

# Software Tools for Efficient Model-Based Design of Energy Optimized Aircraft

Tim C. O'Connell, Kevin McCarthy and Rodney Yeu  
PC Krause and Associates

Paul Pigg and Timothy Bowman  
Boeing Research & Technology

Peter Lamm  
US Air Force

## ABSTRACT

The diverse and complex requirements of next-generation energy optimized aircraft (EOA) demand detailed transient and dynamic model-based design (MBD) to ensure the proper operation of numerous interconnected and interacting subsystems. In support of the U.S. Air Force's Integrated Vehicle Energy Technology (INVENT) program, several software tools have been developed and are in use that aid in the efficient MBD of next-generation EOA. Among these are subsystem model libraries, automated subsystem model verification test scripts, a distributed co-simulation application, and tools for system configuration, EOA mission building, data logging, plotting, post-processing, and visualization, and energy flow analysis.

Herein, each of these tools is described. A detailed discussion of each tool's functionality and its benefits with respect to the goal of achieving successful integrated system simulations in support of MBD of EOA is given. Finally, some identified pitfalls in the MBD of EOA are discussed, both with respect to the described tools, and in general. Suggestions are given for mitigating the effects of, or avoiding altogether, these pitfalls.

**CITATION:** O'Connell, T., McCarthy, K., Yeu, R., Pigg, P. et al., "Software Tools for Efficient Model-Based Design of Energy Optimized Aircraft," *SAE Int. J. Aerosp.* 5(2):2012, doi:10.4271/2012-01-2176.

## INTRODUCTION

### Background

MBD of aircraft has gained momentum in the past three decades as the complexity of aircraft subsystems has increased [1, 2] and the costs associated with discovering design problems during the hardware test phase have escalated. While many of the recent programs in this arena have dealt with More Electric Aircraft (MEA) and the complexities involved in the aircraft electric power system (EPS) [3,4,5,6], interactions between thermal, fuel, mechanical, and additional systems also play a crucial role in the overall energy use of modern aircraft. Thus, to achieve an EOA design all of these energy flows and interactions must be understood.

Reducing energy use in modern aircraft has become a critical goal in both the commercial and military sectors. As the cost of fuel has increased, so has the benefit of reducing

fuel use. In commercial airliners profits are directly and heavily impacted by fuel costs [7, 8], and in the military mission success is often dependent on the availability of fuel, especially in remote and dangerous battlefields.

Depending on the aircraft requirements, the term "Energy Optimized Aircraft" can have many meanings and interpretations [1]. In the simplest terms an EOA is one that uses the least amount of energy (typically, fuel) to accomplish a given mission. However, due to the complexities involved with numerous interacting subsystems, this is not simply a matter of minimizing the energy use of each subsystem individually at certain design points, as has traditionally been done. For example, just looking at the engine itself, it may be more efficient to extract energy via bleed air than by extracting power mechanically from the engine shaft. However, when also taking into account the engine's interaction with other systems (e.g. the environmental control system), it is sometimes more efficient

for the aircraft overall to mechanically extract the energy [7, 9]. Traditional design methods might not find this non-intuitive result. Further, simply minimizing fuel usage may not achieve other system requirements such as reliability targets, system growth potential, and adaptability. For a given aircraft design, then, achieving an EOA requires a detailed understanding of the numerous interactions between all the subsystems and an accurate accounting of energy use. Within the constraints of the design and the goals of the program, the energy use can then be minimized to achieve an EOA design.

The goal, then, of MBD of EOA is to account for and manage all energy flows throughout the aircraft over the entire mission, in order to achieve the minimum energy use. An integrated software model (throughout this paper, the term *model* refers to software models, unless otherwise noted) consisting of dynamic models of each aircraft subsystem is required for this, and this paper addresses the challenges involved with developing, exercising, and using this type of model.

## Challenges

In typical MBD of aircraft, numerous suppliers develop models of the subsystem hardware they will supply, and they provide these models to the system integrator (SI) for creation of the integrated vehicle model (IVM). In this workflow, the SI faces numerous challenges [1]. First and foremost, the SI must ensure that intellectual property (IP) contained in the supplier models is protected appropriately. Particularly, care must be taken when passing results and other model-related information between suppliers so that inadvertent IP disclosure does not occur. Second, suppliers each have their own preferred (and possibly proprietary) platforms for modeling their hardware, including software packages, model libraries, and computer programming languages. These often vary across suppliers, meaning that the SI must create an IVM that allows many disparate models to communicate with each other and simulate simultaneously. General modeling requirements can be specified for a particular EOA program, as has been done in the Modeling Requirements and Implementation Plan (MRIP) [10] for the INVENT program, to help facilitate this, but challenges will still exist. For example, in some cases a subsystem model may be so sensitive that a supplier will not want it to leave the supplier's computer. In this case, the SI must connect models that exist on separate computers, possibly in different geographical locations, while still maintaining the IVM stability, accuracy, and utility.

Configuring a large number of models presents challenges as well. Specifically, setting the initial conditions, boundary conditions, and model parameters for a large number of models, each of which may require inputs in different formats, can be a time-consuming and error-prone process. Thus, the SI must establish a repeatable, efficient process for configuring the IVM so that numerous runs using different configurations can be launched and tracked.

Once IP, connectivity, and configuration challenges have been addressed, the SI must ensure that the IVM has an acceptable run-time so that the necessary studies can be completed in a reasonable time using the available computing resources. For example, if a four-hour EOA mission is to be simulated but the IVM runs ten times slower than real-time, it will take 40 hours to simulate the mission. If numerous missions are to be simulated, this run-time will likely be prohibitive to a timely completion of the study. Thus, the SI must work with subsystem suppliers, and use appropriate system integration strategies [6] to ensure an acceptable IVM run-time. Tightly coupled with this challenge is the challenge of ensuring simulation numerical stability. With numerous subsystem models, each possibly developed in a unique simulation environment and using a unique numerical solver, ensuring stability of the IVM when the models are all connected and simulated together can be a challenge. For example, a simulation can often be sped up by increasing the length of the numerical solver time steps; however, if the time steps are too large the accuracy of the solution, and eventually the numerical stability, will be negatively impacted. Thus, the IVM must meet the challenge of ensuring numerical stability while maintaining a reasonable simulation run-time. These challenges can be mitigated by use of and enforcement of a modeling requirements document such as in [10], but even the use of such a document cannot guarantee that the IVM will meet all the requirements without further work.

Once the IVM is stable and running at an acceptable speed, the SI must effectively manage the vast amounts of data that are generated. The challenges involved in this include organizing, collecting, and storing the simulation data during run-time, visualizing the raw output, both during and after runs, and then post-processing and understanding the results. Because so many data signals are generated and so many interactions are captured in these signals, it is important to streamline the data management process so that the most important signals can be quickly accessed, viewed, processed, and interpreted. This will ensure the EOA design can progress along the necessary timeline.

## Tools

To address the aforementioned challenges, a suite of software tools has been found to be extremely valuable. The tools are listed here briefly, and discussed in detail in the following sections.

Subsystem *model libraries* are used to organize models, for version control, and to prevent the inadvertent disclosure of IP between competing subsystem model suppliers. Automated subsystem *model verification test scripts* exercise individual models and compare their performance against program specifications, allowing subsystem models to be checked before being inserted into the simulation libraries. A *distributed co-simulation application* is utilized both to run individual subsystems in their native simulation environment and to parallelize the simulation, either across multiple



machines in a local or geographically separated cluster, or by utilizing multiple processing cores on a single machine. These capabilities enable diverse subsystem models to be developed independently by suppliers using their preferred modeling software applications, and accommodate the use of different solvers for each subsystem model. A *system configuration tool* is invaluable for initializing the model and configuring the subsystem models, as well as logging configuration settings for archival purposes. An EOA *mission building tool* allows interactive construction of EOA missions, generating the proper time-based mission database that is accessed at run-time by the integrated system model. A *data logging tool* saves the data from the EOA integrated system simulation in a user-specified location at a user-specified sample rate. A *signal viewer tool* allows the loading of saved simulation data and fast, automatic plotting of data signals for quickly scanning large quantities of data, both during and after a simulation. A data *post-processing and visualization script* automatically processes the simulation data, creating plots and summary charts to be saved with the data. Finally, an *energy flow analysis script* calculates and displays the average and peak energy flows throughout the aircraft during each mission segment (takeoff, cruise, etc.) in a convenient spreadsheet format.

## INTEGRATED SYSTEM MODEL OVERVIEW

In the following paragraphs, a general overview is given of the framework for simulating large, integrated EOA systems and processing the data they produce. The tools necessary to implement this framework are then discussed in more detail in the following sections. To better follow the discussion, refer to the flowchart of the IVM workflow shown in [Figure 1](#).

Subsystem suppliers deliver software models (shown as grey blocks labeled A-D in [Figure 1](#)) of their subsystems to the SI, who then verifies the models against a pre-defined suite of requirements before checking the models into model libraries. All supplier models are stored in libraries, with each supplier assigned its own library. Models are given a unique version number, and are named following a standardized format (e.g. "Supplier1.EPS.v2") so that they may be easily identified by the simulation engine. Further, each model is given a solver settings file, which is named similarly to the model itself (e.g. "Supplier1\_EPS\_v2), which is used to initialize the model at run-time.

There will likely be several missions that the EOA design will fly in simulations. These missions are stored in a mission definition file as time-based climb and descent rates, cruising altitudes and speeds, and directional commands for all portions of the missions. This mission definition file can be accessed by all the subsystem models at run-time in order to set necessary boundary conditions throughout the simulation.

A co-simulation tool such as is described in [\[6, 11,12,13\]](#) is used for integrating federated subsystem models into an

integrated, distributed simulation. A system configuration tool is used to configure the study that will be run. Configuration includes specifying which subsystem models to use, according to the standardized naming format, and setting system initial conditions, boundary conditions, parameter values, inter-model data communication rates, data storage rates, and the location of output data files. The cosimulation tool uses the configuration file generated by the system configuration tool to select the proper subsystem models, launch them in their respective native environments, and set their solver settings. The co-simulation tool then connects the models together, managing and coordinating their intercommunication according to the configured communication rates. Each individual model utilizes a standard data logging tool to export selected data to external files during run-time. When the simulation completes, the co-simulation tool closes all the models.

A suite of post-processing tools is then used to analyze and view the results. A signal viewer graphical user interface (GUI) is used to quickly consolidate and plot raw data signals. This is useful for quickly determining whether there are any anomalous data, or whether any errors have occurred. If no immediate problems are detected, chart generator and energy flow analysis scripts are used to post-process the data. The chart generator script processes the raw data to create detailed plots, and then automatically generates PowerPoint charts that display these plots, with labels and other headings. This allows the SI to quickly get a broad overview of the simulation results without having to spend the tedious effort of plotting and creating charts. Analogously, the energy flow analysis script calculates all the relevant average and peak energy flows in the modeled system, and displays them in a tabular form. This allows the SI to quickly view a snapshot of the energy flows in the system, and these values can be used to compare different system configurations, which is essential when trying to optimize the aircraft energy use.

## SOFTWARE TOOLS

The specific software tools necessary to achieve the workflow described above are now discussed in detail. The tools are described here in generic terms, highlighting the key features that make them useful for EOA MBD. Intentionally, no particular software package or modeling language to develop the tools is specified. In certain cases specific examples are given which have been developed in the MathWorks MATLAB/Simulink environment; however, any tool that has the properties described in the following sections should be sufficient to aid the engineer in the ways described.

### Model Libraries

Model libraries are version-controlled repositories for verified supplier models. Any changes to models in the library result in a library version update, as well as documentation of the change. This prevents library models from being changed inadvertently, and allows the SI to keep

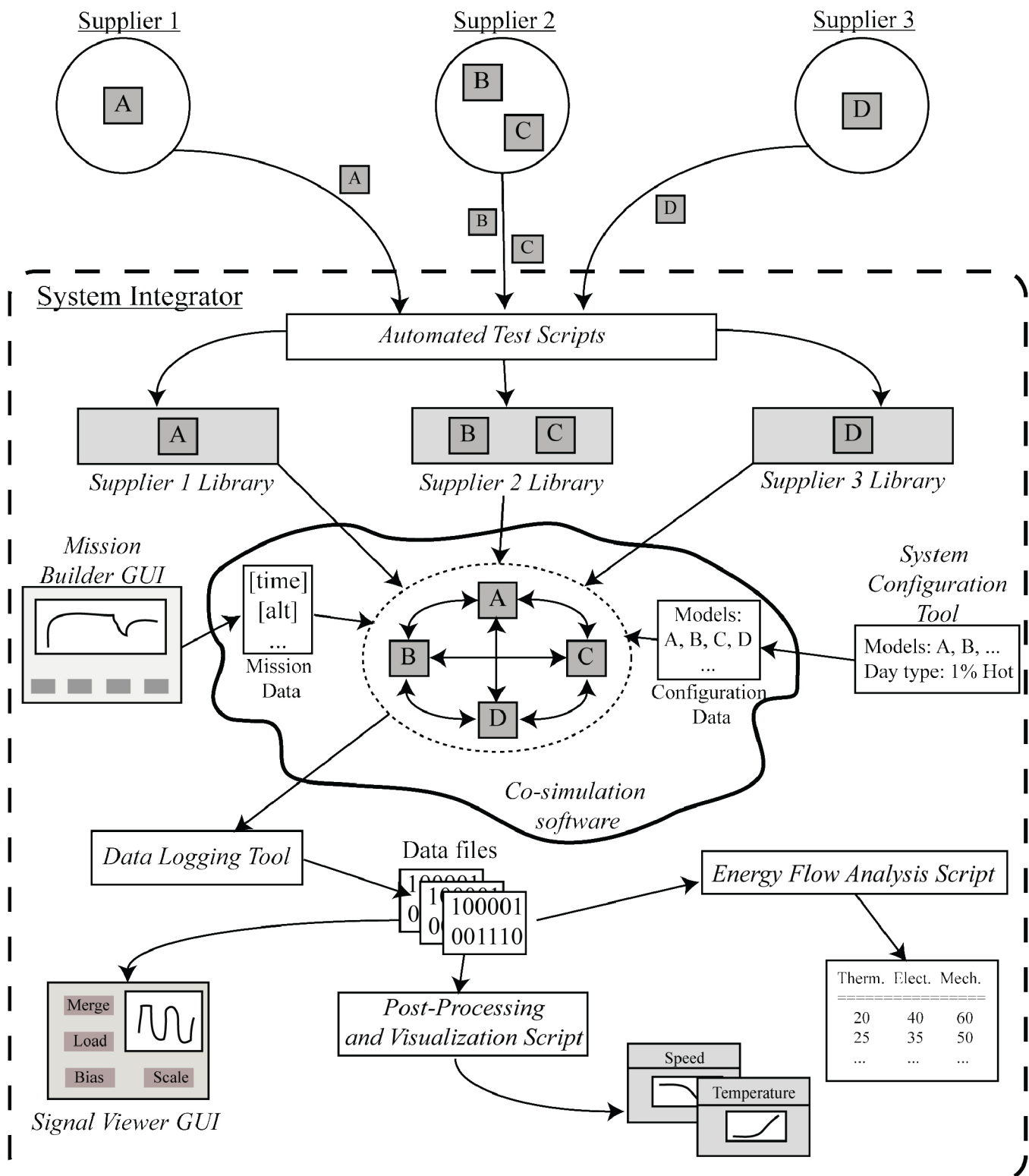


Figure 1. Flowchart of IVM workflow

track of which models were used for each IVM system simulation. Further, when multiple users are working with the IVM simultaneously, the version-controlled libraries ensure that all users are using the same models.

Organizing the supplier models into libraries facilitates organizing a large number of models and helps prevent unintended disclosure of IP between suppliers if model libraries are shared within the team. For example, suppose