Running the Adventium FUSED META-Language Demonstration Models and Scenarios

steve.vestal@adventiumlabs.org
hazel.shackleton@adventiumlabs.org
martin.michalowski@adventiumlabs.org

This paper contains an overview of the various models that are included in the demonstration and the assumed design scenarios in which the demonstration FUSED model composition specifications would be used. The models and scenarios are based on the Lockheed-Martin Desert Hawk (pictured right), a small surveillance UAV. If you have any questions, please email one or more of the above people.

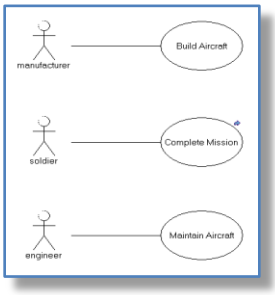


This paper assumes the instructions in the companion white paper "Installing the Adventium FUSED META-Language Demonstrations" have been successfully performed. This paper includes directions to use the installed tools to examine the models and run the demonstration scenario specifications.

Models

The demonstration contains about 12 separate modes (not counting different configurations of the same model) written in 10 different modeling languages.

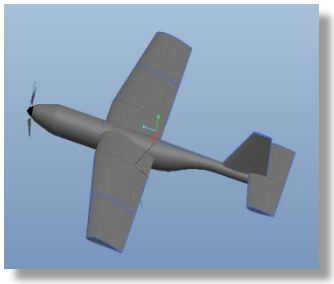
Requirements models, written in the standard SysML language, can be examined by navigating to sysML/Aircraft/Models and double-clicking on the file SystemRequirements.sysmldi. Note that some of these requirements carry the tag "FUSED GT or EQ Constraint." That is the way we have extended the SysML language slightly (using UML profiles) to tag quantitative constraints as model elements that are published so they can be used in FUSED composition specifications.



Equational models, developed using Excel, can be examined by navigating outside of eclipse to DemoWorkspace/tradeoffStudies and double-clicking on AbstractModels.xlsx (we suggest doing this outside of eclipse because we have seen eclipse hang when trying to launch Excel). This is an initial high-level model that engineers might develop very early in a project. There is a second configuration of the model, DetailedModels, that subscribes to mass data published from the solid/geometry model and aero data published from the aerodynamics model in order to refine initial engineering estimates for some of the parameters. Publish/subscribe for spreadsheets make use of native Excel dataset connection features.

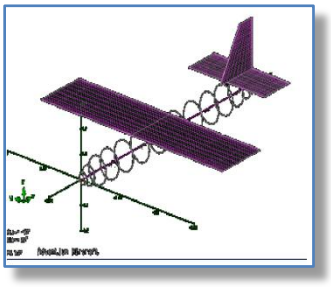
Trade study models, developed using Trade Space Visualizer, can probably best be examined by navigating in eclipse to workFlow/models and double-clicking on abstract_tradeoffs.fused to run that composition specification (as was done to check-out the toolset install). Trade studies models themselves are stored in *.ecf files and are created using a Trade Space Visualizer wizard. The files that end in *.ecf_fused follow guidelines to subscribe to certain data about the design model being studied. There are three different trade space models in this folder, used in three different demonstration scenarios.

Solid/geometry models, developed using Creo/ProE, can be examined by launching the Creo tool from the desktop and navigating to the DemoWorkspace/solidProE/Aircraft/aircraft.asm file. There are actually three configurations of this model: a baseline configuration, a configuration that uses a 4412 airfoil and wing configuration, and a configuration that has erroneous units for one part (to demonstrate units mismatch detection by the FUSED framework).



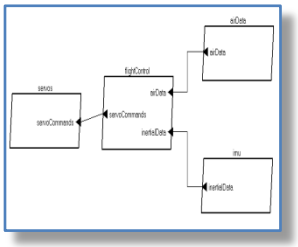
Design constraint models, written in the widely-adopted MiniZinc language, are used to optimally select values for a subset of the design configuration parameters in one of the demonstration scenarios. This model can be examined by navigating in eclipse to constraint_models and double-clicking on optimizeAbstractModel.mzn. Subscribe and publish are done using the MiniZinc properties file format.

Aerodynamics models, written in the textual input language for the Athena Vortex Lattice tool, are analyzed to obtain stability derivatives and related aerodynamics data in some of the demonstration scenarios. This model can be examined by navigating in eclipse to fluidDynamicsAVL and double-clicking on Aircraft_Adventium.avl. This model subscribes to sets of trim point constraints and publishes stability derivatives and aerodynamic coefficients.



Vehicle dynamics models, written in the standard Modelica language, are solved/simulated to obtain trajectories of vehicle state variables, starting from a specified initial vehicle state and simulated for a specified time. This model can be examined by navigating in eclipse to flightDynamics and double-clicking on TestFlight.mos_fused (to see the Open Modelica Compiler model assembly and simulation specification) or any of the files that end in mo_fused (to see Modelica code for parts of the model). The Modelica language has been extended slightly to allow declarations of parameter subscriptions, publications, and uncertainties, which are used in some of the demonstration scenarios.

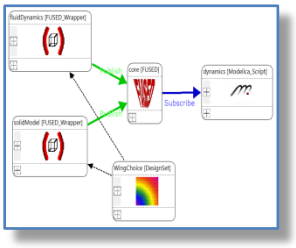
Avionics models, written in the standard Architecture Analysis and Design Language (AADL), specify the software and hardware architecture for the UAV avionics system. This model can be



examined by navigating in eclipse to avionicsAADL/aadl and double-clicking on UAV.aadl. This is the root of the model, the packages folder contains specifications of the various components and hosted functions of the avionics system.

Verification models, written in the widely-adopted SMTLib language, are used to verify a complex consistency condition between two other models in one of the demonstration scenarios. This model, which specifies that a typed subgraph isomorphism property must hold between two models, can be examined by navigating in eclipse to smt_assertions and double-clicking on solidarch.smt2_fused. The SMTLib language has been extended slightly to allow declarations that subscribe to abstract structural representations that have been published for the other two models.

Model compositions, written in our own FUSED model composition language, specify operations that are to be carried out on various subsets of the other models combined in various ways. These can be examined by navigating in eclipse to workFlow/models and double-clicking on files that end with .fused. These will be discussed in more detail in the next section.



Scenarios

The following sections outline five development scenarios and overview the FUSED model composition specifications developed to support them.

Demo specifications generate output files in the workspace. Although you do not need to see them to run the demos, you may want to examine artifacts that are generated. To do this, you must select a project folder, then go to the eclipse Files pull-down and select the Refresh option to tell eclipse to look for new files in the project folder.

Preliminary Trade Space Exploration

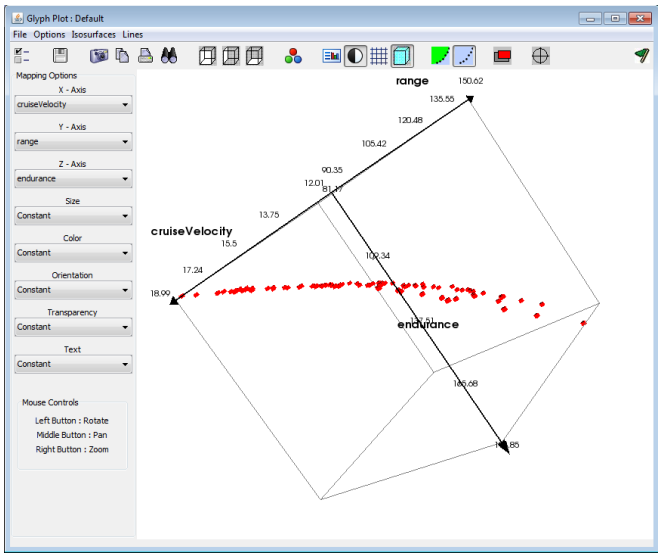
In the early phases of the project, requirements engineers have developed a preliminary requirements specification in SysML, and aeronautical engineers have developed a preliminary model of high-level aircraft design alternatives using basic aeronautical equations. However, there is some uncertainty about what the desires of the end users actually are and what the technology is capable of at what cost. These are issues the requirements engineers are still exploring. For example, this is a hand-launched UAV, and there is significant uncertainty about what the minimum take-off velocity should be for reliable launch under a variety of field conditions and users. (This uncertainty will linger until user experience is gained with a prototype vehicle.)

A FUSED composition specification is created in which the equational spreadsheet model subscribes to ranges of requirements for things like range, endurance, and take-off velocity. These are satisfied using publications from the preliminary SysML requirements model. A Trade

Space Visualizer model subscribes to an abstraction of this combination that allows it to do a statistical sampling of the model trade space. Requirements engineers interact with the Trade Space Visualizer GUI to identify the Pareto frontier and explore trade-offs, perhaps collaboratively with representative end users.

This FUSED specification may be executed by navigating in eclipse to the abstract_tradeoffs.fused specification in the workFlow/models folder and double-clicking on it. Press the red swirly FUSED button on the eclipse toolbar and then press Yes in the next pop-up to run the compiled specification. After several seconds, the script should have opened the equational model in Excel and started the Trade Space Visualizer tool. The Trade Space Visualizer will show two windows, a small toolbar window and a larger Engine Frame window. The latter is used to select sampling methods of the model. For now, increase the number of samples to around 100 and press Start. When the numbers in the spreadsheet stop changing, select the Plots pull-down in the Trade Space Visualizer toolbar and then select any desired visualization display. For example, select Glyph Plot Cube and then use the mapping options on the right to select cruise velocity, endurance, and range as the axes. (The figure below is a screen snapshot where the Pareto filter button has been pressed so that only points on the frontier are shown.)

When you are finished examining a demo, always exit all tools launched by that demo and confirm in the eclipse Console window that the demo has terminated. (Eclipse will only allow one demo to be executing at a time.)



Mixed Fidelity Trade Space Exploration

The preliminary equational model was based on engineer judgment of vehicle mass properties and XFOIL aeronautical data. Both of these are felt to have greater uncertainty than what could be obtained by analysis of more detailed solid and aerodynamic models of the vehicle. At the point in the project where initial solid and aerodynamic models have been developed, the

aeronautical engineers develop a modified version of the equational model that subscribes to mass and aerodynamic parameters obtained from these other models. The resulting equational model is in the tradeoffStudies project with the name DetailedModel.xlxs.

They develop a FUSED specification in publications of mass parameters from the solid model and aerodynamic parameters from the aerodynamic model are used to satisfy subscriptions in the equational model. This composition is then further combined with a Trade Space Visualizer model to support studies using this more detailed model.

This FUSED specification is mixed_fidelity_tradeoffs.fused. It can be executed to launch Trade Space Visualizer to explore the more detailed trade space, just as in the previous scenario.

The first time this specification is executed, it will take several minutes to perform the CFD aerodynamics analyses at all the trimpoints needed for the two vehicle wing configurations. After this, re-executions will be shorter because the FUSED scripts know how to cache analysis results.

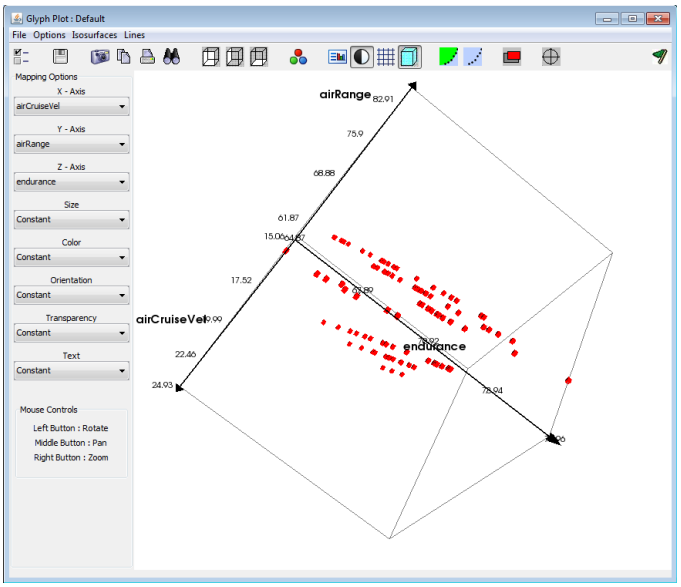
Mixed Initiative Requirements and Design Optimization

The requirements engineers are neither particularly interested in nor qualified to make selections among the various design alternatives identified by the aeronautical engineers, such as choices of battery or propeller or wing design. They want to focus on the uncertain requirements in collaboration with representative end users, and just have a design generated that does a good job for each combination of requirements that is being considered. They only want to explore certain axes of the trade space, those that matter to the end users.

The requirements and aeronautical engineers collaborate to separate out the uncertain requirements parameters to be explored using Trade Space Visualizer from design parameters the engineers will set once they know the requirements. To avoid much manual labor during requirements exploration, the aeronautical engineers write a MiniZinc model that will automatically make an optimal selection of design parameters from a given set of candidate requirements. A FUSED composition specification is created in which the Trade Space Visualizer samples by invoking the optimizer. Another way to think of this is that a composition of an optimizer with the equational model is created that is, in essence, a self-optimizing design model.

This FUSED specification can be executed by navigating in eclipse to optimization.fused and executing it (the companion document “Detailed Walkthrough of Optimization Demo” shows step-by-step screenshots). Trade Space Visualizer will launch as before, except this time there is no Excel model. Instead, when Start is pressed to collect samples, the optimizer will be invoked. Each sample takes longer to collect than with a spreadsheet model, so select a small number of samples (say 5 or 10) just to see something run. An interesting plot can be seen by selecting airCost, airRange, and endurance axes for a Glyph Plot Cube. (In the screen snapshot

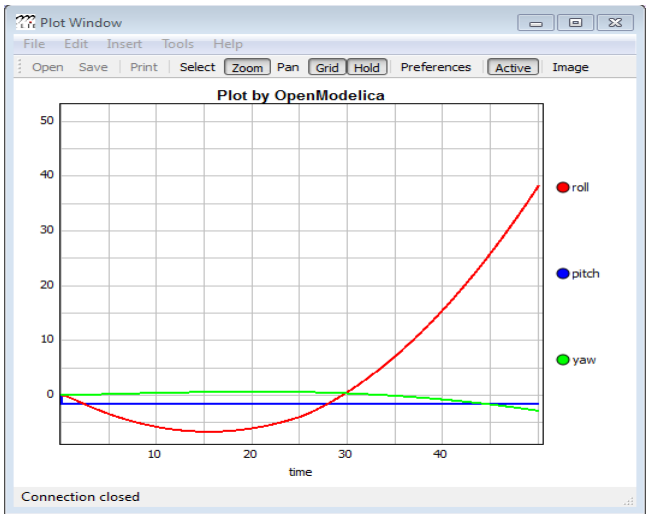
below, we conjecture that the clustering of the Pareto optimal sets reflects optimized decision-making in the discrete design space.)



Multi-Model Simulation

To study vehicle dynamics and handling qualities, the aeronautical engineers create a dynamical model of the vehicle in the Modelica language. This model uses mass properties and stability derivatives, data that is most accurately and easily obtained from the solid and aerodynamics models respectively. Initially, the engineers want to be able to generate simulation traces for various design configurations and vehicle initial states.

A FUSED specification is created in which the three models are composed, with publish/subscribe relations to satisfy the subscriptions in the vehicle dynamics model. This is the solid_aero_and_dynamics.fused specification, which can be executed like the others. After a minute or so (assuming earlier aerodynamic analysis results have been cached), a Modelica plot window will appear. Close this window to end the demo.



Model Consistency Verification

Later, the avionics engineers develop an AADL specification of the digital hardware and software architecture of the system. This specification includes logical hardware resources like processors, busses, sensors, actuators, batteries, motors, radios, etc. The solid model is also detailed so that it includes solid representations of these components. However, there is some concern that these two models may become inconsistent as they evolve during the design.

A FUSED specification is created in which an abstraction of the structure of the solid and avionics models are published by those models. This is a typed graph abstraction in which nodes represent elements of the model (e.g. cards, logical processors) and edges represent relationships between model elements (e.g. part/whole containment, attachment, connection). Nodes and edges are typed using the type system of their respective publishing models. What the engineers desire is that there be a subgraph isomorphism between certain types of nodes in the avionics model to certain types of nodes in the solid model to assure that every logical resource in the avionics model is represented by some part in the solid model, and they are all connected in the same way. This condition is captured in the SMTLib language.

The resulting FUSED specification is graphComparison.fused. The companion document "Detailed Creation of Isomorphic Demo Spec" shows step-by-step how this specification was created.

When executed, structural abstractions of the solid and avionics models will be published, passed to the SMTLib model, and a model-checking tool will be run to verify that the typed subgraph isomorphism holds. When the script completes, the phrase "The two models are consistent" should appear near the end of the Console window output.

To see what happens if the two models are not consistent, you can edit the solidarch.smt2_fused file in the smt_assert project. About half-way down is a line labeled

“Uncomment to introduce an inconsistency.” Uncomment this line, save, and re-execute graphComparison.fused.