GENERIC LIBRARY [GL 1.N]

MODELING RULES

CECILIA 6.3.X

Document Version GL_1_6.3-A

(issue A)

TABLE OF CONTENTS

1. SUBJECT

This document gathers a set of modeling rules that have been followed when building the generic library [GL 1.n] within Cecilia Workshop version 6.3.x.

These rules cover the organization of directories, naming, colors, etc., for the various library elements, as well as how to specify and verify them.

The aim of applying these rules is not only to enable collaborative work on the same project, to enrich the components library, but also to be able to maintain models over time.

2. REVISIONS

Version	Date	Updates
GL_1_6.3-A	09/03/2026	Initial version

3. FIELD OF APPLICATION

These modeling rules apply to version 6.3.x of the Dassault Aviation tool, Cecilia Workshop, whatever “x”. They concern [GL 1.n] generic libraries, whatever “n”.

4. GENERAL

3.1 Naming

Naming is an important part of the modeling process. Explicitly naming the various items used has several advantages:

- quickly identify the content or the function of an item
- make it easy the reading of the results
- enable others to understand the model more quickly

Allowed characters: lowercase letters, uppercase letters, underscore "_", numbers (not allowed at the beginning of a word). Spaces, special characters and accents are not allowed, and avoid too long names!

Names should be as readable as possible:

Avoid:	communicationsystem
Prefer:	CommunicationSystem or Communication_System

3.2 Colors

Here is the list of the colors used:

Color	RGB	HTML code	
Green	0, 255, 0	# 00FF00	🟩
Red	255, 0, 0	# FF0000	🟥
Blue	0, 0, 255	# 0000FF	🟦
Orange	255, 153, 0	# FF9900	🟧
Grey	204, 204, 204	# CCCCCC	⬜

Other colors can be used as needed.

3.2.1 *Enumerate type colors (Flows)*

Links colors are attributes defined for each Enumerate type, commonly named Flows ("Types" tab). Default colors are defined when the flow is created.

These colors can be overloaded in each simulable model (Models in the "Projects" tab), in the "Links colors" sub-tab. Four colors are mainly used:

"Data" flows	"Power" flows	"Zonal" flows	
Nominal	goodPower	no threat	🟩
Loss	noPower	-	🟥
Misleading	badPower	-	🟦
-	-	threat	🟧

3.2.2 *Record type colors (Bus)*

The color of a Record type, commonly named Bus ("Types" tab), is by default the color of the first of its component enumerate flows. This default flow can be selected by checking the "Link to display" box.

<u>Tip:</u> In order to control the color of a bus (in particular to point out that one of the flows is in a different state from the others), it is possible to create an enumerated flow designed to manage this color.

Example:

ElecPowerBus_color	
goodPower	🟩
noPower	🟥
badPower	🟦
mixedPower	🟧

Let's imagine ElecPowerBus_03c (Bus_color, PowerFlow1, PowerFlow2, PowerFlow3):
Type of Bus_color: ElecPowerBus_color (goodPower, noPower, badPower, mixedPower)
Type of PowerFlow: ElecPower (goodPower, noPower, badPower)

The color can be managed at the output of the item as follows:

```
If       PowerFlow1 = PowerFlow2 = PowerFlow3 = goodPower  then Bus_color = goodPower
Else If  PowerFlow1 = PowerFlow2 = PowerFlow3 = noPower    then Bus_color = noPower
Else If  PowerFlow1 = PowerFlow2 = PowerFlow3 = badPower   then Bus_color = badPower
Else     Bus_color = mixedPower
```

3.2.3 *Components and equipment colors*

For components and equipment, the color reflects their intrinsic integrity.

States	Component	Equipment	
Nominal	nominal behavior*	all components: Nominal	🟩
Loss	no longer works*	all components: Lost	🟥
Misleading	misleading behavior*	all components: Misleading	🟦
Degraded	-	at least one component: not Nominal	🟧
Off	-	no power supply	⬜

* due to a failure mode (random failure or development error), the power supply, zonal threats or common cause failures

3.3 Icons

The size of the icons determines the size of the "simulation" mode windows. Thus, special attention must be paid to them.

Taking into account the size and the orientation of the displays, the following distribution of inputs/outputs is recommended (left-right reading):
- Inputs on the left
- Outputs on the right
- ElecPower input at bottom left (close to the small lightning)
- Zonal input, CCF input and State output at top right

The name of the item may be above, below or even inside.

The following naming is recommended:
Edition mode:

- ExplicitName_0	no color

Simulation mode:

- ExplicitName_1	green
- ExplicitName_2	red
- ExplicitName_3	blue
- ExplicitName_4	orange
- ExplicitName_5	grey

3.4 Layers

To facilitate the presentation of models, items can be assigned to different layers, for example:

Layer 1:	Architecture	items and logical links
Layer 2:	ElecPower	electrical power supply links
Layer 3:	HydrauPower	hydraulic power supply links
Layer 4:	CCF	common cause failures items and links
Layer 5:	Zonal	threat impact and zonal links
Layer 6:	Functional	logics and links between organic view and functional view
Layer 7:	Init	initialization artefacts and links
Layer 8:	States	state artefacts and links
Layer 9:	Graphics	graphic items (lines, shapes, images, text, etc.)

It may be judicious to choose other layer names in certain views, for example in the zonal view:

Layer 1:	Zones	zones items
Layer 2:	Filters	artefacts that inhibit a threat on a zone
Layer 3:	GlobalThreats	items that impact several zones (particular risks, etc.)
Layer 4:	Zooms	zoom on a particular zone (with equipment list)
Layer 5:	Graphics	graphic items (lines, shapes, images, text, etc.)
Layer 6:	Links	links and zonal bus output

4. TYPES

There are 2 categories of types: enumerate and record. The "enumerate" types represent various flows that will link the different items. A "record" type gathers several "enumerate" types.

4.1 Directories

Types are distributed in families and sub-families as follows:

Frmk_Generic

Bus	record of generic flows and states
Flows	value of logical flows or measure of physical flows
Misc	modeling artefact flows (zonal, CCF…)
States	states of different items (components, equipement…)

Frmk_Power

Bus	record of power supply flows
Flows	power supply flows (with a direct impact on the behavior of an item)

4.2 Naming

Types globally follow the hereafter naming:

Flows/Misc:	*Name*Flow	Name: the type of logical / physical / artefact flow
	Example:	DataFlow, ElecFlow, CCF_Flow
Bus:	*Flow*Bus_*nn*	Flow: the main type of flows it embeds nn: the number of fields
	Example:	DataBus_03 is a bus of 3 DataFlow
States:	*Name*State	Name: the type of item
	Example:	ComponentState, EquipmentState, ZonalState

4.3 Specification

The specification of each type is defined in the “Properties” tab. The template is the following:

Enumerate type:

[description]
What the type represents.
****
[domain]
Value1: definition
Value2: definition
Value3: definition
****
Copyright if any.
****

Record type:

[description]
What the type represents.
****
Copyright if any.
****

Even if their definition varies from one flow to the other, the following domains are mainly used:

Frmk_Generic/Flows

Nominal	:	good value
Loss	:	no value or detected erroneous value
Misleading	:	undetected erroneous value

Frmk_Generic/Misc

Nominal	:	no impact on component
Loss	:	the component no longer works
Misleading	:	leads to a misleading behavior

Frmk_Power/Flows

goodPower	:	sufficient resource to ensure a nominal behavior
noPower	:	insufficient resource to keep the component working
badPower	:	degraded resource that impacts the behavior

Three different types of States are used:

Frmk_Generic/States/ComponentState

Nominal	:	nominal behavior
Loss	:	no longer works
Misleading	:	misleading behavior

Frmk_Generic/States/EquipmentState

Nominal	:	all components are in a Nominal state
Loss	:	all components are in a Loss state
Misleading	:	all components are in a Misleading state
Degraded	:	at least one component is not in a Nominal state
Off	:	no electrical power supply (or switched off)

Frmk_Generic/States/ZonalState

No_threat		
Bird_strike		
Engine_burst		
HIRF	:	High Intensity Radiated Field
Icing		
Lightning_strike	:	direct as well as indirect effects
Tire_burst		

The different values of the domain represent the zonal threats identified in the ZSA (Zonal Safety Analysis) and in the PRA (Particular Risks Analysis).
These threats may have different impacts on the components:

- no impact	ex: fireproof equipment, electromagnetic shield…
- “loss” impact	ex: the equipment is destroyed
- “misleading impact”	ex: HIRF threat without electromagnetic shield

4.4 Checklist

Here is a checklist of the various steps involved in creating a "Type" item:

Enumerate type	Does the item already exist?
	Create the item in the right directory
	Enumerate choice
	Name the item
	General tab: name the values of the domain
	General tab: define the default colors for each value (§3.2.1)
	Properties tab: write the specification of the item
	Save

Record type	Does the item already exist?
	Create the item in the right directory
	Record choice
	Name the item
	General tab: define the fields (name, type, orientation)
	General tab: define the default link to display
	Properties tab: write the specification of the item
	Save

5. OPERATORS

Operators are logical functions (in the mathematical/computer sense) that manipulate the values of different flows.

5.1 Directories

Operators are distributed in families and sub-families as follows:

Frmk_Generic
    ComponentState
    Icon

More generally, operators are stored in sub-families named after the type of their output (which is unique). *Example*: DataFlow, CCF_Flow, ElecPower, ZonalState…

5.2 Naming

The prefix "op_" is used for the operators, followed by an explicit description, as far as possible.
Example: op_ComputeHW_ComponentState, op_Equipment_Icon…

5.3 Specification

The specification of operators is defined in the “Properties” tab. The template is the following:

[description]
What is the global function of the operator.
****
[logic]
What logic is implemented, in literary format.
****
Copyright if any.
****

5.4 Checklist

Here is a checklist of the various steps involved in creating an "Operator" item:

Operator	Does the item already exist?
	Create the item in the right directory
	Name the item
	General tab: choose the type of the operator
	General tab: name the operands and their types
	Properties tab: write the specification of the item
	Altarica code tab: write and comment the code in AltaRica language
	Syntax and Consistency checks
	Save

6. COMPONENTS

There are 2 categories components:

- basic blocks which can embed states, events to switch from one state to the other (that can represent failure modes, zonal threats, functional actions…) and icons (to reflect the states in simulation mode). These blocks can be used to create equipment.
- graphical operators, which do not embed any event but only logical behaviors

The components are the only items that can embed events.

6.1 Directories

Components are distributed in families and sub-families as follows:

Frmk_Generic	
Functional	basic blocks related to the functional view
Organic	basic blocks related to the organic view
Zonal	basic blocks related to the zonal view
Frmk_Power	
Electrical	basic blocks related to the electrical power supply
Hydraulic	basic blocks related to the hydraulic power supply
Frmk_Tools	graphical operators, arranged according to their output type
CCF_Flow	
DataFlow	
ElecPower	
EquipmentState…	

6.2 Naming

The basic blocks components globally follow the hereafter naming: Name_PossibleStates.

Example: Function_NLM, ElecPower_NLM, HW_Data_NL…
(N for Nominal, L for Loss, M for Misleading)

The graphical operators components globally follow another naming:
Name_PossibleStates(with priority)_NumberOfOperands

Example: Or_NLM_2, Or_MLN_3, Or_GBNo_2…
(G for goodPower, B for badPower, No for noPower)

Inputs / Outputs / Local variables / Events:

For components inputs, use the “i” prefix and for outputs, use the “o” prefix. The rest of the word may be generic or refer to the item from which it comes / to which it goes.

Examples: iElecPower, iZonal, iAntenna1, oData, oState, oTransceiver2…

For local variables, use the “l” prefix, and for events, use the “e” prefix.
Examples: lState, eLoss, eMisleading…

6.3 Specification

The specification of components is defined in the “General” tab, Properties/Comment part. For the basic blocks components, the template is the following:

[description]
What does the component represent.
****

[behavior]
What is the behavior of the component, depending on its events (failure modes, functional action, zonal threats…), in literary format.
****

[assumption]
Assumptions that have been made.
****

[remark]
For which purpose can it be used.
****

Copyright if any
****

For the graphical operators, the template is close to the template of operators:

[description]
What is the global function of the graphical operator.
****

[logic]
What logic is implemented, in literary format.
****

[remark]
For which purpose can it be used.
****

Copyright if any.
****

6.4 Checklist

Here is a checklist of the various steps involved in creating a "Component" item:

Component	Does the item already exist?
	Create the item in the right directory
	Name the item
	General tab: choose the icon in edition mode and define its size
	General tab, Properties part: write the specification of the item
	I/O tab: define inputs/outputs/local variables (name, type, position)
	States tab: define the different states, their type and the default value
	Events tab: define the failure modes, functional actions, zonal threats…
	Icons: define the icons in simulation mode (pay attention to their order) The colors are defined in §3.2.3.
	Altarica code tab: write and comment the code in AltaRica language
	Syntax and Consistency checks
	Save

7. EQUIPMENT

The equipment is, according to Cecilia's definition, a set of components and/or equipment. Therefore, it can either represent a part of a real equipment, a real equipment, a sub-system, a system, the integration of several systems or a view.

Excluding the graphical operators, the choice has been made to use only equipment in a MBSA model.

7.1 Directories

Equipment are distributed in families and sub-families as follows:

Frmk_Generic

Common	equipment shared by some domains
Communication, Avionics…	equipment specific to certain domains

Frmk_Power

Electrical	equipment specific to the electrical power supply
Hydraulic	equipment specific to the hydraulic power supply

A **Generic_Project** family, with **Systems** and **Views** sub-families, gathers the high-level views of the project that presents the metamodel, with functional, zonal and organic views (cf §8.3).

7.2 Naming

The naming of the equipment should be as explicit as possible. The suffixes "system" and "views" are used when necessary.

Examples: Display, Calculator, ControlPanel, Battery, ElectricalSystem, OrganicView…

The naming of inputs, outputs and local variables is identical to that of components (§6.2).
The naming of layers is defined in §3.4.

7.3 Specification

The specification of equipment is defined in the "General" tab, Properties/Comment part. The template is the following:

```
[description]
What does the equipment represent.
***************************************************************
[behavior]
What is the high-level behavior of the equipment,
depending on its components and on zonal threats.
***************************************************************
[assumption]
Assumptions that have been made.
***************************************************************
[remark]
For which purpose can it be used.
***************************************************************
Copyright if any.
***************************************************************
```

7.4 Checklist

Here is a checklist of the various steps involved in creating an "Equipment" item:

Equipment	Does the item already exist?
	Create the item in the right directory
	Name the item
	General tab: choose the icon in edition mode and define its size
	General tab, Properties part: write the specification of the item
	I/O tab: define inputs/outputs/local variables (name, type, position)
	Content tab: drag & drop the components and/or equipment, name and link them
	Synchronizations tab: define the synchronizations, if any (cf §**Erreur ! Source du renvoi introuvable.**)
	Icons: define the icons in simulation mode (pay attention to their order) The colors are defined in §3.2.3.
	Altarica code tab: write and comment the code in AltaRica language
	Syntax and Consistency checks
	Save

8. PROJECTS MODELS

The "projects models" are the only items that can be simulated. It can be either to validate components/equipment or to simulate a complete modeling (several views and/or systems).

The computations (Boolean equations generation or sequence generation) can only be made on these projects models.

8.1 Directories

In the hierarchy, the first level is called Project, the second level is called System. Different items can then created (Model, Tree, DSF and FMEA). The following deals only with Models.

Two projects are present by default, the **Bench** project to test and validate the components/equipment and the **Generic** project that presents the metamodel, with functional, zonal and organic views (cf §8.3).

The other projects may be adapted to your needs.

Example: **Aircraft_A320**	project		
Communication	system		
Comms		model	VHF + UHF + HF + Electrical
VHF		model	VHF system alone
UHF		model	UHF system alone
HF		model	HF system alone
Electrical	system		
Elec		model	Electrical system alone
Hydraulic	system		
Hydrau		model	Hydraulic system alone
FlightControl	system		
FCS		model	Flight Control System alone
Full_FCS		model	FCS + Electrical + Hydraulic

8.2 Specification

The specification of a project model is defined in the “Properties” tab. The template is the following:

[description]
What does the model represent, what is the context.
****

[library version]
Version of the library that has been used.
****

[model version]
Version of the model (number, date…) and
differences with the reference version.
****

[model specification]
Reference of the excel file which embeds the model
specification.
****

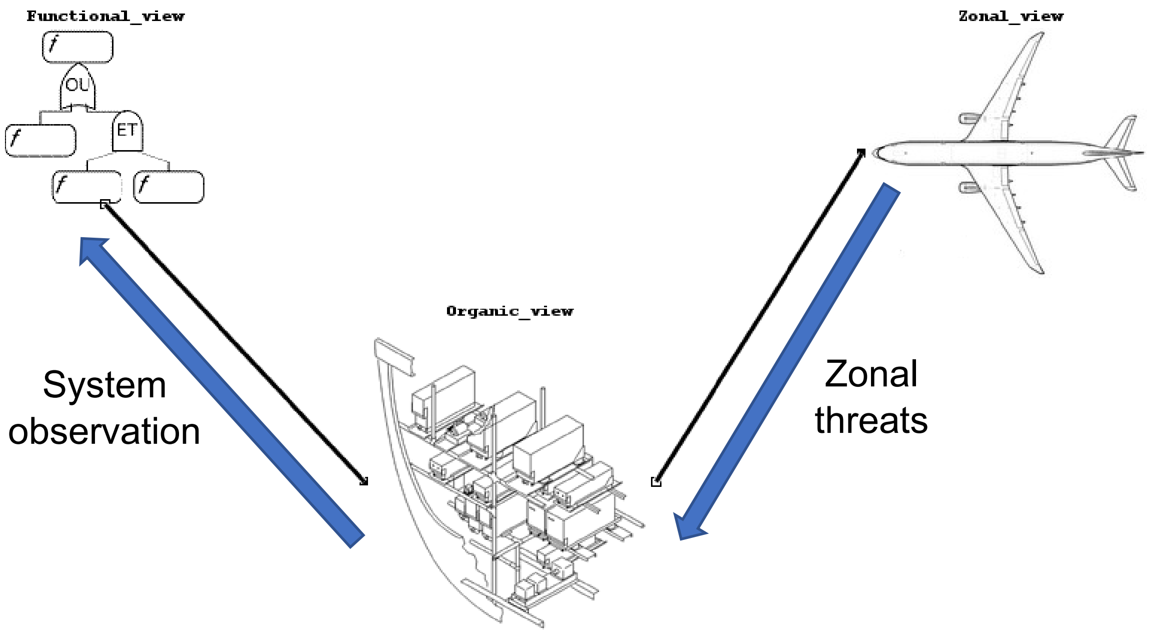
[to-do list]
List of ongoing activities or still to be done.
****

[remark]
Any other information.
****

Copyright if any.
****

8.3 Metamodel

In an aeronautic context, three views are recommended:



Functional view

In this view, functions are only observers of the states of certain equipment or of the value of certain flows placed in the ***organic view***. High-level logics can be implemented.

As FCs (Failure Conditions) correspond to the loss or malfunction of one or more functions, FC observers are placed in this view. They will be the targets of the different computations (Boolean equation generation or sequence generation).

Organic view

This is the view of system architecture, with different sub-systems and resource systems. Each item with a physical existence is linked to a zone in the ***zonal view***.

Zonal view

The areas of the aircraft are divided more or less finely into zones, depending on the threats to which they are subject.

The threats are defined in the Zonal Safety Analysis and in the Particular Risks Analysis. They are represented by events in each "zone" model. Some threats can be filtered in case the zone is not subject to the threat.

If a threat extends to several zones, events are synchronized ("synchronization" type, cf §**Erreur ! Source du renvoi introuvable.**) between the different zones concerned.

8.4 Systems integration

The integration of several systems (as simulable models) must be taken into account at an early stage to minimize the final work.

The example used for the demonstration is a complete communication system, comprising the HF, UHF and VHF subsystems and the electrical system (included in the organic view), plus the zonal view and the functional view.

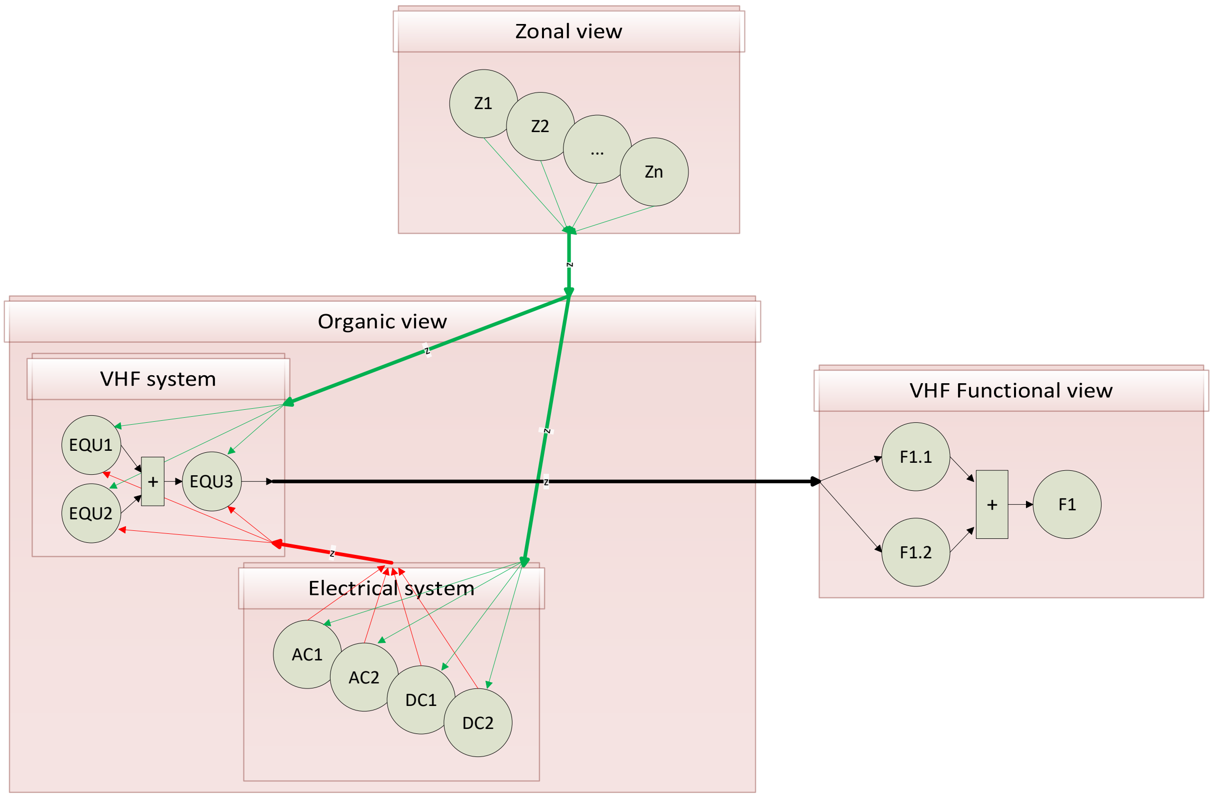
The different views (as well as the different subsystems) are linked through record types (bus).

<u>Issues</u>:

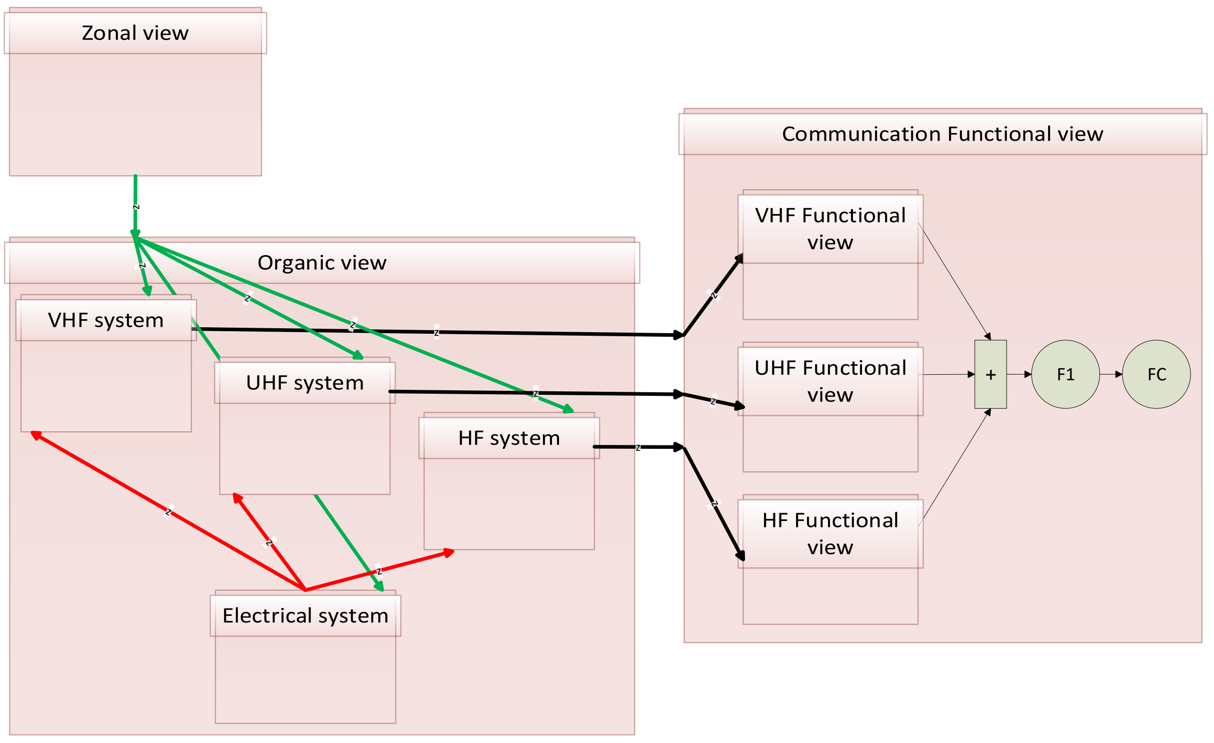
The zonal view is unique, as well as the electrical system.

It is interesting to first test a model:
{functional view + zonal view + electrical system + subsystem}

In the end, we need to test a complete model:
{functional view + zonal view + electrical system + complete system}



Subsystem model overview example



Complete system model overview example