35th Space Symposium, Technical Track, Colorado Springs, Colorado, United States of America
Presented on April 8, 2019

Model-Based Systems Engineering for Space Architecture Trade Studies

Jeffrey Carpenter
IAI LLC, jcarpenter@integrity-apps.com

Jared Clarke
IAI LLC, jclarke@integrity-apps.com

Michael Sobers, Ph.D.
IAI LLC, msobers@integrity-apps.com

ABSTRACT

Systems engineering has continued to evolve over the last few decades from document-based system design and tracking of requirements to what is now commonly known as Model-Based Systems Engineering (MBSE). MBSE is an evolutionary approach in the discipline of systems engineering where rules, descriptive interconnectivity, and physics-based digital models of the structure and behavior of systems replace drawings and documents as a mechanism to manage, develop and integrate system requirements and architecture. By combining processes, tools, methods, and data, MBSE captures a representation of a system or system of systems to help identify risk-informed investment opportunities. Thus, stakeholders can be assured they are making best-value trades across an architecture at a level of fidelity not available with prior analysis methods.

Using a common MBSE framework, developers and operators of space systems can leverage existing systems engineering artifacts, including requirements specifications, system performance metrics, and even physics-based models to optimize ground- and space-based elements of the system architecture. Since MSBE enables manipulation and evaluation of multiple assumptions across an individual system architecture or across a system-of-systems architecture, stakeholders are assured the final architecture is optimized for a variety of use cases instead of locked by a single set of assumptions.

This paper describes an example of one such MBSE implementation, which is used to determine the feasibility, constraints, and operational parameters of a COTS-based situational awareness sensor system. This example demonstrates the feasibility of using MBSE as an approach to create a shared architecture analysis framework, with standardized assumptions and algorithm inputs that can be established to support analysis by multiple customers in order to draw actionable design and business conclusions on a rapid development pace without sacrificing technical and programmatic rigor.

BACKGROUND AND NEED

IAI recognized a common need across the space community for better situational awareness tools and began to develop a suite of visualization products and analysis tools to meet that need. In the course of that development we noticed the lack of moderately priced data sources with global coverage. Internally, we considered that it might be necessary to develop our own suite of sensors to meet this need. Because these sensors would be distributed globally, to address export control concerns, we focused this investigation on only commercially available cameras. We began to consider how to develop modular arrays of cameras that could be rapidly deployed to any environment using only equipment available from internationally-available prosumer product lines.

If IA determined such a system were feasible, it would eventually need to be tested. On 19 December 2018, a series of UAV incursions shut down operations at Gatwick International Airport outside London, UK for several days. Given the proximity of IAI to Dulles International Airport, this led to an idea to test signal to noise discrimination,

proper motion measurement and target discrimination against a background star field using overflying aircraft at night as a test case. The prototype system, called SkyCamera, was thus conceived to demonstrate this capability.

Technical and Operational Concept

Apart from the need to use COTS parts, the driving technical requirement for this system is the ability to resolve a star field well enough that image processing techniques can discern and characterize artificial near-field objects. Additionally, these arrays needed to survive indefinitely in a variety of climates—from coastal to alpine—and be remotely operable. Finally, we had strong motives to ensure remote operation and data processing would be automated with minimal human intervention.

There are a vast number of camera bodies that can be mounted to an optical bench and come equipped with serial interfaces that allow remote operation. It became clear early on that the ability of our image processing algorithms were our key constraint and the performance of camera imaging arrays would be the primary trade space. Therefore, it was important to pair our MBSE-based design approach with imagery simulation software that could generate synthetic imagery for evaluation and scoring.

Camera Enclosure

Because of the variable locations of these camera arrays, the design of the enclosure itself was a substantial sub-project. However, we did not include it in the system model because the requirements could be very well defined and the components were readily available. We modeled an electrical interface that allowed power supplies to be swapped to accommodate the host country and selected low-maintenance temperature and humidity control systems with a ruggedized exterior. The interior data bus included network connection hardware and a light-weight embedded headless control server.

DESIGN INPUTS AND ANALYSIS NEEDS

In order to preserve flexibility in our choice of sensor hardware, we created a generalized sensor model that interfaced with a suite of generic MATLAB image processing modules. Using this model, we can input specific parameters based on camera and lens selection, enabling choice from a variety of COTS hardware. These parameters are then passed through the generalized sensor directly to the MATLAB codes, along with other physical parameters such as location and orientation, to produce a simulated image. Due to the distributed nature of our small design team, coupled with the desire to apply current digital engineering best practices, we decided to build the system model in an MBSE framework that enabled collaboration between remote users.

Modeling Methodology

The nature of the generalized sensor model, with base properties and functions inheritable to a specialized model for each COTS sensor under consideration, led us to consider an object-oriented modeling approach. For this specific application, we desired to create an executable system model within the MBSE framework. As a result, we chose to develop our system model using an extension to the Object-Oriented Systems Engineering Method[1] (OOSEM) known as the Executable Systems Engineering Method[2] (ESEM). This enabled us to quickly model the basic structure and desired behaviors of the system in a framework that allowed execution of generic image processing algorithms directly from the modeling tools. Our generalized system model and executable simulation workflow then enabled trade studies on a variety of COTS hardware prior to the eventual design of SkyCamera hardware and software.

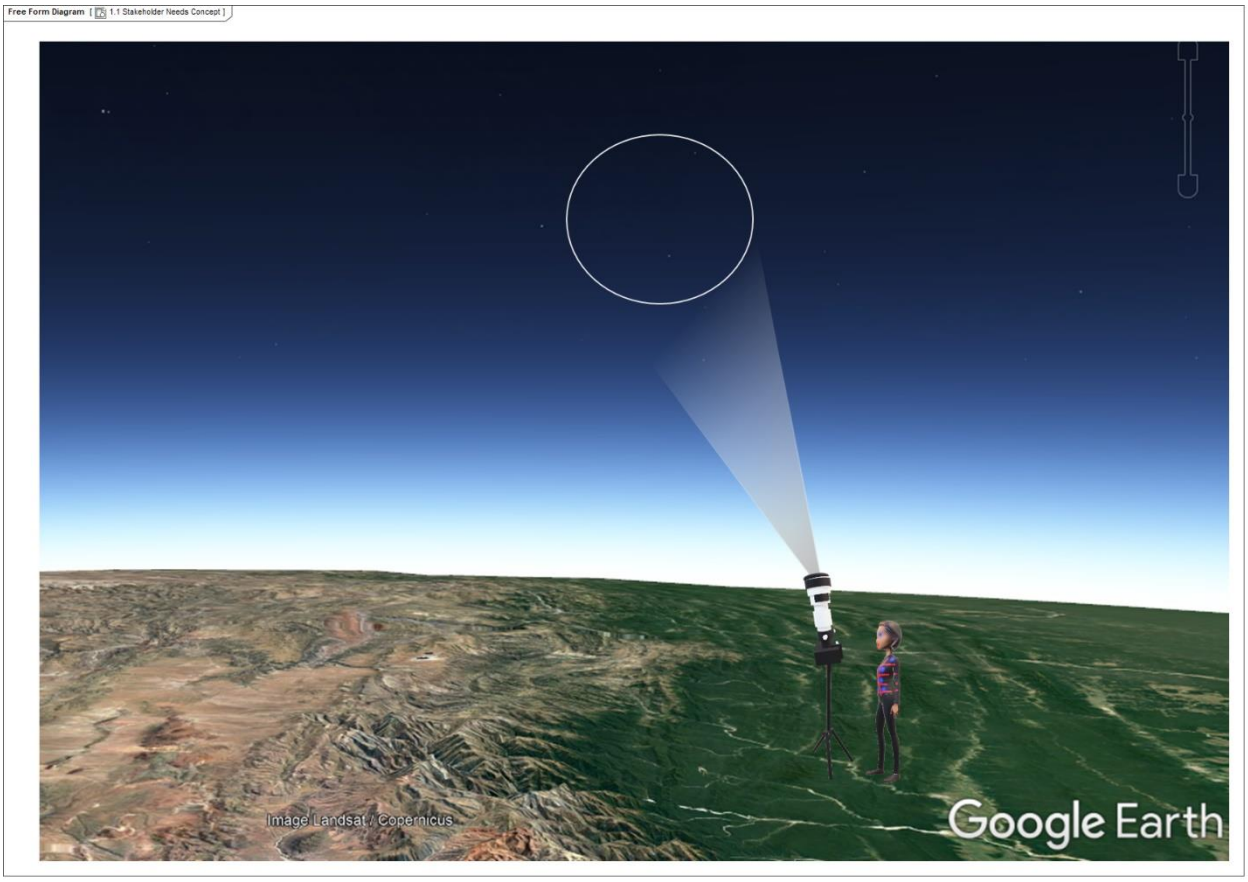

Figure 1: Stakeholder Communication and SkyCamera operational context.

SkyCamera Model Description

After identifying the stakeholder needs and documenting them in the model (see Figure 1), we started modeling the system by building a black-box context consisting of a generic camera and the environment in which it operates. This provided a framework to help us identify which components owned the different parameters that needed to be passed to the MATLAB image processing routines.

Structural Model

Figure 2 below shows the high-level structural decomposition of the system context. We designed generalized components to represent the system structure and functions. The camera, enclosure and supporting hardware, and the environment are all modeled in additional detail as described in Figures 3-4 below. Figure 5 shows an example of how we used MATLAB code imbedded a constraint block to calculate some of the setup parameters during model execution.

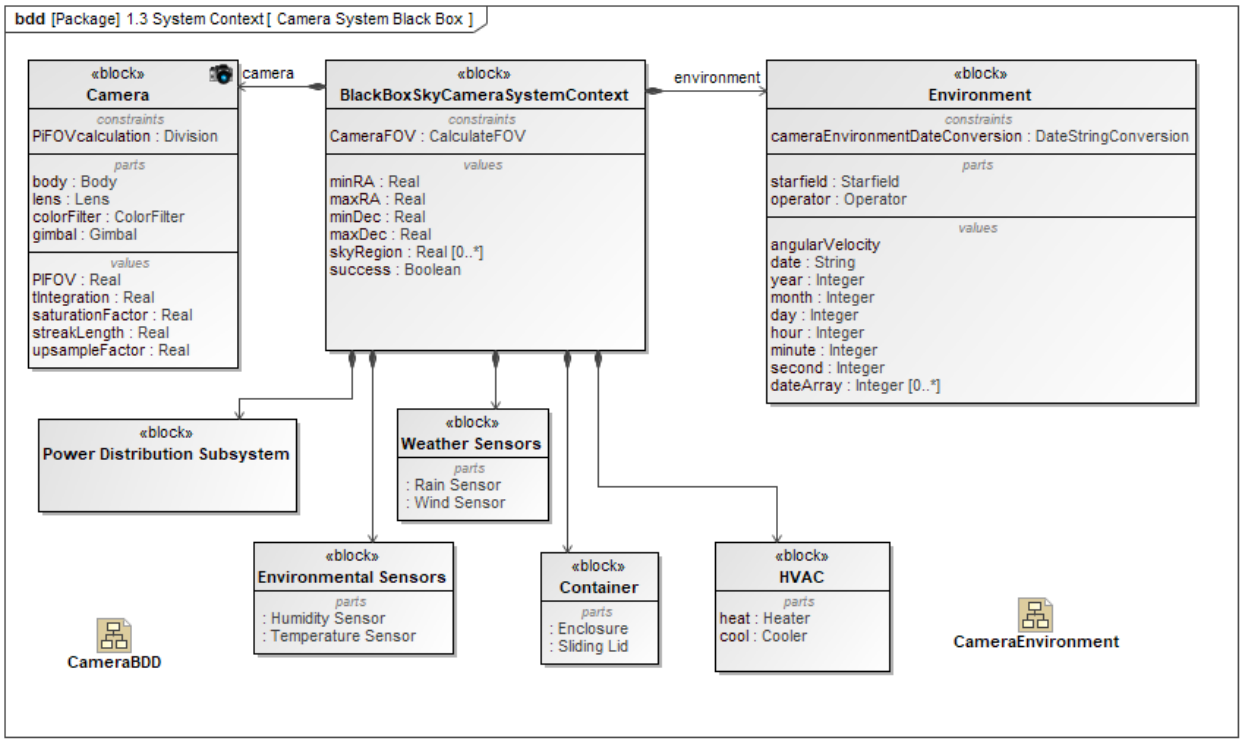

Figure 2: This diagram shows the base system components and model structure. Note, there are several elements not currently used in the analysis case presented in this paper.

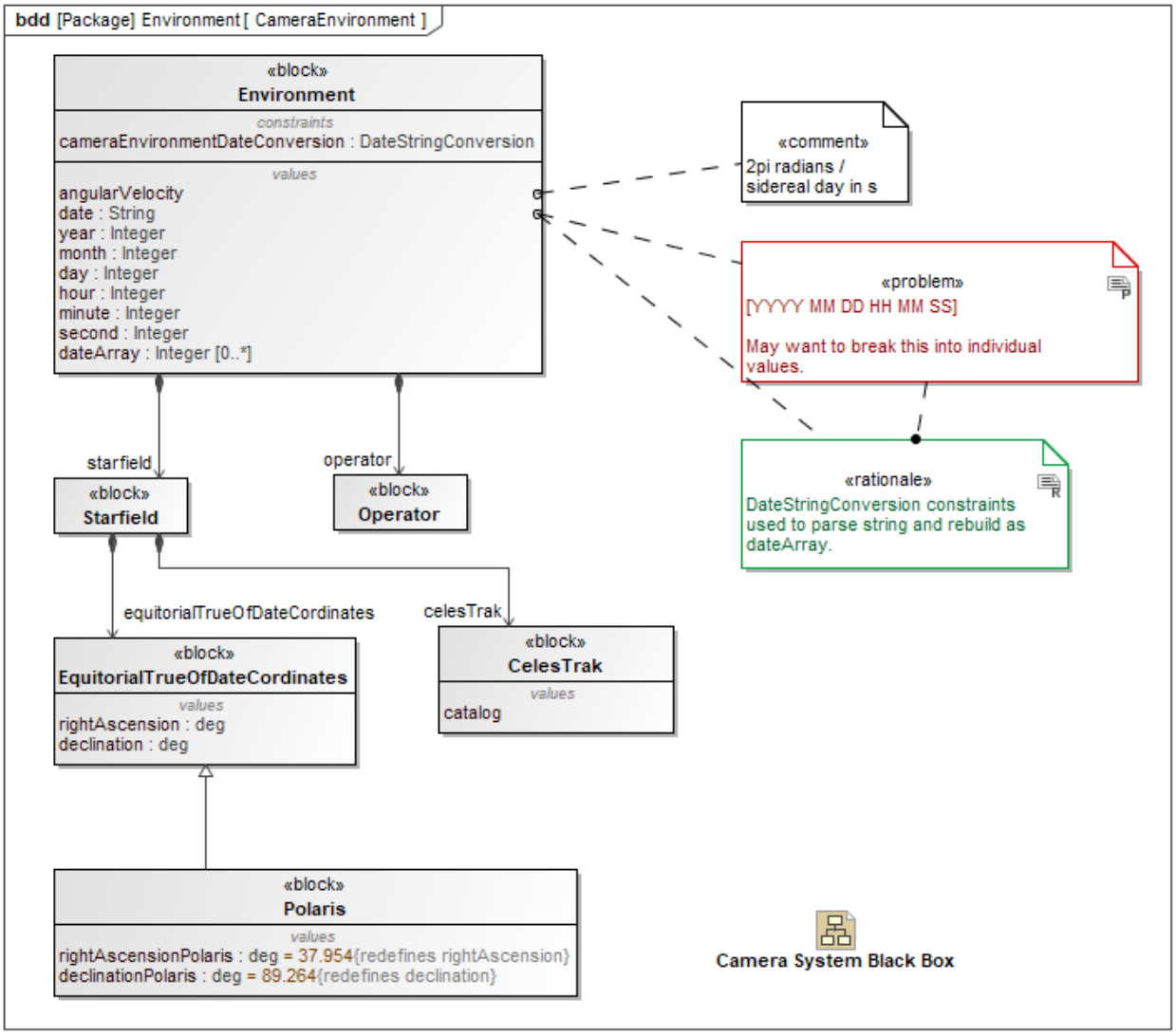

Figure 3: The environmental elements used in the model.

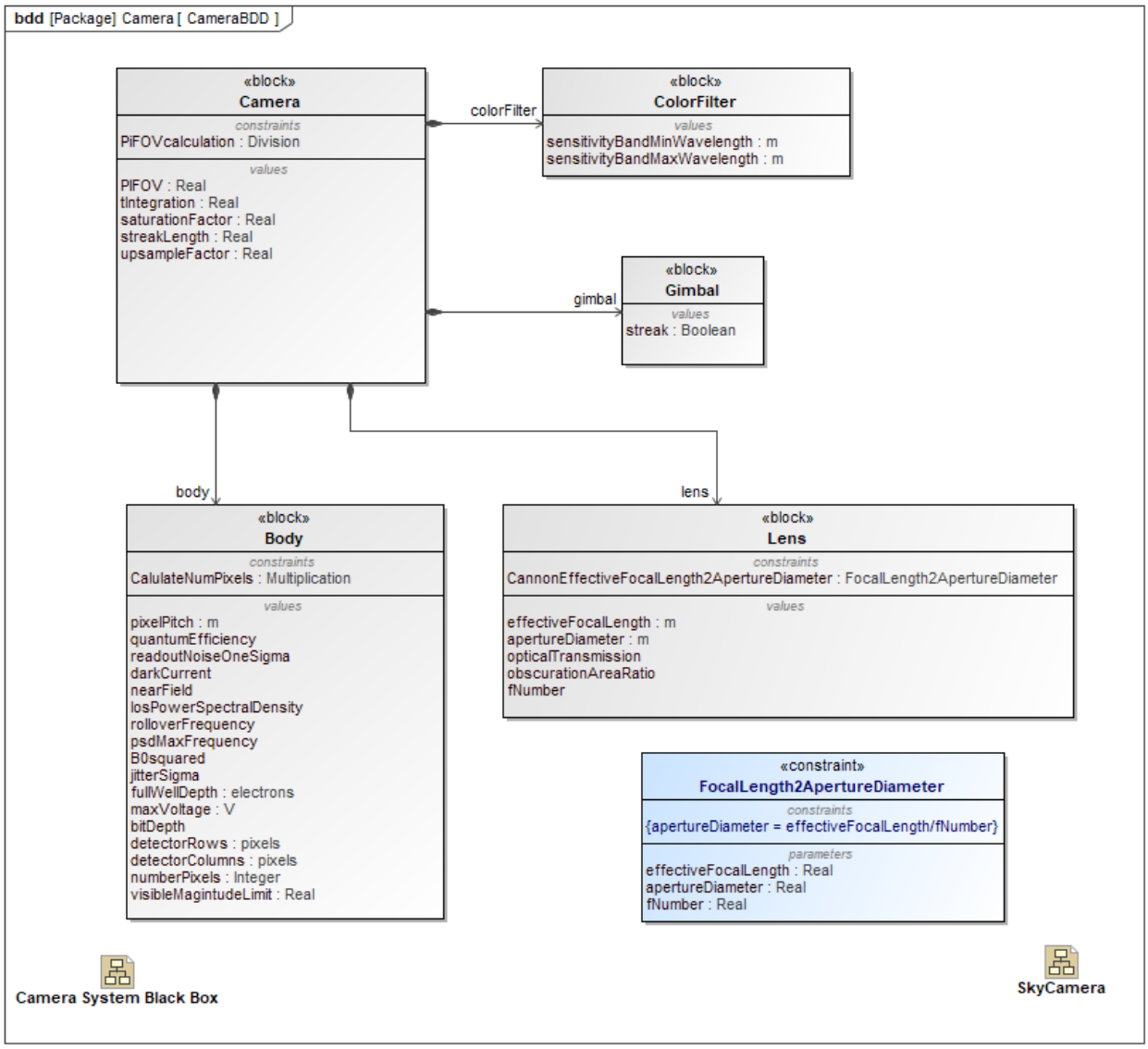

Figure 4: A detailed view of the generalized camera attributes.

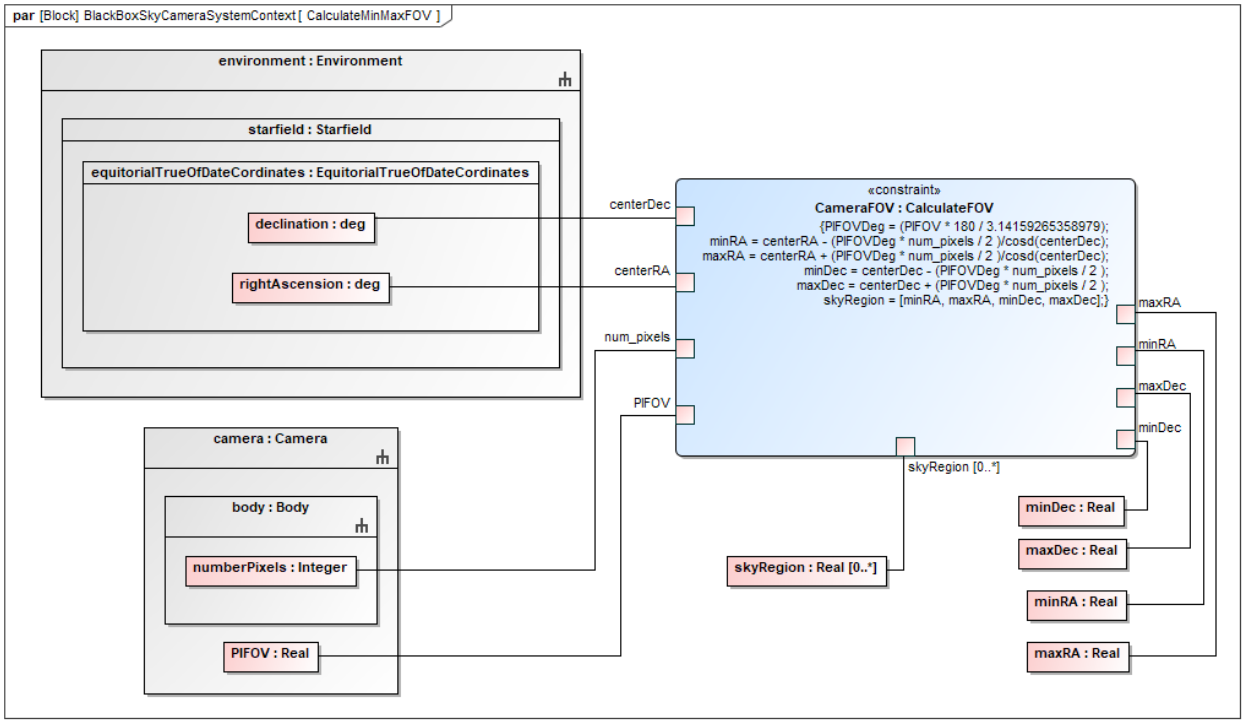

Figure 5: Example of MATLAB code as a parametric constraint for the field of view calculation.

Behavioral Model

We modeled system behaviors using a combination of activity diagrams and state machines. Activity diagrams allowed us to model operational processes, while state machines allowed us to model functional configurations. This enabled us to dynamically switch model output and behavior based on finite parameters within our trade space. The state machine shown in Figure 6 illustrates this for the relatively simple case of optical filter selection.

As the optical filter state machine executed in line with the model, we selected a set of filters depending on the particular test case or functional scenario being analyzed. This also allowed us to change filter definitions as needed since these state machines themselves could be parameterized and driven by inputs from other tools. This simplified configuration management since we did not need to maintain long lists of named filter response files.

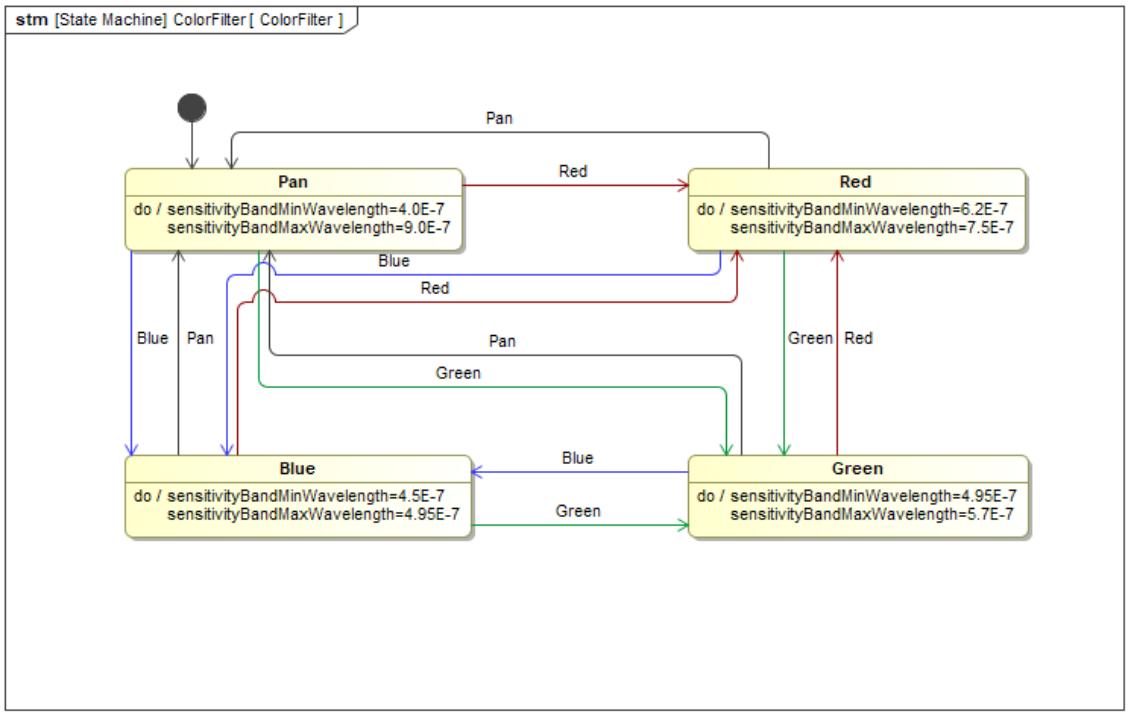

Figure 6: A state machine that shows the color filter classifying behavior. We used a similar state machine to select the star catalog and to model the camera tracking gimbal states.

Figures 7 and 8 illustrate some of the high-level behaviors we engineered to further automate model execution and simplify the orchestration of behaviors between different model elements. Black box and white box behaviors allowed us to use classifier behavior to pre-define some of that orchestration without necessarily requiring that we specify each parameter within the model on every run—we could execute it from a set of pre-defined states.

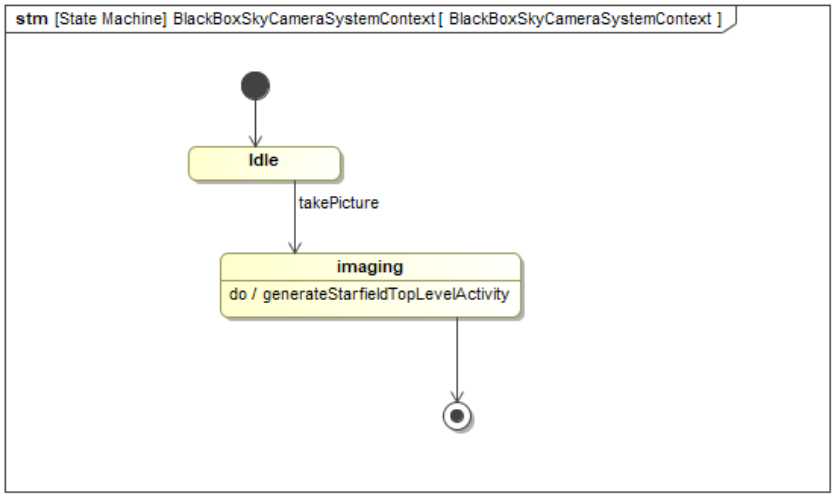

Figure 7: This is the classifying behavior of the BlackBoxCameraSystemContext block. The system model starts in an idle state to allow the user to set camera parameters before signaling the camera to 'take a picture.'

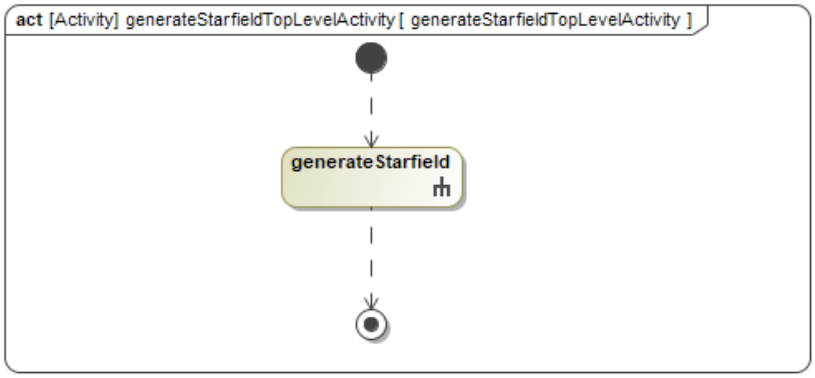

Figure 8: This is the top-level activity created by assigning an activity to the "do/" portion of the state machine BlackBoxCameraSystemContext.

Model Execution

Using the state-based approach allowed us to model in a way that represents variable system settings, enabling quick setup for analysis of multiple use cases. When executed, the *generateStarfield* activity calls the MATLAB image processing chain using the precalculated parameters as inputs to produce a simulated starfield. Integrating the camera properties and environmental parameters needed by the image processing chain in constraint blocks allowed us to execute the model with a specific sensor and environmental context without needing to specify everything for each run and to keep records of the configurations run along with the resulting imagery.

The Activity shown in Figure 9 executes the image generation code, providing the connective tissue between model parameters and COTS simulation tools, delivering the output we need to make trade space decisions. It reads all the necessary runtime-calculated value properties and feeds them to the MATLAB image processing chain to generate the synthetic image.

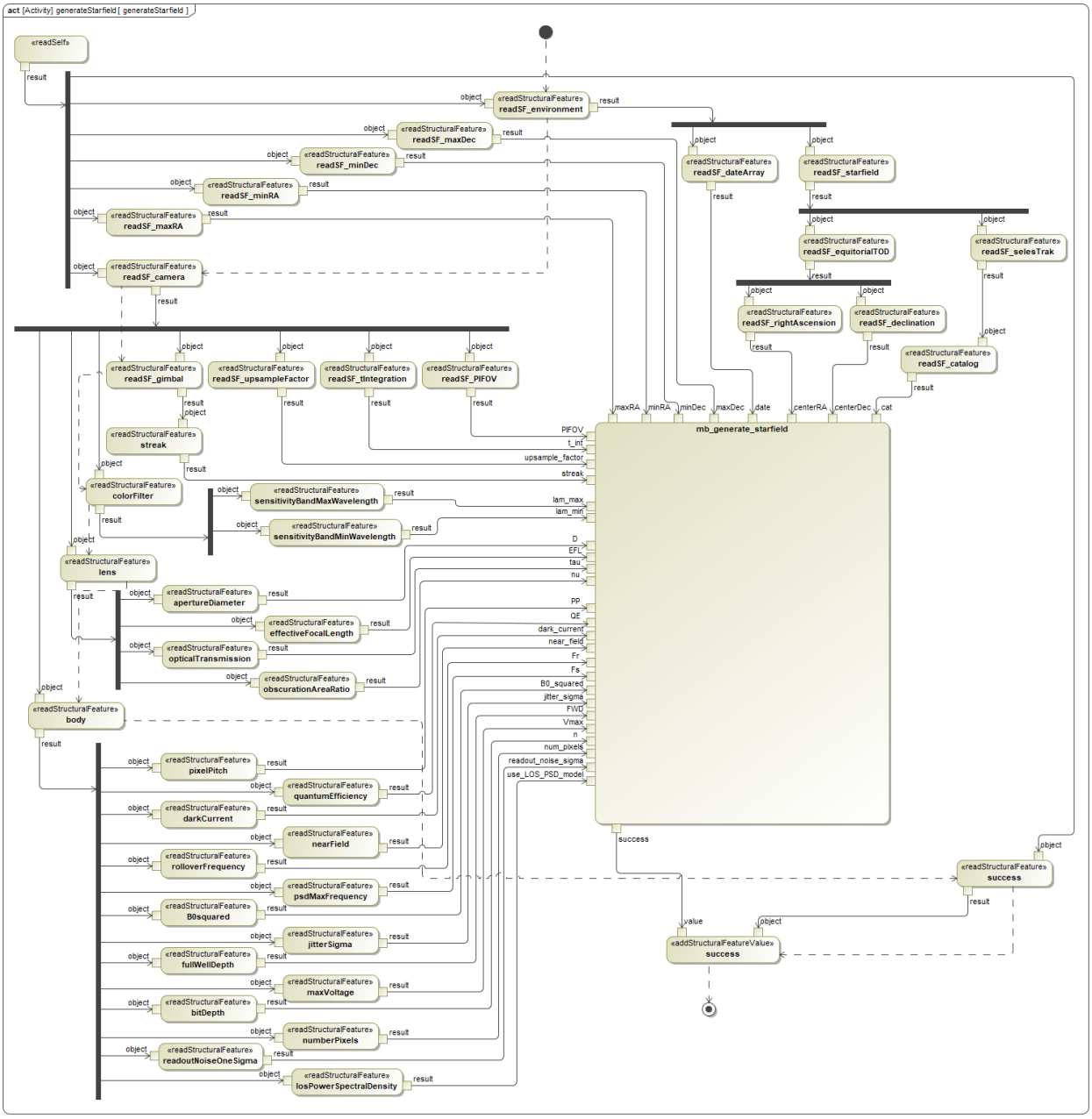

Figure 9: This activity diagram illustrates the complexity needed in an executable model. The inputs are pulled from calculated model constraints, as well as user selections for setting, filter, gimbal, and star catalog.

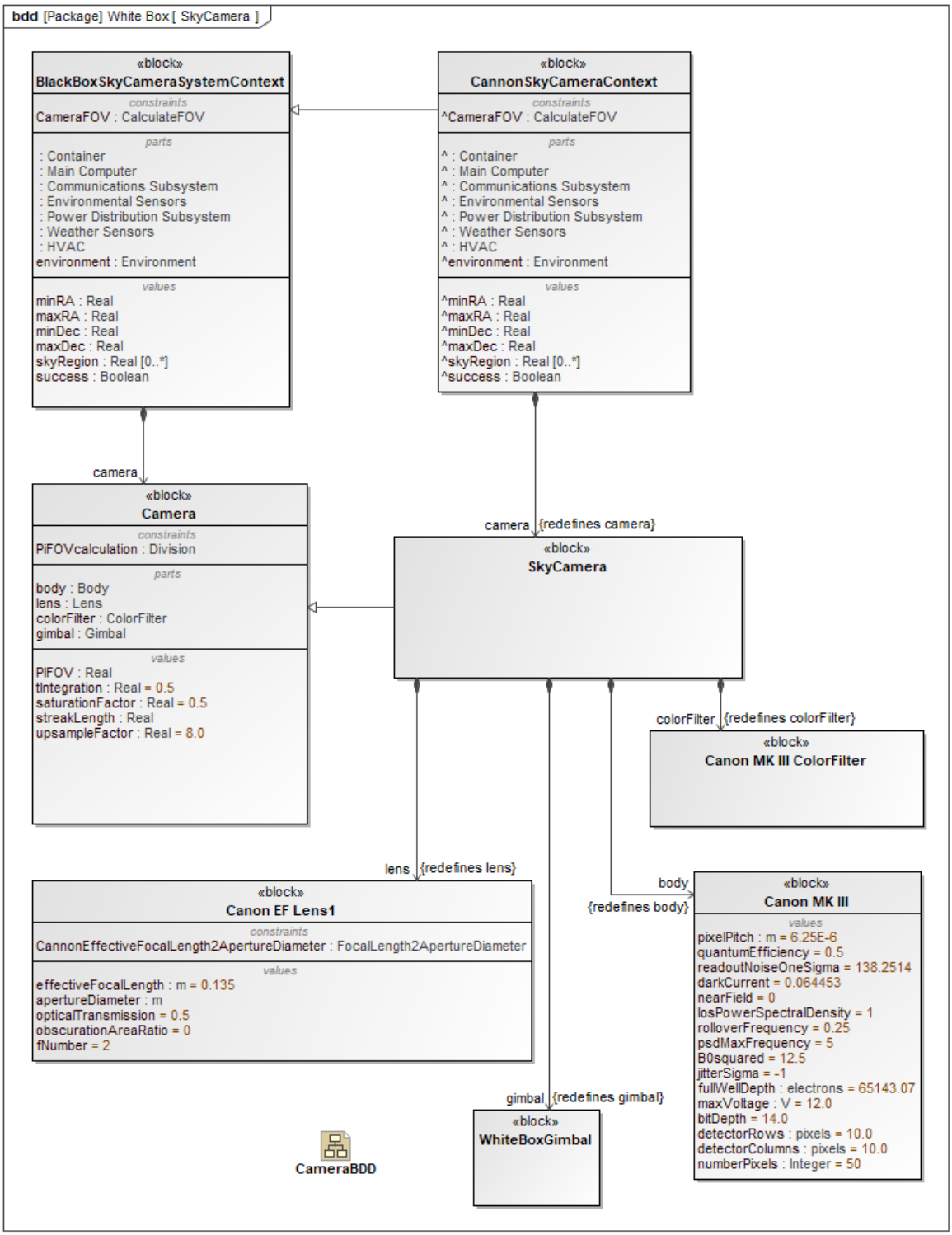

Figure 10: This diagram shows the specification of a camera design from the 'abstract' camera model. The specific camera components can then be defined as shown to allow multiple configurations to be specified from the base design. This approach enables the engineer to assess individual performance or optimize parameters during a Multi-Disciplinary Analysis and Optimization study.

Model Output

Ultimately, the purpose of this model was to determine if it could execute the code needed to produce imagery that would meet our needs. Once completed, the simulated imagery allowed us to begin exploring the trade space of COTS hardware by determining the sensitivity of the output to variations in the performance specifications of different combinations of camera bodies and lenses. Figure 11 below is an example of the simulated imagery produced by the SkyCamera model.

Figure 11. This is a sample output generated by the executable digital engineering model, based on design choices selected within the model at runtime.

CONCLUSION

Our motivation for developing this project using the OOSEM and leveraging MBSE was very simply to test the feasibility of a technically intricate project with a limited prototyping budget using a small, distributed team. Conventional engineering methodologies and architectural modalities have proven insufficient for large scale technical projects for some time. The need to enforce good engineering practice and balance stakeholder needs in an environment of distributed teams, diverging standards increasing technical complexities have created significant inefficiencies. Projects such as the Thirty Meter Telescope[3] have addressed these technical challenges and the needs of diverse stakeholders using MBSE tools. The attempts to develop tools, technologies and methods to overcome these problems have resulted in the methodologies and software tools that our team leveraged for this project.

Using this approach, we successfully demonstrated the feasibility of our proposed system and generated synthetic test data. This enabled us to validate the concept prior to moving to a prototype phase. Most significantly, this allowed us to rapidly iterate ideas, vet off-the-shelf parts and test concepts before committing to a hardware

design or bill of materials. Further, it allowed us to distribute work among a team which was not co-located and to communicate with senior leadership, providing concrete answers to ensure we had represented their equities.

The next prototype phase will be used to test operational and assembly processes, image processing techniques, and to validate untestable assumptions in the engineering model—namely the effects of camera array alignment and lens optical aberrations. This prototype will allow us to focus on accommodation of requirements and concepts of operation rather that trouble-shooting camera hardware or integration with our post-processing toolset.

[1] Friedenthal S, Moore A., and Steiner R. 2015. A Practical Guide to SysML 3rd Ed. Morgan Kaufmann OMG Press.

[2] Karban, R. , Jankevičius, N. and Elaasar, M. (2016), ESEM: Automated Systems Analysis using Executable SysML Modeling Patterns. INCOSE International Symposium, 26: 1-24.

[3] Nakawatase, J. 2017. Telescope Modeling Challenge Overview. INCOSE MBSE Leadership Meeting.