Analyzing the Operational Behavior of the Alignment and Phasing System of the Thirty Meter Telescope using SysML

Sebastian J. I. Herzig[a], Robert Karban[a], Gelys Trancho[b], Frank G. Dekens[a], Nerijus Jankevičius[c], and Mitchell Troy[a]

[a]Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109, USA
[b]Thirty Meter Telescope, Pasadena, CA 91124, USA
[c]No Magic, Inc. Kaunas, LT-51480, Lithuania

ABSTRACT

The Alignment and Phasing System (APS) of the Thirty Meter Telescope (TMT) is responsible for positioning individual segments of the primary mirror, as well as the secondary and tertiary mirrors. Given its essential role, understanding the as-specified behavior and verifying related requirements is vital to the correct operation of the TMT. Analyzing the behavior of APS is challenging due to the variety of interactions with other subsystems. This paper presents results from developing an integrated system model that captures the structure, behavior, and requirements in a formal modeling language to enable automated verification using appropriate solvers. Specifically, demonstrated and discussed are the results of applying a Systems Modeling Language (SysML™) based approach in which operational modes, behavior specifications and use case scenarios are used for the purpose of verifying requirements on timing, power, and pointing error through system-level simulation using a single, integrated model.

Keywords: Model-based Systems Engineering, Verification, Requirements, Modeling and Simulation, TMT

1. INTRODUCTION

The Alignment and Phasing System (APS) of the Thirty Meter Telescope (TMT) is a Shack-Hartmann wavefront sensor responsible for the sensing and commanding of the pre-adaptive-optics wavefront quality.[1] In order to produce wavefronts of acceptable quality, APS will adjust pistons and tip/tilts of each of the 492 segments of the primary mirror (M1), adjust the segment surface figure (via warping harness adjustments), and rigid body degrees of freedom of M2 (secondary mirror) and M3 (tertiary mirror) (piston/tip/tilt or alternatively piston/x and y translation). APS interacts with numerous other components of TMT, including the M1 control system (M1CS), telescope control system (TCS), and common services (CS). APS also receives input from operators. Other TMT sub-systems require knowledge about how APS is specified to operate. This is particularly important to help develop, derive and verify interfaces to external subsystems, requirements on APS itself, and requirements on other telescope subsystems. This, combined with the complex internal behavior of APS makes designing, developing and verifying APS a challenge. In this paper, we present the latest findings on our model- and use-case-based approach to analyzing the operational behavior of APS for the purpose of verifying requirements on timing, power, and alignment error.

Analyzing the operational behavior of APS requires a variety of information about APS and its interfacing components. In common systems engineering practice, this information is spread across a number of documents (including spreadsheets and presentation slides) and isolated analysis models. These analysis models, which may include discrete event simulations or algebraic models, refer to information captured in a variety of documents, but this link is typically not explicit. Any change to the system specification - which may include changes in requirements, interfaces, procedures and protocols and, in some cases, changes in structure - usually necessitates an update to other artifacts, including any analysis models. This is typically an error-prone process, and artifacts are synchronized often only infrequently.

Further author information:
Sebastian J. I. Herzig: E-mail: sebastian.j.herzig@jpl.nasa.gov, Telephone: +1 818 393 3059

In our work, we follow the Model-based Systems Engineering paradigm. The Systems Modeling Language (SysML™) is used to create a holistic *system model*. This system model offers an integrated view on the requirements, structure, behavior, and parametric relationships between APS, its components, and interfacing components of TMT. The advantage of using a formal language such as SysML™ is its well-defined semantics and the ability to interpret the model computationally. This enables, e.g., solvers such as the Cameo Simulation Toolkit for the SysML™ modeling tool NoMagic MagicDraw to *execute* and simulate the system in the time domain. An additional advantage of using system modeling languages such as SysML™ is the ability to maintain a consistent view of the system and single source of authority for systems engineering related information.

The system model development is heavily driven by *use cases*. Use cases represent high level procedures that APS uses to perform various telescope alignments and calibrations (i.e., operational scenarios). With these use cases we are able to analyze, e.g., both power usage and the length of time a given procedure requires to complete. This paper adds to previously published work by offering a complete view on operational analysis of APS, and introduces a novel approach to error budget analysis in SysML™ which is demonstrated through application to analyzing pointing errors.

The remainder of this paper is structured as follows: in section 2 our approach to modeling operational scenarios using SysML™ is introduced. Section 3 details how our system model is used for the purpose of analyzing the performance of, and verifying requirements on APS. Towards the end, we present a review of related work in section 4. The paper closes with a summary of the main insights and conclusions from the work.

2. MODELING OPERATIONAL SCENARIOS IN SYSML

Operational scenarios are high level procedures that APS uses to perform various telescope alignments and calibrations. As such, we use operational scenarios to verify requirements on the performance of the telescope as specified. Primarily, this includes verifying requirements on (a) timing, (b) electrical power, and (c) pointing errors. It should be noted that, thus far, we have not yet covered use cases that are specific to Assembly, Integration and Verification (AIV) or commissioning, or off-nominal use cases.

We have modeled a number of operational scenarios, the current set of which is briefly introduced in the following. The scenarios were modeled using SysML™ by following the principles of the *Executable Systems Engineering Method* (ESEM).[2] ESEM prescribes the functional and physical decomposition of the system into a nested tree of components, as well as the specification of the behavior of each. Requirements are formally captured through manual translation to mathematical constraints and the binding of any variables in these to behavioral or structural properties of the system. This method is detailed in the second part of this section using *Maintenance Alignment* as a guiding example. The system model is open source [*].

2.1 List of Modeled Operational Scenarios

To cover the full spectrum of interactions with other subsystems of TMT, and to accurately predict the performance of APS, a number of operational scenarios have been modeled. Currently, this set consists of:

- **Post Segment-Exchange Alignment:** re-align the telescope after new segments have been installed or exchanged. It is estimated that 8 segments will need to be exchanged in a single night every two weeks.
- **Maintenance Alignment:** re-align the telescope in between segment exchanges.
- **Rigid Body M3 Alignment:** align M3 in rigid body motion. The main impact of M3 motion is pupil motion, which APS can measure. M3 will be aligned at a single elevation angle.
- **Off-Axis Wavefront Measurements:** perform off-axis wavefront measurements at any point in the telescope field of view. This scenario is primarily used to diagnose telescope problems as well as confirm the telescope performance off-axis.

[*]Available at http://www.github.com/Open-MBEE/TMT-SysML-Model

- **On-Sky Measurement of Segment Warping Harness Influence Functions:** on-sky measurements of the segment warping harness influence functions. The control matrix for the warping harness is generated and used in the correction of rigid body and segment figure.
- **APS Self Test:** execute tests using internal light sources to confirm that it is functioning correctly.
- **M1CS Sensor Calibration with Post Segment-Exchange Alignment:** perform calibration of the M1 control system (M1CS) sensors. In this scenario, post segment exchange alignment at a single elevation is run, followed by an alignment at two additional elevation angles.
- **M2 and M3 Rigid Body Gravity Calibration:** align M2 and M3 at multiple elevation angles in order to provide the data needed for generation of calibration of M2 and M3 motion with telescope elevation angle. APS is responsible for collecting this data, but the other sub-systems are responsible for using this data to generate the needed calibrations.

2.2 Components and Interfaces

Both the structure and behavior of APS is captured in the system model. Structure includes the encapsulation of properties of a system, and the decomposition of a system by either function or physical boundaries. Interfaces formally capture any and all exchange of information, energy or matter between components. The captured behavior of each component specifies how the state and properties of a component change, and when and how components interact. Requirements are imposed on both structure and behavior. Since behavior is defined for each component, the first logical step in creating the system model is modeling the structure.

In ESEM, the structure of a system is modeled by starting with the operational domain. This operational domain specifies the context in which the mission (here: aligning and phasing TMT) exists. The mission encompasses all elements which are needed to achieve the goals. This includes elements that are external to APS

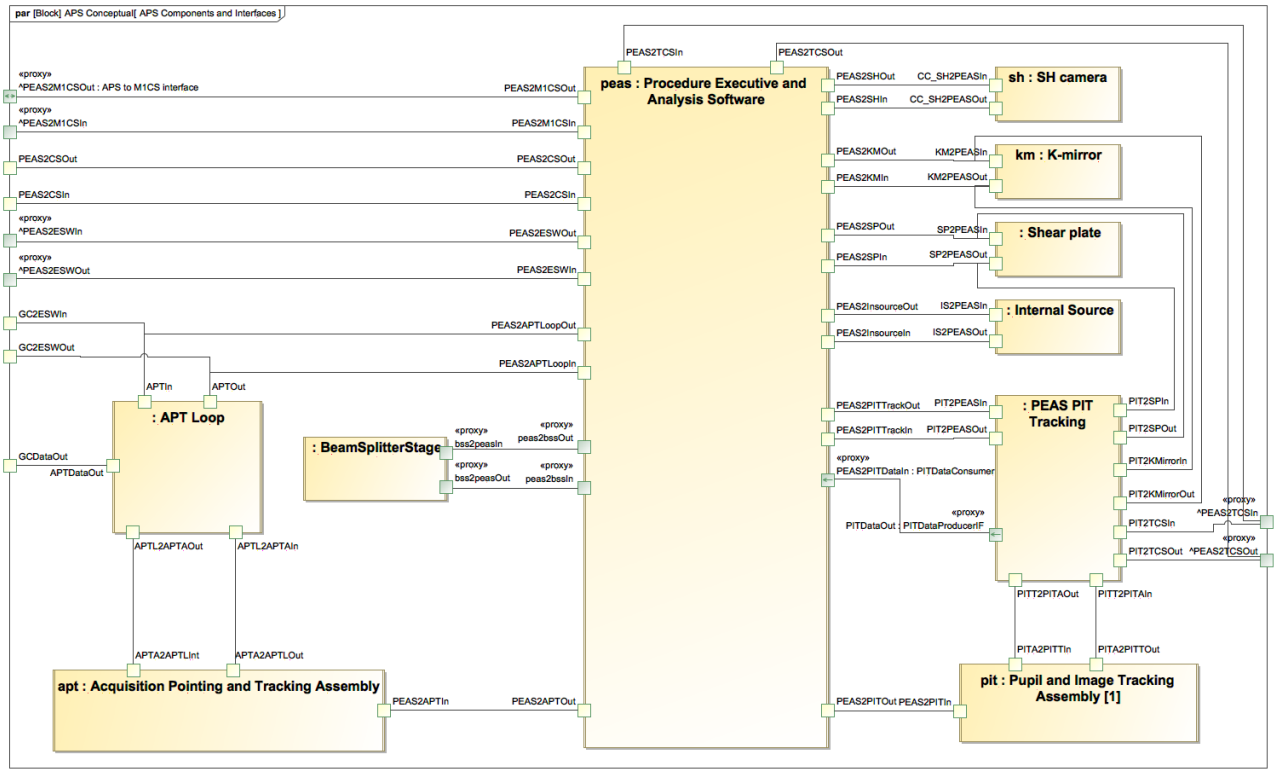

Figure 1. Conceptual model: interfaces, and components of APS with their respective APS-internal interfaces.

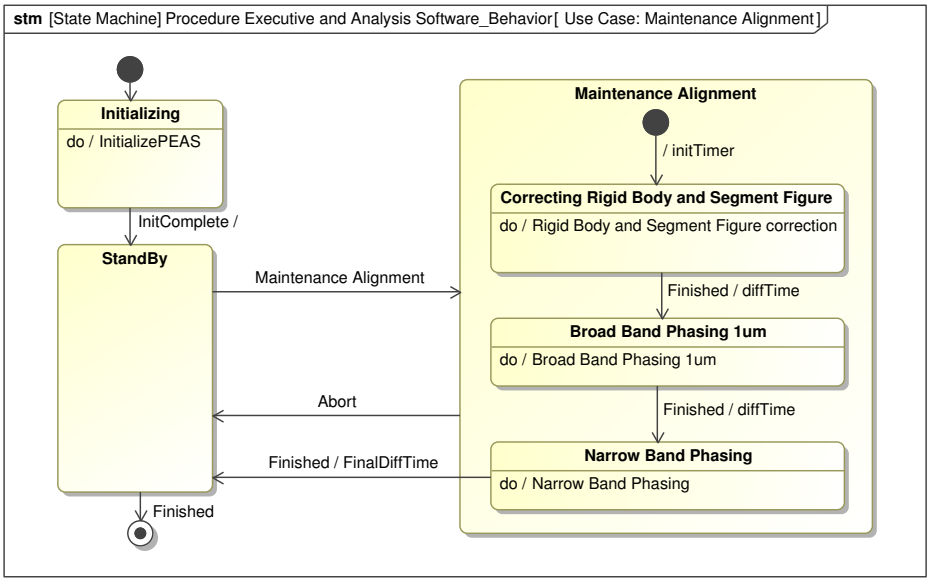

Figure 2. Partial behavioral specification of the *Procedure Executive Analysis Software* (PEAS) of APS.

(e.g., the M1 control system and Telescope control system) and APS itself.[3] Since APS is the system of interest, it is further broken down into sub-components. Figure 1 illustrates a view on this structural decomposition, which also shows the various interfaces of each component (here: data flow) and of APS with external components (at the diagram boundary). These "outer" interfaces are part of the specification of the interfaces with components in the overall mission that are external to APS.

2.3 Operational Behavior of APS Components

In specifying the behavior of components, we distinguish between the *lifecycle behavior* and distinct, isolated activities of a component. The lifecycle behavior describes the behavior of a component from beginning to end-of-life, and describes the state of a component at any instant in its operational lifetime. In SysML™, the lifecycle behavior is the *classifier behavior*.

SysML™ supports multiple behavioral formalisms, and uses three primary notations for describing behavior: state machines (state charts), activity diagrams (flow charts), and sequence diagrams. We have found state machines to be most suitable for specifying the lifecycle behavior of components. Primarily, this is due to state machines offering a natural way for specifying the various possible modes of a system (including failure modes) using states, as well as the conditions for transitioning between them. For capturing specific tasks that represent only a part of the overall behavior of a component, activity diagrams have shown to be most practicable. The nature of such flow charts allow the specification of sequential and parallel behavior, decision points and synchronization points among other features, allowing for an intuitive description of intended behavior.

The specification of the operational behavior of APS and its components is driven by the operational scenarios introduced in section 2.1. In most cases, the *Procedure Executive Analysis Software* (PEAS) plays a central role in their description. Figure 2 illustrates the specified behavior of PEAS during the operational scenario *Maintenance Alignment*. After initialization, PEAS is in a `StandBy` state. An external signal `Maintenance Alignment` (which may be provided by a user) triggers it to transition to the *Maintenance Alignment* state. PEAS is specified to only be able to leave this state if an `Abort` signal is provided (internally or externally). Otherwise it proceeds with first correcting the rigid body and segment figure, then performs broad band phasing and finally narrow band phasing. Once completed, PEAS returns to the `StandBy` state.

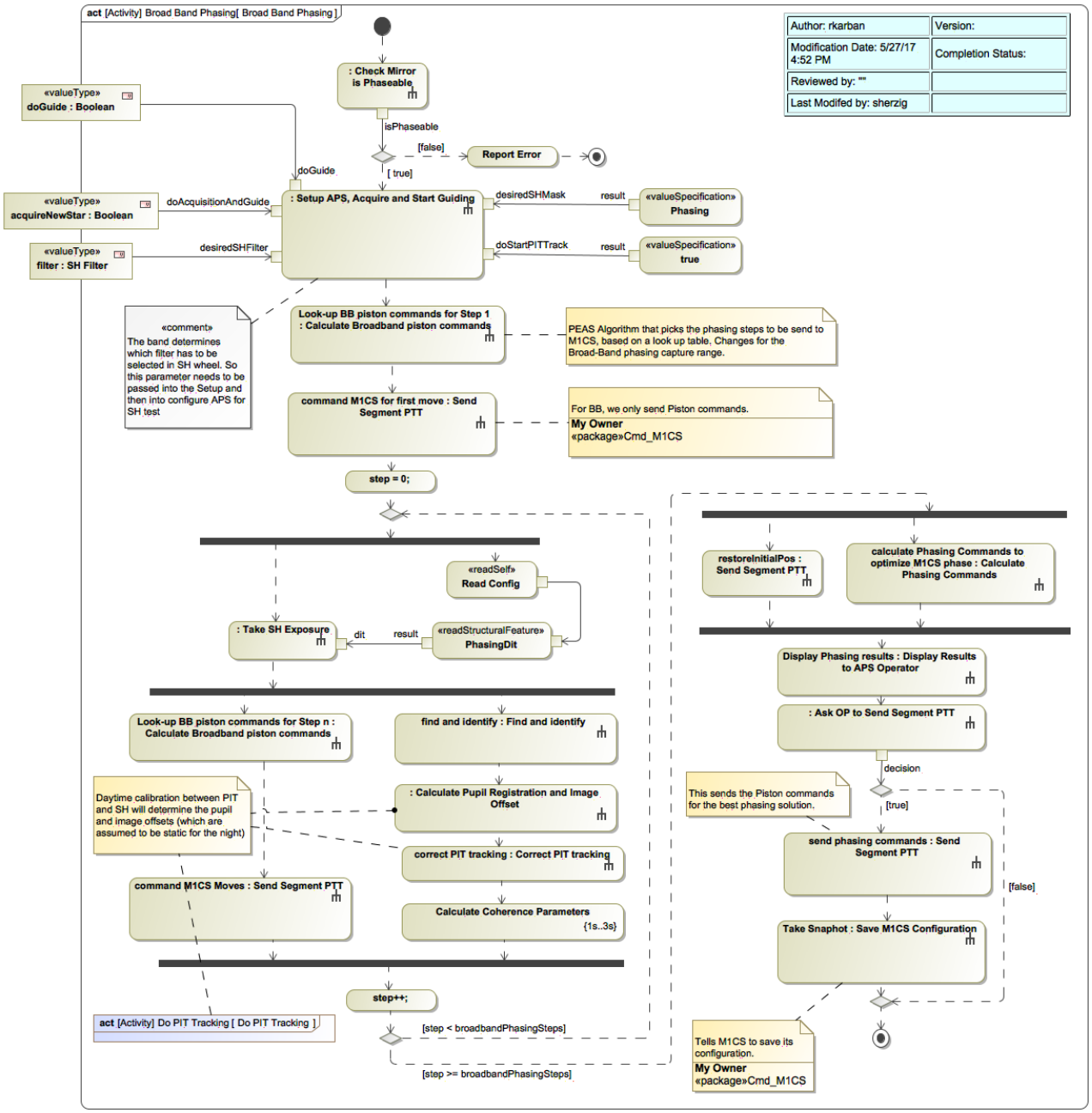

Figure 3. Specified behavior of PEAS during the *Broad Band Phasing* activity.

The "do" behavior of each state (or here: phase of the use case) specifies the behavior of PEAS when in that particular state [†]. Figure 3 illustrates this "do" behavior, defined using an activity diagram, for broad band phasing (see *do / Broad Band Phasing 1*µm in Figure 2). The goal of this activity is the reduction of the steps (or piston errors) between the individual mirror segments to less than 1µm. In the APS for TMT, this is accomplished mainly using the Shack-Hartmann camera and correlating individual subimages, similar to the implementation at the Keck observatory.[4] The activity begins with setting up APS, acquiring a start and start guiding. Parameters for this activity include the Shack-Hartmann filter to be used and the desired Shack-Hartmann mask to be used. Note that this part of the overall flow of events is a *call* to another activity, where the setup procedures are specified in more detail. This method of calling other activities allows for the reuse of isolated, distinct tasks within multiple contexts without having to redefine the flow of events. After initialization, the Shack-Hartmann camera is instructed to take a first image. The desired exposure time is passed in as a parameter, and is defined in the property *PhasingDit*, which is a property defined in the context of the PEAS component directly. This is executed in a loop, with the coherence parameters being calculated after every image is taken while, in parallel, the mirror segments are moved. After a number of images are taken (the number of images is also defined in a property in the context of the PEAS component), the initial position of the telescope is restored, and the results of the main broad band phasing activity displayed to an operator. The operator is then asked to confirm (or disprove) the computed new segment positions, after which the telescope is moved back to the position previously identified.

2.4 Interactions Between Components

Figures 2 and 3 specify the behavior of PEAS. Notice how in Figure 3 a number of exposures are specified to be taken using the Shack-Hartmann camera. Yet, according to the structural breakdown (see Figure 1), PEAS and the SH Camera are two separate components. They are, however, connected. Similarly, PEAS is specified to request M1CS to move segments of the primary mirror, which is a component outside of APS.

Formally, we specify such interactions over interfaces using the concept of *ports* and *connectors* in SysML™. As introduced earlier, we specify the lifecycle behavior of each component using state machines, where events such as signals (i.e., messages) can trigger state transitions. In SysML™, signals can be passed over ports, and can have data attached to them. Actions with the semantics of receiving and waiting for signals are elementary to SysML™, allowing for communication between components using just signals.

Figure 4 illustrates the sending of a signal `SH_Take_exposure_Cmd` over the port `PEAS2SHOut` of PEAS (see Figure 1). An excerpt of the state machine defining the lifecycle behavior of the SH Camera is shown in Figure 5 (left). If the SH camera is currently not in a state in which it is taking an exposure, receiving the signal

[†]SysML™ also allows for the behavior when *entering* and *exiting* the state to be defined.

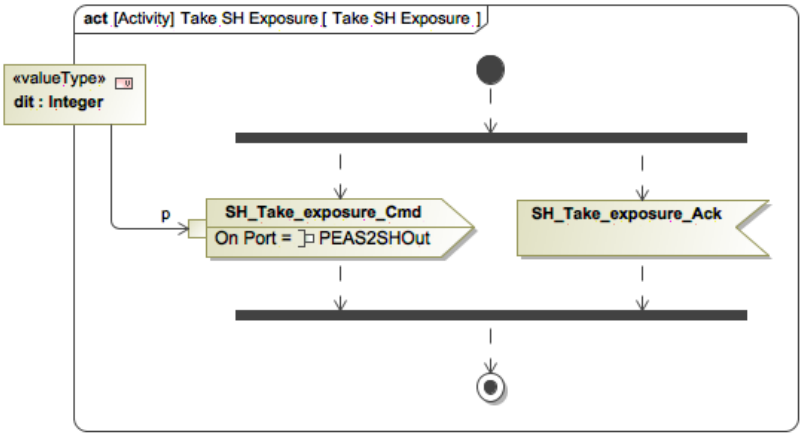

Figure 4. Interaction with Shack-Hartmann Camera: *Take SH Exposure* activity.

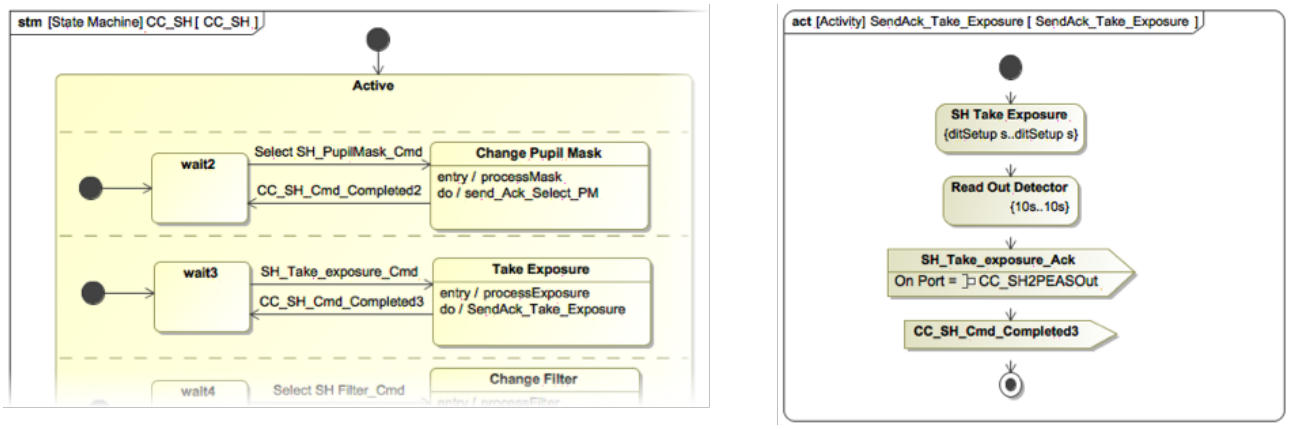

Figure 5. Specified behavior of SH filter (partial): (a) lifecycle behavior and (b) taking exposure.

`SH_Take_exposure_Cmd` leads to a transition into the *Take Exposure* state. This, in turn, triggers the "do" behavior *SendAck_Take_Exposure* (see Figure 5 (right)), in which an exposure is taken, the detector is read out, and the results are sent back. Note that the exposure time is attached to the incoming signal as a payload (see Figure 4). Similarly, the exposure image itself is sent back as a payload of the signal that PEAS is waiting for (namely, `SH_Take_exposure_Ack`), over the port `CC_SH2PEASOut` which connects the SH camera to PEAS.

2.5 Scenario Execution Drivers

To simulate an operator initiating a particular operational scenario, signals must be injected that trigger appropriate behavior of the various components. For instance, in the case of *Maintenance Alignment*, the signal `Maintenance Alignment` triggers the start of the maintenance alignment scenario. In our approach, we have chosen to substitute an operator with an *Analysis Driver* that interacts with APS (or one of its components) simply by sending a signal to it. Practically, this can be done using sequence diagrams. Figure 6 illustrates this: first, we abort any potentially running scenarios. Then, after 60s, maintenance alignment is initiated. As a failsafe, the operational scenario is aborted by sending the `Abort` signal after 2000s.

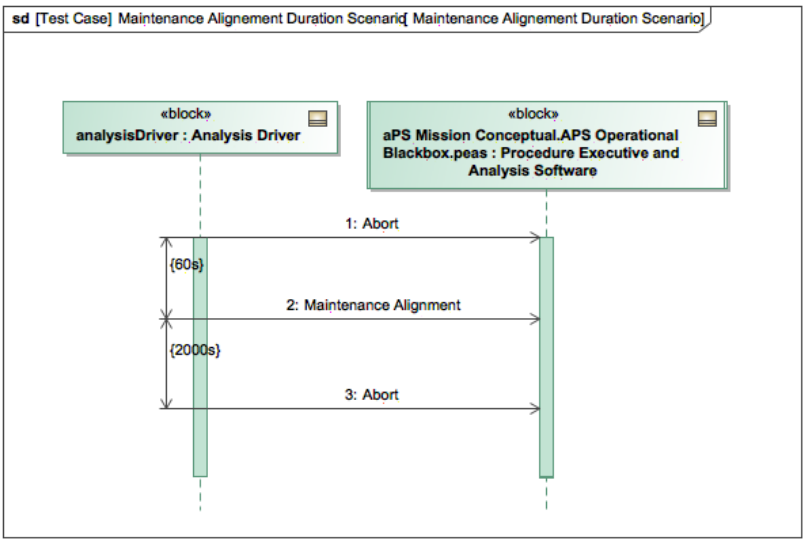

Figure 6. Flow of events initiating the operational scenario *Maintenance Alignment*.

3. VERIFYING REQUIREMENTS ON OPERATIONAL BEHAVIOR IN SYSML

SysML^{TM} behavioral models have reasonably well defined execution semantics.[5] This allows for the execution of the models of the operational scenarios introduced in section 2, thereby allowing for complex system simulation verification of a variety of operational performance requirements by analysis. In this section we focus on performing three types of analysis of operational behavior for the purpose of requirements verification: timing (duration) analysis, power usage analysis, and error budget analysis.

3.1 Tying Operational Scenarios to Requirements

In the system model, APS is defined from two perspectives: the *customer perspective*, and the *supplier perspective*. The customer perspective is modeled as a *"black box"* system that is *specified* by a number of requirements, interfaces, and externally visible behavior. The supplier perspective is modeled as a specialization of the customer perspective and captures supplier-specific decisions and technical solutions that meet the requirements.

Customer provided requirements are textual in nature, and are stored in an external requirements management system. These requirements are imported into the SysML^{TM} model. Given that the requirements are all expressed using natural language, their use in computation is limited. Therefore, each requirement that is relevant to analyzing the performance of APS is manually translated into a mathematical constraint. In our approach, SysML^{TM} blocks *refine* natural language requirements, and are composed of one or more mathematical constraints, which are modeled using SysML^{TM} constraint blocks. The blocks representing requirements are then modeled as being a composite part of the specification of APS. This formalization of requirements using mathematical constraints and SysML^{TM} -specific constructs is illustrated in Figure 7.

The definition of executable behavior and a formalization of requirements using mathematical constraints provides the key ingredients for checking whether a particular performance requirement can be met. However, they are seemingly still disconnected, also from the system structure. This connection can be formalized using a SysML^{TM} parametric diagram. Given that the system structure is modeled as the structural decomposition of the "black-box" element that also has a direct reference to the requirements, a common context is present in which the two can be related. Behavior simulations influence properties of the structure: e.g., the value of a property "tFinal", which is a property of the structural block PEAS and denotes the value of the simulation clock at the end of the simulation, is set when PEAS transitions back into the *StandBy* state (see Figure 2. These structural properties can now be formally related using *binding connectors* (which equate source and target).

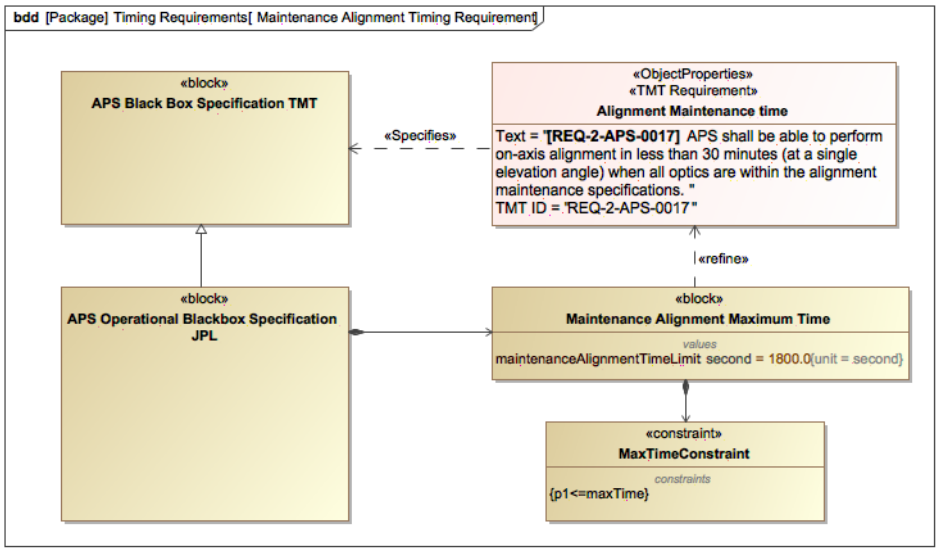

Figure 7. Requirement as imported from DOORS, and formalization using mathematical constraints.

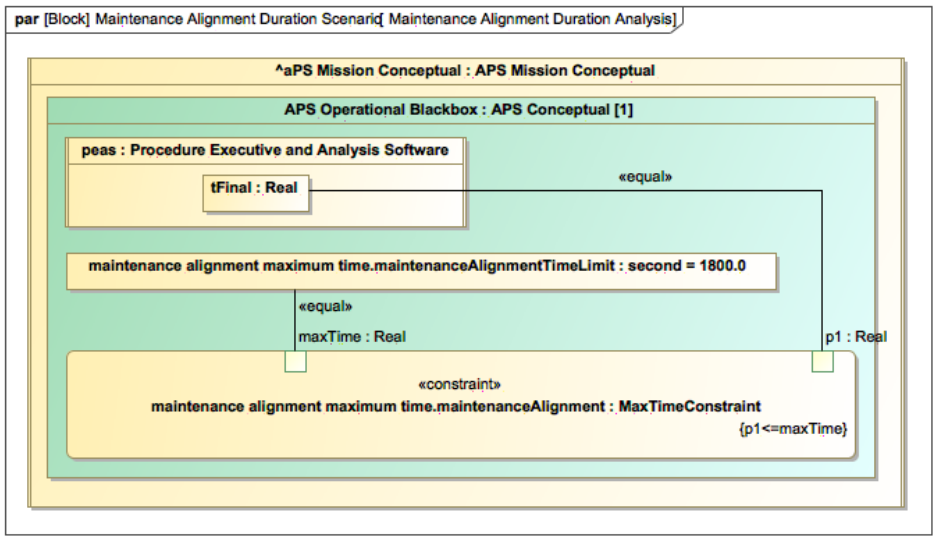

Figure 8. Connecting properties of APS to formalized requirements.

This is illustrated in Figure 8. Running a particular operational scenario using an execution engine such as the Cameo Simulation Toolkit would now lead to the value "tFinal" being set, and trigger checking of the constraint. The requirement is considered verified (by analysis) if the constraint is not violated.

3.2 Verifying Timing Requirements

SysML^{TM} defines *duration constraints* which may be specified to have a lower and upper bound. These duration constraints are typically applied to elements in behavioral diagrams and have the effect of the simulation engine interpreting the particular step in the simulation to take a particular amount of time. For instance, consider the activity diagram in the (right part of) Figure 5. Both *SH Take Exposure* and *Read Out Detector* have duration constraints applied: for instance, *Read Out Detector* has a lower and upper bound of 10 seconds. Simulation engines can typically be configured to either use the lower or upper value of a duration constraint, or use a random value (assuming a distribution) in each run. The latter is used, e.g., in the design of the Adaptive Optics System (AOS) of TMT to run a Monte Carlo simulation for determining with which certainty a particular requirement can be met.[6]

Advanced simulation engines, such as the Cameo Simulation Toolkit, are capable of taking into account effects on duration constraints when parallel behavior is analyzed. This allows for highly complex operational behavior to be analyzed. Table 1 summarizes the results of applying the described pattern to the various operational scenarios from section 2. The table contains both the predicted timing (using only the upper bound value), and the requirement (if any).

Table 1. Results of timing analysis.

Operational Scenario	Predicted Timing	Requirement
Post-Segment Exchange Alignment	1h 15min	<2h
Maintenance Alignment	26min	<30min
Rigid Body M3 Alignment	79s	TBD
Off-Axis Measurements	50min	TBD
On-Sky Measurement of Segment Warping Harness Influence Functions	10h	TBD
Self Test	10min	1h downtime / yr
M1CS Sensor Calibration with Post-Segment Exchange Alignment	2h 20min	<3h
M2 and M3 Rigid Body Gravity Calibration	1h 10min	TBD

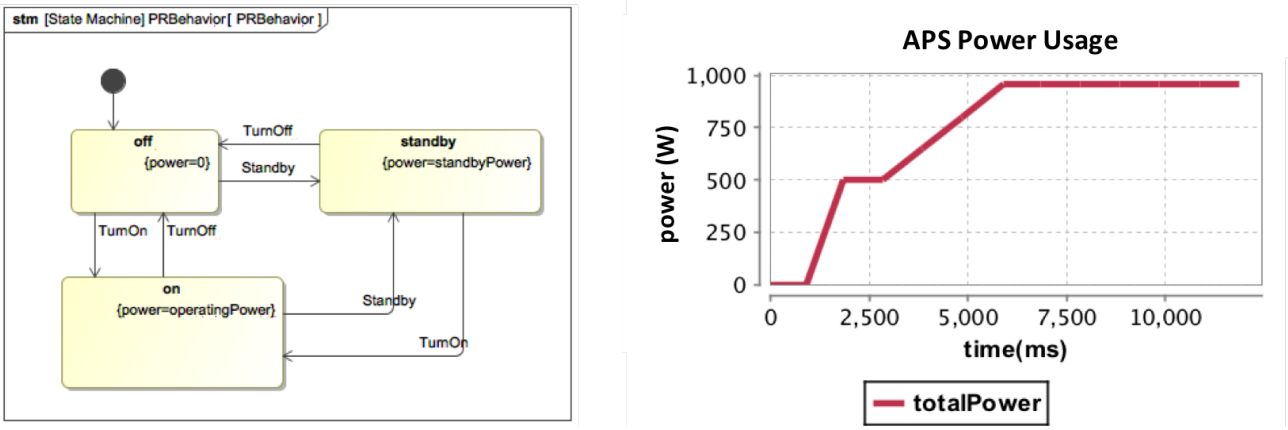

Figure 9. Peak power consumption analysis: (a) general pattern used and (b) result of executing analysis.

3.3 Verifying Power Requirements

Similar to performing timing analysis, requirements on the power consumption can also be analyzed. While SysML^{TM} does not provide an explicit construct for power as a resource, the power usage of a component at any instant in time (i.e., $P(t)$) can be modeled as a numerically valued property of the component. Since power consumption can often be associated with the state that a component is in, it is prudent to model power consumption as a *state invariant*. This is illustrated in Figure 9 (left).

What is often of interest in analyzing the performance of a complex system, is the aggregate power usage over time. For instance, APS, and TMT in general, consists of a number of components (SH camera, PEAS, motors, controllers, etc.). But, not all components are active in their most power consuming modes at all times. However, given the precise definition of the operational behavior of all components, adding the additional state invariants defining the power usage of a component that is in a particular state provides one with the necessary information to plot total power consumption over time[‡]. Specific power scenarios can also be defined - e.g., a *peak power scenario*, where a certain set of instruments is turned on in quick succession. The result can then be plotted using a diagram similar to Figure 9 (right).

3.4 Pointing Error Analysis

In systems engineering practice, technical resources (such as cost, mass, data, or power) are often managed using the concept of *budgets*. That is, at any point in the development lifecycle, there is a maximum possible, maximum expected, and current best estimate (CBE) for any technical resource. The maximum possible is often equal to an *allocated* value. The current best estimate evolves as the design matures. Margin is defined as how much *growth* of a technical resource usage is possible. In telescope applications, *error budgets* play a central role. Similar to other technical resources, portions of an overall acceptable error are allocated to various aspects of the telescope. Verifying requirements on the operational behavior includes verifying that the aggregate expected error is below the maximum allowed error.

As illustrated in Figure 10, we model the current best estimate, allocated value and margin as properties of a SysML^{TM} block. We follow the following pattern: an element that is, at some level in the inheritance hierarchy, a *ErrorRollUpPatternElement* inherits three properties referring to the CBE and allocated value, as well as the margin. All three can be derived from the values of its sub-components (using either a root sum squared, product, or sum of values of the sub-component), or any may be directly specified. For instance, the CBE value will often be specified directly, while the margin will typically be derived.

Decomposition is a natural mechanism for allocating error budgets. In practice this decomposition can be distinctly different from a structural decomposition. For instance, in the example given in Figure 10 (right),

[‡]Aggregate numbers are simply formulated using parametric blocks, similar to how mass and other roll-ups are performed in the TMT model.[2]

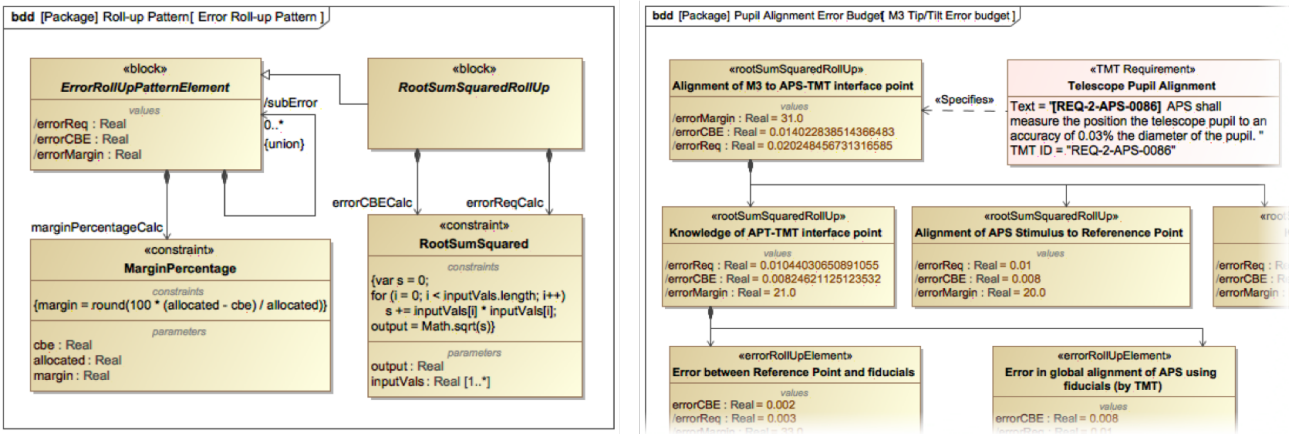

Figure 10. Pointing error budget analysis: (a) general pattern used and (b) example application of pattern.

most errors refer to properties of a (mechanical) interface rather than a single structural component. Whenever relevant and possible, properties related to the components involved can be bound to the error budget using SysML^{TM} parametrics, similar to how required values are bound to the relevant *errorReq* properties.

4. RELATED WORK

With SysML^{TM} being the de-facto standard in applied Model-based Systems Engineering practice, there are numerous publications describing the use of SysML^{TM} for modeling complex systems. However, only few publications discuss the use of the execution semantics of SysML^{TM} for purposes of system analysis and requirements verification. In the following, we focus on related publications within the telescope domain.

Karban *et al.*[7] created a comprehensive system model in SysML^{TM} of an actual operational demonstrator for mirror phasing in the context of the European Extremely Large Telescope (E-ELT). This work was performed in the context of the *International Council on Systems Engineering* (INCOSE) *SE*^{2} MBSE Challenge. The goal of the work was to provide examples of SysML^{TM}, common modeling problems and approaches, and to build a comprehensive model that is to serve as the basis for providing different views for different engineering aspects and associated activities. While the SysML^{TM} model was not mean to be executed directly, Karban *et al.* describe the use of model transformations for the purpose of simulation model and code generation.

Selvy *et al.*[8] describe their use of SysML^{TM} for the development of the operational plan of the Large Synoptic Survey Telescope (LSST). Specifically, all systems engineering planning and definition activities that have historically been captured in paper documents are captured in a SysML^{TM} model. Model transformations aid in integrating with external tools such as specialized project management tools. The approach lead to full traceability from initial requirements to scheduled, costed, and resource loaded activities.

Filgueira *et al.*[9] describe a end-to-end modeling approach using (textual) domain-specific (modeling) languages. These domain-specific languages expand into several knowledge domains including control, data processing and observatory operations. The use of domain-specific languages gives access to a more precise vocabulary for defining particular aspects of an overall system. The authors use SysML^{TM} primarily for purposes of visualizing the textual artifacts to communicate quantitative information if needed.

5. CONCLUSION

In this paper, we discuss the use of SysML^{TM} for creating a comprehensive system model in which aspects of behavior, structure and requirements are integrated formally. We also present how this system model can be used for analyzing the operational behavior of the system by making use of the well defined execution semantics of behaviors modeled in SysML^{TM}. Specifically, we introduce how requirements on timing, power and alignment errors can be verified, and formally linked to requirements and the system architecture.

We have found the approach to be highly effective in designing APS. For instance, behavioral specifications can be easily modified, and the impact of the changes on timing, power usage, and other operational properties quickly evaluated. Unfortunately, the approach still requires manual translation of artifacts specified in natural language (primarily requirements) to a formal representation (e.g., requirements as mathematical constraints). This introduces a possible source for inconsistencies. Such inconsistencies could be avoided if all systems engineering related information were captured only using formal (i.e., computer-interpretable) languages. However, this would require a significant paradigm shift.

Furthermore, we have found SysML^{TM} to work very well within the context of certain system level analyses, but not generally a suitable replacement for more specialized analyses. For instance, applications requiring geometric reasoning, or finite element analyses, are best expressed using tools and languages specifically created for that purpose. However, for many other tasks traditionally completed using spreadsheets or document, SysML^{TM}, and MBSE in general, offer a plethora of advantages, including better traceability, less ambiguity, and the ability to generate other artifacts automatically. While more effective, the use of SysML^{TM} comes at a price: the learning curve is steep. However, this is also the case for many frameworks and languages (e.g., programming languages, or even many branches of mathematics).

ACKNOWLEDGMENTS

This research was carried out in part at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. The authors gratefully acknowledge the support of the TMT collaborating institutions. These are the California Institute of Technology, the University of California, the National Astronomical Observatory of Japan, the National Astronomical Observatories of China and their consortium partners, the Department of Science and Technology of India and their supported institutes, and the National Research Council of Canada. This work was also supported in part by the Gordon and Betty Moore Foundation, the Canada Foundation for Innovation, the Ontario Ministry of Research and Innovation, the Natural Sciences and Engineering Research Council of Canada, the British Columbia Knowledge Development Fund, the Association of Canadian Universities for Research in Astronomy (ACURA), the Association of Universities for Research in Astronomy (AURA), the U.S. National Science Foundation, the National Institutes of Natural Sciences of Japan, and the Department of Atomic Energy of India.

REFERENCES

[1] Troy, M., Chanan, G., Michaels, S., Bartos, R., Bothwell, G., Giveon, A., Hein, R., Radin, M., Roberts, J., Rodgers, J. M., Scherr, L. M., Seo, B.-J., and Zimmerman, D., "A Conceptual Design for the Thirty Meter Telescope Alignment and Phasing System," in [*Proc. SPIE*], **7012**, 70120Y (2008).

[2] Karban, R., Jankevičius, N., and Elaasar, M., "Esem: Automated systems analysis using executable sysml modeling patterns," in [*INCOSE International Symposium*], **26**(1), 1–24, Wiley Online Library (2016).

[3] Karban, R., Dekens, F. G., Herzig, S., Elaasar, M., and Jankevicius, N., "Creating system engineering products with executable models in a model based engineering environment," *Modeling, Systems Engineering, and Project Management for Astronomy VI, SPIE, Edinburgh, UK* (2016).

[4] Chanan, G., Troy, M., Dekens, F., Michaels, S., Nelson, J., Mast, T., and Kirkman, D., "Phasing the mirror segments of the keck telescopes: the broadband phasing algorithm," *Applied Optics* **37**(1), 140–155 (1998).

[5] Object Management Group, "Semantics of a Foundational Subset for Executable UML Models." Online (Jan. 2016).

[6] Trancho, G., Wang, L., Herzig, S., Karban, R., Boyer, C., Herriot, G., Anderson, D., and Ellerbroek, B., "Analyzing the Operational Behavior of NFIRAOS LGS MCAO and IRIS Acquisition on the Thirty Meter Telescope using SysML," in [*Adaptive Optics for Extremely Large Telescopes (AO4ELT)*], (2017).

[7] Karban, R., Hauber, R., and Weilkiens, T., "Mbse in telescope modeling," *Insight* **12**(4), 24–31 (2009).

[8] Selvy, B. M., Claver, C., and Angeli, G., "Using sysml for verification and validation planning on the large synoptic survey telescope (lsst)," in [*SPIE Astronomical Telescopes+ Instrumentation*], 91500N–91500N, International Society for Optics and Photonics (2014).

[9] Filgueira, J. M., Bec, M., Liu, N., Peng, C., and Soto, J., "End-to-end observatory software modeling using domain specific languages," *Software and Cyberinfrastructure for Astronomy III* **9152**, 91521O (2014).