2024 NDIA MICHIGAN CHAPTER
GROUND VEHICLE SYSTEMS ENGINEERING
AND TECHNOLOGY SYMPOSIUM
<SESSION NAME> TECHNICAL SESSION
AUGUST 13-15, 2024 - NOVI, MICHIGAN

MBSE STRATEGIES: AVOIDING CYCLIC PROJECT USAGE DEPENDENCIES BY USING BLACK BOX AND WHITE BOX MODELS

Jason Kolligs, PhD[1], Stuart Masterson[1], Eric Thome[2], Christian Knutson[2]

[1]Strategic Technology Consulting, Aberdeen, MD
[2]General Dynamics, Bothell, WA

ABSTRACT

Every digital engineering framework and modeling approach will include benefits and concerns. It is important to customize the response, within reason and based on the available resources, to the needs of the project and contract. For this case, the consideration of a large, singular model was overturned for a distributed model. The potential for a cyclic usage, which can be catastrophic in both performance issues and data loss, was mitigated by an innovative approach that allowed for two (2) systems models – one (1) Black Box and one (1) White Box – using a novel model federation strategy. The concerns of having two (2) system models were mitigated via acceptance and understanding that each system model would play its part appropriately based on model function, system development, and contract deliverables.

Citation: J. Kolligs, S. Masterson, E. Thome, C. Knutson, "MBSE Strategies: Avoiding Cyclic Project Usage Dependencies using Blackbox and Whitebox Models," In *Proceedings of the Ground Vehicle Systems Engineering and Technology Symposium* (GVSETS), NDIA, Novi, MI, Aug. 13-15, 2024.

1. INTRODUCTION

Systems engineering is often charged with changing the perspective on a project to facilitate solution generation. Large, complex systems that involve multiple partners and external actors require flexible digital engineering approaches that allow for modeling to include input from those partners while supporting data quality and configuration management. Using project usages in a model is a viable approach, however it can pose challenges. In the case described herein, the planned contents of the models were well understood while the project usage architecture posed a challenge that was rife with potential for cyclic usages or dependencies. This paper establishes background information (Section 2) to prepare the reader to better understand the challenge (Section 3) and then describes the solution and rationale (Section 4,5, & 6).

2. BACKGROUND

The concepts of Project Usages, Cyclic Usages (or Cyclic Dependencies), and Black

Box/White Box Modeling are described here to provide context in defining the challenge and solution.

2.1. Project Usages

Project usages are used within Cameo and other modeling tools to import a model into another project [1]. Project usage models cannot be modified in the current project, but their elements are available for usage in the project, i.e. project usages are read-only. For example, Model A (a System Model) could import Model B (a Requirement Set) as a project usage. Model A would be able to see and relate the requirement elements in Model B without compromising the configuration management of Model B. The practice of project usages is common and is often used to include custom organizational profiles, reference materials such as regulations and standards, and other template or style guide type projects.

2.2. Cyclic Usages

Cyclic usages, or cyclic dependencies, are project usages within model-based systems engineering (MBSE) that reference each other in a circular manner, i.e. Project A has a project usage of Project B which has a project usage of Project A [2]. Cyclic dependencies create an infinite loop or references that will eventually lead to resource issues that can be realized as model crashes or even model corruption at the extreme cases. Figures 1 and 2 are examples of simple cyclic usages.

Cyclic usages can be accidently created and difficult to identify in more complex model federations. For example, in Figure 1, Cameo system recognizes the cyclic usage and applies a warning icon while in Figure 2 the Cameo tool is unable to identify the cyclic usage despite its obvious presentation in the diagram.

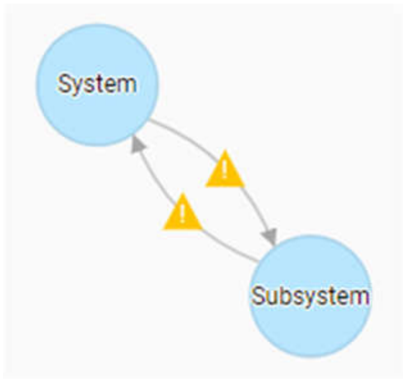

Figure 1: Example of Cyclic Usage with Two (2) Components (Visualized using Cameo Resource Map)

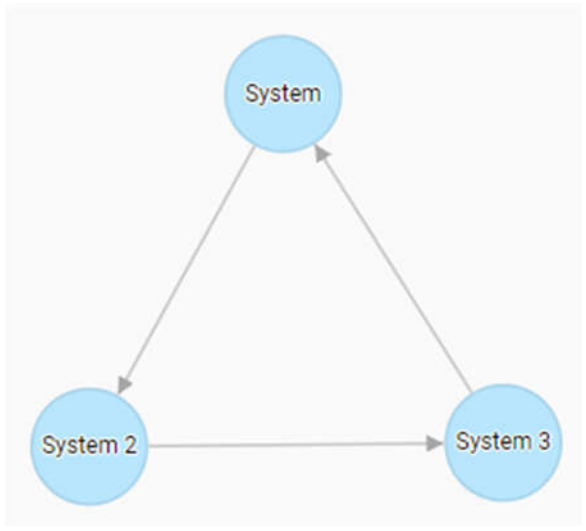

Figure 2: Example of Cyclic Usage with Three (3) Components (Visualized using Cameo Resource Map)

2.3. Black Box and White Box Modeling

Black box modeling, also referred to as a black box view, focuses on the system's external behaviors, such as the interfaces and data flows, and does not reveal the internal workings. More formally, a black box "facilitates discussing a system at an abstract level with a focus on input and output rather than the details of how inputs are transformed into outputs" [3]. Other major advantages of black box modeling include the establishment of system boundaries and the intentional obfuscation of complexity and/or Intellectual Property (IP).

White box models or views are the reciprocal of black box models because white box shows the internal workings of a system while acknowledging the same external behaviors as the black box. Figure 3 is a simple example depicting a black box and white box model.

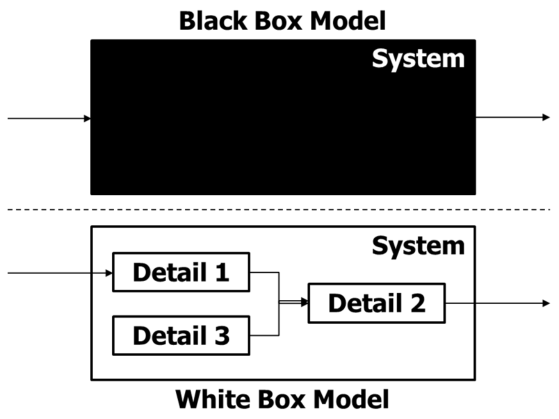

Figure 3: Examples of a Black Box Model and White Box Model

3. THE CHALLENGE

The authors were presented with a challenge in which the stakeholders of a new engineering effort wanted to develop a model utilizing the feedback from external partners such as subcontractors. The concept was to drive lower-level design and development based on high-level needs and expectations. Additionally, those lower-level designs would influence the higher-level design and therefore will have to be consolidated into the higher-level model to facilitate the final design. In short, a preliminary model that established the system boundary, desired inputs, and expected outputs was needed by the external partners so that they could develop their subsystems which would then be integrated into the system model. The proceeding sections describe the benefits of this approach and why the cyclic usage challenge had to be addressed.

3.1. One Big Model vs. Distributed Projects

The first goal was to determine if there would be a single, larger model with everyone having access or if the project would be distributed to the external actors for their contributions. The one big model allows for all the data to be centrally located, while the distributed project would provide some increased performance at scale. Each of these approaches carry obvious logistical challenges. The large model needs to be accessed by different companies using an agreed-to and accessible tool. The distributed project also requires some of the same agreed-to, accessible tool but also introduces data availability issues, i.e. data is only available to the original system developer when iterations are sent.

From the MBSE perspective, a single, large model with many people working in it will invariably lead to more iterations or revisions. More revisions translate to more work for the modeling tool to maintain the changes, or deltas, between different revisions. A higher revision count will eventually reach performance issues as those revisions are loaded or modified. A higher revision count also has a drastic negative impact on more intensive model operations such as merges and publishing to a cloud. The advantage would be a fully integrated model with full data accessibility.

Distributed projects, or an approach in which multiple companies work on their portion of the overall project but only submit major revisions or an end product, mitigated the potential performance issues of the larger singular and, for this case, the team did not require or really benefit from full data accessibility. The challenge was to integrate the distributed models into a cohesive final system model. The initial thought was to include each distributed subsystem model as a project usage into the system model, but the team would later learn of cyclic usages and their issues.

3.2. Configuration Management

To properly execute the distributed project approach, the team needed to provide a configuration managed version of their system model to external actors to facilitate subsystem development. It was straightforward to provide an iteration of the original system model to each subsystem developer, but the CM assurances were a concern. The solution was to mandate that the original system model be imported into the subsystem model as a project usage. Recall that a project usage is read only which provides configuration management.

3.3. Cyclic Usages

The combination of solutions to mitigate the performance issues of a large model and provide adequate configuration management gave birth to a potential for cyclic usages. If the subsystem models are using the original system model as a project usage (which is preferred), then the original system model cannot import the subsystem models as project usages (which is preferred) as it would create the cyclic usage depicted in Figure 1. A project usage architecture solution was needed to mitigate this cyclic usage challenge.

Many modeling tools offer another option like project usages that involves multiple model branches that are then merged back into the trunk. When considering branches and merges there are two (2) general factors to consider, and unfortunately, they are conflicting. First, keep revisions of the trunk low, or in other words, use branches as long as possible. This consideration is good for long term model health and future operations such as migrations, exports, and managing history. This is like the issues of a singular, large model. The second factor is to keeping branches small enough to allow for merging. Merging, as currently available in the accessible modeling tools, is a memory intensive process. If the branches get large and the differences from the trunk grow, then the memory requirements of a merge increase. The merge capabilities of the modeling tool being used by the team were difficult and inconsistent, which led to that option being removed from consideration.

4. FINDING A SOLUTION

There are two (2) potential solutions to the cyclic usage issue. The first is to ignore it which translates to serial process in which the original system model that is distributed to the external actors is never updated as the system design process iterates. This solution is not viable in a multi-partner systems engineering effort. The second solution is to not have the usages be cyclical – but what does that architecture look like and is it rational?

4.1. Another System Model

The reason that the project usages in this case might be perceived as potentially cyclic is based on belief that there can only be a single system model. This team considered the potential of a second system model that would integrate the details from external partners. While it is true that the original system model disseminated to external actors is describing the same system as the final integrated system model, they serve different purposes. The original model is the expansion of customer needs into requirements and logical design concepts. This represents divergent systems thinking as the problem space is being defined and the solution space is being opened. The integrated system model is the convergence of subsystem design details that home in on a system solution.

4.2. Black Box and White Box Principles

The different content expressed in the original and integrated system models align with black box and white box modeling as described earlier in this paper. The original model is concerned with how the system will interact with its environment and represents a black box perspective of the system. These concepts facilitate system architecture design, requirements development, analyses based on the inputs and outputs, and external interfaces maturation.

The integrated system model includes the details as provided by the external actors which is in line with a white box model. The inherited subsystem models provide further detail to the system to create the white box perspective. As the integrated system model matures it provides the necessary data to perform analysis on subsystem interactions, lower-level interfaces, requirements traceability across the subsystems, verification method traceability, and potentially subsystem level test data.

4.3. Path Forward

This revelation led to an extensible project usage architecture that is depicted in Figure 3. Recall that an arrow means that the source project is using the contents of the destination project. For example, from Figure 3, System WB (White Box) is using Subsystem A which is using System BB (Black Box).

The architecture includes a project usage from System WB to System BB which mitigates two (2) minor issues:

1. Designating which subsystem project usage supports the System WB project need for data in the System BB model.
2. Nested project usages, which involves updating each echelon of project usages.

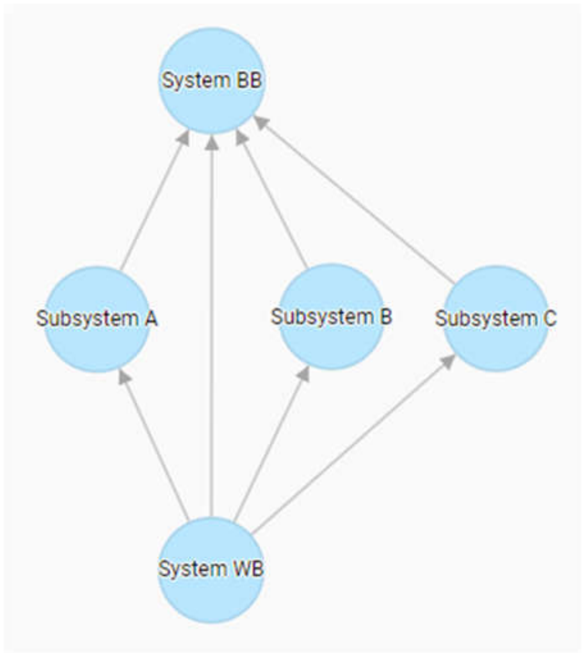

Figure 4: Project Usage Architecture Solution

5. DESIRED BENEFITS

Every and any approach comes with benefits and issues that must be accepted by the adopting engineering effort and their circumstances. The team decided that the following benefits were highly desired:

- Distributed – mitigates the performance issues of a potentially massive model.
- Configuration Management – inherently protects the System BB model from external actors and Subsystem models from System WB usages.
- Extensible – supports many subsystems and external actors.
- Reusable – can be templated and used throughout the organization with teams that have similar situations.
- IP Obfuscation – the system integrator can maintain control of IP while disseminating a meaningful model to support partner needs.
- Product Line Engineering (PLE) – the customer has plans for PLE that are well supported by the WB model details

tracing to the requirements in the BB model.

6. ACCEPTED ISSUES

The primary concerns or issues of this approach come directly from the issues of a distributed model. As these were accepted at the onset of the approach, each was mitigated according to the team's needs and interests.

6.1. Non-Real-Time Data

Since external data is imported via project usages in lieu of a singular model, data is not available in near real time. This means that changes from external actors require additional steps to update the imports of the System WB model. While this could be considered a large concern to some engineering efforts, this team accepted that this activity would be rare and easily dealt with during those exceptions.

6.2. Multiple System Models

The second concern was having two (2) system models: a black box and a white box. As discussed above, this was mitigated early by maintaining that each continue to serve their primary purpose. The issue becomes accessing the correct model for the purpose of the modeler or viewer. If they are trying to access the system to modify requirements or another black box activity, they must access the System BB model. Conversely, if the modeler or reviewer are seeking white box model features, they must log into the System WB model. The challenge is if there are activities that requires an update to both models – in which the case is that the BB model must be updated first, committed to the server, then the WB model needs to update the BB project usage before proceeding onto the task at hand.

The team decided to accept this risk. Their intent had been that once the System BB model was released, changes to that model would be undesirable and limited to critically important iterations. The concept was that changes to the System BB model would also have to be propagated to the external actors who may have additional impacts. Likewise, the System WB model is predominantly comprised of data shared from other models, and therefore does not update often. It is mostly being leveraged for the assessment of the models, their dependencies, some analysis, and their exchanges of information.

The two (2) system model approach was also evaluated regarding contract deliverables. The project needed to ensure that individual contract deliverables could reside in either of the system models, not split amongst both. This was not perceived as challenging since the models, as explained above, serve different purposes and those purposes align with the contract deliverables.

6.3. Issue Summary

In summary, the issues were: Non-Real-Time Data and Two (2) System Models. Both issues were accepted by the team in this case and for this business product. The consensus was that the benefits outweigh the potential issues.

7. Summary and Conclusion

Every digital engineering framework and modeling approach will include benefits and concerns. It is important to customize the response, within reason and based on the available resources, to the needs of the project and contract. For this case, the consideration of a large, singular model was overturned for a distributed model. The potential for a cyclic usage, which can be catastrophic in both performance issues and data loss, was mitigated by an innovative approach that allowed for two (2) systems models – one (1) Black Box and one (1) White Box. The concerns of having two (2) system models were mitigated via acceptance and understanding that each system model would play its part appropriately based on model function, system development, and contract deliverables.

8. REFERENCES

[1] "Start using a project in your project - MagicDraw 19.0 LTR - No Magic Documentation," CATIA No Magic, [Online]. Available: https://docs.nomagic.com/display/MD190/Start+using+a+project+in+your+project. [Accessed 2024].

[2] "Identifying Package Dependencies - MagicDraw 19.0 LTR - No Magic Documentation," CATIA No Magic, [Online]. Available: https://docs.nomagic.com/display/MD190/Identifying+Package+Dependencies. [Accessed 2024].

[3] M. Green, "The Application of Black Box Theory to System Development," in *System Engineering in DC Proceedings (SEDC)*, Washington, DC, 2014.

.