ngVLA Architecture Modeling Plan

020.10.20.00.00-0003-PLA

Status: **RELEASED**

PREPARED BY	ORGANIZATION	SIGNATURE
R. Hiriart	ngVLA, NRAO	Rafael Hiriart

APPROVALS	ORGANIZATION	SIGNATURES
P. Kotzé, Systems Engineer	ngVLA, NRAO	Pieter Kotzé Pieter Kotzé (Jul 16, 2024 14:35 MDT)
E. Murphy, Project Scientist	ngVLA, NRAO	Eric J. Murphy
R. Selina, Project Engineer	ngVLA, NRAO	RS Rob Selina (Jul 16, 2024 14:57 MDT)
W. Hojnowski, Project Manager	ngVLA, NRAO	William Hojnowski William Hojnowski (Jul 16, 2024 15:30 MDT)

RELEASED BY	ORGANIZATION	SIGNATURE
W. Esterhuyse, Project Manager	ngVLA, NRAO	

Change Record

Version	Date	Author	Reason
1	20 Jan 2021	T. Küsel	First high-level draft for comment by Architecture team.
2	3 Mar 2021	T. Küsel	Second draft with inputs from Architecture team.
A	15 Mar 2021	A. Lear	Prepared PDF for approvals and release.
B	4 Mar 2024	P. Kotzé	Section 10 updated to include Interface identification numbering, and ICD framework definition

Table of Contents

1 Introduction

1.1 Scope of this Document

This document defines the system architecture modeling approach for the ngVLA project. The modeling work is in support of the Systems Engineering activities as defined in the SEMP [AD01]. This document defines

- the objectives of the modeling,
- scope of work,
- model outputs,
- model inputs,
- model implementation,
- roles and responsibilities, and
- phasing of the model development.

The purpose is to set clear goals and guidelines before starting with the modeling effort, to ensure an effective and efficient model implementation.

2 References

2.1 Applicable Documents

The following list of documents is applicable to this document to the extent specified. If not stated otherwise, the latest released version of the document in the repository is applicable.

Precedence is indicated in the table below as either "this doc", indicating that this document takes precedence, or "ref doc", in which case the reference document takes precedence.

Ref. No.	Document Title	Precedence	Document Number
AD01	ngVLA Systems Engineering Management Plan	Ref doc	020.10.00.00.00-0001 PLA

2.2 Reference Documents

The following documents are referenced within this text or provide supporting context.

Ref. No.	Document Title	Document Number
RD01	ngVLA Product Breakdown Structure	020.10.20.00.00-0004 DSN
RD02	ngVLA N-Squared Matrix	020.10.40.00.00-0001 DWG

3 Objectives

The purpose of the architectural modeling activity is to support the Systems Engineering effort as defined in [AD01], mainly during the design phases (Conceptual, Preliminary, and Final design phases). The purpose is also to support the development and management of the ngVLA software development.

The goals of the activity fall broadly into the following categories:

1. Support the development of system and subsystem ***requirements specifications***. The model should enable the central management of all requirements from L0 to L2 and lower levels as applicable. This includes the ability to capture requirements, allocate them to functions and products, and maintain full traceability across the system hierarchy. The model should serve as the source of requirement specification documents, including all the relevant contextual information such as context diagrams and interface identification.
2. The model should be used as a central reference for the ***definition of the system architecture***. This includes the product's structural architecture and all managed system interfaces. It also includes the functional architecture and behavior (functions, functional flow control, data flow, states, and modes) and functional interfaces.
3. The model should be used to ***verify the completeness and internal consistency*** of the system architecture. The model should reflect the relationships and traceability between the different model elements (requirements, functions, and structure) that will allow the cross-checking and detection of inconsistencies and gaps in the architecture definition.
4. Support the development of system and subsystem ***verification*** plans. The model should enable a structured approach for developing verification plans with full traceability to product requirements. This includes the definition of verification requirements and methods, and the definition of verification events.
5. Support the evaluation of ***impact of engineering changes*** to the system requirements and system architecture. The model should enable easy tracing of impact of changes to requirements and architectural elements.
6. Support the ***software architecture development*** by enabling the modeling of the software architecture alongside the system architecture with traceability between them.
7. Harmonize the system model with other ***project data models***. One clear example is the M&C data model, where the system model will be used as a centralized mechanism to define and control the M&C interfaces across the system. Other models may include the Science data model, Calibration data model, Proposal Management data model, and Telescope Configuration data model.

System modeling can take up a significant amount of resources and time, and caution should be taken to not over-complicate the model. A further objective is thus to keep the model as simple as possible while achieving the above stated goals. More detailed Subsystem models may be kept as separate models with import/export links to the System model.

4 Scope of the Architectural Model

An overview of the scope of the architectural model is illustrated in Figure 1 below, showing the elements of the model, key inputs and outputs on the various levels of the system hierarchy, responsibilities, and project timing.

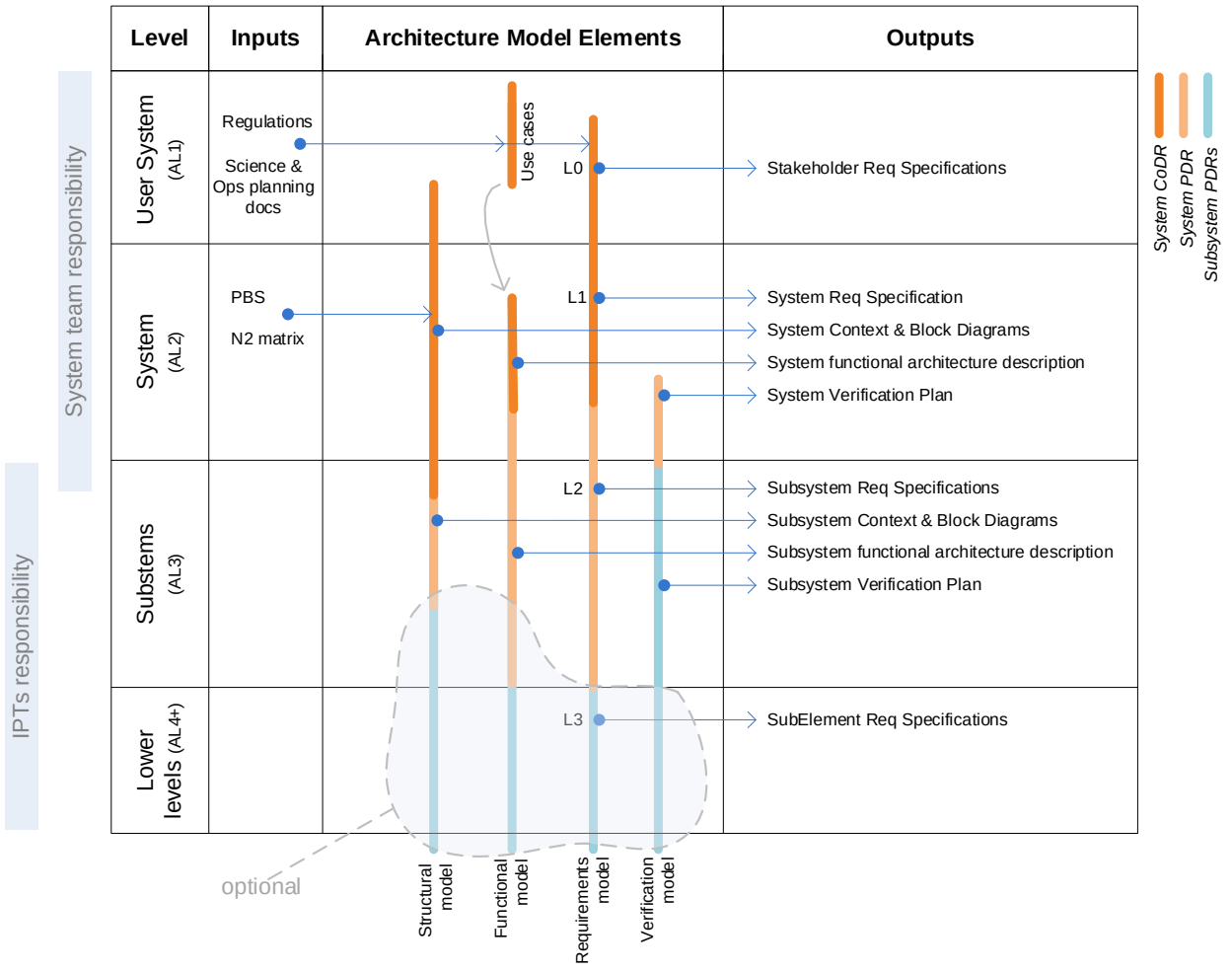

Figure 1: Overview of architecture model elements, inputs, and outputs.

4.1 Subsystem Models

For hardware subsystems, a top-level functional and structural model is required as a minimum. More detailed modeling of a hardware subsystem's structural and functional architectures is optional. However, for software subsystems it is required to develop detailed behavioral models as part of the software architectural design effort. The software models may be developed separately, but there shall be a method to synchronize the system-level behavioral model with the software behavioral model.

4.2 Definition of Architecture Layers

The Architecture Layers are identified in [RD01] and are broadly defined as follows:

1. Architecture Layer 1 (AL1) is the Observatory-level architecture, which defines the interfaces and interaction between the Telescope, Enabling Systems (training systems, development systems, etc.), User Systems (Science user systems, maintenance & support systems, etc.) and external systems.
2. Architecture Layer 2 (AL2) defines the interfaces and interactions between Telescope subsystems (e.g. Antenna system, correlator-beamformer, time and frequency subsystem, infrastructure, buildings).
3. Architecture Layer 3 (AL3) defines the interfaces and interactions internal to major subsystems, for example all the interfaces on the Antenna.
4. Architecture Layer 4 (AL4) defines the interfaces and interactions internal to components of a subsystem (e.g. interfaces internal to the Water Vapor Radiometer).

5 Modeling Language and Tool

SysML was selected as the modeling language and Cameo Systems Modeler ™ was selected as the modeling tool for the ngVLA. The reasons for the selection are as follows:

1. There is a legacy of using the language and the tool in the organization and a significant amount of training has been invested in the team.
2. SysML is a widely recognized modeling language for Model Based Systems Engineering. It is also flexible and allows a wide range of interpretation and application.
3. SysML is easily translated to UML for software implementation, thus creating a smooth path between system models, software models, and software implementation. This is a key value proposition of the modeling approach that is proposed for the ngVLA.

Risks of the approach include

1. Currently there are no expert users in the organization who have applied the tool to a large project such as ngVLA and the team will have to develop expertise during the build-up of the model.
2. The flexibility of the tool makes it more difficult for non-expert users to understand the best way of using it and how to avoid modeling pitfalls.

6 Model Inputs and Outputs

The key input documents for the model include the following:

1. User-level (L0) inputs:

- Science Requirements doc 020.10.15.00.00-0001-REQ
- Operations Concept doc 020.10.05.00.00-0002-PLA
- Stakeholder Requirements doc 020.10.15.01.00-0001-REQ

2. System-level (L1) inputs:

- Product Breakdown Structure [RD01]
- N-Squared Diagram (interface identification matrix) [RD02]

Note that this is not an exhaustive list and may be expanded as the need for more inputs arises.

Table 1 lists a summary of the model outputs, which are defined in more detail in Sections 8 to 10.

Table 1: Summary of architectural model outputs.

Category	Model output	Used where	Archtitecture Layer			Priority	Description/notes
			Observatory (AL1)	Telescope (AL2)	Subsys (AL3)		
Requirements							
	Requirements	Requirements Specifications		y	y	1	Requirements Specification Documents for System and Subsystems.
	Requirements Traceability matrix	Traceability checking		y	y	1	Table showing traceability between two levels of requirements. Used to check the requirements coverage up and down.
Verification							
	Verification method table	Requirements Specifications & Verification Plans		y	y	1	Table that shows for each requirement what verification method will be used for verification (Test, Analysis, Inspection or Demonstration)
	Verification requirements	Verification Plans		y	y	1	For each requirement, have a high level statement of how the requirement will be verified for final design review. Should specifiy the scope of the testing. Forms the seed for the test procedures.
	Verification events	Verification Plans		y	y	2	Defines Verification Events and which requirements are verified for each event.
Structural Architecture							
	Product Hierarchy	Requirements Specifications & Design documents	y	y	y	1	Hierarchical decomposition of an item in line with the Product Breakdown Structure.
	Context Diagram	Requirements Specifications	y	y	y	1	Diagram showing all the external interfaces of an item.
	Block Diagram	Design documents	y	y	y	1	Block diagram showing the internal structure of an item, including lower level components, interfaces between components.
Functional / Behavioural Architecture							
	Functional flow diagrams	Design documents		y	y	1	Diagrams that show an item's functions, control flow and object(data) flow between functions. Functions are modeled as a hierarchical decomposition with allocation to products.
	States & Modes diagrams	Design documents		y	y	2	Only used in limited cases as needed.
	Sequence diagrams	Functional ICDs		y	y	2	Required for modeling complex functional interfaces.
	Behavioural model	Software architecture & implementation		y	y	1	Model to be synchronized with SW implementation models.
	Software interface definitions	Software architecture & implementation		y	y	1	Model to be synchronized with SW implementation models for definition of data exchange interfaces.

7 Requirements Model

The scope of the requirements model includes

1. Capturing of all requirements on L0, L1 and L2.
2. Allocation of all requirements to the products.
3. Allocation of functional requirements to Functions (optional).
4. Requirements traceability across the product hierarchy from L0 to L2.

7.1 Requirements Model Outputs

User-level requirements model outputs:

- L0 Requirements Specification.

System-level requirements model outputs:

- L1 Requirements Specification.
- Requirements Traceability Matrix: L0 ← → L1
- Requirements Traceability Matrix: L1 ← → L2

Subsystem-level requirements model outputs:

- L2 Requirements Specification.
- Requirements Traceability Matrix: L2 ← → L3 where needed.

Lower-level requirements model outputs:

- L3 Requirements Specifications where needed.

7.2 Requirements Model Implementation

Figure 2 shows the implementation of requirements model entities and relationships in SysML. Downward expansion to Level 3 requirements is optional and up to the discretion of the L2 system designer.

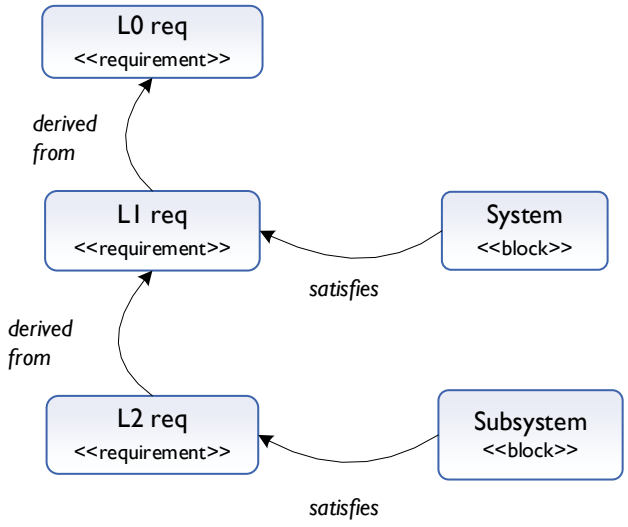

Figure 2: Requirements model implementation schema.

7.3 Requirements Model Package Structure

The proposed package structure for the requirements model is shown in Figure 3. This structure is not strictly prescriptive—it may evolve as the model matures.

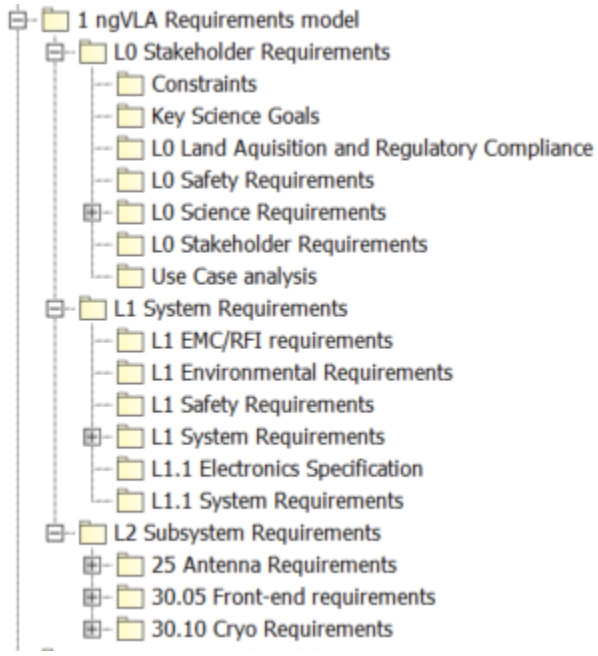

Figure 3: Requirements model package structure.

8 Structural Model

The scope of the structural model includes

1. A hierarchical decomposition of the System in line with the Product Breakdown Structure (PBS) as defined in [RD01]. Not all elements of the PBS are modeled, but only architecturally significant items. The naming and numbering of the items in the model shall correspond to the name and number allocated in the PBS.
2. Interfaces between PBS items, including

- Interfaces that need to be managed through Interface Control Documents (ICDs).
- Software-to-software interfaces that will be defined through a formal software interface definition language.

The identification of interfaces should be consistent with the interface identification defined in the N-squared matrix [RD02]. The model should contain a mapping between identified interfaces to ICDs.

8.1 Structural Model Outputs

8.1.1 User-Level Structural Model Outputs

User-level structural model outputs include

- Telescope context diagram, showing the interaction between the Telescope and the user systems and all significant external interfaces.

8.1.2 System-Level Structural Model Outputs

System-level structural model outputs shall include

- Subsystem context diagrams where needed (e.g. Antenna).
- Product structure (PBS) from the telescope level (AL2) and AL3, with Subsystems as leaf nodes.
- System Block diagrams from telescope level (AL2) and AL3, down to Subsystems as leaf nodes.

8.1.3 Subsystem-Level Structural Model Outputs

Subsystem-level structural model outputs shall include the following:

- Subsystem (AL4) context diagram.
- Subsystem (AL4) decomposition diagram.
- Subsystem (AL4) internal block diagram.

8.2 Structural Model Implementation

The product decomposition is modelled in SysML using the block definition diagram (bdd). The internal structure of a product, including its sub-parts, interfaces, and connections is modelled using the internal block diagram (ibd). The ibds are also used for context diagrams. The implementation of the structural modeling in SysML is shown in Figure 4.

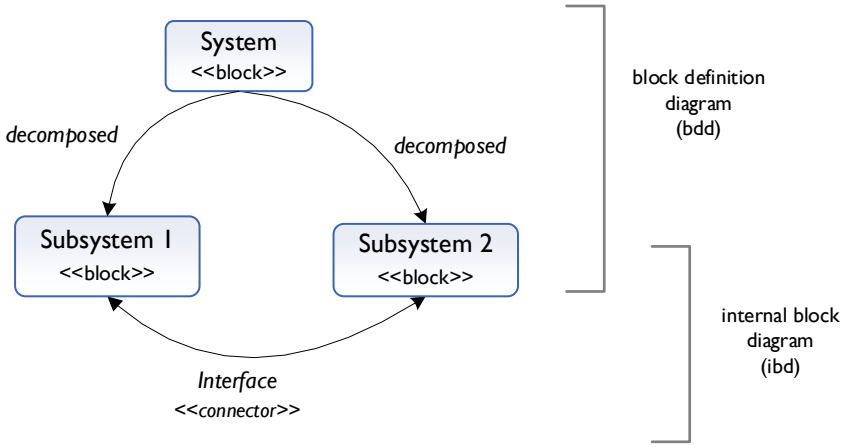

Figure 4: Structural model implementation schema.

Figure 5 (next page) shows a proposed set of block diagrams that should be developed, but this is not prescriptive and should evolve as the model matures. It is recommended to generate separate block diagrams for physical, M&C, signal path and LRT interfaces. Interfaces will be identified using the N^2 matrix [RD02].

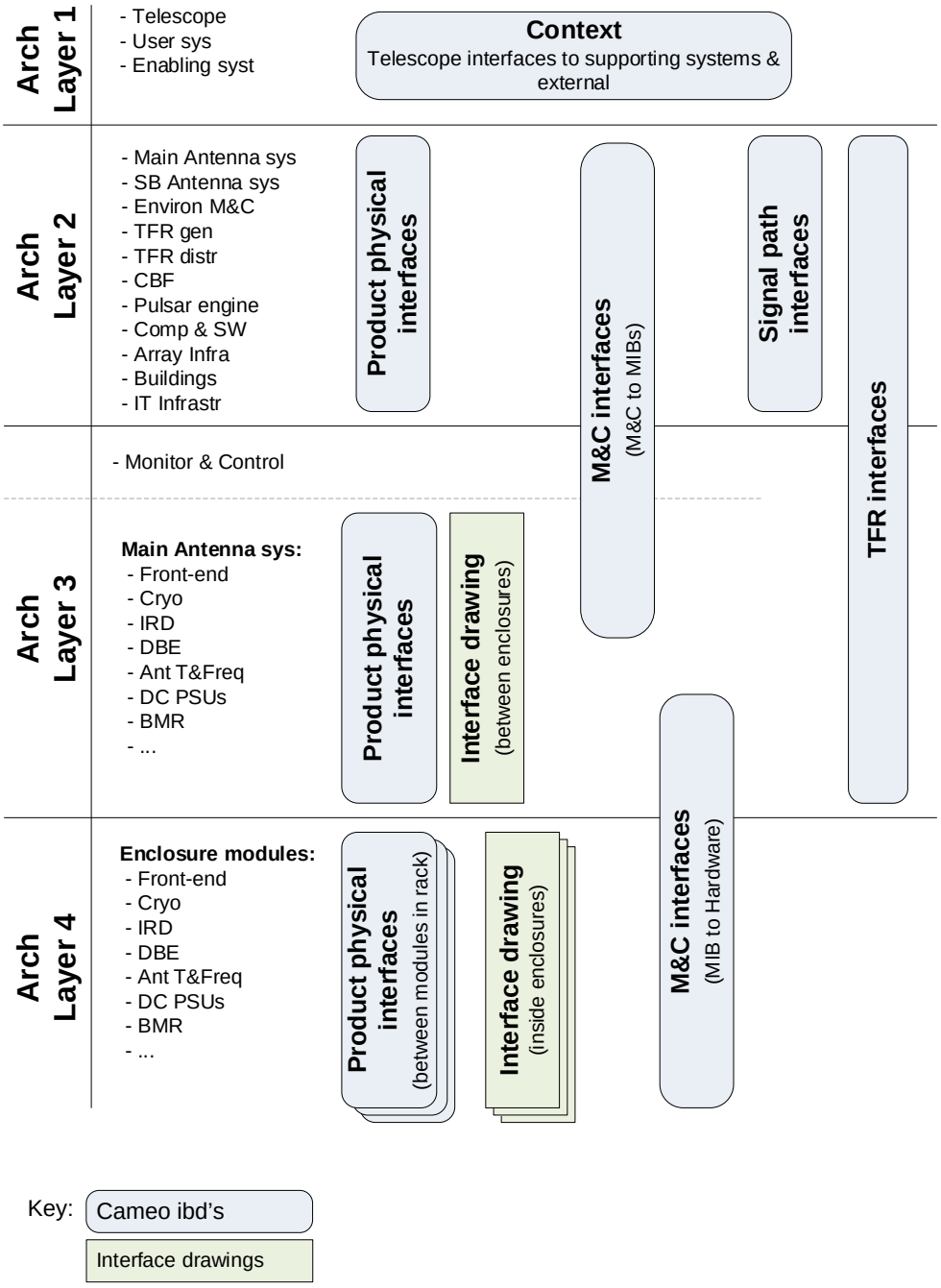

Figure 5: Preliminary identification of required ibds.

8.3 Structural Model: Package Structure

The proposed package structure for the structural model is shown in Figure 6 below. This is not strictly prescriptive and may be refined as the models matures.

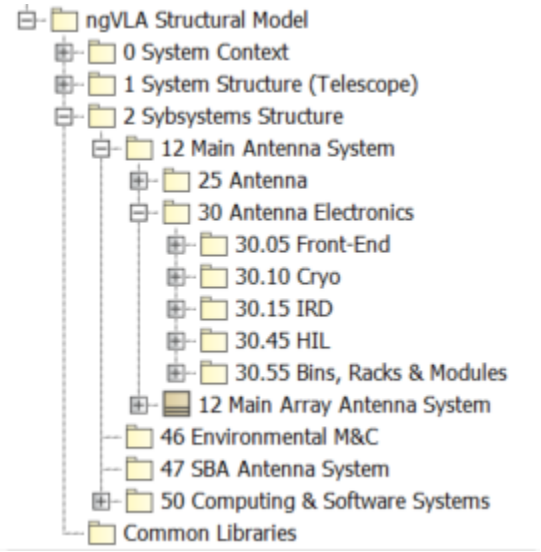

Figure 6: Structural model package structure.

9 Functional/Behavioral Model

The scope of the functional/behavioral model includes

1. Functions (SysML activities).
2. Functional concurrency and control flow (SysML activity diagrams).
3. Data flow between functions.
4. Allocation of data flows to functional interfaces.
5. Functions allocated to Product items.
6. Functional requirements allocated to functions (optional).
7. Functional interfaces only where complex functional interfaces exist that we need to manage.
8. States and modes where needed (e.g. Antenna system and software subsystems).

9.1 Functional/Behavioral Model Outputs

9.1.1 User-Level Functional Model Outputs

- Optional: Use case diagrams and use case specifications, showing interaction between telescope and telescope users. Use cases are developed for the purpose of eliciting or clarifying system requirements and system functions. Use cases should only be developed to the detail needed for this purpose.

9.1.2 System-Level Functional Model Outputs

- Telescope functional hierarchy diagrams from top-level telescope functions down to where functions can uniquely be allocated to Subsystems. Functions that are performed by multiple subsystems are

regarded system-level functions which should be decomposed until they can be fully allocated to one subsystem.
- Functional control flow diagrams (activity diagrams) for AL2 and AL3. This should include important data flowing between functions.
- List of data items carried by functional interfaces.
- Sequence diagrams for complex functional interfaces on AL2 and AL3.
- States and modes diagrams on AL2 and AL3 where needed.
- M&C interface definitions to be used in the M&C software models and code generation.

9.1.3 Subsystem-Level Functional Model Outputs

For each subsystem, at least the top-level functional model will be created from the system allocated functions. More detailed subsystem functional models are optional and only required for functionally complex subsystems that require the development and review of a functional architecture. This is applicable to software subsystems. Such model outputs should include

- Functional decomposition diagrams.
- Functional control flow diagrams (activity diagrams) with critical data flows.
- Data lists for functional interfaces.
- Sequence diagrams for complex functional interfaces as needed.
- States and modes diagrams on AL4 as needed.

9.2 Functional/Behavioral Model Implementation

The behavioral model is represented by a hierarchical functional structure implemented in SysML through activity diagrams. Functions are represented with activities. System-level functions are decomposed into lower-level activity diagrams until all functions can be allocated to subsystems. Complexity on any single activity diagram should be limited and a guideline is to have no more than nine activities in a single activity diagram.

Item flow or data flow between functions is represented through object flow. The Item Flow Manager is used to allocate data from the data model to (a) the object flow in the behavioral model and (b) to interfaces in the structural model. The implementation schema is shown in Figure 7 (next page), and an implementation example is shown in Figure 8 (next page).

Data items can either be represented as Value Types or Blocks. Complex data structures should be represented by Blocks, which will enable the software developers to use the data model for code generation. All M&C data shall be represented by Blocks. M&C interfaces shall use Proxy Ports to interface with products in the structural model. Proxy Ports are used for non-physical interfaces such as signals and data. Full ports are used to model physical interfaces such as mechanical interfaces and connectors.

Detailed activity diagrams will show explicitly how functions call the M&C interfaces. Calling these interfaces is represented by call actions (actions are atomic behavioral elements inside activity diagrams). These actions are directly connected with the M&C interfaces in the structural model, providing another explicit link between the structural model and behavioral model, besides data and flows items described above. The M&C interfaces will be specified in a normalized way using SysML stereotypes, enabling the generation of M&C ICD documentation and code.

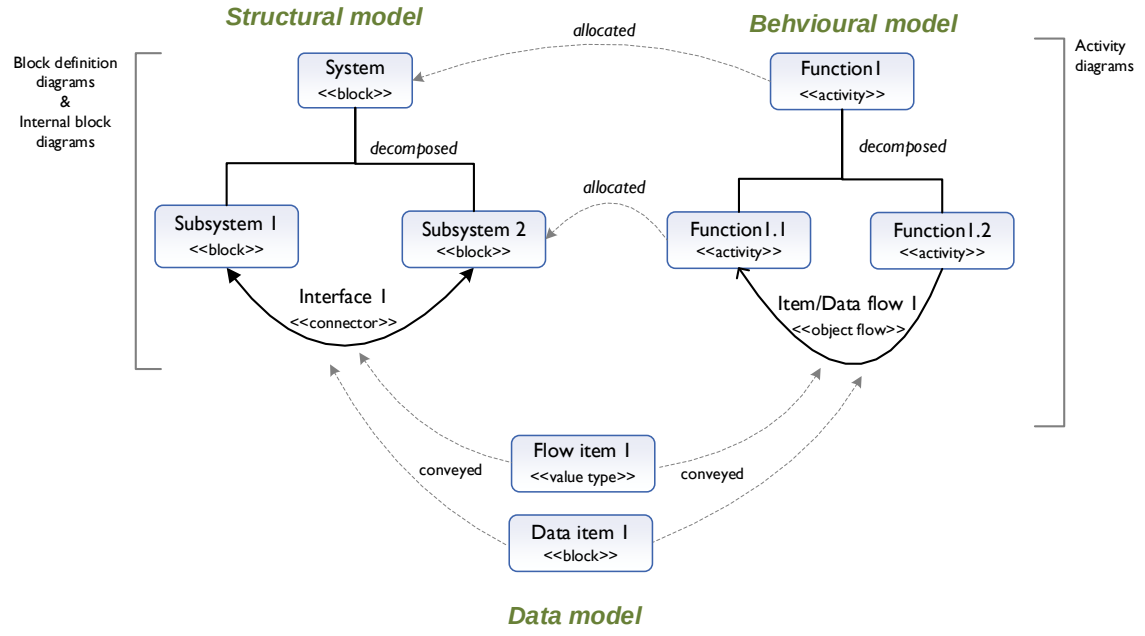

Figure 7: Behavioral and data modeling schema.

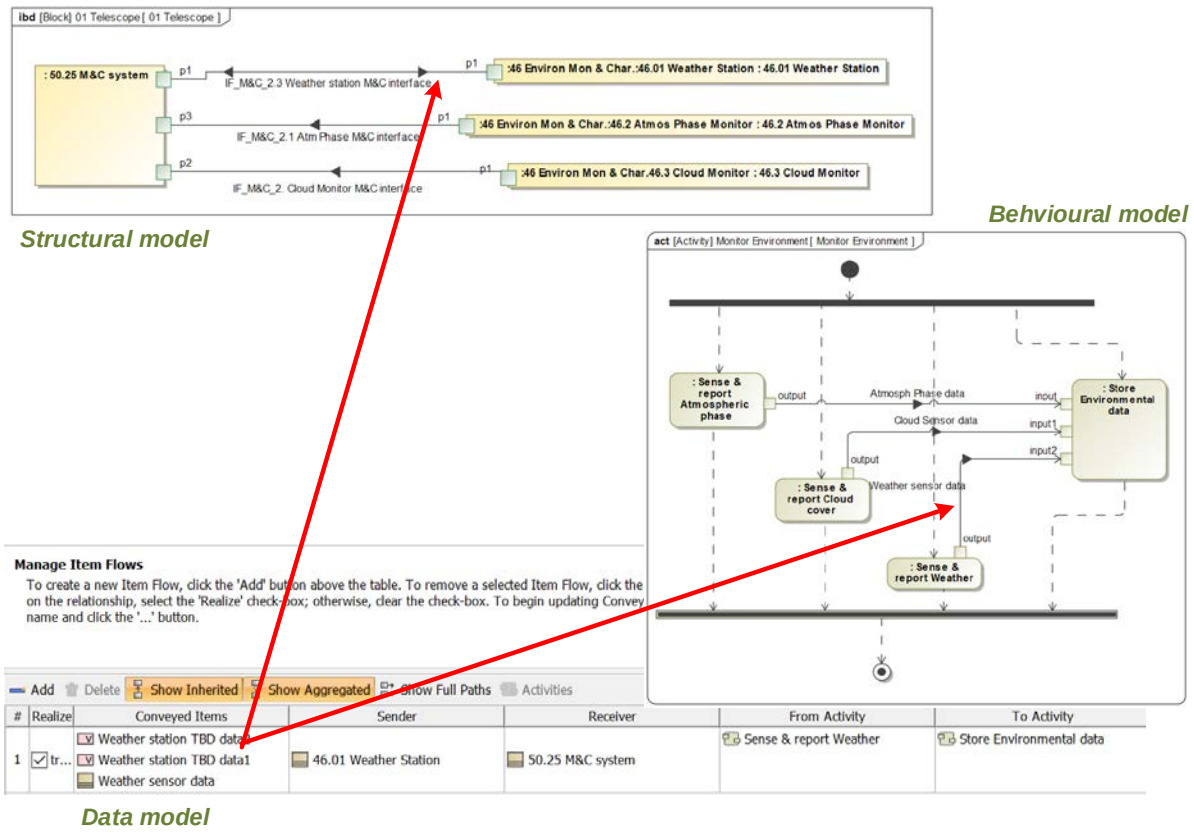

Figure 8: Behavioral and data modeling implementation example.

9.3 High-Level Functional Structure

A proposed structure for the top-level functional model is shown in Figure 9. This structure is based on a design pattern that is commonly used in systems modelling. It is structured around the core functional flow, which in this case is the end-to-end signal path. Signal path support functions are secondary functions that are required to operate and calibrate the signal path. The control and monitoring functions provide interfacing functions between the users and the real-time M&C functions. This is not a strictly prescriptive framework, but is suggested as a structure for the activity diagrams at the top levels.

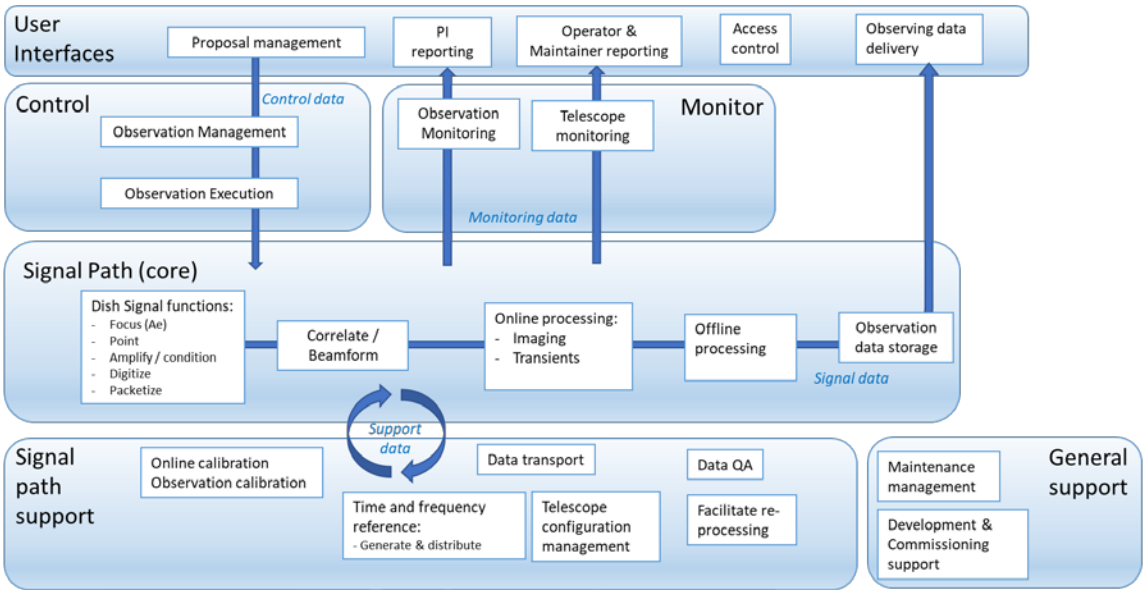

Figure 9: High-level functional structure.

9.4 Functional/Behavioral Model Package Structure

The package structure for the behavioral model is shown in Figure 10. The system model is kept in a separate package, which should contain all the system-level functions with sub-folders as needed to organize the functions. Subsystem functional models should be populated in their own separate folders and will vary in detail, depending on the need of the subsystem. For each subsystem, at least one top-level activity diagram should be built from all the functions that are allocated in the system model.

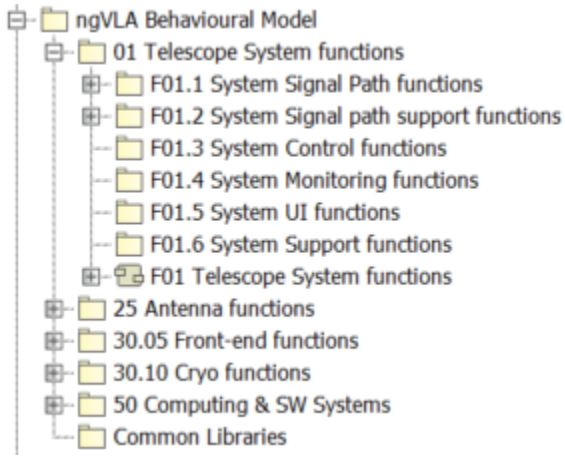

Figure 10: Behavioral model package structure.

10 Interface modelling definition

10.1 Interface identification numbering

In the CAMEO model for system level interface identification, numbers where identified using 4 digits numbers assigned in the N^2 diagram. In CAMEO connectors are used to model identified interfaces. For further modelling of interfaces the following approach is suggested:

- System level interfaces as identified in the N^2 diagram between sub-systems are named according to their number NNNN e.g. 0012 for CRY to FED as shown in the figure below (top connector between CRY and FED).
- Where these system level interfaces are further decomposed into individual sub-interfaces the identification numbering suggested uses NNNN_nnnn e.g. 0012_0001.

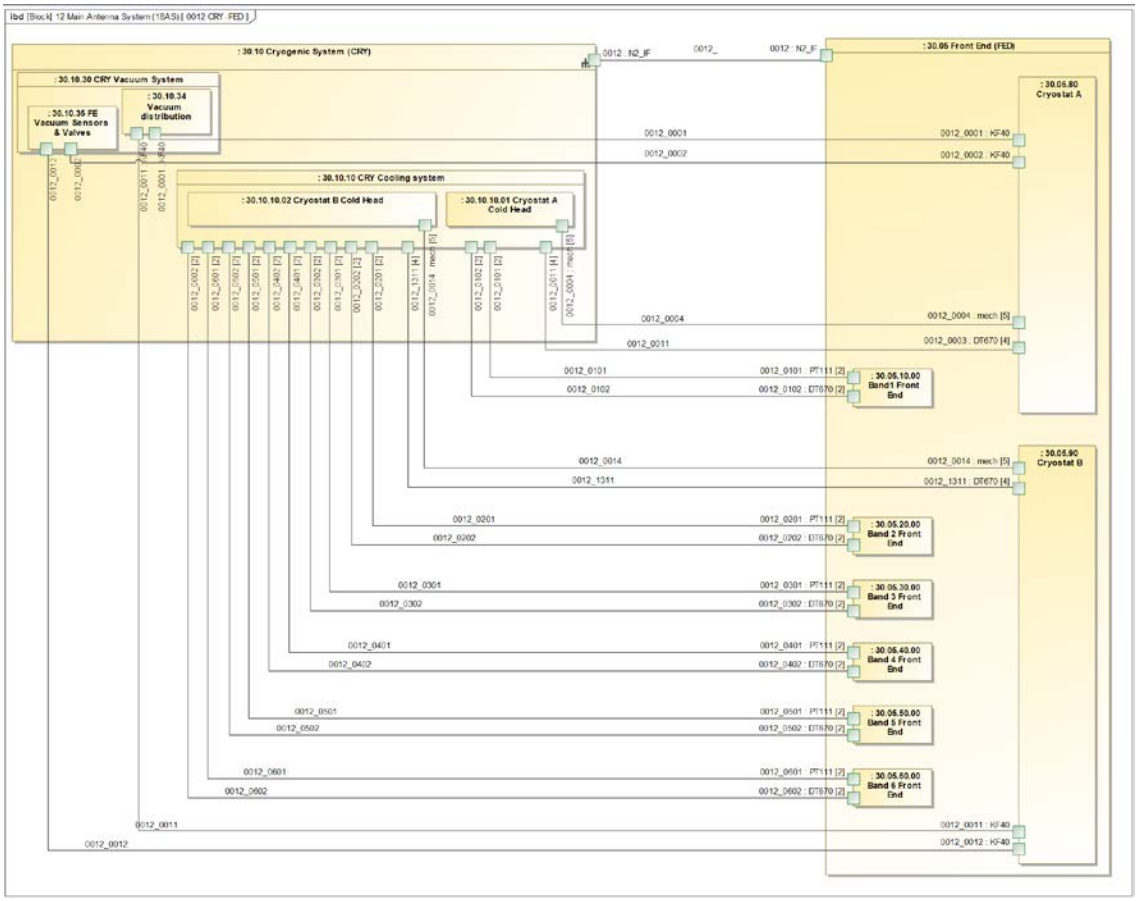

Figure 11: Suggested interface identification numbering of sub-interfaces.

- Internal to a sub-system the N^2 diagram numbers do not apply, and sub-systems may allocate their own numbers. As an example, the sub-system PBS number could be used instead of the NNNN prefix of the individual identified interface. See the CSP internal IBD example below where 020.40 indicates the owning sub-system (CSP) of the interface between SBP and PSE

identified as 020.40_0001.

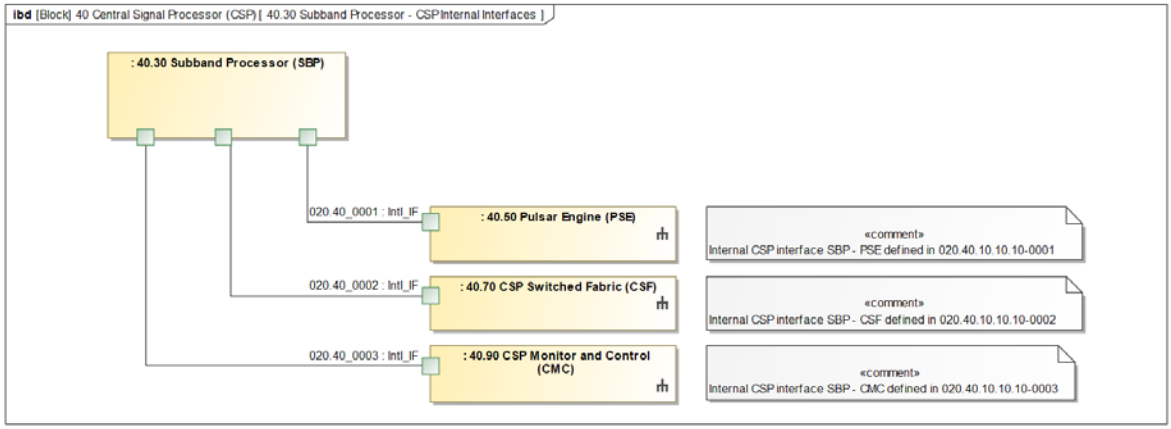

Figure 12: Sub-interface prefix for internal sub-system interfaces

- A further example of a lower level Antenna Electronics sub-system the Antenna Fiber Distribution uses 30.70_nnnn as its interface identification prefix. 30.70.

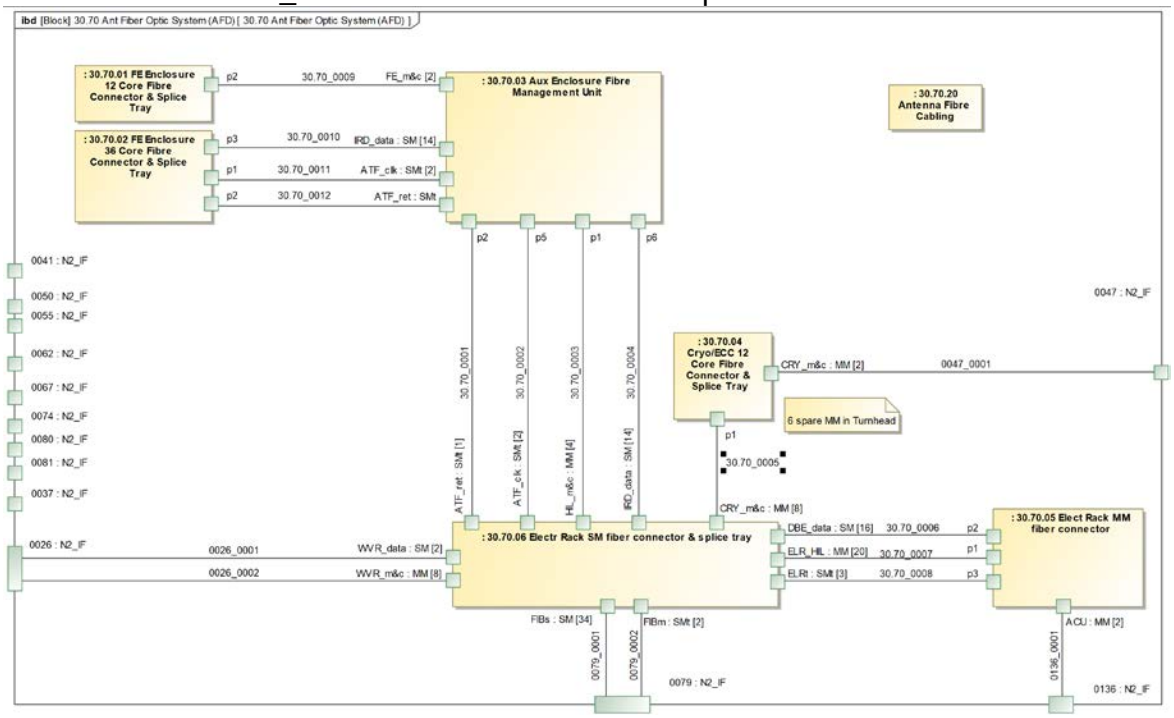

Figure 13: Use of both internal and external sub-interface numbering

Notes:

- CAMEO does not prevent duplicate connector names in the model.
- Port naming needs to be unique on a block, this modelling plan does not give any guidance on these apart from stating that proxy ports should be used for modeling interfaces.
- When connecting two blocks in CAMEO with ports using a connector: CAMEO will prompt the user to create a second port of the same name, type and multiplicity on the second destination block if you already created and named the port on the first block.
- Item flow can exist between blocks without having a port or connector defined.

10.2 Communications interface modelling framework

In order to gain maximum benefit from the system interface model, ideally it should serve as the source for any related system documentation including interface definitions. With this goal in mind the following interface modelling strategy is suggested. We use the DBE M&C interface (interface 0104 in the N^2 diagram) as an example.

This is a complex interface, so it is decomposed in several sub-interfaces. Some of these sub-interfaces are shown in Figure 14. Note that the sub-interfaces are represented by a SysML interfaceBlock and their names end with "I/F" for clarity.

Each one of these sub-interfaces includes many properties, along with their corresponding metadata. For example, as shown in Figure 14, the DBE Health I/F sub-interface incorporates the Processor Temperature property, which is a single temperature value in Celsius, represented as an 8-bit Integer.

Cameo includes the units defined the ISO-80000 standard and additional quantity types and units can be defined if (and only if) necessary. The types associated with each property can be simple types, like the ones shown in the DBE Health I/F sub-interface, or complex types like the properties shown in the DBE BIT Time Code Generator Configuration I/F.

Complex types are modeled as blocks, as shown in Figure 15. Note that both the sub-interfaces and the complex-types defined for the DBE interface are associated with the main "0104 – MCL DBE I/F" interface by composition, as they can be considered to be "part of" this interface. Sub-interfaces can also include functions, which are modeled as SysML operations.

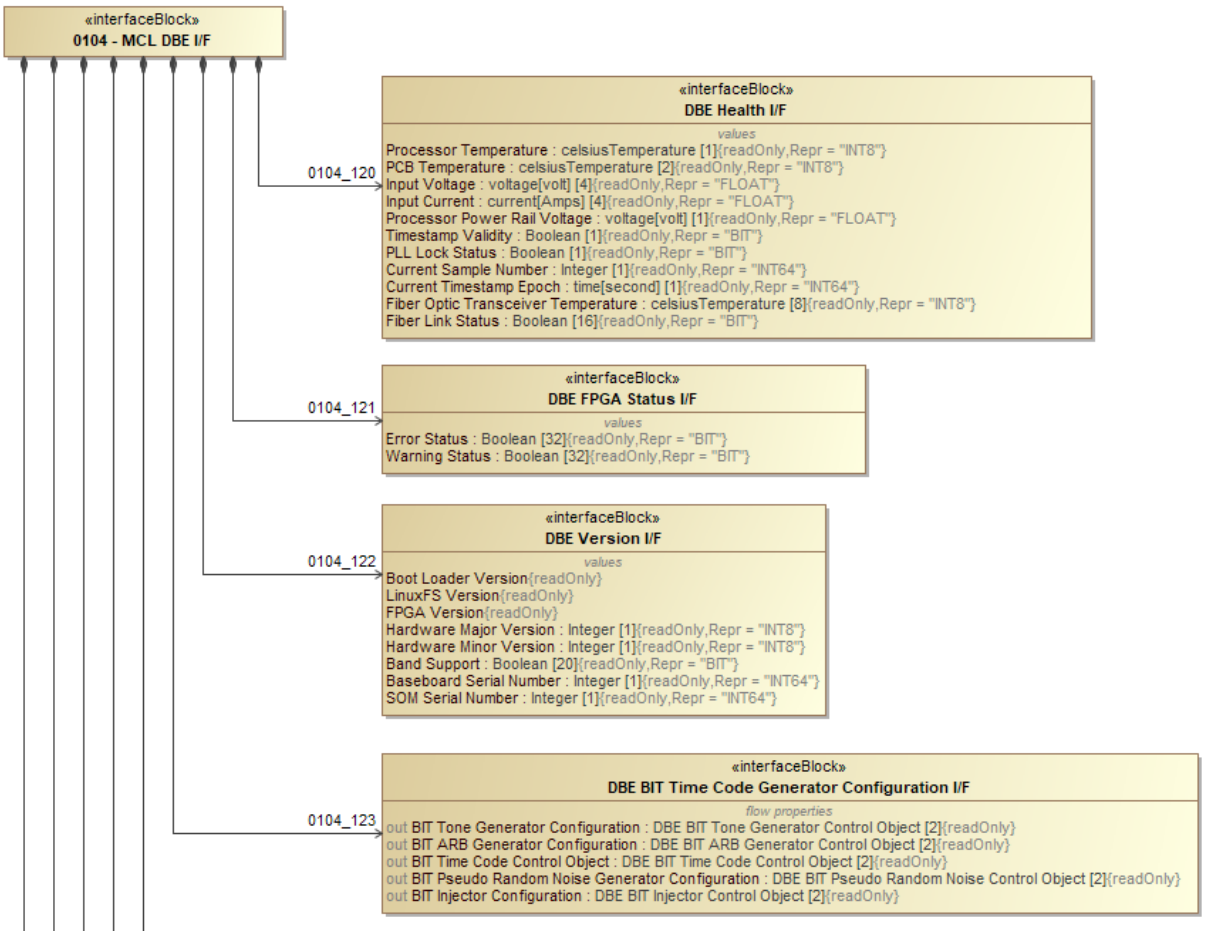

Figure 14. Example decomposition of the DBE interface in sub-interfaces.

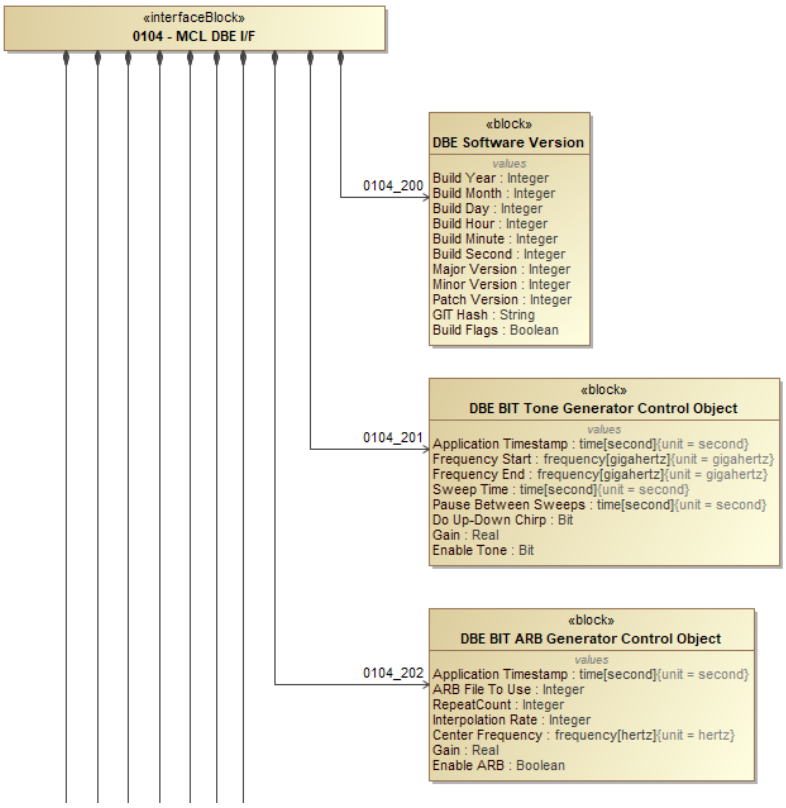

Figure 15. Some complex types used by the sub-interfaces are defined as blocks.

Once the interfaceBlocks and Blocks that constitute the interface have been defined, these types can be used to define the ports of the element that implements the interface, in this case the DBE. This is shown in Figure 16. Note that in this diagram both the main interface (0104) and its sub-interfaces (0104_XXX) are shown associated to the same DBE block. This may or may not be the approach to follow, depending on the case. In Figure 11, for example, the sub-interfaces are owned by internal elements between CRY and FED.

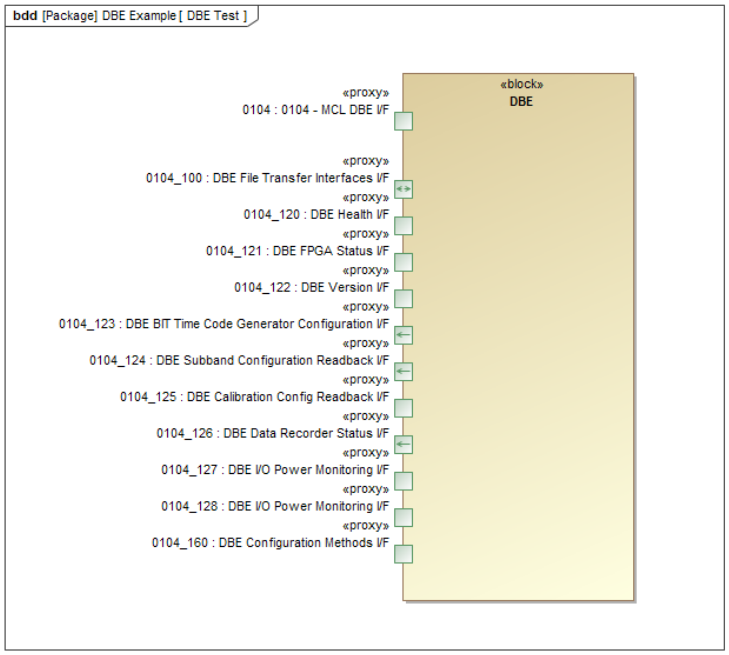

Figure 16. This diagram shows the DBE ports. Each proxy port is typified by the interfaceBlocks defined earlier.

This modelling approach allows the generation of an ICD table (Figure 17) in Cameo. Macros are used to output the ICD tables to an interface control document with the information captured in the model. Similarly, it is possible to generate software code implementing the interface from the model, which can be helpful to streamline software development activities, and avoid translation errors.

#	Port Name	Port Type	Type Features	Direction	Documentation
2	0104_100	DBE File Transfer Interfaces I/F	in DBE FPGA Image : File in LinuxFS : File out BIT Waveform : File	inout	
3	0104_120	DBE Health I/F	Current Sample Number : Integer [1] Current Timestamp Epoch : time[second] [1] Fiber Link Status : Boolean [16] Fiber Optic Transceiver Temperature : celsiusTemperature [8] Input Current : current[Amps] [4] Input Voltage : voltage[volt] [4] PCB Temperature : celsiusTemperature [2] PLL Lock Status : Boolean [1] Processor Power Rail Voltage : voltage[volt] [1] Processor Temperature : celsiusTemperature [1] Timestamp Validity : Boolean [1]	N/A	
4	0104_121	DBE FPGA Status I/F	Error Status : Boolean [32] Warning Status : Boolean [32]	N/A	
5	0104_122	DBE Version I/F	Band Support : Boolean [20] Baseboard Serial Number : Integer [1] Boot Loader Version FPGA Version Hardware Major Version : Integer [1] Hardware Minor Version : Integer [1] LinuxFS Version SOM Serial Number : Integer [1]	N/A	
6	0104_123	DBE BIT Time Code Generator Configuration I/F	out BIT ARB Generator Configuration : DBE BIT ARB Generator Control Object [2] out BIT Injector Configuration : DBE BIT Injector Control Object [2] out BIT Pseudo Random Noise Generator Configuration : DBE BIT Pseudo Random N out BIT Time Code Control Object : DBE BIT Time Code Control Object [2] out BIT Tone Generator Configuration : DBE BIT Tone Generator Control Object [2]	out	
7	0104_124	DBE Subband Configuration Readback I/F	out Subband Configuration Readback : DBE Subband Configuration Control Object	out	
8	0104_125	DBE Calibration Config Readback I/F	C Coefficient Band A : Integer [8192] C Coefficient Band B : Integer [8192] D Coefficient Band A : Integer [8192] D Coefficient Band B : Integer [8192]	N/A	

Figure 17: ICD table example

Additional parameters for each one of the value or flow properties (column "Type Features" in Figure 16) can be added as tags defined in custom stereotypes. This allows the definition of the interface independent of the protocol that will be used. The OPC UA specific tags would be defined in its own stereotype. Other protocols such I2C or SPI would have their own stereotypes to define their specific tags. This approach would even allow migration to a different protocol if necessary in future.

The interfaces defined in this way can be used to model and document behavior in activity or sequence diagrams.

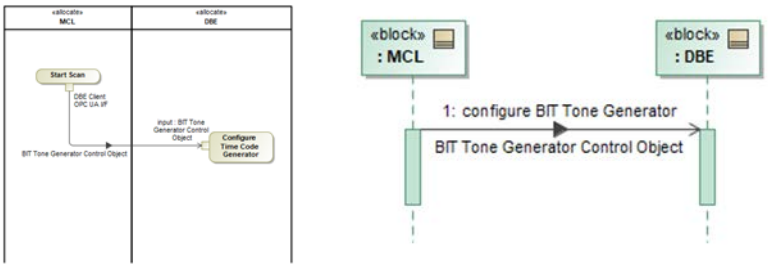

Figure 18: Interface behavior modelling

11 Verification Model

The Verification model includes

1. Verification Requirements, which are high-level statements of how each requirement will be verified. Verification Requirements are later used as a seed to develop the verification procedures.

There should be traceability between each requirement and its verification requirement on all L1 and L2 requirements.

2. Verification Events define programmatic events at which the verification will be executed. Each Verification Requirement shall be traceable to a Verification Event. Typically, multiple verification requirements would be associated with a single Verification Event. Verification Events should be aligned with the AIV/CSV planning and are typically associated with:

- A specific life cycle phase (e.g. qualification phase leading up to CDR).
- Location (e.g. supplier facility, laboratory, or site).
- Test setup, which requires specific facilities and equipment. For example, an RFI test event would be defined for all the RFI tests, which requires specialized test chamber and receiver equipment. Another example would be the environmental testing which would be performed at an environmental test facility.

11.1 Verification Model Outputs

11.1.1 User Level

- None

11.1.2 System Level

- L1 Verification Requirements matrix, listing all the requirements and the corresponding verification requirement.
- L1 Verification Events matrix. For each Verification Event, listing all the verification requirements.

11.1.3 Subsystem Level

- L2 Verification Requirements matrix, listing all the requirements and the corresponding verification requirement.
- L2 Verification Events matrix. For each Verification Event, listing all the verification requirements.

11.2 Verification Model Implementation

The verification model is implemented in SysML as shown in Figure 19 (next page). Verification Requirements are modelled as requirements and linked to the product requirements through the "verified by" relationship. Verification Events are modeled as activities and are linked to the Verification Requirements through the "satisfied by" relationship. Multiple Verification Requirements would typically be mapped to a single Verification Event.

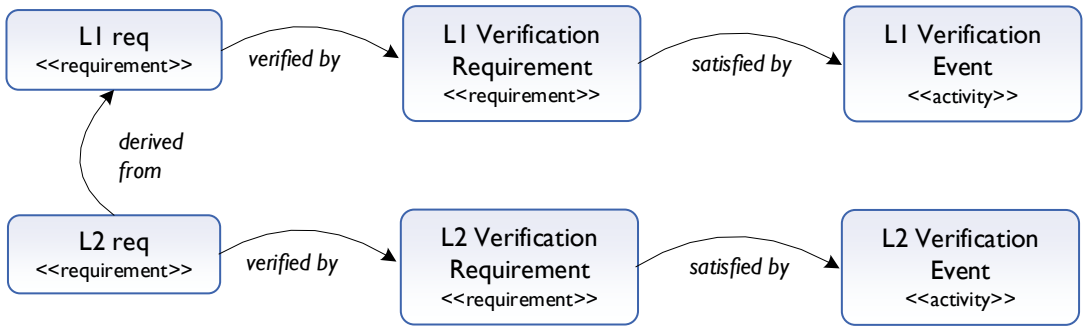

Figure 19: Verification model implementation.

12 Responsibilities

The responsibilities for developing the model are defined as follows:

1. The System IPT is responsible for User-Level and System-Level modelling.

- Requirements: L0 & L1
- Product Structure: from Observatory (AL1) down to and including Subsystems (AL3) at leaf-level.
- Behavioral/Functional model: from top down to where functions can be uniquely allocated to a Subsystem.
- Interfaces from top down to all interfaces connecting Subsystems.
- Verification: L1 Verification Requirements and Verification Events.

2. IPTs are responsible to develop the following for their subsystems:

- L2 Requirements with traceability to L1 requirements.
- Product Structure: at least one level below Subsystem level (AL4).
- Behavioral/Functional model: Expansion of the Subsystem allocated functions as required to define functional architecture (this will vary between subsystems).
- Interfaces: at least one level below Subsystem level (AL4) to generate the subsystem internal block diagrams.
- Verification: L2 Verification Requirements and Verification Events.

13 Phasing

The phasing of the implementation of the model is defined to support the program deliverables for major project reviews as follows.

13.1 System CoDR

1. System IPT:

- Requirements model: L0 & L1
- User-level and system-level structural model with identification of system-level interfaces.
- Preliminary user-level and system-level functional model.

2. Subsystem IPTs:

- Requirements model: L2.

13.2 System PDR

1. System IPT:

- Final user-level and system-level functional models.
- Verification model: L1 requirements.

2. Subsystem IPTs:

- NA

13.3 Subsystem PDRs

1. System IPT:

- Updates to align with subsystem design work as needed.

2. Subsystem IPTs:

- Subsystem structural model.
- Subsystem functional model.
- Verification model: L2 requirements.

14 Appendix A: Acronyms

Acronym	Non-Abbreviated Reference
ALx	Architecture Layer x
bdd	Block definition diagram
ICD	Interface Control Document
ibd	Internal block diagram
Lx	Level x
M&C	Monitoring & Control
PBS	Product Breakdown Structure
TBD	To Be Determined
LRT	LO Reference and Timing
SEMP	Systems Engineering Management
VE	Verification Event
VR	Verification Requirement

020.10.20.00.00-0003_PLA_ngVLA_System_Modeling_Plan

Final Audit Report 2024-07-16

Created:	2024-07-16
By:	Pieter Kotzé (pkotze@nrao.edu)
Status:	Signed
Transaction ID:	CBJCHBCAABAAgi6MQ_ZkHFzzDSxonZzaFSVrOJNq1Ml-

"020.10.20.00.00-0003_PLA_ngVLA_System_Modeling_Plan" History

Document created by Pieter Kotzé (pkotze@nrao.edu)
2024-07-16 - 7:55:45 PM GMT- IP address: 174.56.85.49

Document emailed to Rafael Hiriart (rhiriart@nrao.edu) for signature
2024-07-16 - 7:56:55 PM GMT

Email viewed by Rafael Hiriart (rhiriart@nrao.edu)
2024-07-16 - 8:29:45 PM GMT- IP address: 66.173.53.171

Document e-signed by Rafael Hiriart (rhiriart@nrao.edu)
Signature Date: 2024-07-16 - 8:33:46 PM GMT - Time Source: server- IP address: 66.173.53.171- Signature captured from device with phone number XXXXXXX2427

Document emailed to Pieter Kotzé (pkotze@nrao.edu) for signature
2024-07-16 - 8:33:48 PM GMT

Email viewed by Pieter Kotzé (pkotze@nrao.edu)
2024-07-16 - 8:35:30 PM GMT- IP address: 174.56.85.49

Document e-signed by Pieter Kotzé (pkotze@nrao.edu)
Signature Date: 2024-07-16 - 8:35:40 PM GMT - Time Source: server- IP address: 174.56.85.49

Document emailed to Eric Murphy (emurphy@nrao.edu) for signature
2024-07-16 - 8:35:41 PM GMT

Email viewed by Eric Murphy (emurphy@nrao.edu)
2024-07-16 - 8:40:09 PM GMT- IP address: 71.63.60.73

Document e-signed by Eric Murphy (emurphy@nrao.edu)
Signature Date: 2024-07-16 - 8:40:18 PM GMT - Time Source: server- IP address: 71.63.60.73

Document emailed to Rob Selina (rselina@nrao.edu) for signature
2024-07-16 - 8:40:20 PM GMT

Email viewed by Rob Selina (rselina@nrao.edu)
2024-07-16 - 8:57:00 PM GMT- IP address: 199.117.229.234

Document e-signed by Rob Selina (rselina@nrao.edu)
Signature Date: 2024-07-16 - 8:57:33 PM GMT - Time Source: server- IP address: 199.117.229.234

Document emailed to whojnowski@nrao.edu for signature
2024-07-16 - 8:57:34 PM GMT

Email viewed by whojnowski@nrao.edu
2024-07-16 - 8:58:33 PM GMT- IP address: 216.24.212.221

Signer whojnowski@nrao.edu entered name at signing as William Hojnowski
2024-07-16 - 9:30:24 PM GMT- IP address: 216.24.212.221

Document e-signed by William Hojnowski (whojnowski@nrao.edu)
Signature Date: 2024-07-16 - 9:30:26 PM GMT - Time Source: server- IP address: 216.24.212.221

Document emailed to Willem Esterhuyse (westerhu@nrao.edu) for signature
2024-07-16 - 9:30:28 PM GMT

Email viewed by Willem Esterhuyse (westerhu@nrao.edu)
2024-07-16 - 10:42:54 PM GMT- IP address: 73.31.51.29

Document e-signed by Willem Esterhuyse (westerhu@nrao.edu)
Signature Date: 2024-07-16 - 10:51:28 PM GMT - Time Source: server- IP address: 73.31.51.29

Agreement completed.
2024-07-16 - 10:51:28 PM GMT