Resource Analysis and Automated Verification for the Thirty Meter Telescope using Executable SysML Models

(Invited Talk)

Nerijus Jankevicius
No Magic Europe
Kaunas, Lithuania
nerijus@nomagic.com

Abstract— **SysML and its supporting modeling tools have evolved to support a precise execution semantics for an automated verification of design models. This paper gives an overview of the application of such analyses in the context of the Thirty Meter Telescope (TMT), one of the next generation giant optical ground based telescopes.**

Keywords—SysML; Simulation; Model Execution; Verification and Validation; Requirements Verification; Automation; System Requirements; Roll-Up; Test Cases; Analysis; MBSE

I. Introduction

Thirty Meter Telescope (TMT) [10] is under development by the TMT International Observatory (TIO). The examples in this paper are extracted from a production-level SysML model which involves the modeling and analysis of the Alignment and Phasing System (APS) for the TMT, developed by Jet Propulsion Lab (JPL) with the help of the tools, patterns and methods developed by No Magic Inc. The APS team pursues an MBSE approach to analyze the requirements and to demonstrate that the specified design satisfies the requirements. Based on the customer supplied requirements and derived use cases, the goals are to analyze operation scenarios to ensure that power, mass, and duration requirements are always met by the specified system design. The TMT model [12] uses SysML [3] as modeling language.

II. Standards and Tools

Executable models are executed with the help of an execution or simulation engine. The execution of the models was performed using the Cameo Simulation Toolkit (CST) [2], which is a plugin to MagicDraw [1] enabling model execution for early system behavior simulation.

CST is a simulation platform based on OMG fUML standard [4], which defines precise model execution semantics and a virtual machine for UML [5], enabling compliant models to be transformed into various executable forms for verification.

CST uses fUML as a foundation to plug in additional standard engines, such as W3C SCXML (State Chart XML) engine for state machines [7], JSR223 for scripting-action languages, and a parametric solver based on PSCS (Precise Semantics of UML Composite Structures) [6].

The resource analysis makes particular use of the parametric solver, and the behavior (fUML and SCXML) execution engines of CST.

The parametric solver uses the fUML standard to create objects of blocks (UML classes stereotyped with SysML Block) and set their attribute (property) values. Also, as SysML's parametric diagram is based on UML's composite structure diagram, ports (properties at the boundary of classes) and connectors (between properties) are part of the execution model.

One notable kind of connector is a binding connector which makes the values of properties at both ends of the connector equal. If one value changes, the change propagates to the opposite end. These semantics allow the "given" values (of pre-bound attributes) to immediately propagate after fUML object instantiation and update any "target" values (of unbound attributes) after constraints evaluate. Initially, at instantiation of objects and attributes, CST analyzes the causality of attributes, i.e., determines which attributes are given and which are target in the parametric equation. Initial solving provides the given values and derives the target values.

Whenever the value of an attribute that is bound to constraint parameter changes, the constraint is re-evaluated and updates all related variables, creating a cascade effect, which may trigger more and more related constraints evaluations. If that happens during behaviors execution (state machine or activity), CST re-evaluates the parametrics and considers an entire cascade as part of the run-to-completion step (an atomic action which happens at the same instant of time). CST is flexible in dealing with underspecified models as it fills in automatically some gaps concerning initial conditions (e.g. not all parameters of activities value properties need to be fully specified). Based on this feature of CST dynamic (behavior based) roll-ups using parametrics is built in the running model.

III. Automated Requirements Verification and Systems Analysis With SysML

The Executable System Engineering Method (ESEM) [9] defines steps and patterns to construct an executable model to carry out resource analysis:

1. Formalize Requirements
   Requirements are specified by SysML Blocks and Constraint Blocks where a Boolean expression defines the requirement.
2. Specify Design
   The system design is specified by decomposition trees of its functional and physical aspects
3. Characterize Components
   The individual components are augmented by a standard modeling pattern, adding several properties (e.g. power, mass, duration) used in the analysis
4. Specify Analysis Context
   The Analysis context binds formal requirements to the as-designed system properties for analysis.
5. Specify Operational Scenarios
   Operational scenarios elaborate customer use cases as a sequence or activity diagram. Operational scenarios exercise different configuration; e.g. for power consumption
6. Specify Configurations
   The initial state of the operational scenarios (in this case, e.g. the initial power consumption configurations) is specified with different decomposition trees of instance specifications
7. Run Analysis
   The configured analysis is run using a simulation engine, such as CST. This provides several outputs such as: a) timeline of the states of the individual components which shows state changes of individual components during the simulation (Fig 1), b) rolled up power and margin, and c) value profiles, which show the total value over time of the rolled up value (e.g. total power of dome installation part of the APS) – see Fig 2. These products represent different views that can be used by the engineer to analyze the results.

As soon as the results of the different scenarios are available, the satisfaction of the original customer requirement has to be demonstrated. This is done with a separate analysis context.

A typical scenario to verify the as-designed system is when a requirement changes (Fig 3).

A change (1) of a TMT requirement (kept in DOORS (2) as management tool for textual requirements) is propagated to the SysML model, managed in a model repository, where it is formalized into so-called property-based requirements (3), which allow for a formalized trace of requirements into the design. The as-specified conceptual design (4) and/or the realization design (5) are verified against the changed requirement, resulting in pass or fail (6). The systems modeling environment and change scenario is described in more detail in [11].

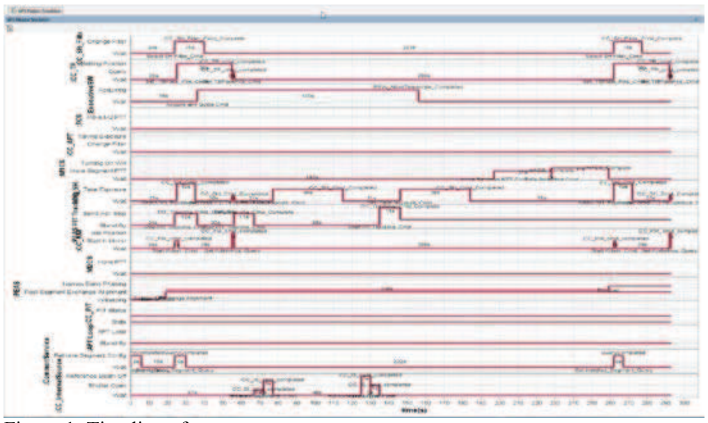
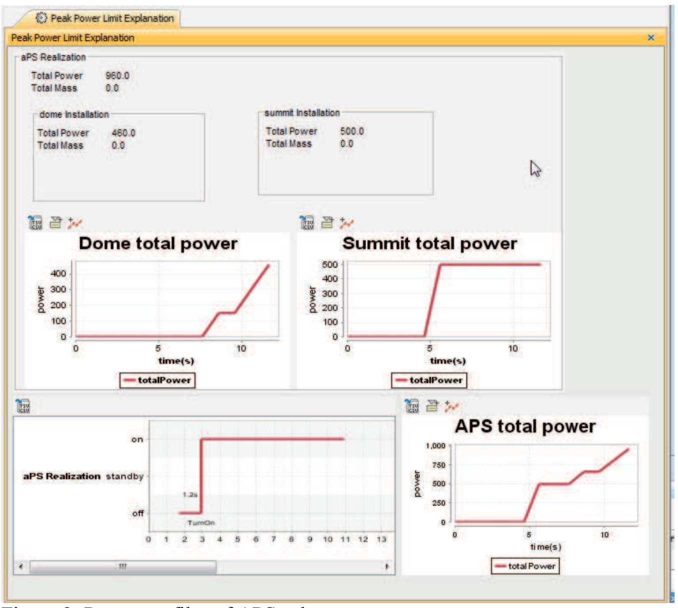


Figure 1. Timeline of component states


Peak Power Limit Explanation

Peak Power Limit Explanation

aPS Realization

Total Power	960.0
Total Mass	0.0

dome Installation

Total Power	460.0
Total Mass	0.0

summit Installation

Total Power	500.0
Total Mass	0.0

Figure 2. Power profiles of APS subsystems

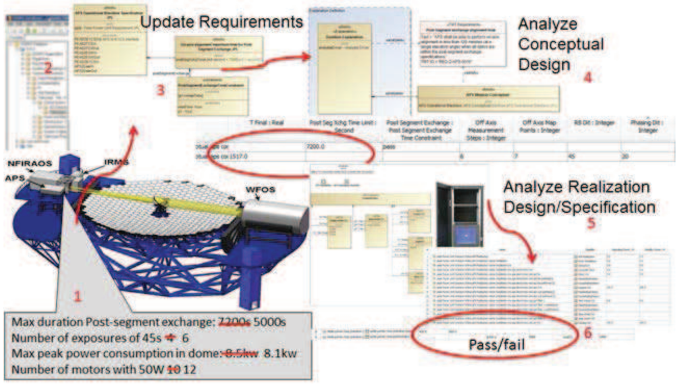

Update Requirements
Analyze Conceptual Design
Analyze Realization Design/Specification
Pass/fail
NFIRAOS
APS
IRMS
WFOS
1
2
3
4
5
6
Max duration Post-segment exchange: ~~7200s~~ 5000s
Number of exposures of 45s ~~4~~ 6
Max peak power consumption in dome: ~~8.5kw~~ 8.1kw
Number of motors with 50W ~~10~~ 12

Figure 3. Propagation of a change in the requirements of APS

IV. Lessons And Experience Gathered From The Executable TMT Model

By driving the analysis directly from data in the model, the inconsistency that often emerges when data is also maintained in analysis tools (e.g., Excel) was eliminated. In addition, performing the analysis by simulating the system design captured in the model has the added benefit of having all the necessary information (e.g., the state of the system at relevant time points) readily available to check. This makes the development of analysis patterns (e.g., rollups) to check the satisfaction of requirements reasonably straightforward.

Different rollups can be applied to different characteristics of a system like power and mass. They all follow the same basic pattern. Originally the mass and power roll-ups for the APS were maintained and calculated in Excel spreadsheets for different (simplified) scenarios and updated whenever a requirement or the system design changed. However, it was completely disconnected from the SysML system model which captures the different operational scenarios, the system design, and the requirements. Integrating the roll-ups simplified the maintenance and consistency of the data (change was only in one place: the system model) and helped automate the checking of requirement satisfaction. Another added benefit is that performing the analysis on the system model directly allows the analysis result to stay in the model, which makes the engineering document that would be generated from the model stay up to date and consistent.

Obviously, not all kinds of system analysis (e.g., finite element analysis) can be performed using the semantics of SysML alone. These analyses would require the augmentation of SysML with other profiles that capture the missing information. Also, the reliability of the analyses that can be performed using this method relies on the reliability of the SysML simulation tools, which may be less mature than their general-purpose counterparts. However, this is often mitigated by integrating the former with the latter.

V. Conclusions and Future Works

Requirement verification is an important kind of analysis that is often performed in the context of MBSE. In this paper, a running example derived from an industrial case study was presented. In the sample a practice is used where automated requirements verification with a set of executable SysML modeling patterns is deployed. These patterns integrate parametrics with the execution semantics of behavioral diagrams.

The presented practice integrates the standard pattern of an analysis context with the behavior execution pattern by specifying a scenario as the behavior of the analysis context's classifier. The instance of the analysis context contains the current state of the scenario, and the parametrics compute the values of selected properties as the scenario runs. The initial state for the analysis context comes from a tree of instance specifications and their property values (a configuration), and the output is another tree that displays the computed values of the analysis.

There are plans to improve some of the patterns related to behavior modeling to support executability. For example, there is a need to find a more formal way to tie constraints on value properties for each power mode to the state. It is desirable to improve the roll-up pattern that deals with state specific constraints, by avoiding hard coding constraint values in state invariant or other constraints. Furthermore, running simulations generates a large amount of results which are currently stored in instance specifications, which make the design model grow unnecessarily. Those results (sequence diagram recording, instance snapshots, simulation logs) should be stored outside of the system model in dedicated data repository.

VI. Acknowledgements

The author would like to acknowledge the major ongoing contributions of Robert Karban to the evolution of system level behavior specification and analysis with executable models.

References

[1] No Magic Inc., 2016. “MagicDraw”. http://ww.magicdraw.com.
[2] No Magic Inc., 2016. “Cameo Simulation Toolkit”. http://www.nomagic.com/products/magicdraw-addons/cameo-simulation-toolkit.html.
[3] OMG, 2014. “Systems Modeling Language (SysML) Version 1.4”.
[4] OMG, 2014. “Semantics of a Foundational Subset for Executable UML Models (FUML) Version 1.1”.
[5] OMG, 2015. “Unified Modeling Language (UML) Version 2.5”.
[6] OMG, 2015. “Precise Semantics of UML Composite Structures (PSCS) Version 1.0”.
[7] W3C, 2015. “State Chart XML (SCXML): State Machine Notation for Control Abstraction. W3C Recommendation 1”.
[8] W3C, 2016. “World Wide Web Consortium”. http://www.w3.org.
[9] Karban, R., Jankevičius, N., Elaasar, M., “ESEM: Automated System Analysis using Executable SysML Modeling Patterns”, Annual INCOSE International Symposium (IS 2016), Edinburgh, UK, 2016.
[10] TMT, 2016. “Thirty Meter Telescope”. http://www.tmt.org.
[11] Karban, R., Dekens, F., Jankevičius, N., Elaasar, M., “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.
[12] “TMT model” https://github.com/Open-MBEE/TMT-SysML-Model.