represented in the 2D diagram and in the treeview, the Corridor object itself has not graphical counterpart.
However, if you would delete the Corridor from the Treeview here, all crossing points in that corridor would be removed as well.

Non 2D objects in 2D viewpoints

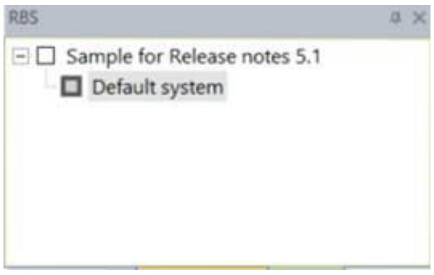
If we come back to the previous picture, we see that "Some Requirement" is allocated to the Room in the Treeview but is not displayed in the Internal Block Diagram. Logical objects without 2D type are not displayed in 2D views, this allows to keep them readable even in the context where hundreds of requirements are allocated to rooms, equipments, networks…

Much easier Excel interface

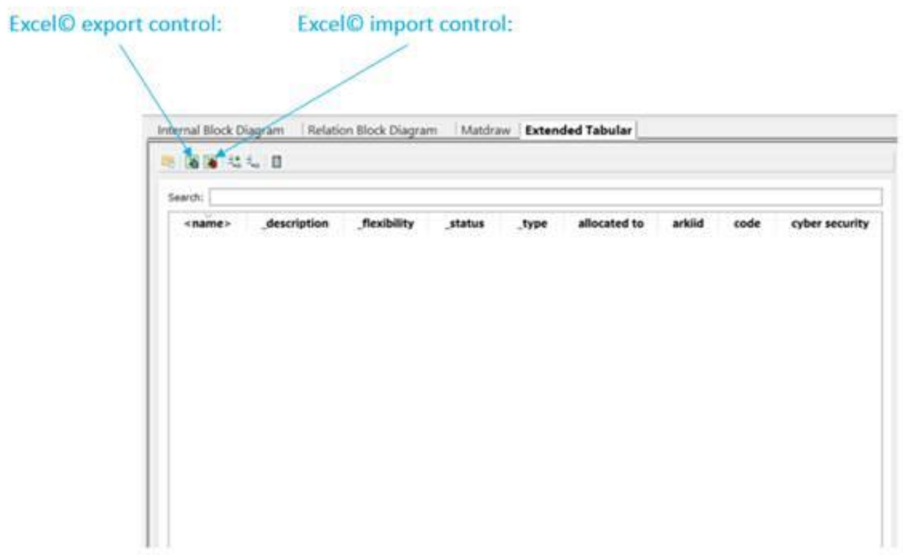
There exist now a set of directories in your arKItect local variables that capture all your Excel files import and export.
These directories are organized according to the arKItect model structure. This means each time you import and export a file, by default, arKItect is already pointing to the right file without the need to navigate in your file system either to record or to open.

Let's consider a simple example:
Suppose I'm in a requirement view and select a System and want to add requirements on it.

Here is my empty view:

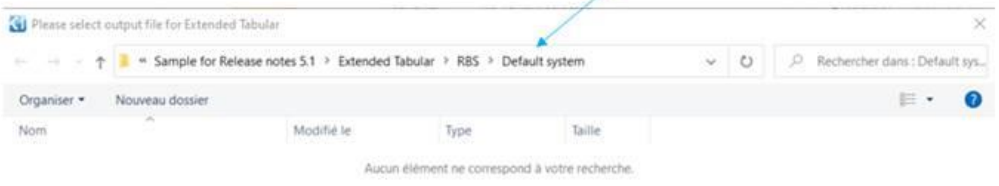


I can visualize my Extended tabular view and see that Default system gets no attached requirements:

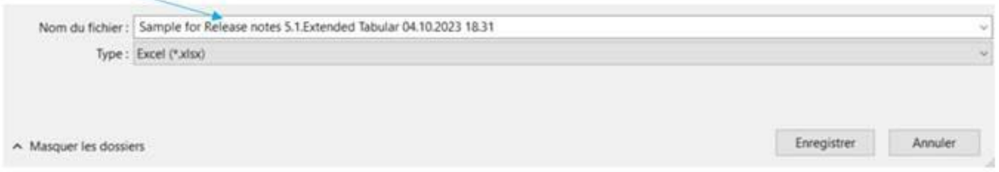


If now I export my empty file in Excel, I get:

This path is specific to my project, view and object

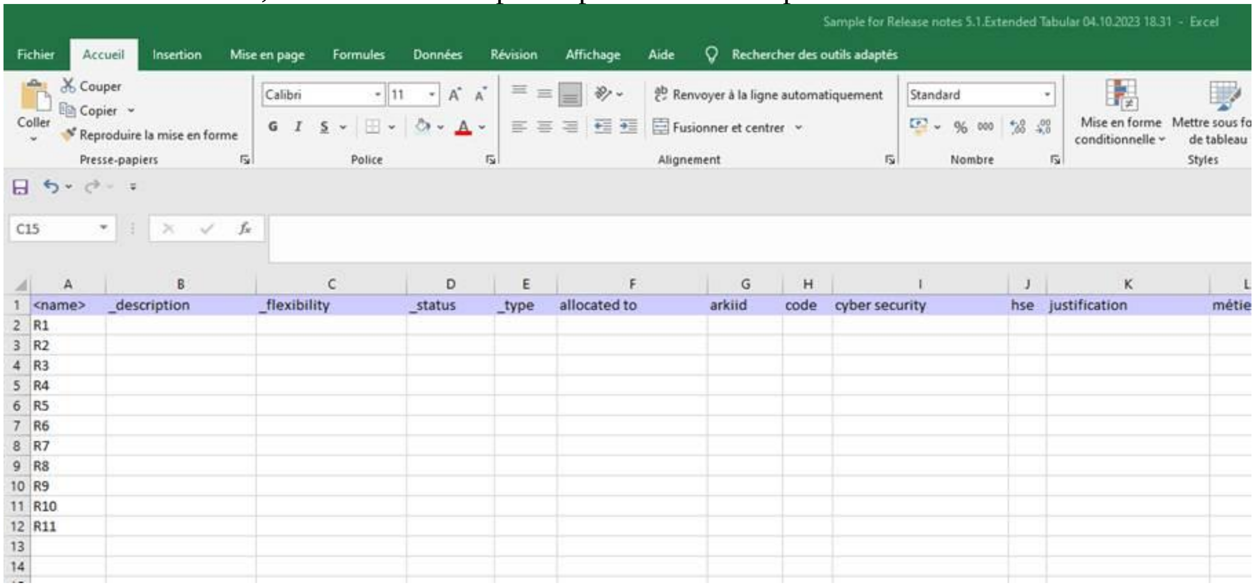

Please select output file for Extended Tabular
Sample for Release notes 5.1 > Extended Tabular > RBS > Default system
Rechercher dans : Default sys...
Organiser
Nouveau dossier
Nom
Modifié le
Type
Taille
Aucun élément ne correspond à votre recherche.

File name is automatically generated


Nom du fichier : Sample for Release notes 5.1.Extended Tabular 04.10.2023 18.31

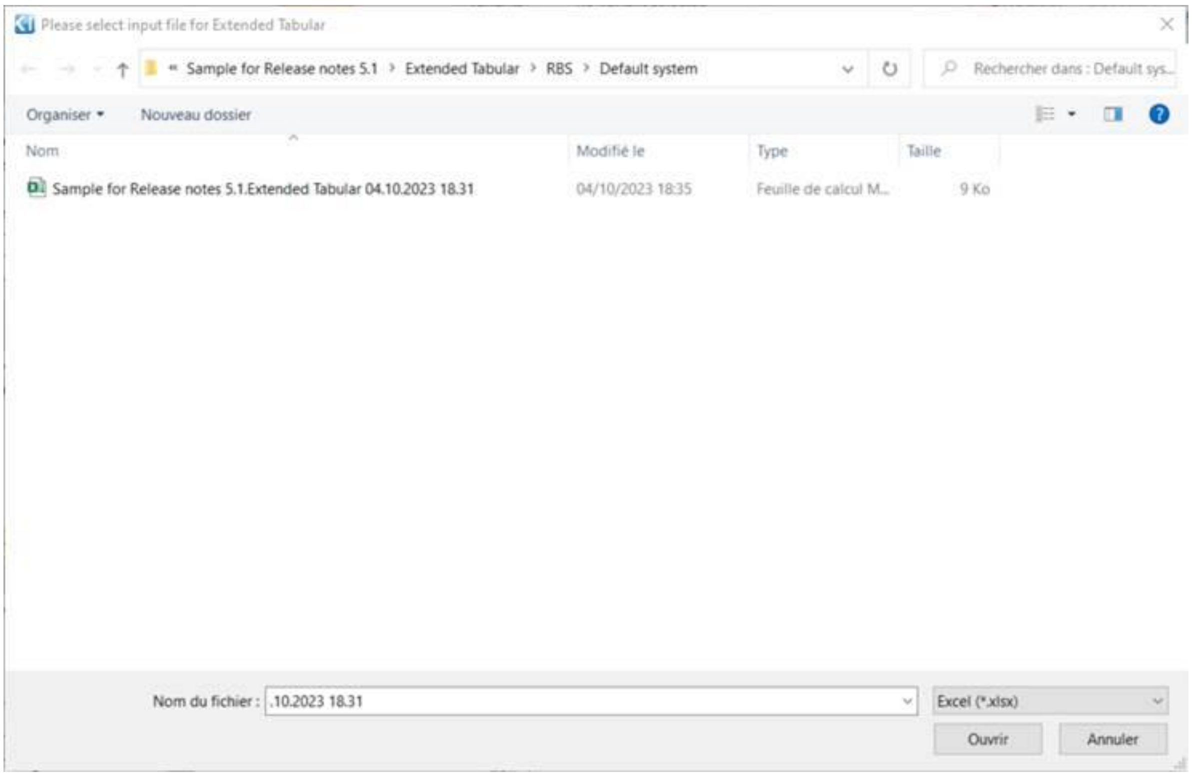
Type : Excel (*.xlsx)

When I save the file, the Excel© file opens up and I can complete it fast:


Sample for Release notes 5.1.Extended Tabular 04.10.2023 18.31 - Excel
Fichier
Accueil
Insertion
Mise en page
Formules
Données
Révision
Affichage
Aide
Rechercher des outils adaptés
Couper
Copier
Reproduire la mise en forme
Coller
Presse-papiers
Calibri
11
Police
Renvoyer à la ligne automatiquement
Fusionner et centrer
Alignement
Standard
Nombre
Mise en forme conditionnelle
Mettre sous forme de tableau
Styles
C15

	A	B	C	D	E	F	G	H	I	J	K	L
1	<name>	_description	_flexibility	_status	_type	allocated to	arkiid	code	cyber security	hse	justification	métie
2	R1											
3	R2											
4	R3											
5	R4											
6	R5											
7	R6											
8	R7											
9	R8											
10	R9											
11	R10											
12	R11											
13												
14												

Let's play and create some fake requirements
We save our file and close it.
Now going back to arKItect and importing from Excel menu we get:


Please select input file for Extended Tabular
« Sample for Release notes 5.1 › Extended Tabular › RBS › Default system
Rechercher dans : Default sys...
Organiser ▾
Nouveau dossier
Ouvrir
Annuler

Nom	Modifié le	Type	Taille
Sample for Release notes 5.1.Extended Tabular 04.10.2023 18.31	04/10/2023 18:35	Feuille de calcul M...	9 Ko

Nom du fichier : .10.2023 18.31

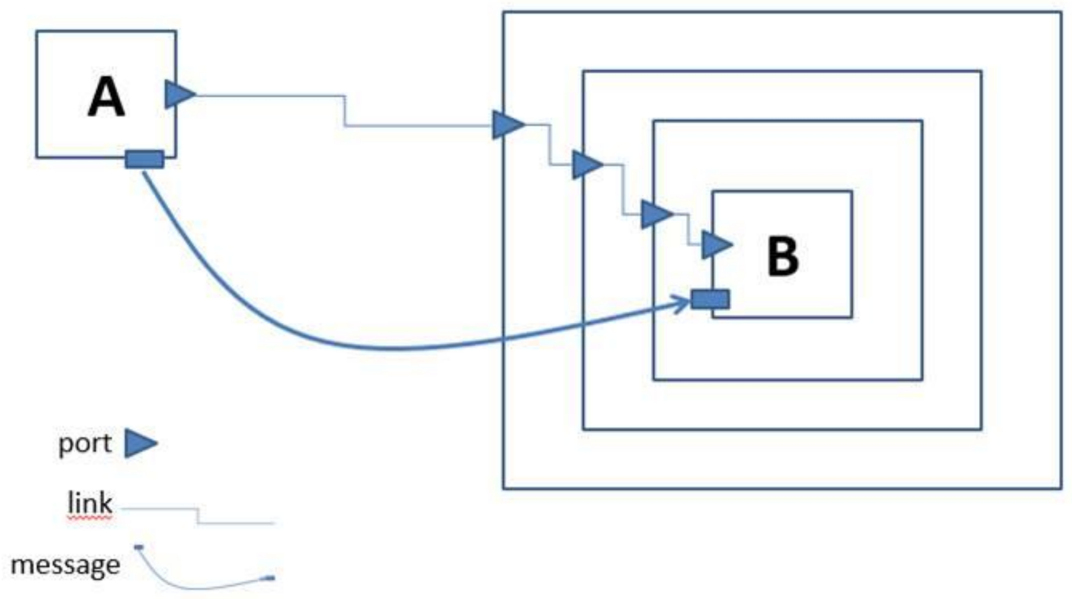
Excel (*.xlsx)

We now come back to the exact file we just saved and we can see all other files we may have defined at the same location. We just need to click "Open" and everything is going to be imported just fine and fast.

This behavior is implemented for any Excel© import and export with "Extended Tabular". You can use it for any specific script as well.

Improved Ports handling

arKItect is basically a "Port less" tool and our approach has always been to avoid ports and links and manage everything with messages. This explains why our technology is so simple and agile w.r.t. most of Object Oriented tools and methods. The diagram below explains this:



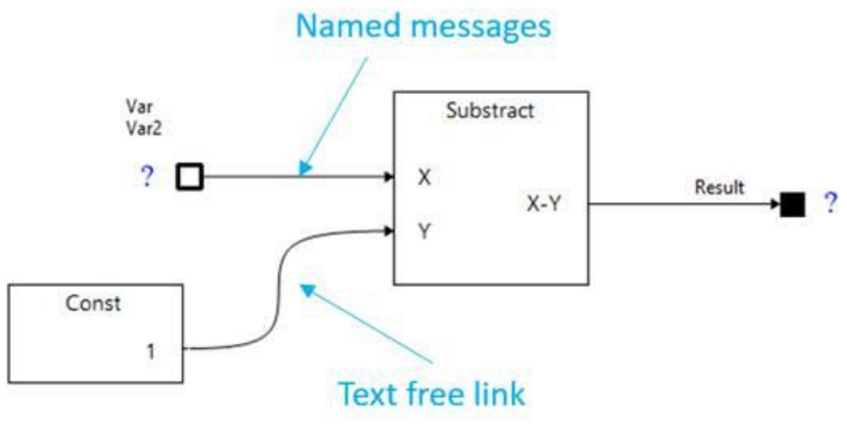
In this example, to describe with ports and link one message sent from A to B, you need 5 ports and 4 links. This is why port based tools and models are so fat: 9 objects versus 1 here!

But this is not all! If you want to move block B somewhere else in the architecture, you will have to delete and recreate many ports and links: this is hyperstatic. Once you model is complex this is very cumbersome.

There are other advantages for messages versus ports which are out of the scope of this release notes.
Of course, a message is a higher-level concept. In arKItect, messages are "first class citizens" or atomic objects in the arKItect platform. Object Oriented languages usually support only one concept of class and objects as the basis for everything. As a consequence, defining and using messages is usually very complex to configure and not available.
But if you consider most of the visual languages for simulation tools like Matlab/Simulink. They suffer from the same drawback.

There are however particular cases where ports are really useful. It's typically to describe black box objects with fixed interfaces. In particular this is the case for specific computational objects like subtraction.

With the latest version of arKItect, you will get the best of all worlds. As far as we know this makes our platform unique.



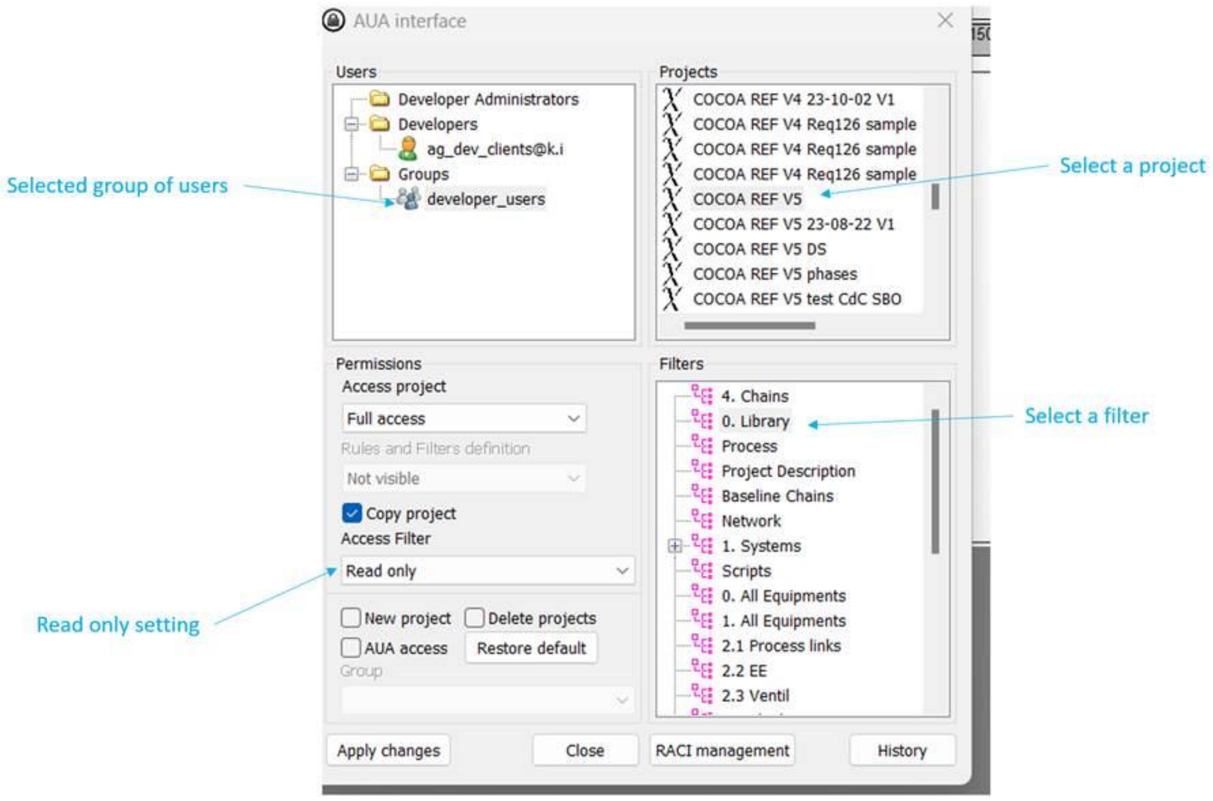
You can easily mix link style and message style model.
As addition is associative and commutative, there's no problem authorizing several input message on an addition port, you can even mix links and messages on same port.

If you used to copy a message name on various different links with Matlab/Simulink for instance, you will immediately understand the advantage of this.

You now benefit of the ability to reorganize your architecture freely without having to update many ports and links display.

Read-only views

It is now possible to set a view as "Read Only" in your project by using the AUA.
AUA allow setting a filter (viewpoint) as "Read Only" for all users.
This property is saved when you copy this project or instanciate it for creating new projects.
Although view is read-only, drag&drop from this read-only view to another will be available.
This is very usefull if you use a filter/viewpoint and his treeview as a library from which you can drag&drop your off the shelf items and instanciate them in your project. You could see this feature as practical help for implementing a library/instanciation mechanism in arKItect.



AUA interface

Users
- Developer Administrators
- Developers
  - ag_dev_clients@k.i
- Groups
  - developer_users

Projects
- COCOA REF V4 23-10-02 V1
- COCOA REF V4 Req126 sample
- COCOA REF V4 Req126 sample
- COCOA REF V4 Req126 sample
- COCOA REF V5
- COCOA REF V5 23-08-22 V1
- COCOA REF V5 DS
- COCOA REF V5 phases
- COCOA REF V5 test CdC SBO

Permissions
Access project: Full access
Rules and Filters definition: Not visible
[✓] Copy project
Access Filter: Read only
[ ] New project [ ] Delete projects
[ ] AUA access Restore default
Group

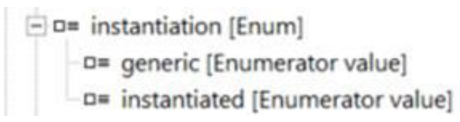
Filters
- 4. Chains
- 0. Library
- Process
- Project Description
- Baseline Chains
- Network
- 1. Systems
- Scripts
- 0. All Equipments
- 1. All Equipments
- 2.1 Process links
- 2.2 EE
- 2.3 Ventil

Apply changes | Close | RACI management | History

Libraries management

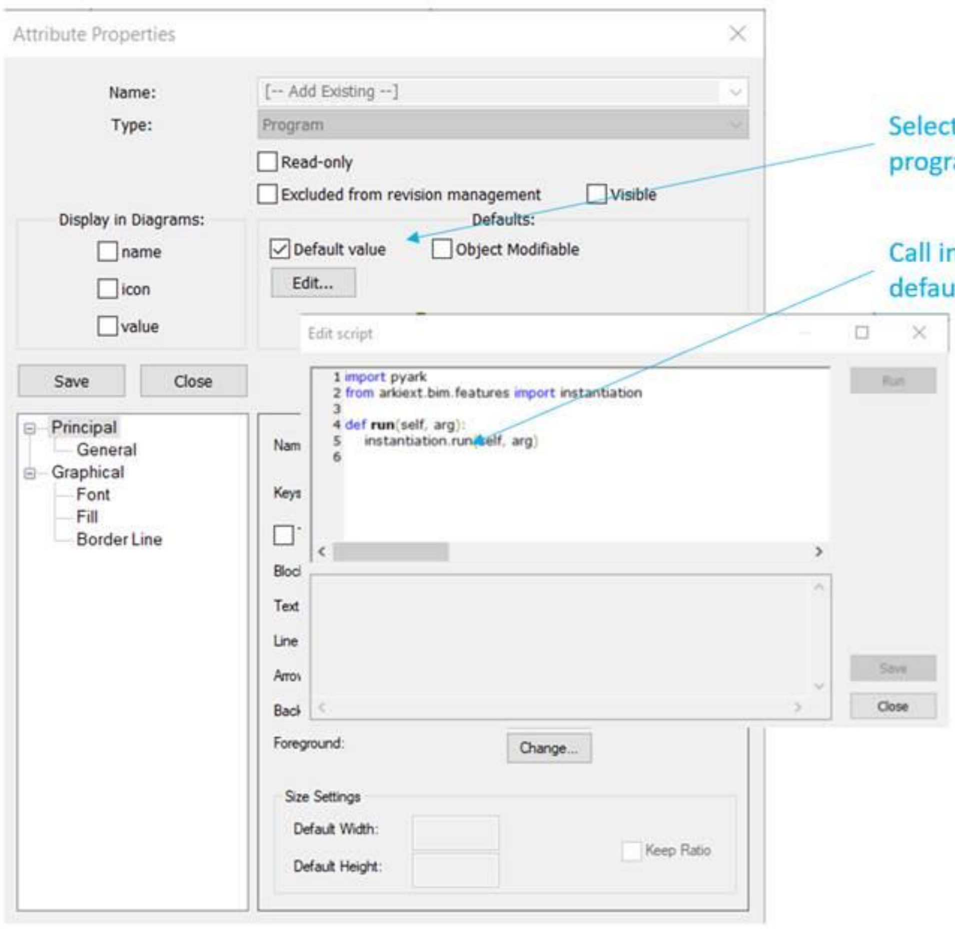
Suppose we want to define a standard library of components and allow a user to instanciate those components in some view. For instance, the user may decide to add some off the shelf electric transformers in a specific room in a plant.
Assume we already have standard description of some standard equipments. The topic here is to define a library of standard objects (with the interfaces) and instantiate them for a particular application.

For this you will need to add an attribute to distinguish between standard equipements and instantiated ones.



- instantiation [Enum]
  - generic [Enumerator value]
  - instantiated [Enumerator value]

Then you will need to add a programming attribute to equipements as below:


Attribute Properties

Name: [-- Add Existing --]

Type: Program

☐ Read-only

☐ Excluded from revision management ☐ Visible

Display in Diagrams: ☐ name ☐ icon ☐ value

Defaults: ☑ Default value ☐ Object Modifiable

Select default value for the program

Call instantiation library in default value

Edit script

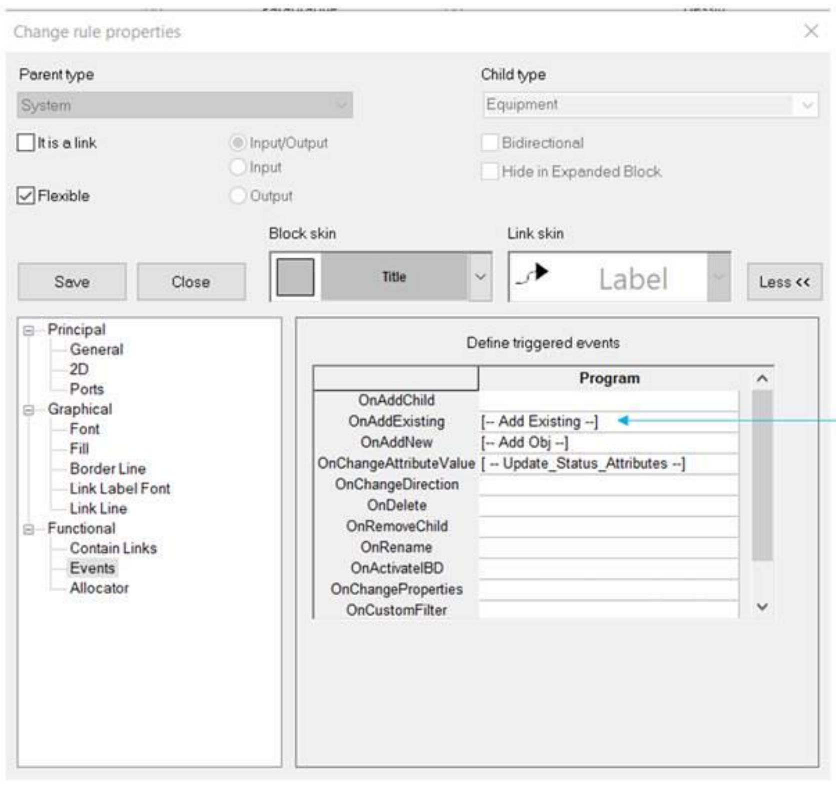
```
import pyark
from arkiext.bim.features import instantiation

def run(self, arg):
    instantiation.run(self, arg)

```

This programming attribute will call a multi purpose instantiation python library that will replicate both the object and its interfaces and manage replication of attributes and naming issues.

Then you have to configure the event manager of arKItect so that the instantiation is performed each time you add a generic equipement to another view. Typically this will look like this:



Change rule properties

Parent type: System

Child type: Equipment

☐ It is a link

● Input/Output

○ Input

○ Output

☐ Bidirectional

☐ Hide in Expanded Block

☑ Flexible

Block skin: Title

Link skin: Label

Save | Close | Less <<

- Principal
  - General
  - 2D
  - Ports
- Graphical
  - Font
  - Fill
  - Border Line
  - Link Label Font
  - Link Line
- Functional
  - Contain Links
  - Events
  - Allocator

Define triggered events

	Program
OnAddChild	
OnAddExisting	[-- Add Existing --]
OnAddNew	[-- Add Obj --]
OnChangeAttributeValue	[-- Update_Status_Attributes --]
OnChangeDirection	
OnDelete	
OnRemoveChild	
OnRename	
OnActivateIBD	
OnChangeProperties	
OnCustomFilter	

Once we drag & drop an object from library to another view, this program is making an instanciation and substitute generic object with instantiated one

Word Document generation: simplification

Although python based document generation exists for a long time in arKItect (15 years), we recently updated the supporting libraries to help users doing this easily. This however requires some python skills.

You will find a documentation with samples in the distribution:
ArkiScripts\arki\utils\wordexport
Update of wiki documentation for this item will be performed afterwards.

Multi-language

We now have multi-language support on the platform.
This requires of course Multi-language translation of many items:

- arKItect commands, viewpoints
- Script related messages
- Data model types and attributes

We ask you to contact us for integrating that in your models.

Improved Web export

Now display of attributes is faithful to what you get in arKItect views
Icons in treeviews are now exported
Global quality of export is major.

Hints and Bug fixes:

- Improvement of orthogonal lines as flow shape.
- Changes in selection management in the man machine interface, up to now, selection of a flow object in Internal Block Diarams would point to that flow in corresponding treeview, that was cumbersome for very big projects when hundreds of flows could exist and you could get lost in treeview. Now you need double-click on a flow to get it in the treeview. In the meantime, you can move, allocate this flow in Internal Block Diagram without changing the display of the corresponding treeview.
- Several Improvements in Mgw and extended tabular: management of empty cells, management of various floats representations (e.g. 0,0 versus 0.0).
- Support of keys in python APIS allows getting rid of naming issues in scripts.
- Options and variants: Options are no longer recursive.
- Important changes for generic chains: when drag&dropping an object from a treeview, it considers only the showed flows and subobjects of that treeview.
- It's now possible to set an attribute active in a view but not visible in object properties.
- When using our Model Gateway tool to create imports exports, you can do this with developer credentials now while you had to be designer before...
- Improved selection of objects in expanded view. Before when you were clicking in an expanded object, you would select the expanded object all the time. Now you need to be closed to the border to do that. This allows multi selection of objects in an expanded object.
- A lot of improvements have been done for baseline chains. For instance you now get automatically a graphical diff of two revisions of a baseline chain together with the list of changes in the change management process.
- Snap to grid is now not selected by default for new views.
- It's now possible to display several icons horizontally on an object in IBD, in the past you would have one line per attribute.
- Export of image from a view used to be a little truncated in certain configurations. This is fixed.
- When selecting available views for a filter(viewpoint) in the past, IBD was mandatory. This constraint no more exists.
- By default, first level of every tree view is always unfolded. Was not the case before.

Contact information:
contact@k-inside.com