SE2

Cookbook for MBSE with SysML

ISSUE 1

	Name	Date	Signature

Authors

Name	Affiliation

Change Record

Issue	Date	Section / Paragraph affected	Reason / Initiation Documents / Remarks

Table of Contents

Part I. MBSE in Telescope Modelling

Chapter 1. Introduction

In the framework of INCOSE's strategic initiative, the Systems Engineering Vision 2020, one of the main areas of focus is model-based systems engineering. In keeping with this emphasis, the European Southern Observatory (ESO; http://www.eso.org/) is collaborating with the German Chapter of INCOSE (http://www.gfse.de/) in the form of an "MBSE Challenge" team. The team's task is to demonstrate solutions to challenging problems using MBSE. The Active Phasing Experiment (APE; see Gonte et al. 2004), a European Union Framework Program 6 project, was chosen as the subject of the SE^2 Challenge Team (http://mbse.gfse.de/). Many technical products in the telescope domain show an increasing integration of mechanics with electronics, information processing, and also optics, and can therefore be rightly considered as optomechatronic systems.

This article presents the results of model-based systems engineering using the Systems Modeling Language (SysML; see Ogren 2000), drawing on experiences within the MBSE Challenge project and also the European Extremely Large Telescope (E-ELT) project. For the former project, SysML models were created by reverse engineering from existing documentation and from interviews with systems engineers, whereas for the latter project, the practices were applied to a new system. We will make use of Ingmar Ogren's concept of a common project model (Ogren 2000) to establish a common understanding of the system.

Chapter 2. Project Description

Our system case study is the Active Phasing Experiment technology demonstrator for the future European Extremely Large Telescope, which is a high-tech, interdisciplinary optomechatronic system in operation at the Paranal observatory (see ESO 2009). The next generation of telescopes needs to collect significantly more light than older models, therefore requiring bigger reflecting surfaces that consist of many individual mirror segments. Due to different disturbances (such as vibrations, wind, and gravity), the segments must be actively controlled to get a continuous mirror surface with a phasing error of only a few nanometers over the main mirror's diameter of 42 m. The main challenge is to correctly detect the positioning errors of the segments via specific phasing sensors in order to create a continuous mirror surface.

APE was developed to evaluate those sensors, and was installed on one of the 8 m telescopes that constitutes part of the Very Large Telescope in Chile (VLT) for sky tests. APE can be seen as the black box in Figure 2.1, “Active Phasing Experiment at the Very Large Telescope” . For the installation it had to comply with various mechanical, electrical, optical, and software interfaces. APE consists of about two hundred sensors and actuators such as wheels, translation stages, lenses, detectors, mirrors, light sources, an interferometer, and twelve computing nodes for control. Since APE had to be deployed in the test lab and in an already existing telescope, for each context it was necessary to model variants of function, interfaces, and structure. All of these characteristics made APE well suited to evaluate the potential of SysML in tackling similar issues.

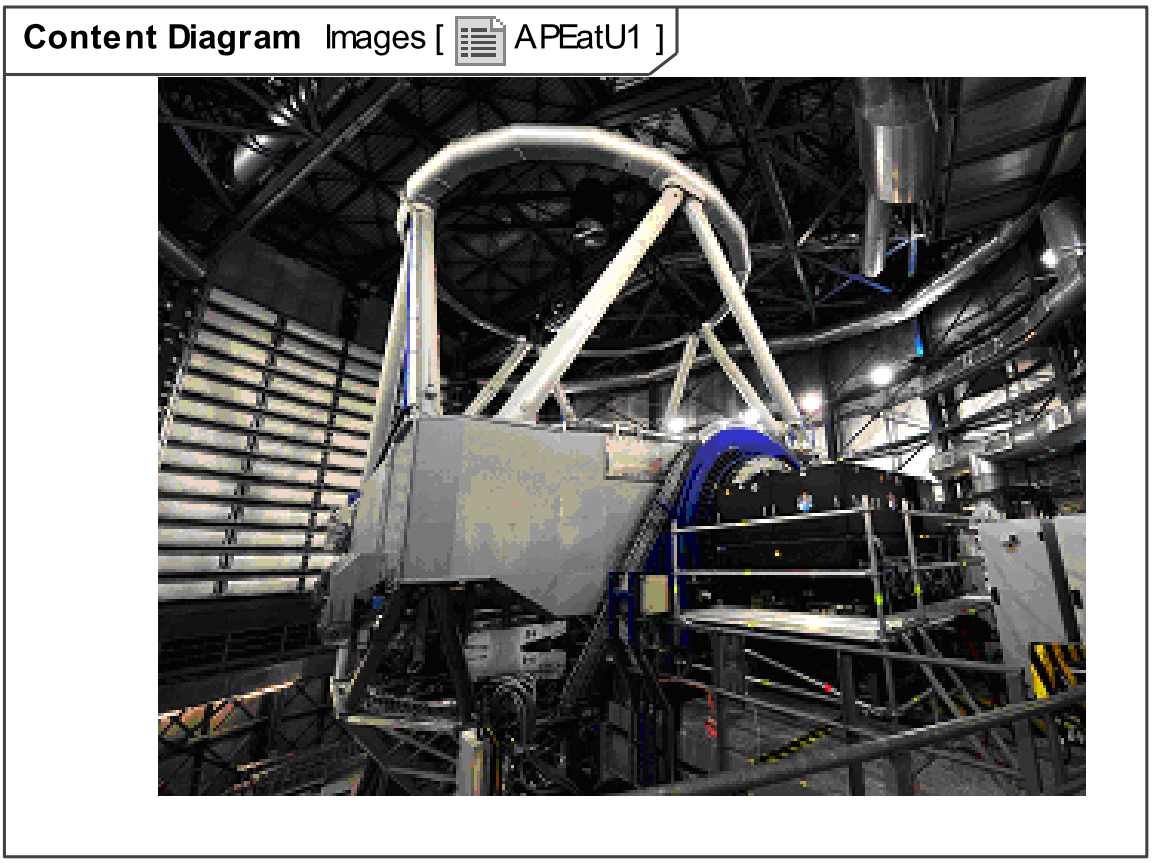

Figure 2.1. Active Phasing Experiment at the Very Large Telescope

Chapter 3. MBSE Challenge Goals

SysML is only a graphical language and defines a set of diagrammatics, modeling elements, a formal syntax, and semantics. Like any language (formal or informal), it can be used in many different ways, including many wrong ways. Most notably, it is possible by misusing the language to create incorrect models. The main goals of the SE^2 MBSE Challenge Team are to

- create modeling guidelines and conventions for all system aspects, hierarchy levels, and views;
- provide examples in SysML, solving common modeling problems;
- build a comprehensive model, which serves as the basis for providing different views to different engineering aspects and subsequent activities; and to
- demonstrate that SysML is an effective means to support systems engineering.

The SE^2 team has provided their guidelines for modeling on the "frequently asked questions" page of their Web site (http://mbse.gfse.de/documents/faq.html). A SysML model, as described in the next section, illustrates the results of their comprehensive modeling. The SysML model is not merely a mental abstraction, but a collection of complex data structures that can be edited, augmented, queried, and reported on by means of a suitable tool, which is an indispensible pillar for MBSE.

Chapter 4. Model Structure and Overview

Modeling is all about abstraction (reducing the complexity of a system), improving communication and understanding of the system, and providing reusable system elements. However, capturing a lot of different aspects like requirements, structure, interfaces, behavior, and verification in a model of a complex system like APE leads to a large model. Therefore, the first challenge is to find a clear, intuitive structure for the model, since a well-structured model is crucial for controlling complexity.

In the APE model we applied two techniques to establish a good and easily understandable structure: recursive structure patterns and views. In SysML, packages are the structuring mechanism to group model elements, which were used by the authors for both techniques. Packages are a mechanism to group model elements in higher-level units, similar to the way folders organize files in a computer's file system.

Chapter 5. APE Model Structure Pattern

5.1. Introduction

5.2. Overview Diagrams

To further improve the understandability of the APE model content diagrams are provided, which describe the system by showing all the different aspects captured by the model. The top-level overview diagram is a "project content" diagram Figure 5.1, “APE project-content diagram” and serves as an entry point and an anchor for navigation through the complex model.

The System Structure View ("APE_Structure" in Figure 5.1, “APE project-content diagram”) is used to decompose the system and provide the recursive modeling pattern within the subsystem package; for example, the "APE_Structure" sub-package "InternalMetrology" contains the same view packages, the "InternalMetrology" sub-package "PhaseModulator" Figure 5.2, “Subsystem content diagram” contains again the same view packages, and so on.

For every system decomposition element (like the nested structure view for the "InternalMetrology" in Figure 5.2, “Subsystem content diagram”), a package exists together with an overview diagram that shows the aspect packages of the respective element. The arrows between the packages show their dependencies. This recursive model structure provides an intuitive look-and-feel navigation capability within the APE model.

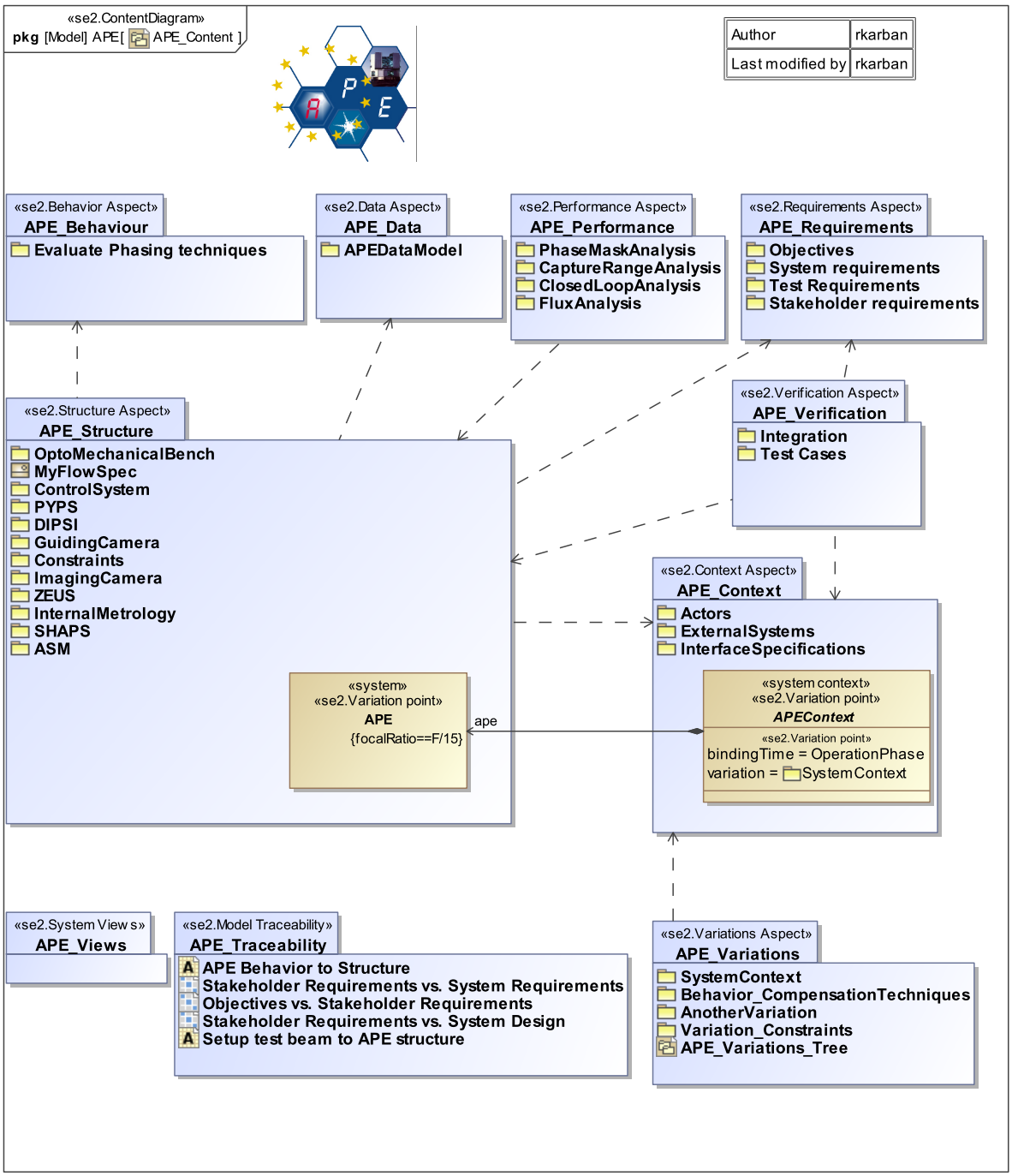

Figure 5.1. APE project-content diagram

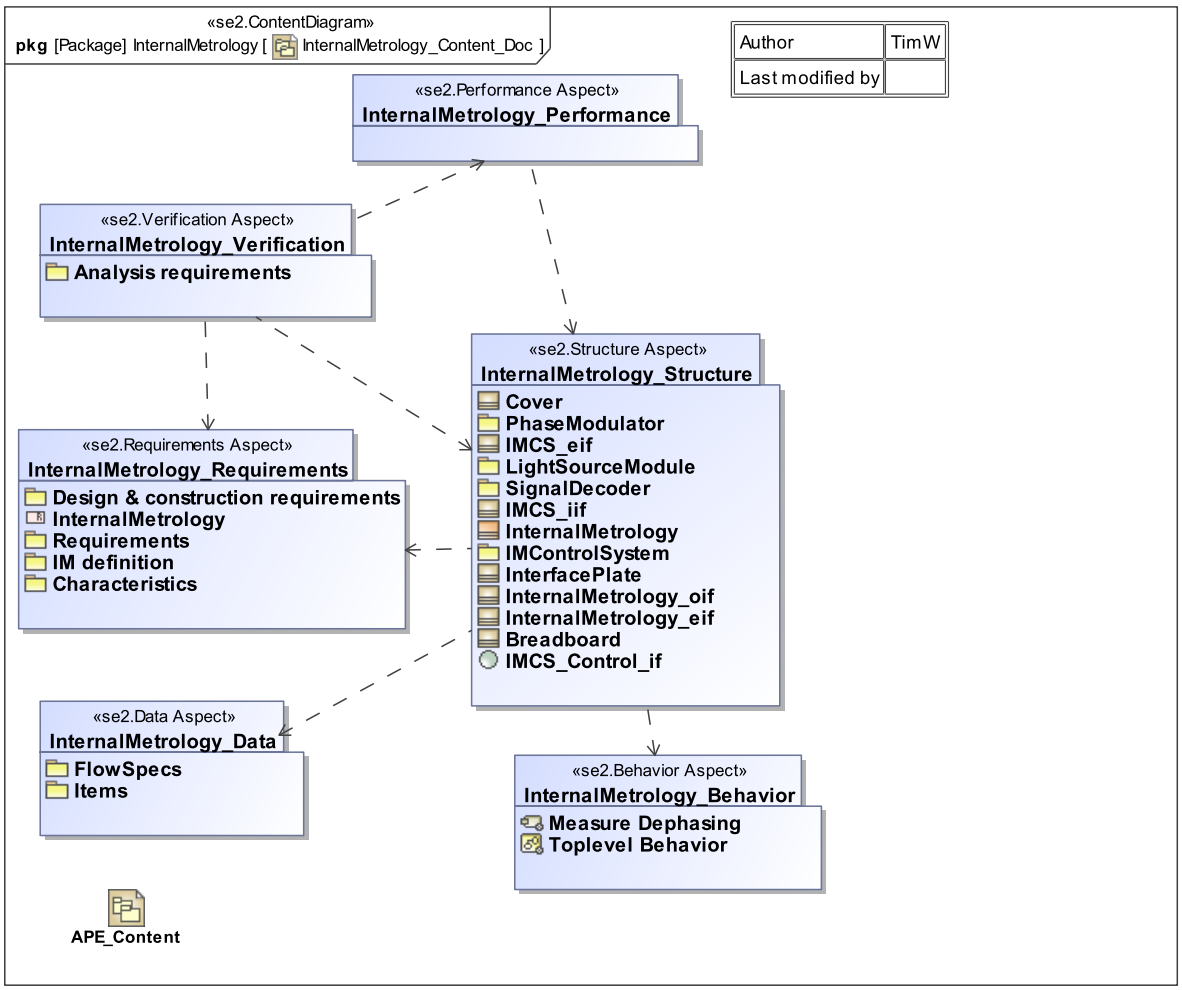

Figure 5.2. Subsystem content diagram

5.3. Aspects

5.4. Objectives and Requirements

APE, like any complex system, has a large number of functional, performance, physical, and interface requirements that have to be satisfied. This implies the need for formal requirements management during the project. APE has about 50 high-level system requirements. The control system has also about 50 requirements, refined by 150 use cases. We used the SysML requirements diagrams to show the main objectives of APE (Figure 5.3, “APE objectives diagram”). The limitations of standard text-based requirements-management tools were overcome by visualizing key requirements and their impact on system design and verification in an intuitive way.

The following user defined types extend the SysML requirements modeling features to organize and trace objectives, business and system requirements of APE. Figure 5.4, “Traceability among different types of requirements” shows the dependencies of the requirements and automatically created traceability matrices. In the figure, objectives are project-specific stereotypes of a class to capture the objectives for the projects (see Figure 5.3, “APE objectives diagram”); stakeholder requirements are project-specific stereotypes of a SysML requirement to capture the high-level requirements for the projects; and for the system requirements, SysML requirements are used to capture the detailed requirements for the projects.

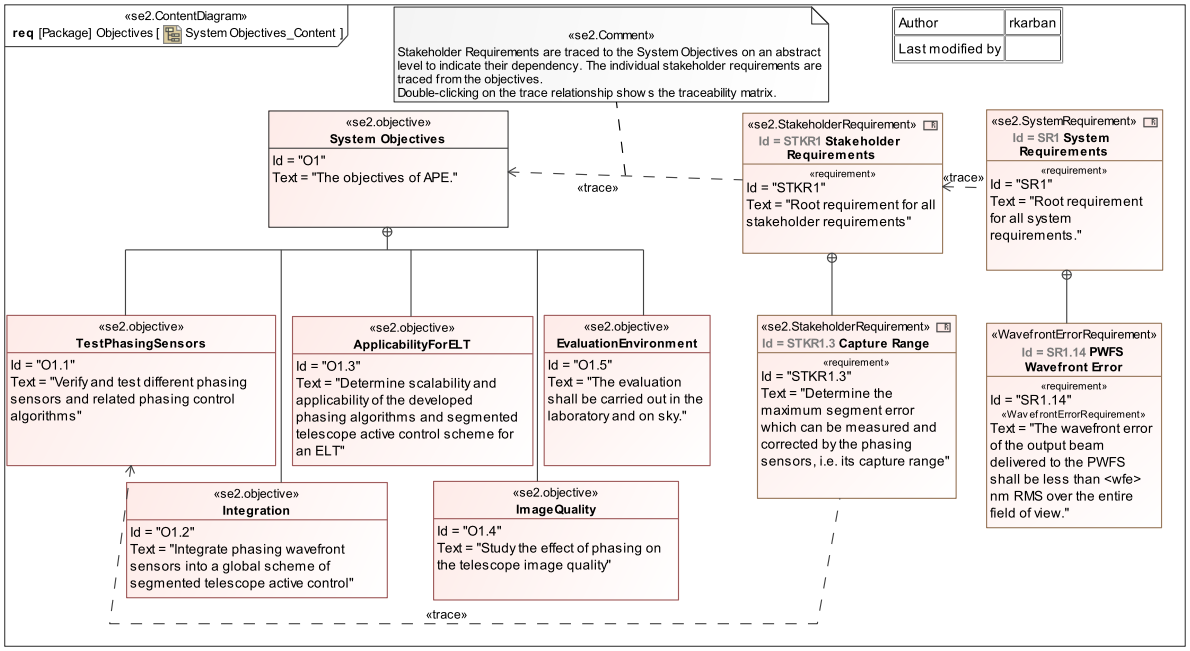

Figure 5.3. APE objectives diagram

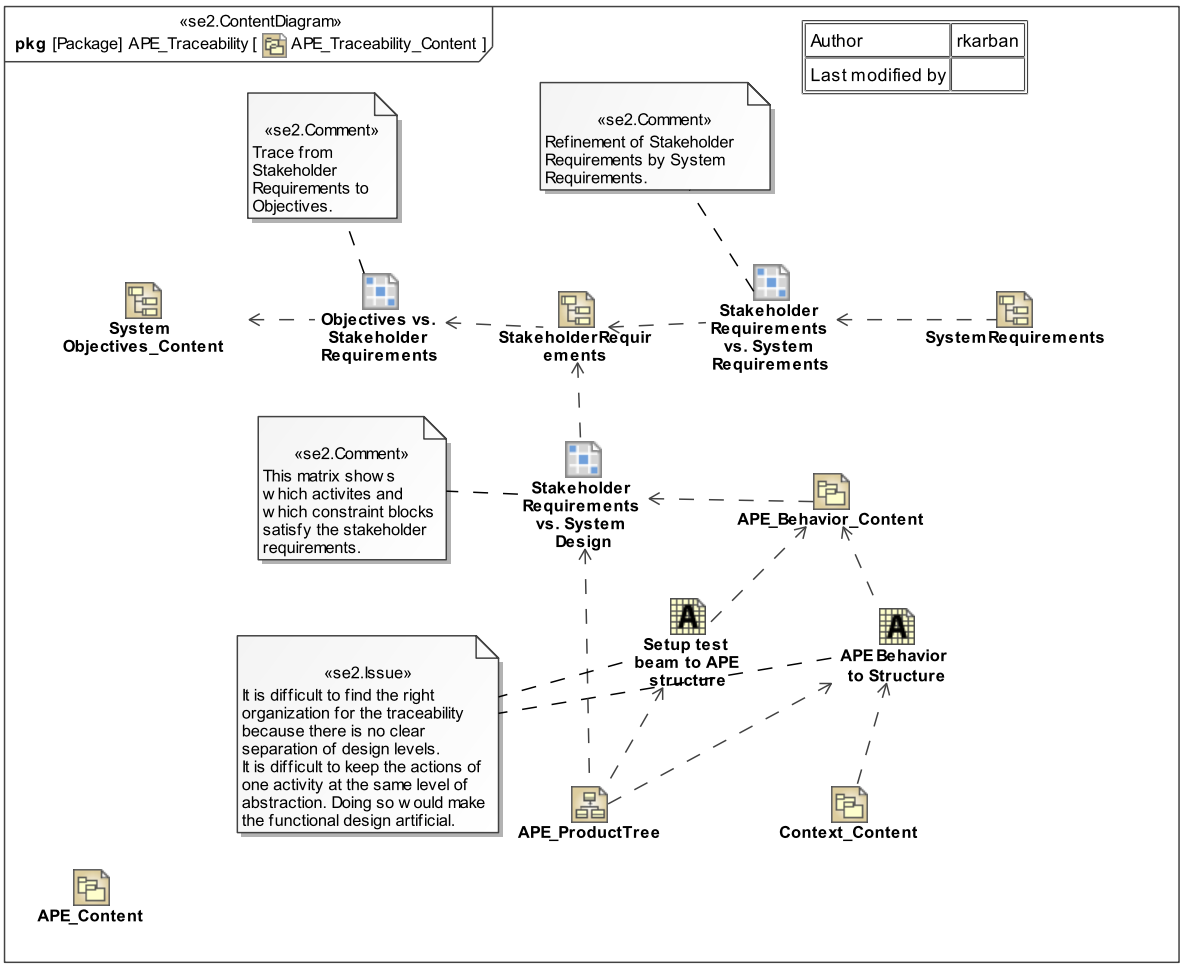

Figure 5.4. Traceability among different types of requirements

5.5. Context

The system's context defines the system's boundaries and is modeled using SysML internal block diagrams. A SysML internal block diagram (IBD) shows a block, its parts, and its interfaces. For the system context our main focus is on system interfaces.

SysML uses ports to model the interfaces of a block. SysML provides a standard port for service-like interfaces and flow ports for physical interfaces. However, there are different possibilities to use these ports for capturing system interfaces (see Figure 5.5, “Internal block diagram of electrical context”):

- Standard ports to model abstract interfaces as representation of an interface control document (ICD) as shown for port "scp"
- Flow ports for a combination of mechanical and flow interface at block level (model physical and logical properties at the border of a block, hiding its internals), as shown for port "15-N-A"
- Model mechanical and flow interface directly at the specific part level, as shown for "coolantReturn" and "coolantSupply," crossing the border of the outer part
- Model mechanical and flow interface at block and part level, as shown for port "15-N-C"

To ensure consistency between the interface control document and the model, the latter serves as the basis for the former. <token> shows the context of the telescope from an electrical viewpoint (ports on control system side are not modeled).

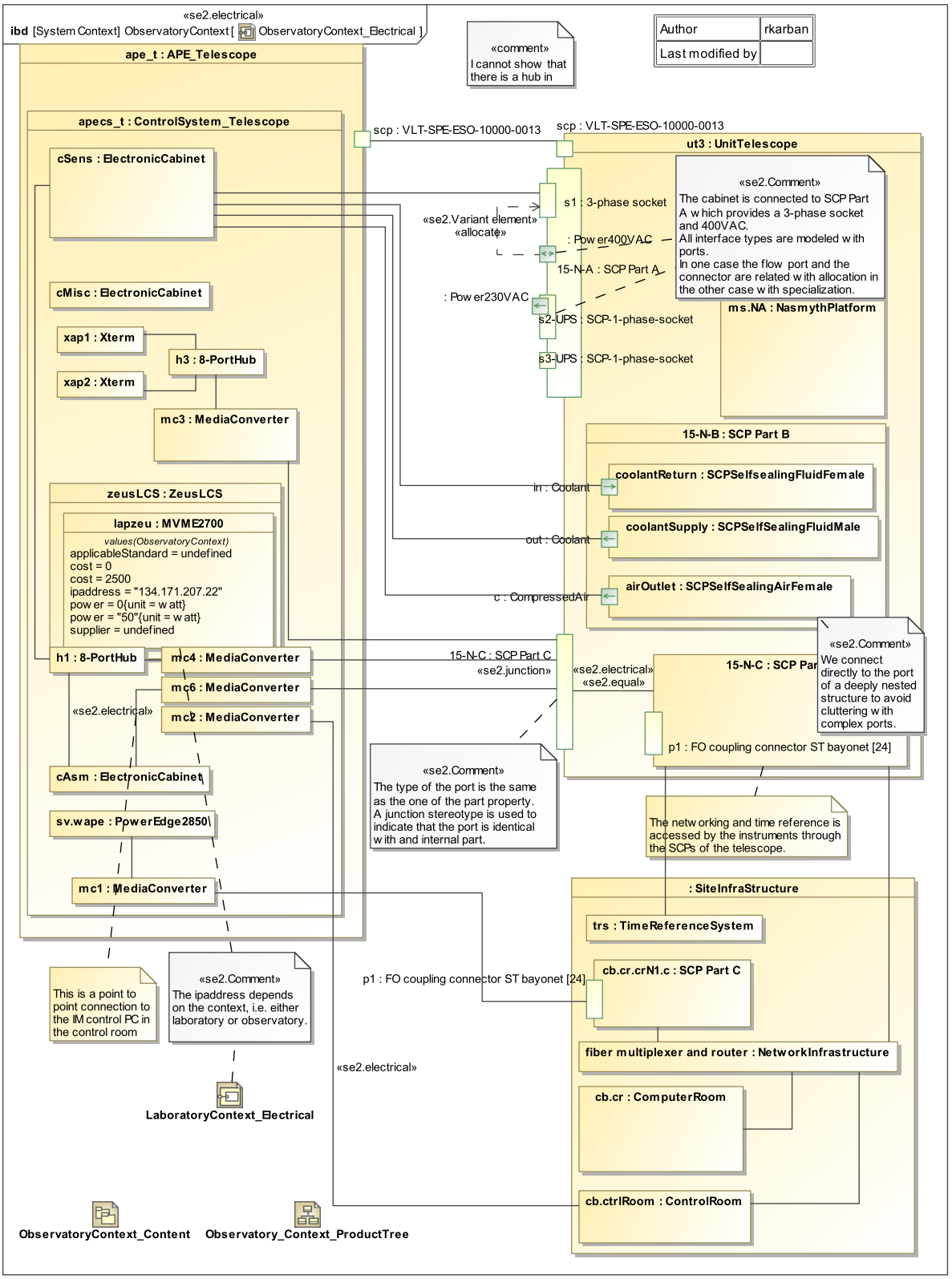

Figure 5.5. Internal block diagram of electrical context

5.6. System Structure

The system structure is maybe the most self-evident aspect to model. We have used SysML block definition diagrams (BDDs) to model the product tree and internal block diagrams to model the structure and interfaces of APE and its subsystem. The APE hierarchical breakdown is based on the product tree. It has several levels, going from the highest level into more and more details, using decomposition of its elements.

On the other hand, a complex system has much more than just one internal structure. There are multiple views showing electrical, optical, and mechanical elements that are interconnected, and therefore multiple structures exist. The same components can be connected in different views in different ways. We have used IBDs to show the electrical, optical, and mechanical layout of APE and its components at different levels. Figure 5.6, “IBD of the APE optical layout” shows the optical layout in an abstract manner. The connectors are stereotyped as optical, but they are elided for readability.

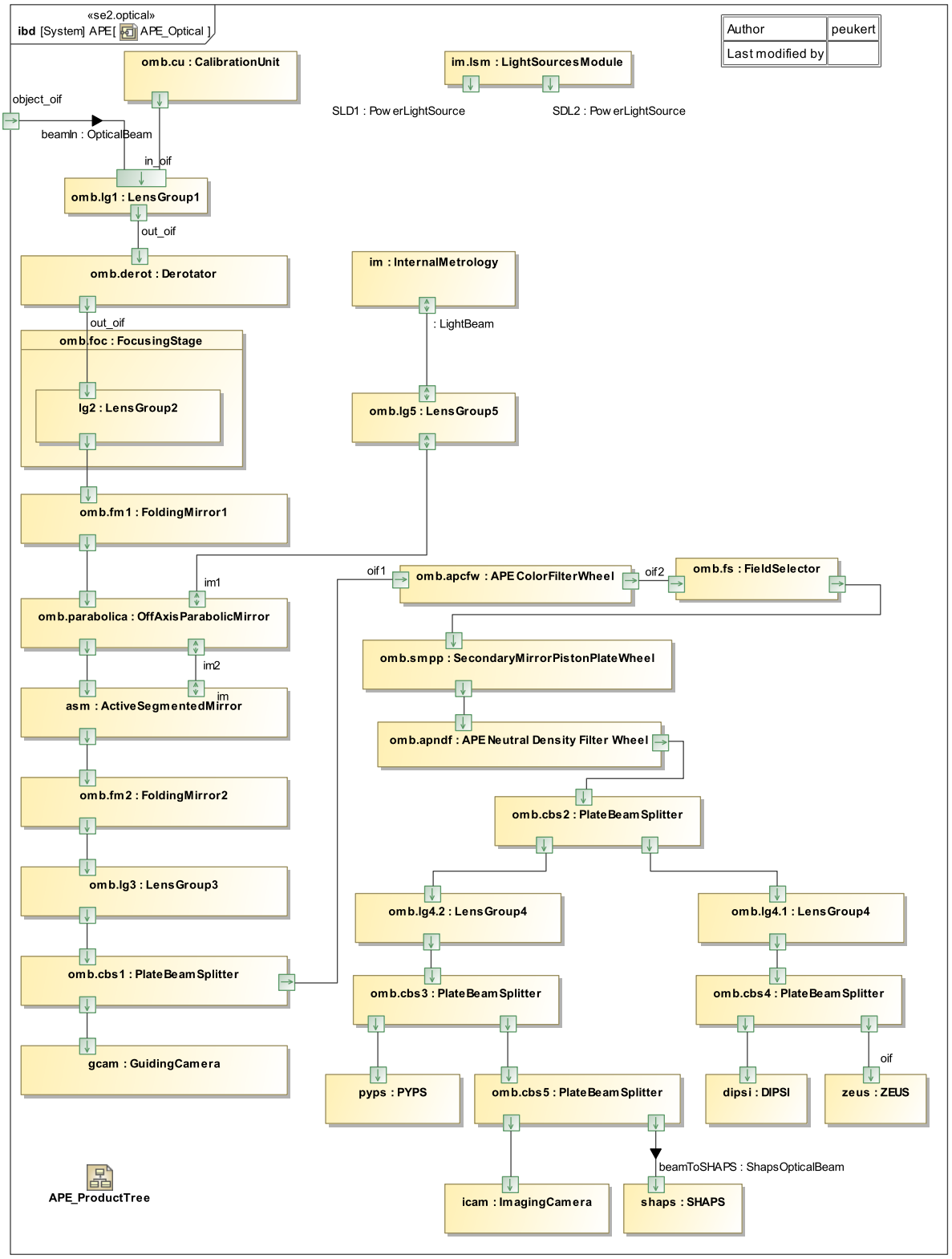

Figure 5.6. IBD of the APE optical layout

5.7. Behavior

A complex system is much more than just the collection of its elements and their structural architecture, because its behavior derives from the collaboration of its parts. Therefore it is essential to capture the

behavior of the system to be able to understand it. We have used activities to model the behavior of APE and its subsystems. SysML activity diagrams can be used to show the actions taken by the system and its data and control flow. Figure 5.7, “Activity diagram for APE wave-front control” shows the wave-front control of APE. It shows at the same time the physical effects of the system (like distortion of the wave front), as well as sensing, actuating actions, and control flows.

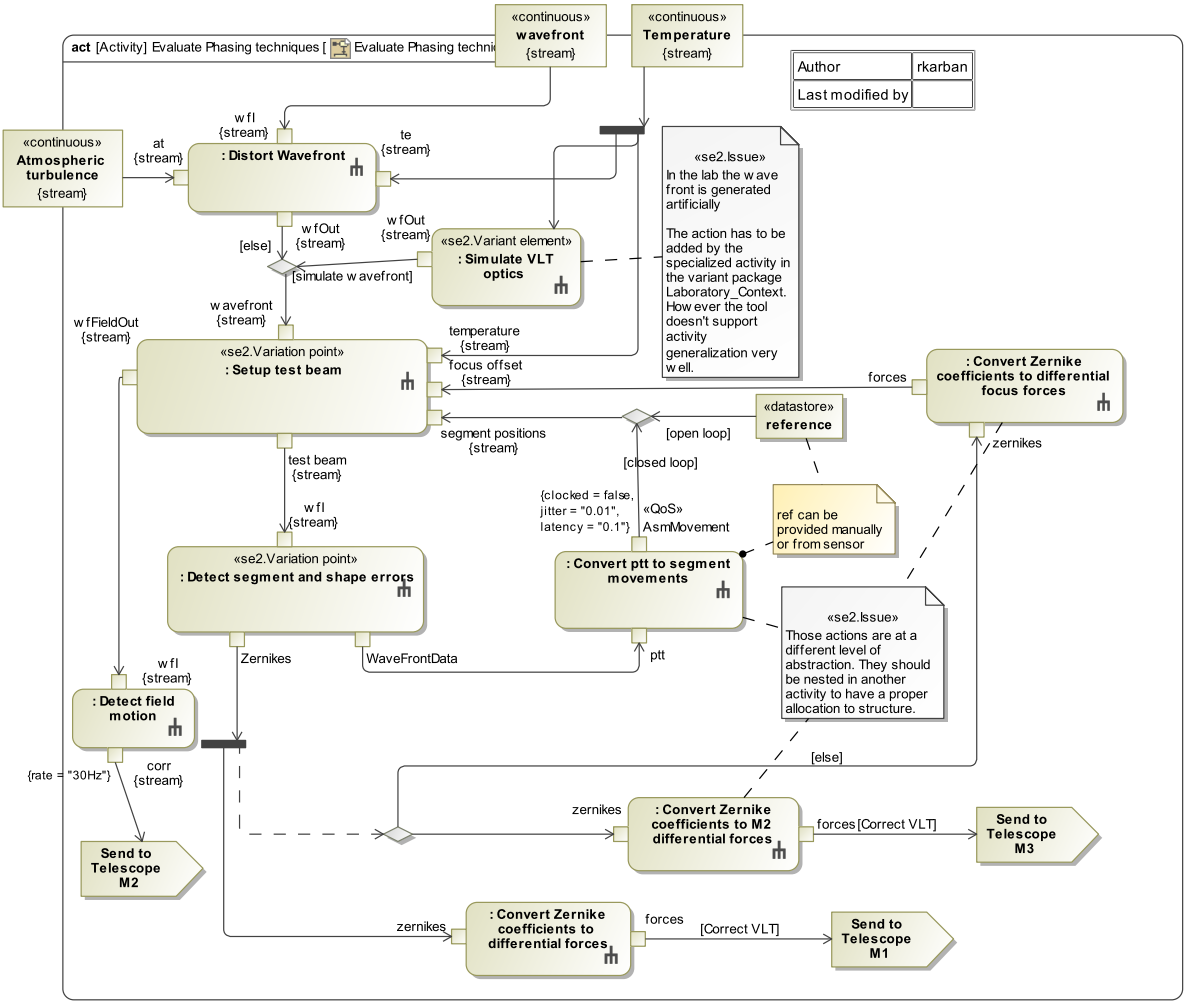

Figure 5.7. Activity diagram for APE wave-front control

5.8. Data

Another aspect is the information or data handled by the system. SysML provides block definition diagrams for the definition of data, and it provides internal block diagrams, activities, and sequence diagrams for data usage and flow. APE uses SysML "dataTypes" to define the data of APE and its subsystems. Figure 5.8, “Block definition diagram of APE wave-front data” shows the composition structure of a measurement and its relation to movements of the segmented mirror ("AsmMovement"). Those data types are used to define ports in IBDs and objects in activity diagrams.

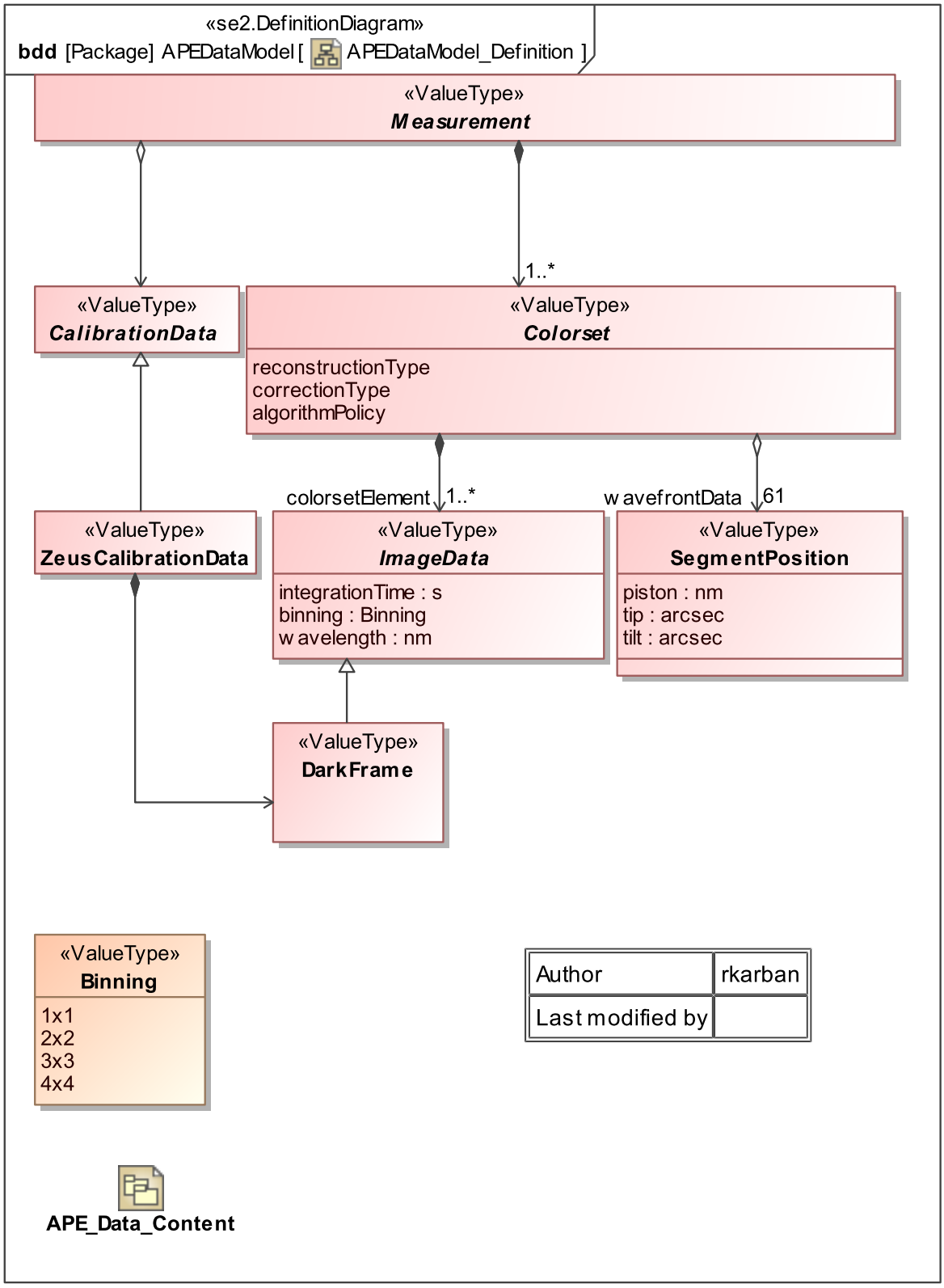

Figure 5.8. Block definition diagram of APE wave-front data

5.9. Verification

For every large system, verification is an essential part of the system acceptance in order to prove that the system meets its requirements. SysML supports the modeling of test cases with a specific model element, "testCase". The "testCase" element can be used at different levels for component integration, software integration, and system integration. Furthermore, APE has two different test contexts: verification in the lab and in the sky.

Chapter 6. Model Library and Systems-Engineering Profile

Besides the modeling of the different aspects, the APE modeling project provides a model library and an SE^2 profile. Data types and model elements that are frequently used are modeled in a model library to increase reuse. Abstract types are used as place holder for specific building blocks. They are classified in different catalogue packages (Figure 6.1, “APE abstract types”).

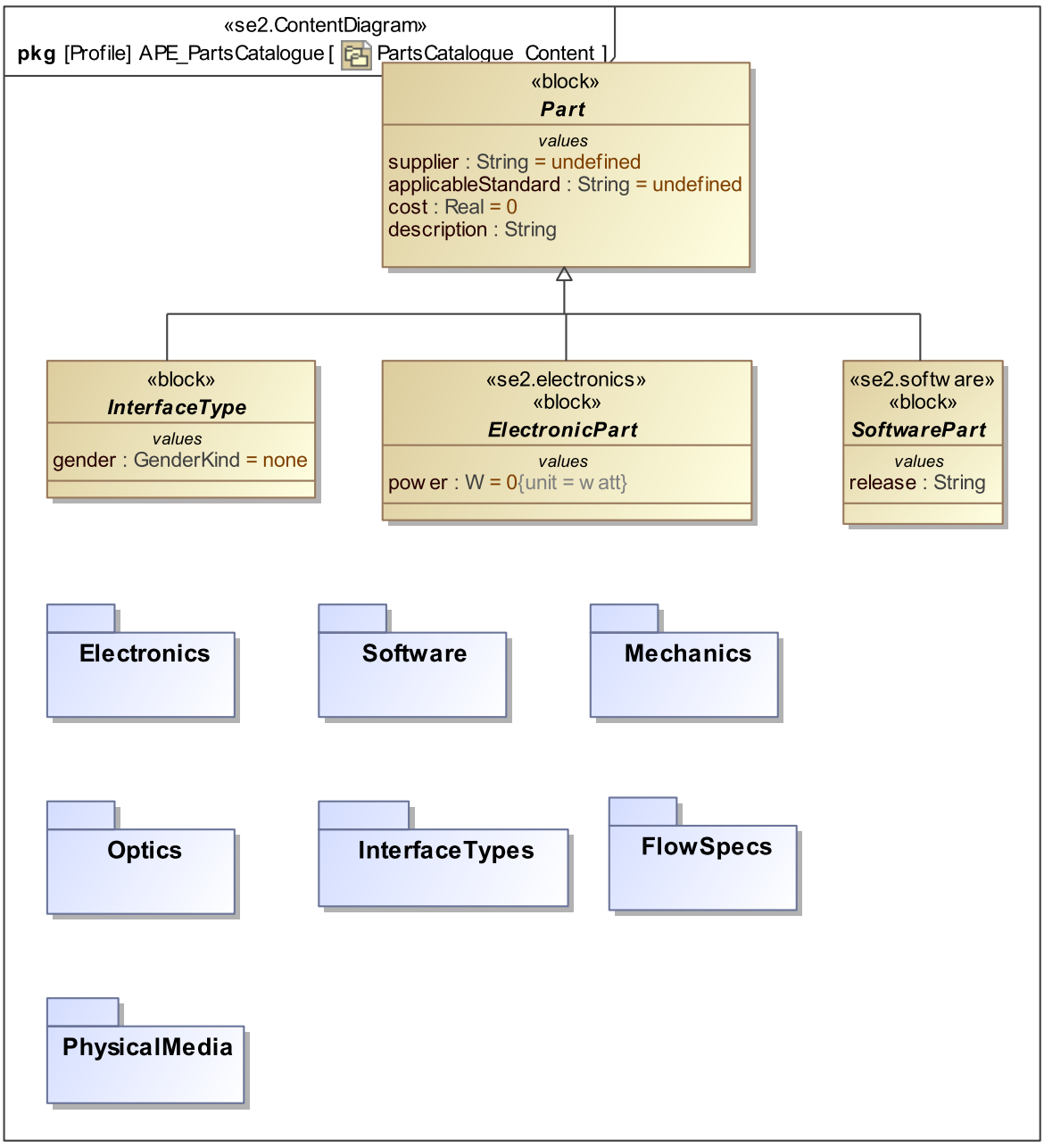

Figure 6.1. APE abstract types

Catalogues can be easily extended by using inheritance. Furthermore, the preliminary design of a system can initially work with an abstract type (when the detailed requirements are yet unknown) and decide later which specific type to use for the implementation. A generic connector gets a different context-specific pin assignment by inheritance. For each specific assignment a separate specialization is needed. The SE^2 profile provides the project-specific extensions to SysML: stereotypes like "objective", constraints in element usage, enumerations, and so on.

Chapter 7. Modeling Challenges

7.1. Introduction

7.2. Notation: Connection of Nested Blocks

Figure 7.1, “Nested connectors” shows the connection of control-system elements with telescope elements: for example, the sensor cabinet ("sensCabinet") is connected with the "coolantReturn" and "coolantSupply." The solid line is a nested connector. It crosses the boundaries of the encapsulated system blocks.

If the modeler would like to hide the internal structure of the telescope in Figure 7.1, “Nested connectors” , the nested connector would also be hidden. Typically we still want to see that there is some kind of relationship between the telescope and APE. SysML doesn't provide a presentation option to show the link in this case.

As a workaround we propose a standard port with stereotype junction that divides the nested connector at each relevant boundary crossing in separate connectors. Figure 7.2, “Junction ports for nested connector” shows that the junction port resolves the issue (see also Figure 5.5, “Internal block diagram of electrical context”). We propose that SysML support the concept of junction ports, which would enable tools to offer convenience functions for nested connector modeling (we have issued this proposal to the SysML Revision Task Force).

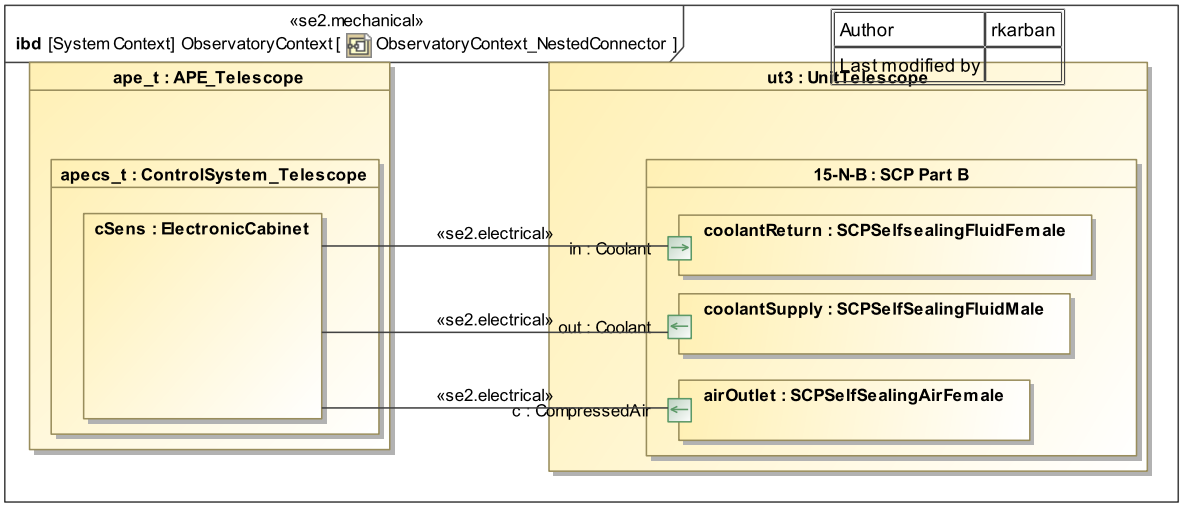

Figure 7.1. Nested connectors

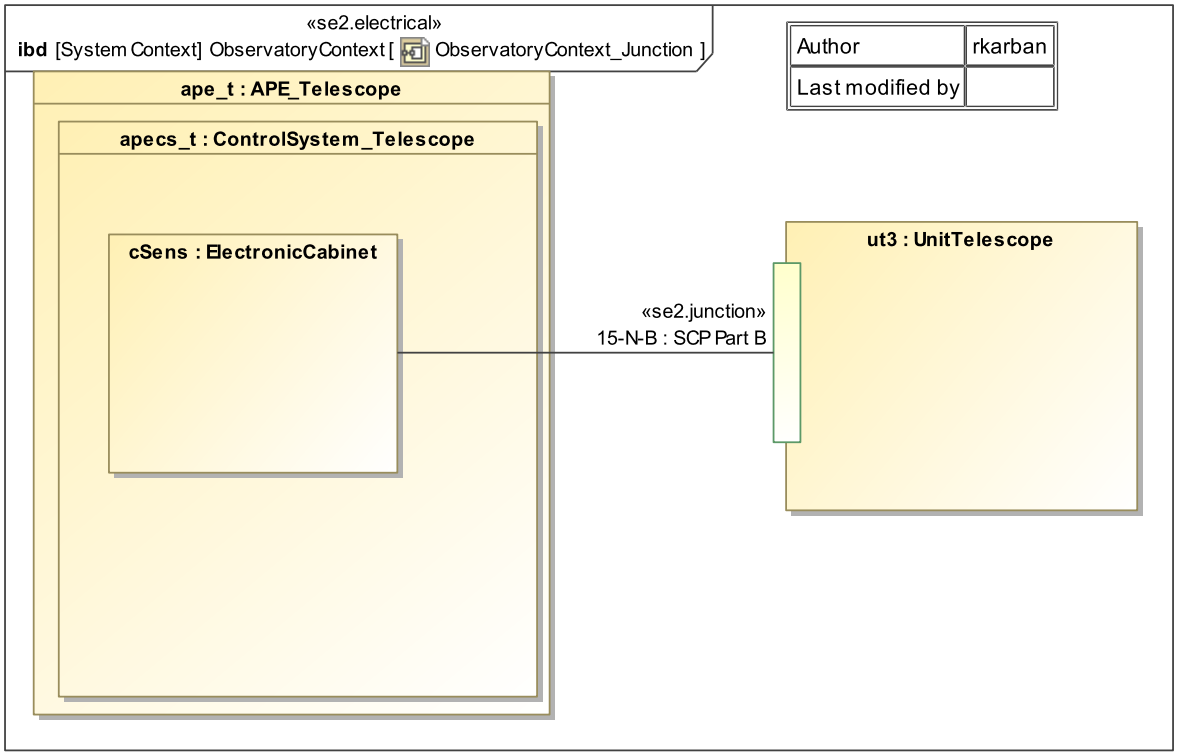

Figure 7.2. Junction ports for nested connector

7.3. Model

SysML supports two kinds of ports, that is, interaction points between a system block and its environment. The standard port provides or requests services, whereas the flow port specifies the flow of items inside or outside the system block. SysML has no explicit support of more complex ports that combine single, reusable ports. We propose nested ports for SysML. They allow a decomposing structure, the potential for re-usability and individual delegation of item flows that go through this port (see Figure 5.5, “Internal block diagram of electrical context”). Nested ports conform to the abstract syntax of SysML. Therefore some tool vendors already provide this important modeling feature. SysML doesn't say anything about the semantics of nested ports. Because we issued this proposal to the SysML Revision Task Force, a forthcoming SysML version will include a special kind of this concept called nested flow ports.

7.4. Tool

SysML has some modeling areas where creating and editing model information in a diagram is cumbersome. For example for requirements, which are often entered in a bulk, it is easier to use a table than a diagram to enter or modify the data. The same is true for relationship modeling like the allocation of behavioral elements to structural system elements. It is easier to assign the relationships in a matrix than in a diagram with lots of arrows. SysML already defines table and matrix formats. Most tools provide them as limited read-only views on the model. What we need are table and matrix views with the ability to create and modify model elements.

7.5. Methodology

SysML is a pure language and does not prescribe any methodology. For example, SysML allows one to model the "allocate" relationship between nearly any model elements. But where is this feature useful in a specific project? How can the relationship be determined from the model? What are the consequences of having an "allocate" relationship between two elements? You don't find answers to these questions in

the SysML specification. While there are some books that describe methodologies for modeling systems with SysML (Friedenthal 2008; Weilkiens 2008), our MBSE Challenge Team has found some best practices and modeling guidelines to complement them. Last but not least, each project needs its own specific set of methods.

7.6. Configuration and Quality Control

As soon as a common project model is created and more than one person uses it, configuration control becomes a fundamental requirement. In particular, consistent linking among model elements must be ensured. Individual changes must be traceable as well as creating visual differences to follow in detail what has changed where. Due to the extensive linking, side effects (introduced by changes) can go unnoticed and corrupt the model. This can only be mitigated by establishing rigorous configuration-management practices and using tools that allow rollbacks.

Chapter 8. Experiences from a New Project - E-ELT Telescope Control System

The European Extremely Large Telescope is a telescope with a primary mirror of 42 m in diameter, composed of 984 hexagonal segments, and four other mirrors with diameters ranging from 2.5 m to 6 m. Figure 8.1, “Size comparison of the European Extremely Large Telescope (left), the Very Large Telescope (center) and the Roman Colosseum (right). (Image created by ESO; reprinted by permission.)” shows the E-ELT in comparison with the Very Large Telescope and the Roman Colosseum, which might be an indicator for its complexity.

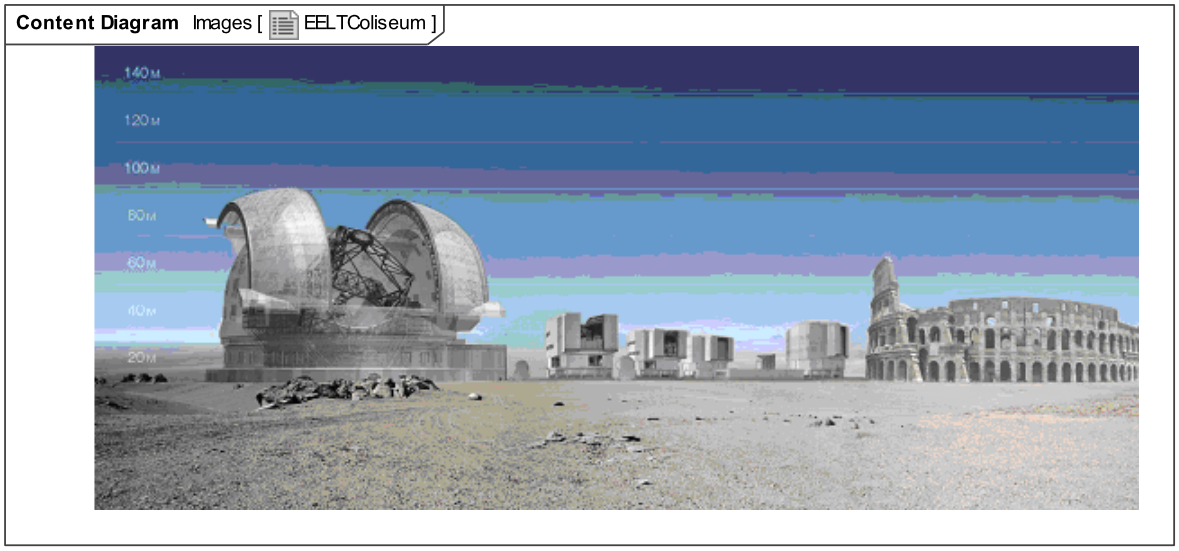

Figure 8.1. Size comparison of the European Extremely Large Telescope (left), the Very Large Telescope (center) and the Roman Colosseum (right). (Image created by ESO; reprinted by permission.)

The telescope consists roughly of 10,000 tons of steel and glass in a structure the size of a big football stadium; it needs 20,000 actuators, some of which have to be controlled to location accuracies within a nanometer and to angular accuracies within 0.02 degrees. It requires high-performance computation up to 700 Gflop/s, and data transfers rates of up to 17 Gbyte/s. The control system has to deal with about 60,000 I/O points, 15 subsystems (one particular subsystem requires the coordination of 15,000 actuators alone), and interacting, distributed control loops with sampling rates ranging from 0.01 Hz to several kHz.

The telescope control system (TCS) includes all the hardware, software, and communication infrastructure required to control the telescope and the dome. Many subsystems will be contracted and have to be properly integrated. Therefore, the TCS defines for the subsystems the interfaces and requirements as well as standards for the field electronics, software, and hardware to be used.

The common project model, where all TCS-relevant components of the observatory are logically represented, enables the development team to

- share a common, consistent view of the system, like the context in figure 14;
- structure requirements, in combination with a requirements management tool;
- describe the complex behavior of the system;
- define the architecture and detailed design;
- standardize meta-architecture across the various subsystems;

- allocate function to structure and allow full traceability; and
- analyze different design variants.

TCS architecture is highly data-oriented, making it well suited for representation with SysML constructs like activities and pins, blocks and ports, and state machines. The design of the TCS covers different views as control, electrical, and even mechanical because components have to be mechanically mounted.

Diagrams are extracted from the model to create paper-based documentation, as required by the project. The reporting and plug-in facilities of the modeling tool allow the engineer to automatically create the recursive structure as defined by the guidelines, and to make cost estimates using a predefined parts catalogue and estimates of the required communication infrastructure to accommodate the necessary throughput.

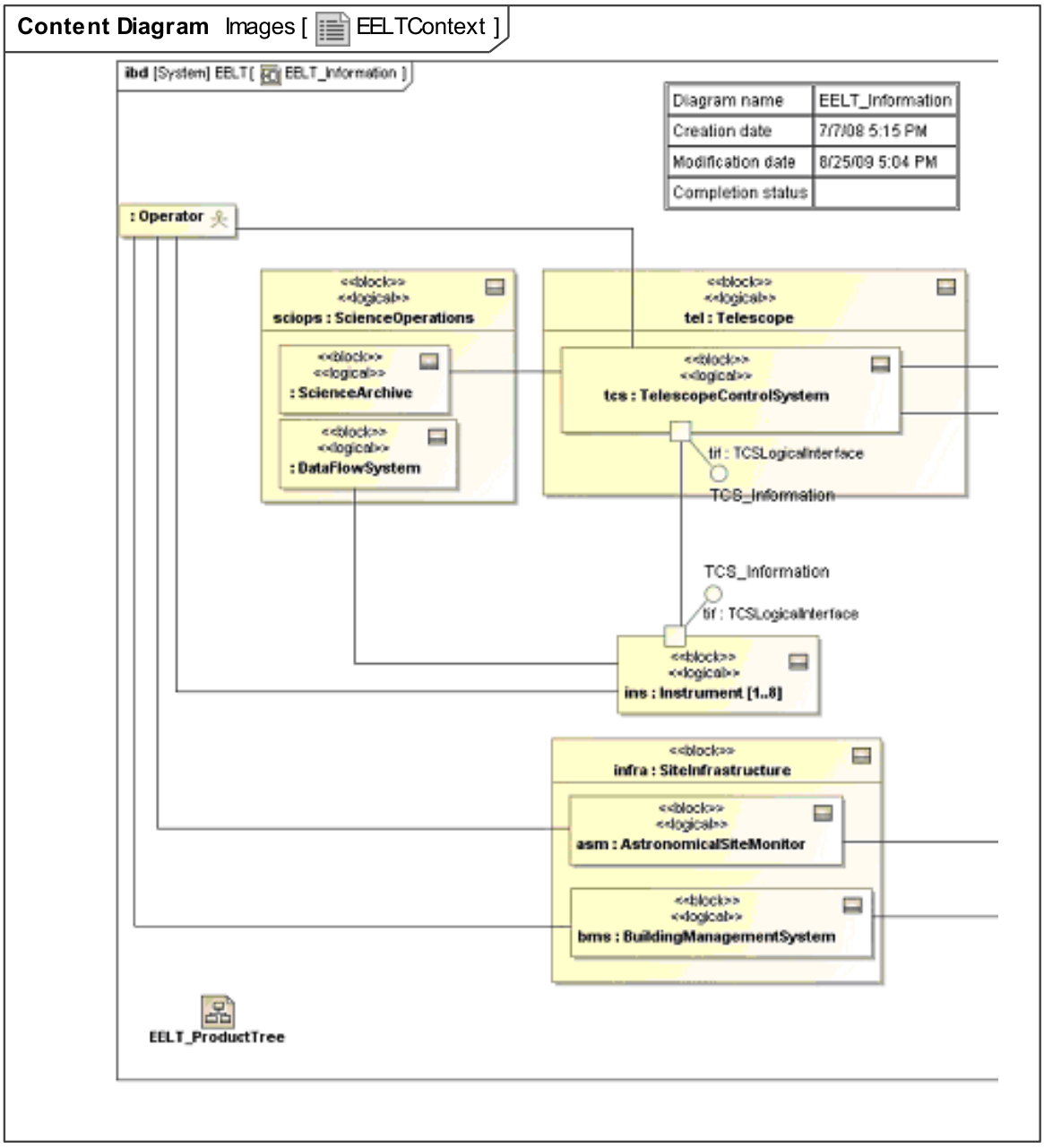

Figure 8.2. TCS context diagram

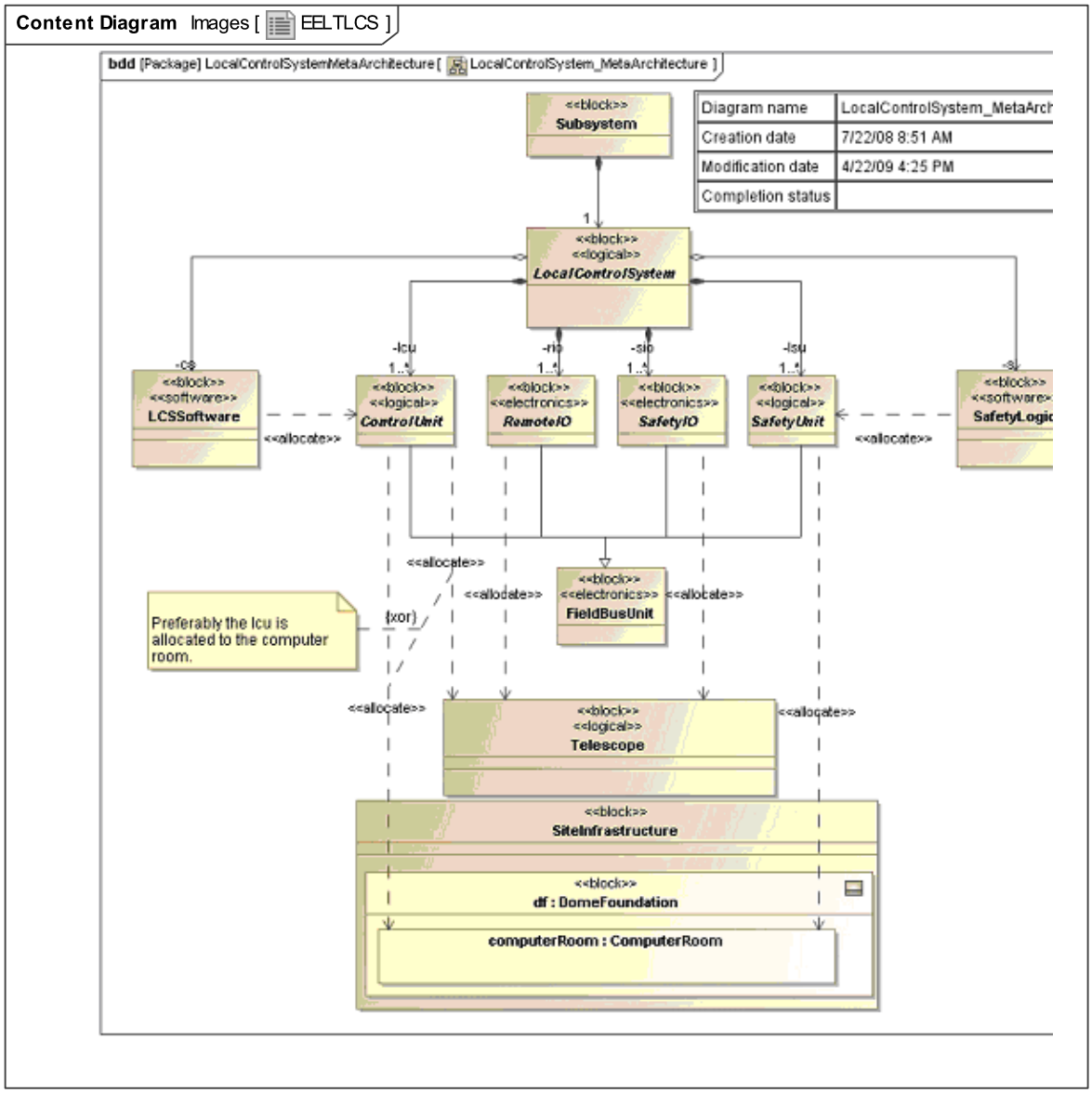

Figure 8.3. Meta-architecture of the local control system

The successful application of MBSE to a project of this scale was only made possible by the existence of the guidelines produced by the SE^2 Challenge Team. We consider those guidelines, together with their accompanying examples, a precondition to allow one to focus on the content, rather than on hands-on technicalities.

Chapter 9. Conclusions

The SE^2 Challenge team used an existing, complex system to create a comprehensive SysML model and solve common daily modeling problems. The engineering team of the E-ELT TCS is able to successfully apply the established guidelines, model structures, and modeling procedures to a new large-scale system in the optical telescope domain. The results demonstrate that SysML is an effective tool to document the complexity of requirements, interfaces, behavior, and structure, and is instrumental to enhance the traceability between requirements, design, and verification and validation.

A formal language and an adequately strict tool enforce structured thinking and a detailed description of the problem at hand. This increases the consistency and reveals undefined or unclear parts of the problem. Some limits of SysML were reached, because it does not offer out-of-the-box concepts for optical or electrical engineering. However, this can be overcome by extending the language using domain-specific profiles.

The most important value in a systems-engineering project is to have a common understanding among all stakeholders; therefore, fancy SysML constructs should be avoided when starting up. Even if SysML is a very good tool for communication and improving understanding, not all aspects of systems engineering can be fully covered by modeling with SysML, and systems engineers will still need the expressiveness and level of detail of traditional engineering activities like numerical analysis, simulation and prototyping to handle non-functional aspects like safety, security, reliability, usability, and logistics.

Chapter 10. References

ESO (European Southern Observatory). 2009. Very large telescope information. ESO (Web site). http://www.eso.org/public/astronomy/teles-instr/paranal.html (accessed 3 Nov. 2009).

Friedenthal, S., A. Moore, and R. Steiner. 2008. A practical guide to SysML: The systems modeling language. Burlington, MA: Elsevier/Morgan Kaufmann.

Gonte, F. Y. J., N. Yaitskova, P. Dierickx, R. Karban, A. Courteville, A. Schumacher, N. Devaney et al. 2004. APE: A breadboard to evaluate new phasing technologies for a future European Giant Optical Telescope. In Ground-based Telescopes, ed. Jacobus M. Oschmann, Jr. (vol. 5489 of Proceedings of SPIE [The International Society for Optical Engineering]), 1184-1191. Bellingham, WA: SPIE.

Ogren, I. 2000. On principles for model-based systems engineering. Systems Engineering 3 (1): 38-49.

OMG (Object Management Group). 2008. OMG Systems Modeling Language (OMG SysML): version 1.1. OMG document no. formal/08-11-02. http://www.omg.org/spec/SysML/1.1/ (accessed 3 Nov. 2009).

Weilkiens, T. 2008. Systems engineering with SysML/UML: Modeling, analysis, design. Amsterdam: Morgan Kaufmann OMG Press/Elsevier.

Part II. Recipes and best Practices

Chapter 11. Tool support

11.1. Templates for Model Structure

A getTemplate facility is supplied to act upon a package, and will produce a number of subpackage stereotyped by the corresponding SE2 stereotypes specified in this manual.

These packages cover all the aspects of system engineering specification, e.g. Requirements, Structure, Behavior, Performance, and several others. For each of these subpackages a content diagram (BDD) stereotyped by "se2.ContentDiagram", is created, where the user is supposed to add further decomposition as sensible (Chapter 5, *APE Model Structure Pattern*).

The diagram is intended to be used as a "entry point" for the package in which it is located, with all the packages appearing in it and containing hyperlinks for ease of navigation.

Most notably, the structure package is the one which is used for recursion, where a SysML block standing for the "system", and stereotyped by "se2.physical" is created and added to the product tree Diagram. The block also contains 4 different IBD, for Electrical, Mechanical,Optical and Software (stereotyped with the respective SE2 stereotypes).

The user is supposed to add packages which corresponds to the various Part Properties, and reapply the "get Template for System Element" facility to those packages.

An example of the its application for a "B" element can be seen in Figure 11.1, “Example Structure for package B” .

Also, for the requirements package an element of type "Requirement" is added (the Requirement ID counter is unique throughout the model) and a Requirement diagram containing it.

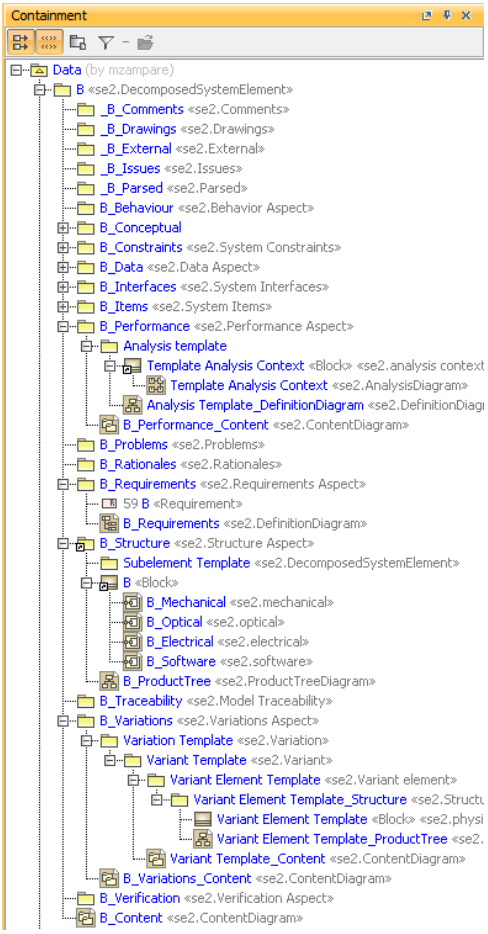

Figure 11.1. Example Structure for package B