CHESS

CHESS V3.1 News

EDT TEAM

Backward compatibility

The following actions need to be manually performed upon CHESS models created with CHESS tool v <3.1 (Papyrus editor can be used):

- Apply CHESS::Dendability::MitigationMeans profile to the ComponentView Package
- Apply MARTE::MARTE_DesignModel::SRM::SW_Concurrency profile to DeploymentView Package

Flow Port (PIM – Component View – Functional View)

Impact on CHESS-ML:

FlowPort can be used to model event-based interactions between ComponentImplementation and the middleware, not for interactions between components.

Constraints:

- In the ComponentView, FlowPort cannot be connected.
- In the ComponentView, FlowPort must not be bidirectional.
- They cannot have any extra functional information attached.

Managing property/port with multiplicities > 1 (schedulability analysis)

Impact on CHESS-ML:

- In composite structure diagram (component view only) it is allowed to model property and port with multiplicities > 1.
- We assume as valid the following cases only (defined by MDH)

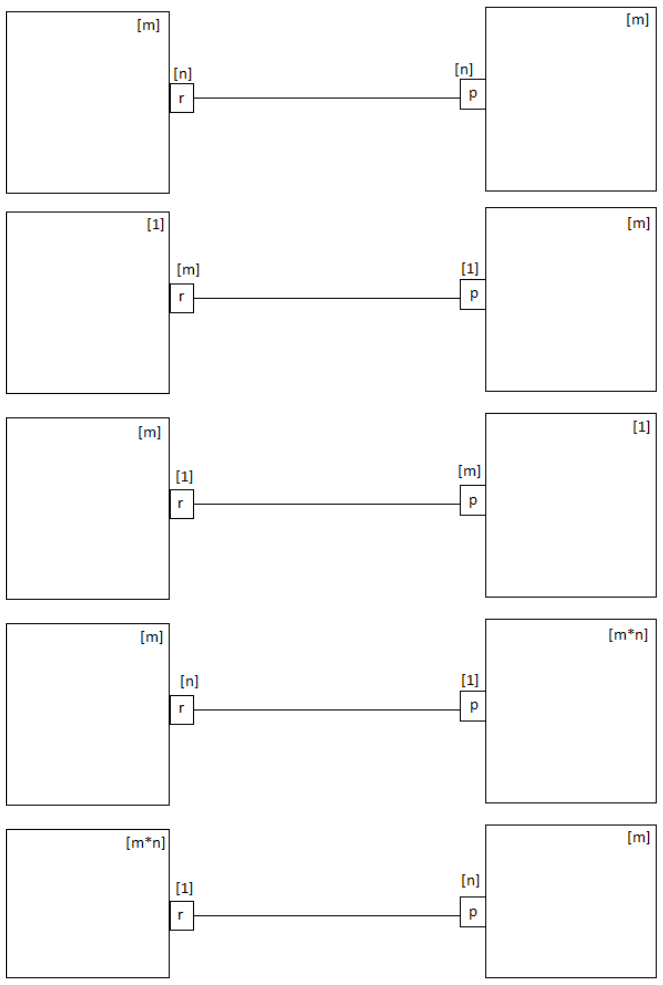


Figure 1: allowed property/port multiplicities

- The same extra functional annotations applied to a given Port or Property having multiplicities > 1 are assumed for all the instances mapping on that Port or Property

- Properties in CHRtSPecification used to store information resulting from the analysis have [0..*] as multiplicities, this in order to be able to store in the model analysis results for each instance represented by a Property with multiplicities N > 1, in particular
    - respT[0..*], where respT[k] = (ID=k, value=XX.X) , with k the k-th instance, 0≤k<N
    - blockT[0..*], where blockT[k] = (ID=k, value=XX.X), with k the k-th instance, 0≤k<N

Figure 2 shows an example of CHRtSPecification in case of Property with multiplicity > 1. Note that in this case the values for blockT and respT in Papyrus will be available through the property editor.

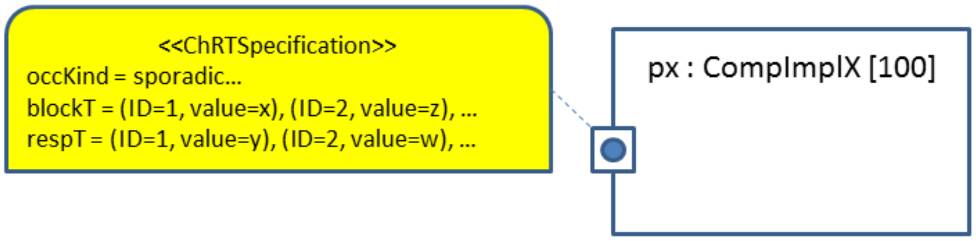

<<ChRTSpecification>>
occKind = sporadic...
blockT = (ID=1, value=x), (ID=2, value=z), ...
respT = (ID=1, value=y), (ID=2, value=w), ...
px : CompImplX [100]

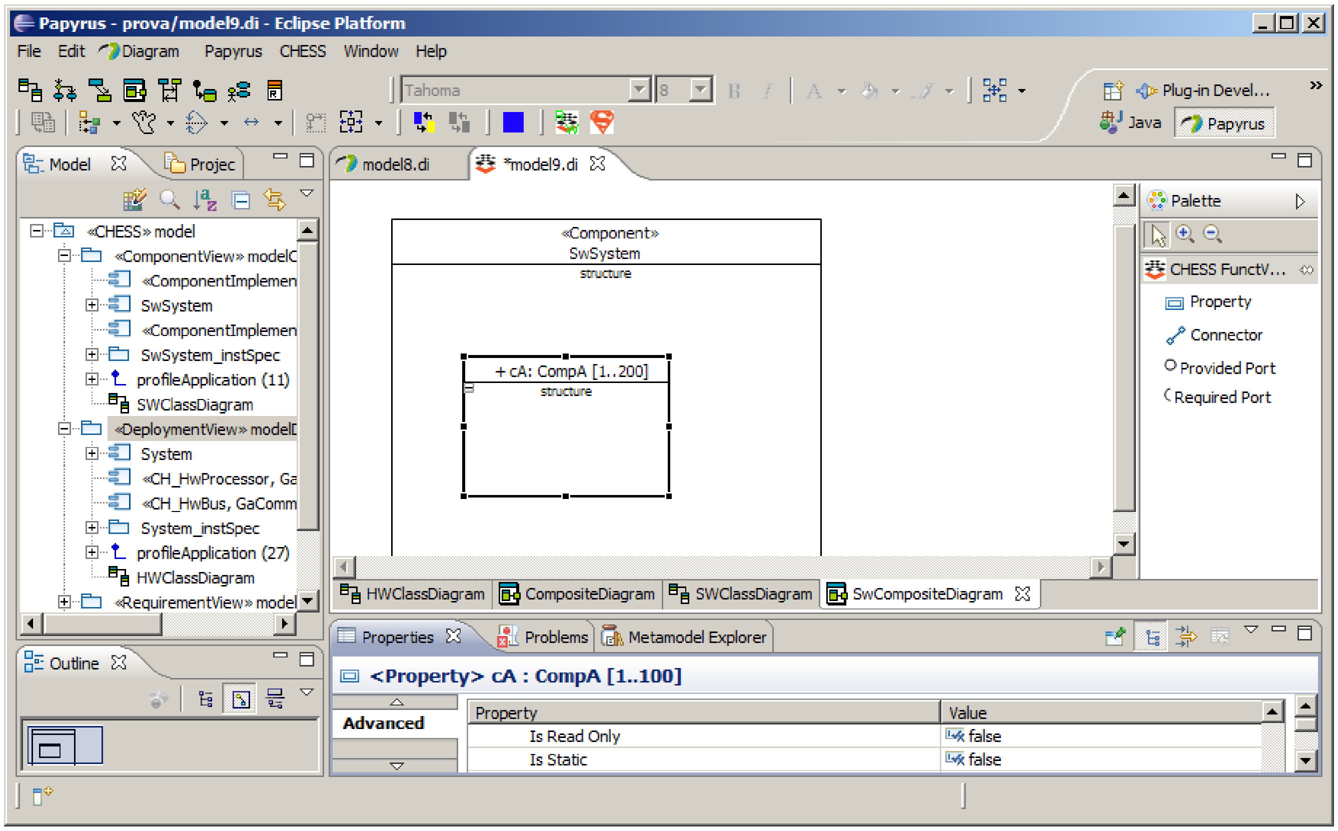
Figure 2: Example of CHRtSpecification annotation for Property with mult>1

About Deployment of Multiple Instances

Through the Build Instance only one InstanceSpecification fro Property is generated even if the Property has multiplicity greater than 1; then this InstanceSpecification can be chosen by the user as value for the *Assign.from* field and the modeler can use **NfpConstraint** to specify the proper set of instances to be allocated on the same hardware.

Then when the user has finished with the deployment and just before PIM2PSM transformation, a PIM-PIM transformation is automatically applied to derive the proper set of InstanceSpecifications (invisible to the modeler) which are then given in input to PIM to PSM transformation, e.g. to allow schedulability analysis.

Figure 3 shows an example of this solution: let’s suppose to have a Property of type CompA with multiplicities 200



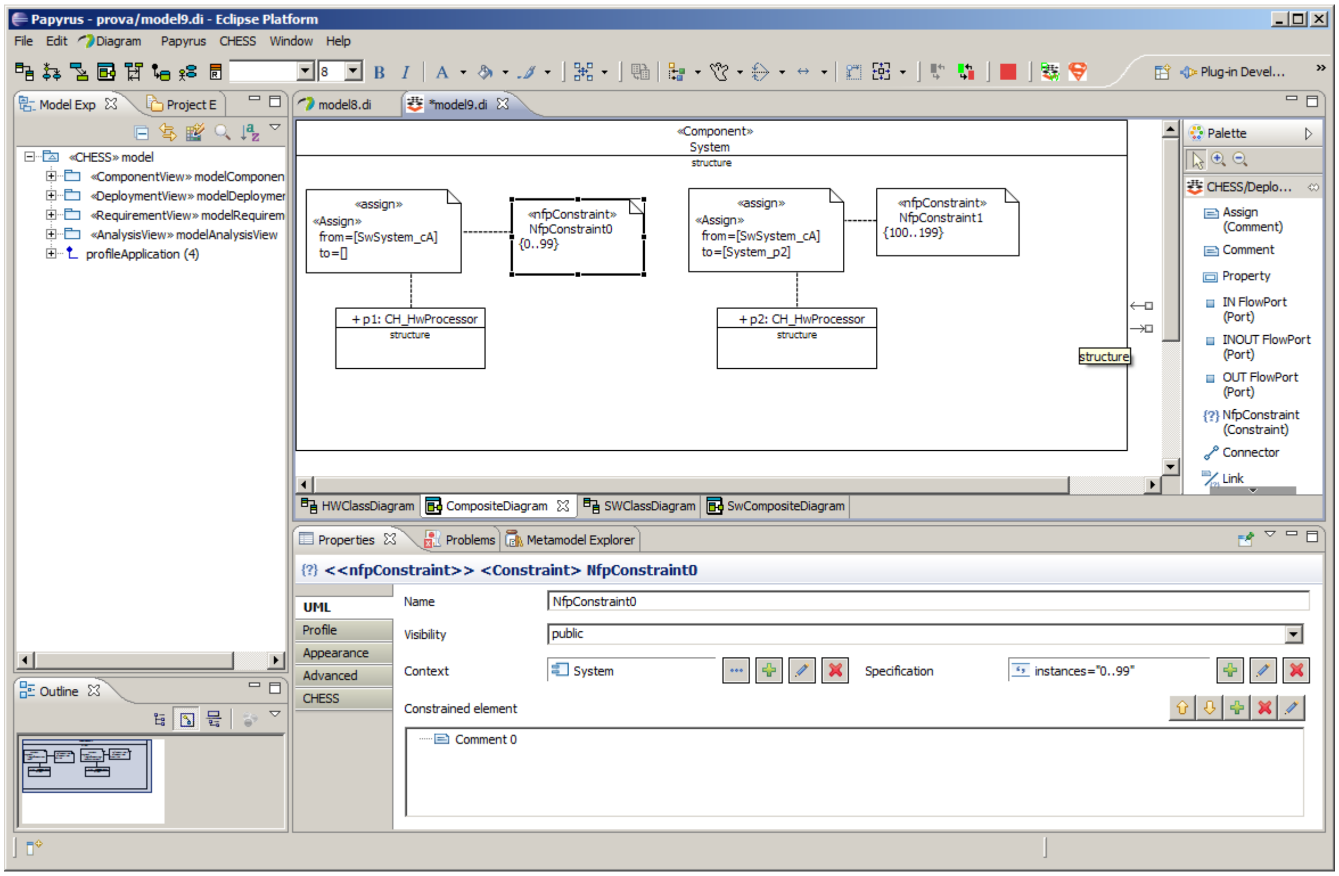
<Property> cA : CompA [1..100]

Advanced

Property	Value
Is Read Only	false
Is Static	false

Figure 3 – example of Property with multiplicity > 1

Then to model the scenario where instances from 0 to 99 are deployed on a processor (or partition, see below) and instances from 100 to 199 are deployed on a different processor (or partition on the same processor, see below), then we can use the NfpConstraint as in the following figure.



<<nfpConstraint>> <Constraint> NfpConstraint0

Name	NfpConstraint0
Visibility	public
Context	System
Specification	instances="0..99"
Constrained element	Comment 0

Figure 4: deploying a multiple instance

Note: NFPConstraints for multi-instance deployment must contain a string of the form "[x..y]".

Note: Currently the Assign.impliedConstraint attribute cannot be set in Papyrus 0.8.1, so Links have to be used to associate the NFPConstraint to the given Assign.

So, in order to model the deployment of a multiple instance the following steps have to be followed:

1. create the InstanceSpecification through the Build Instance command,
2. in the Assign.from field chooses the InstanceSpecification representing the multiple instance,
3. uses the NfpConstraint to specify the actual set of instances to deploy,

then, when the PIM-PSM transformation is invoked, a PIM-PIM transformation is automatically executed to generate, starting from the InstanceSpecification representing the multiple instance, all the corresponding InstanceSpecifications.

Deployment: partitioning SW component instances

Impact on CHESS-ML:

MARTE::SRM::MemoryPartition stereotype (applied to Component) is used to model logical partition (Figure 6).

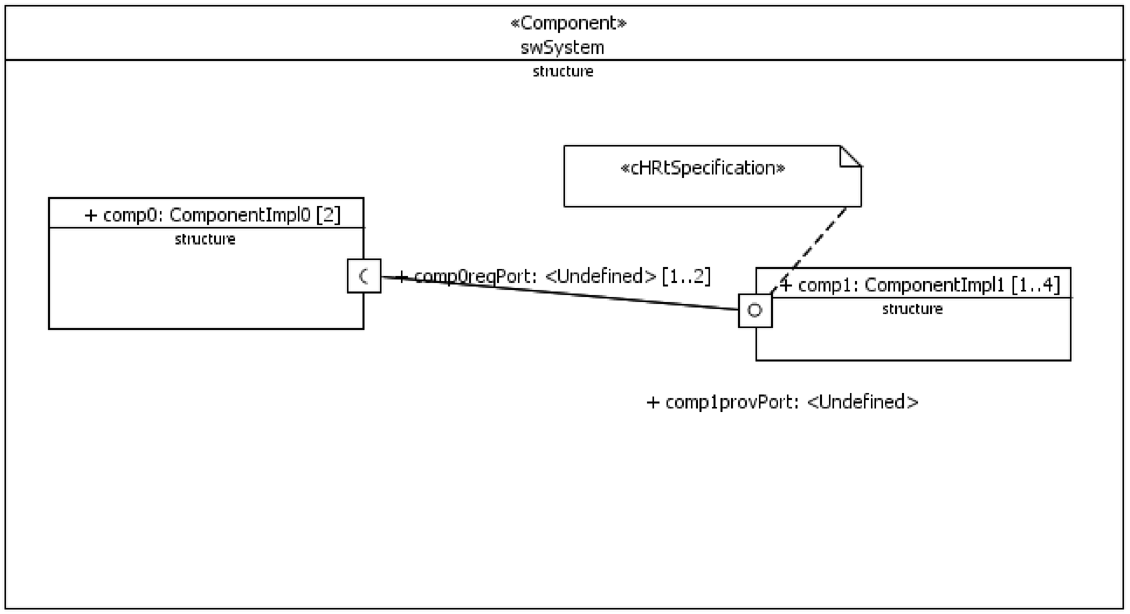

«Component»
swSystem
structure
«cHRtSpecification»
+ comp0: ComponentImpl0 [2]
structure
+ comp0reqPort: <Undefined> [1..2]
+ comp1: ComponentImpl1 [1..4]
structure
+ comp1provPort: <Undefined>

Figure 5: example of SW Property/Port with mult > 1

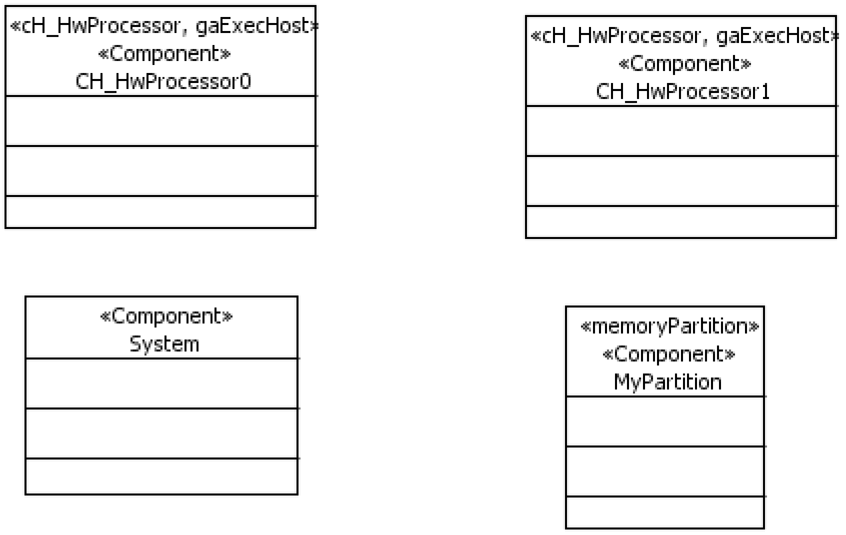


Figure 6: modeling partition (as type)

Then MemoryPartition (its representing InstanceSpecification) can be allocated to HwProcessor through Assign/Allocate. Moreover Assign can be used to map sw instances to partition instances (Figure 5 and Figure 7).

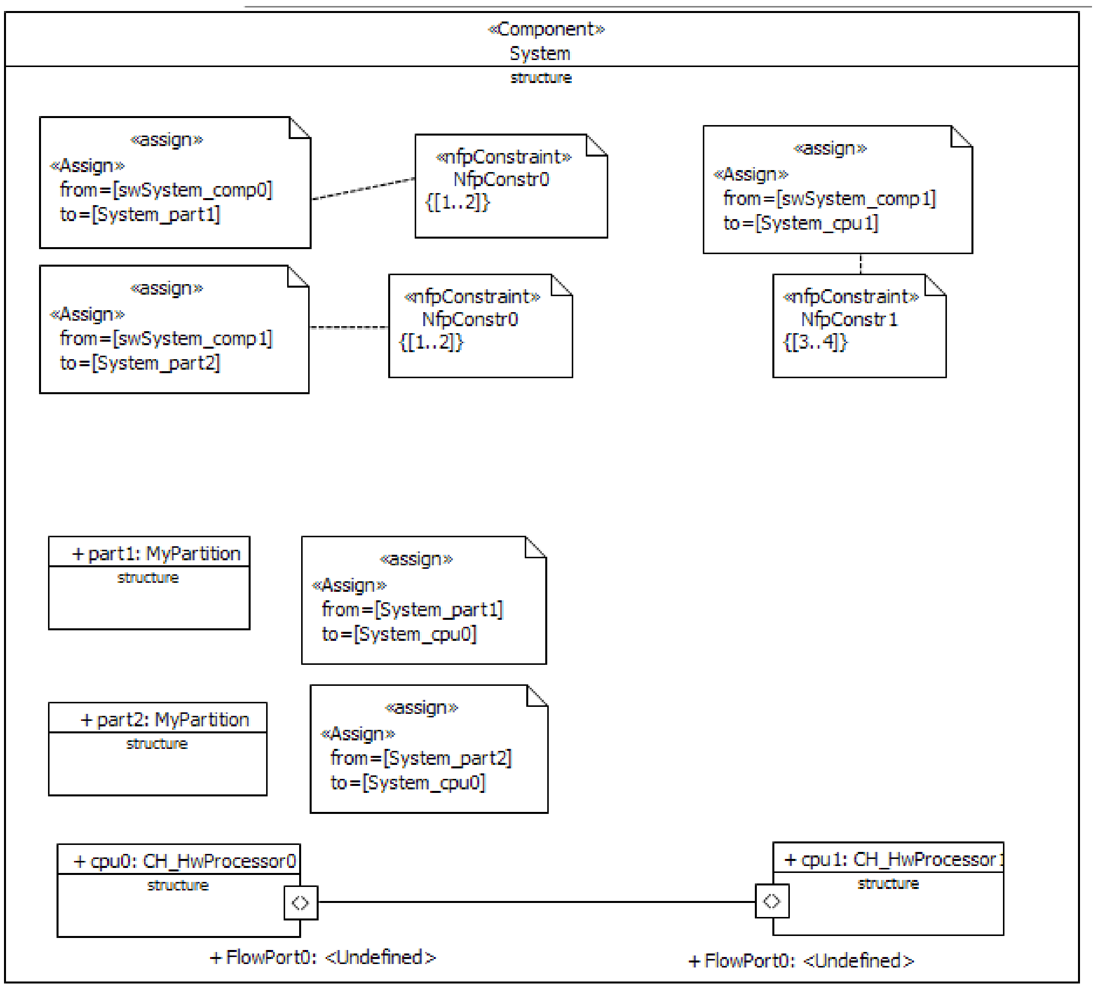


Figure 7: Example of Deployment with multi-instances and partitions

Note for Telecom domain: interactions between SW instances allocated on the same partition are implemented as function (synchronous) calls. Interactions between instances allocate on different partitions are implemented as signal (asynchronous) calls (which, in the CHESS component model jargon, are mediated by Connectors).

So, for example, in order to have the kind of communication depicted in Figure 8 it is necessary to map NciHandler_inst, NetConn_inst, Scihandler_inst and NodeConn_inst onto the same partition, while the remaining instances NCIClient_inst, AAL2RIClient_inst and SPAS_inst have to be mapped onto different partition(s).

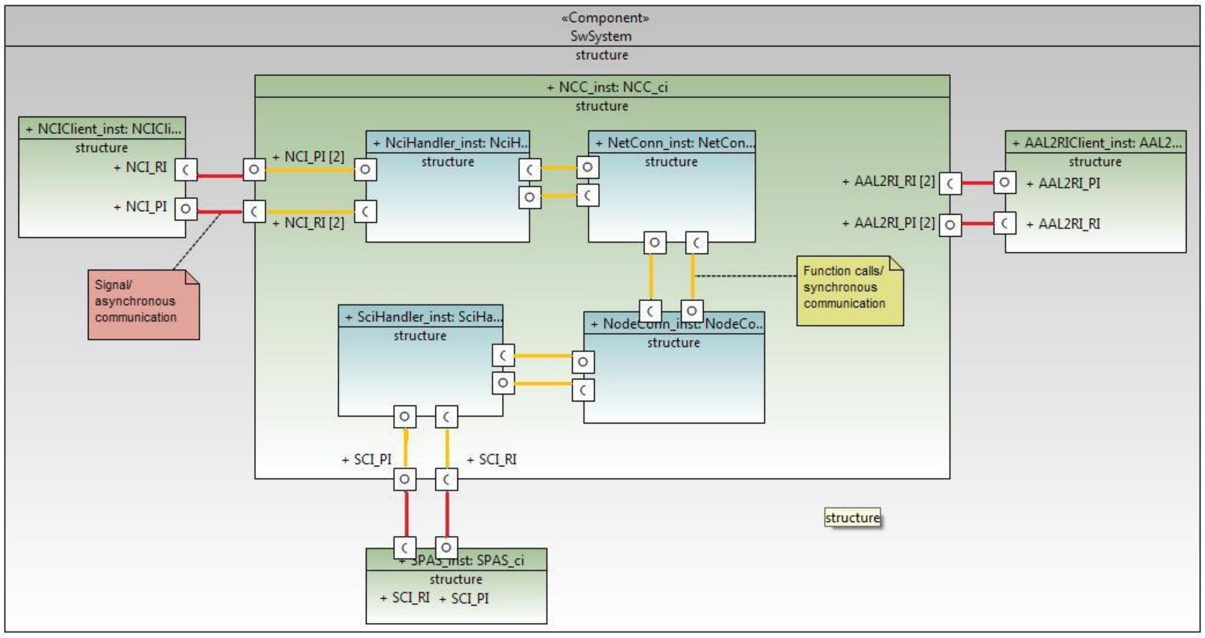


Figure 8: example from Telecom

In CHESS, the modeling of composite ComponentImplementation (as in Figure 8**) is currently not allowed. This is surely a feature that could be added in a CHESS+ project.**

However it is worth noting that the process of mapping a set of ComponentImplementation instances to a logical partition, that will be available in CHESS, can be compared to a process through which a composite parent entity is created in a bottom-up way.

Other new refining/extension of batch validations

Given a ComponentType realizing an Interface X, the component has to define the same operations defined in X at least.

A ComponentImplemention has to define the same operations defined in the realizing ComponentType(s)

Railway View

A domain specific view for Railway has been implemented on top of the ComponentView/ExtraFunctionalView: this view allows to specify transmissions between software component which need specific protocol (e.g. CRC).

FI4FA Analysis (MDH).

See documentation provided for user training.

Simulation Timing Based Analysis (FhG)

See documentation provided for user training.

Code generation taking into account deployment partition (MDH)

See documentation provided for user training.

Ada infrastructural code generation (UPD)

New features:

- Component implementations written in C or comprising imperative C++ code are now supported;
- Added protected code sections in Ada component implementations to define local variables, package body declarations and private declaration (in the implementation specification);

Bug fixes:

- Fixed two mistakes in one conditional branch for the generation of OBCS and sporadic operations.

Current limitations:

- No support for <<MemoryPartition>> yet;
- Partial support for generation of types (Integer, Float and Enumeration tested with parameter passing across Ada and C components);
- No support is planned for instances with multiplicities;

Plan for next incremental release:

- Mapping of railway-specific stereotype <<TrasmissionWithProtocol>> to code patterns;
- Support for structured and constrained datatypes.