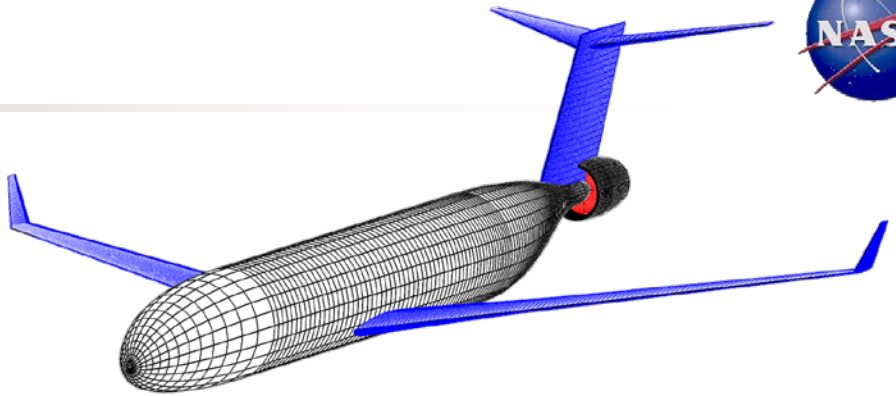
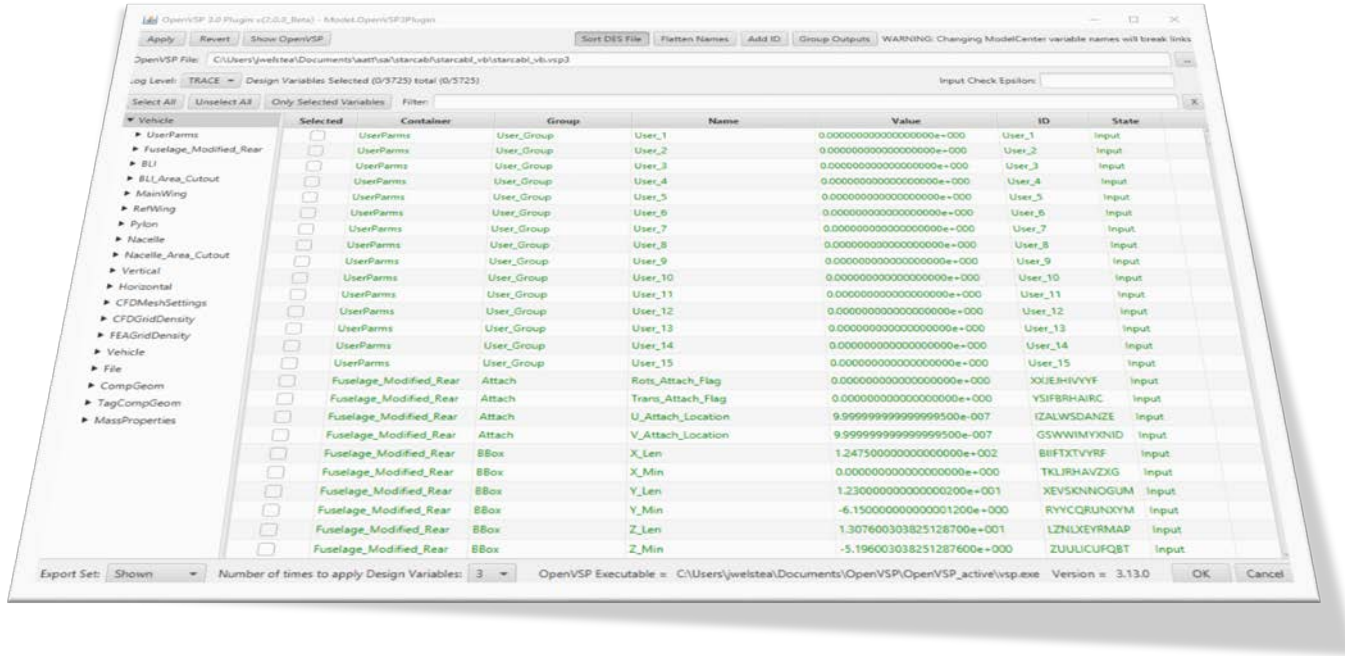
# OpenVSP 3 Plugin



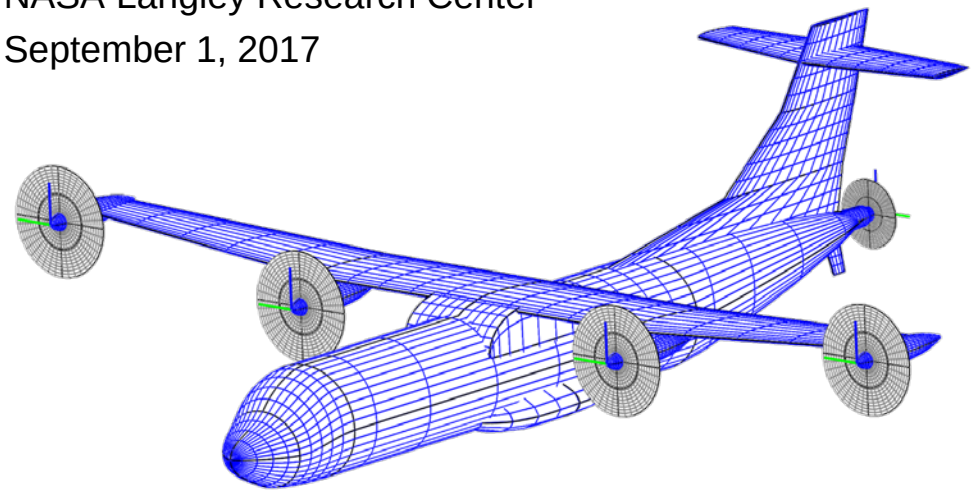


Jason Welstead

Aeronautics Systems Analysis Branch

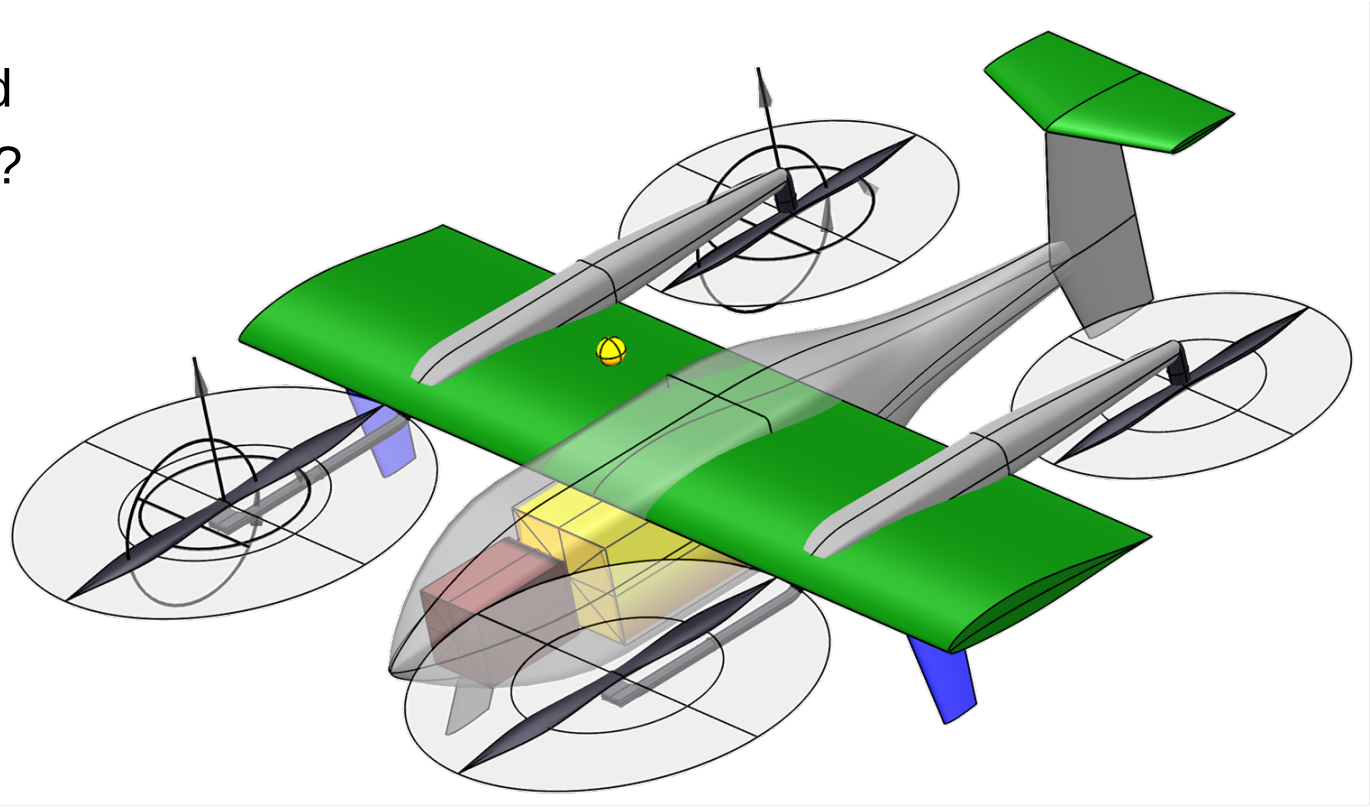
NASA Langley Research Center

September 1, 2017



# Presentation Overview

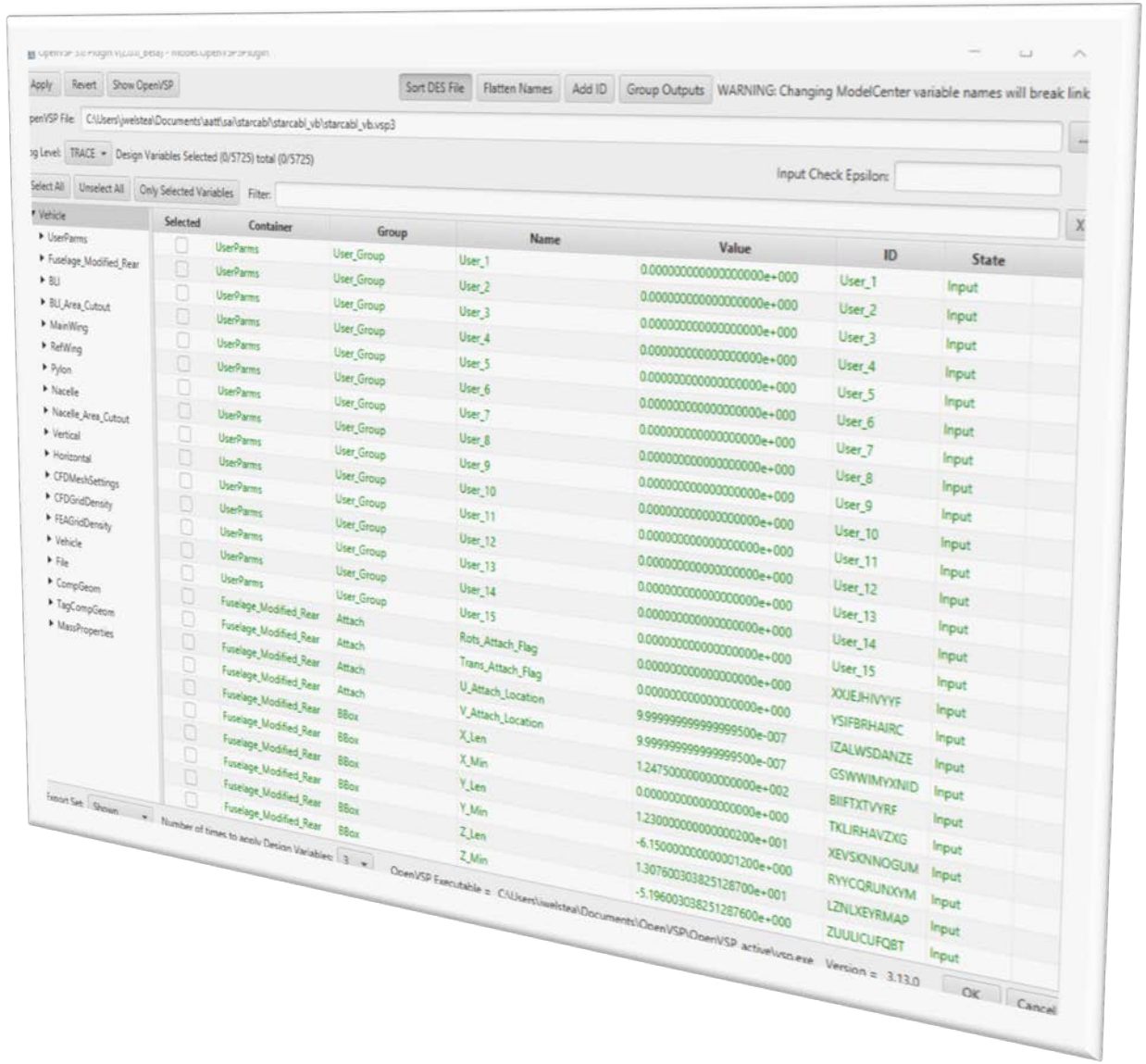
- Introduction
- Plugin Overview
- Graphical User Face (GUI) Walkaround
- Behind the Scenes – How does it work?
- Brief Primer on Installation
  - A ModelCenter Perspective
  - An OpenMDAO Perspective
- Best Practices
- Lessons Learned
- Next Steps
- Acknowledgments



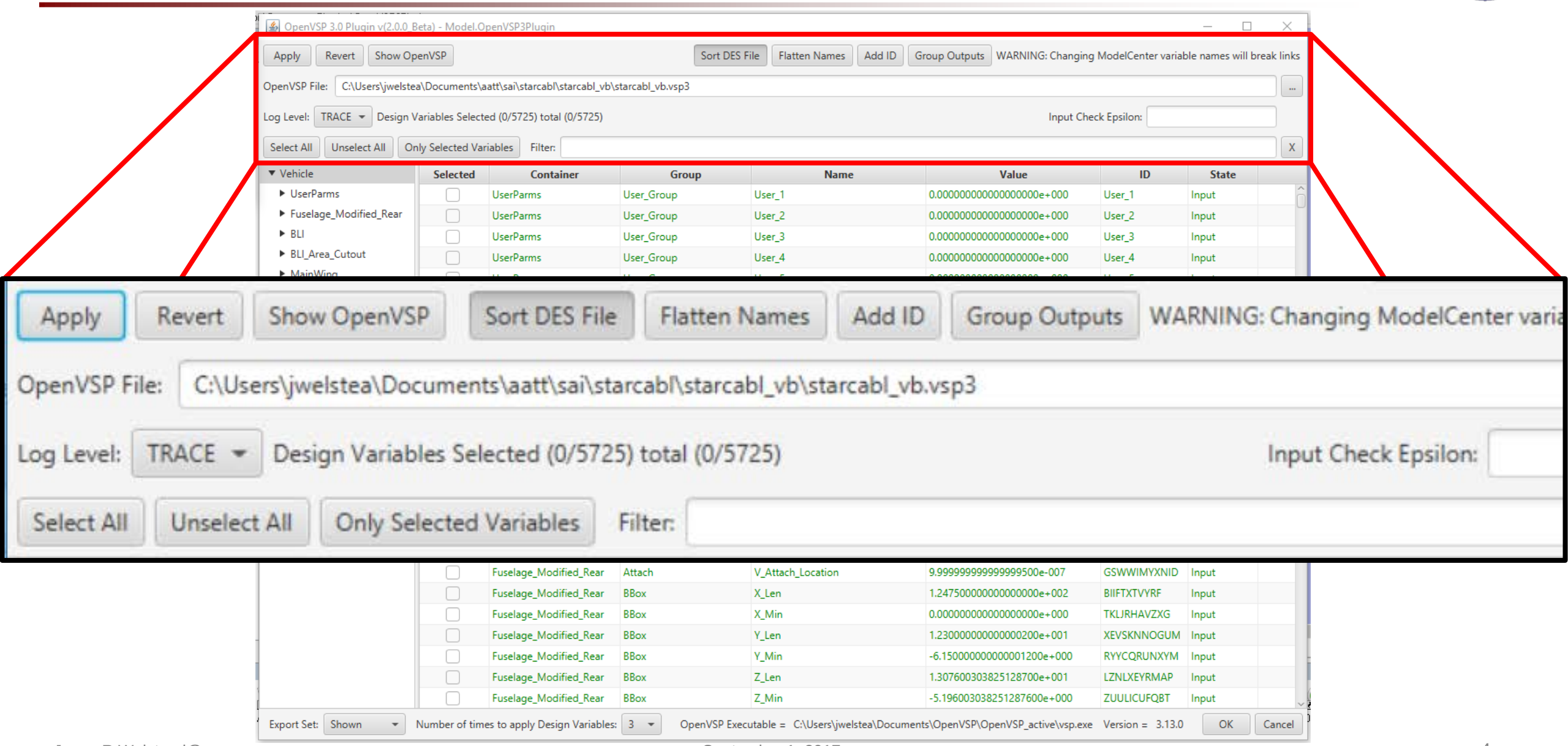
Example OpenVSP v.3 Geometry

# Plugin Overview

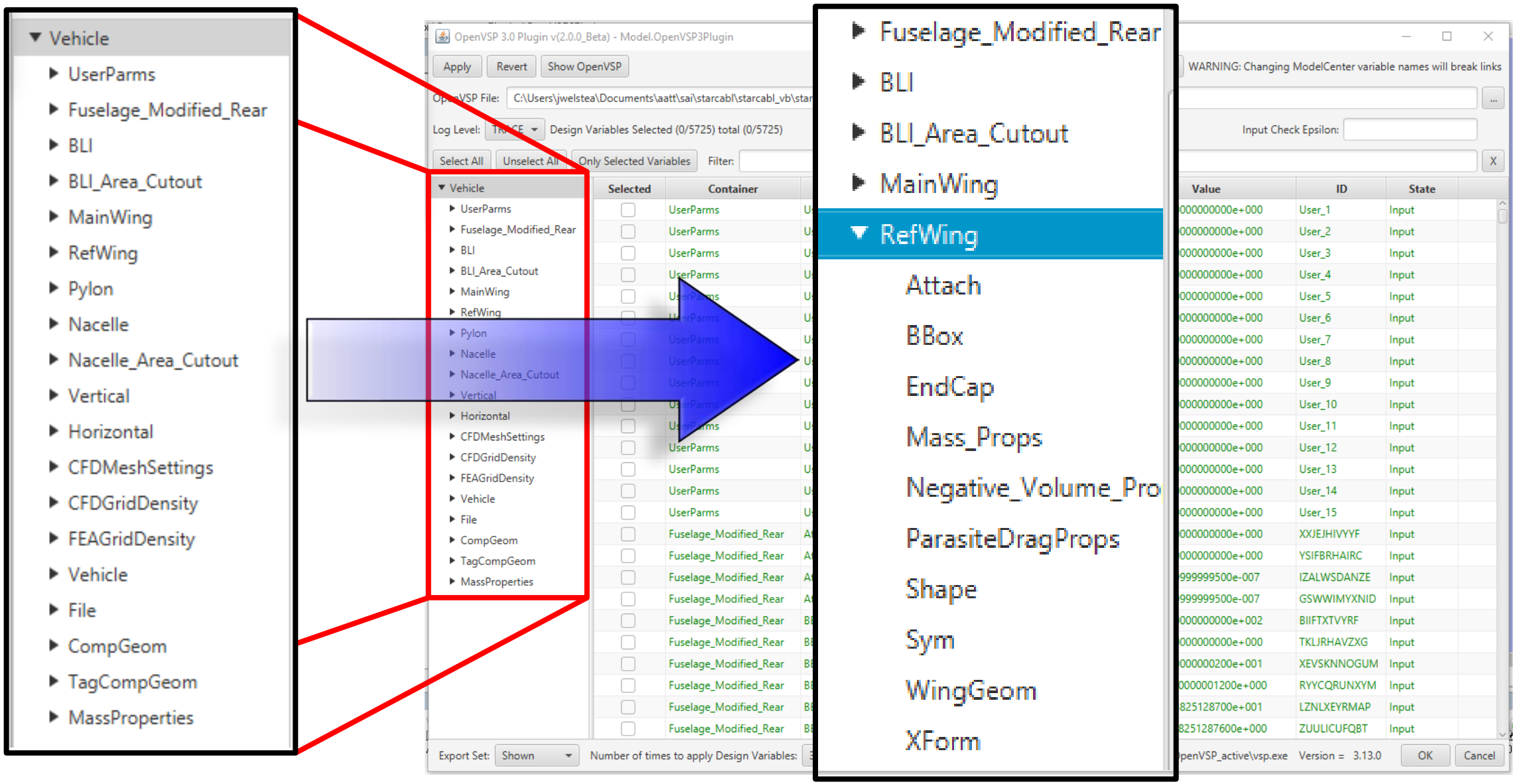
- What is it?
  - JAVA tool for any platform with at least JAVA version 8u40 installed
  - Designed specifically for OpenVSP v.3
- What does it do?
  - Interface between OpenVSP and an analysis framework (ModelCenter, OpenMDAO, etc.)
  - Allows user to interact with an OpenVSP v.3 model through the selection of design variables
- Why did we make it?
  - OpenVSP is an intuitive aircraft modeler for aircraft design
  - OpenVSP is highly integrated into many of our internal design processes
  - Desire to interface OpenVSP v.3 with ModelCenter through a plugin



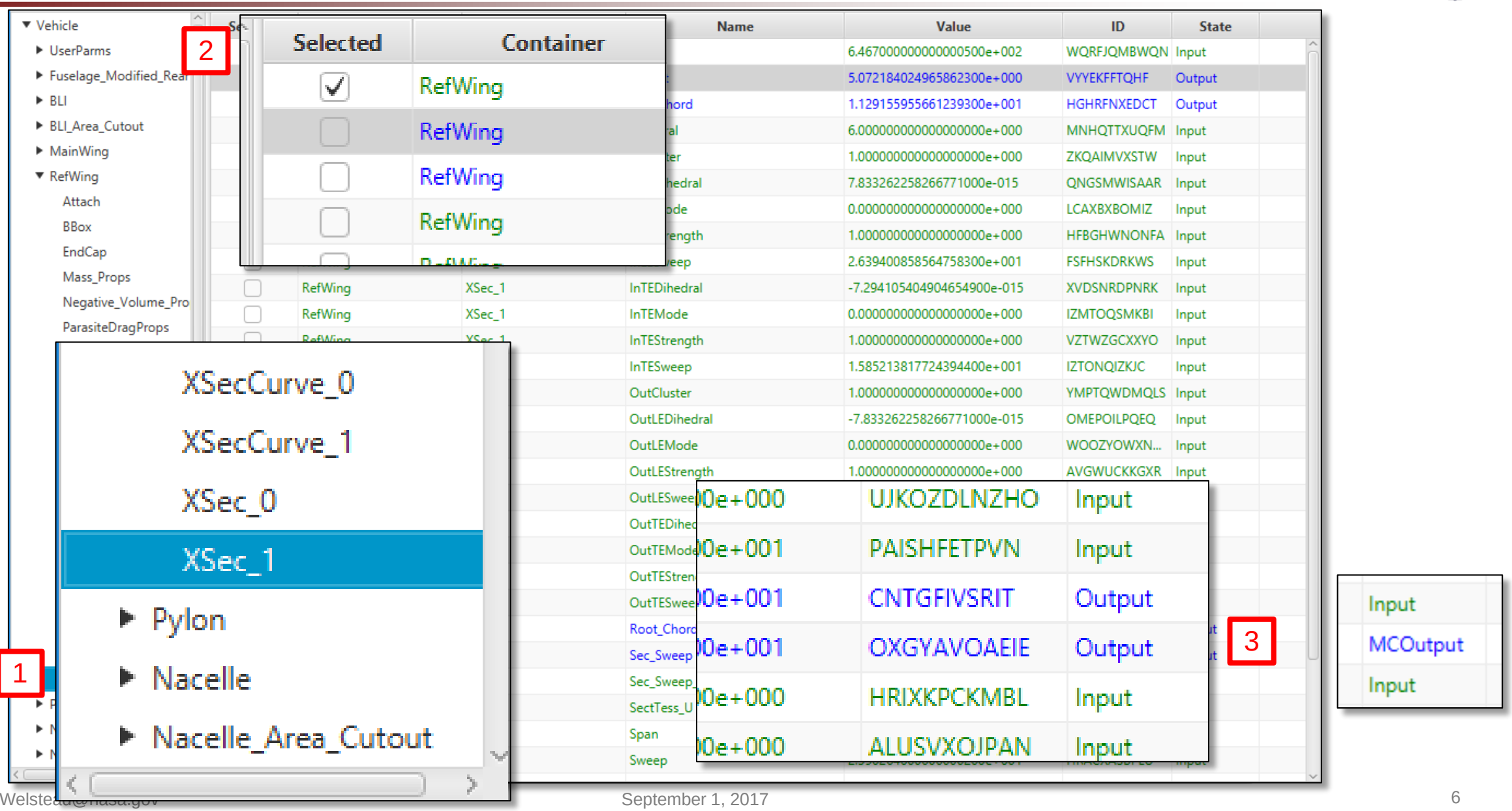
# Graphical User Interface Walkaround – Header



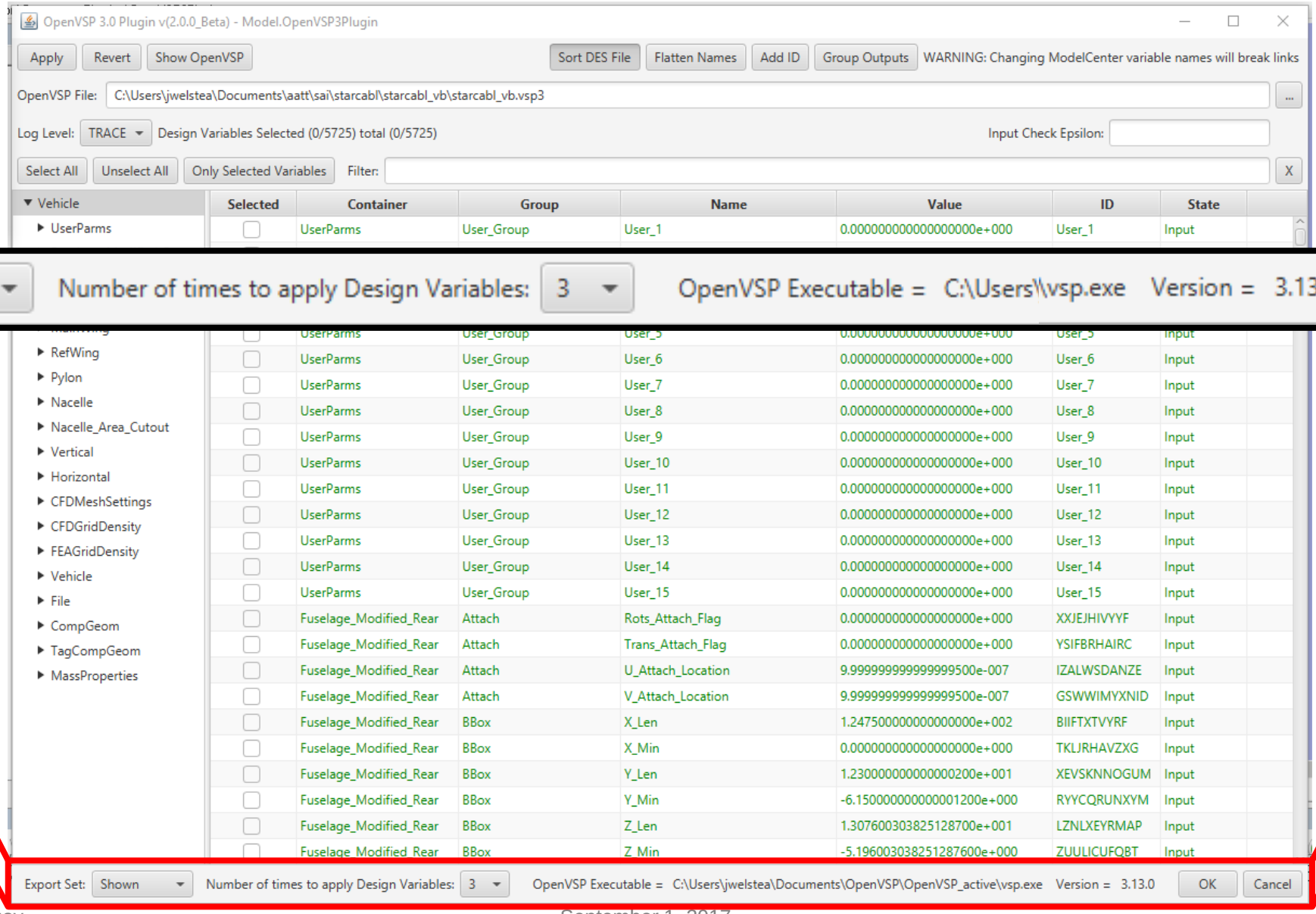
# Graphical User Interface Walkaround – Component Tree



# Graphical User Interface Walkaround – Variable Table



# Graphical User Interface Walkaround – Footer



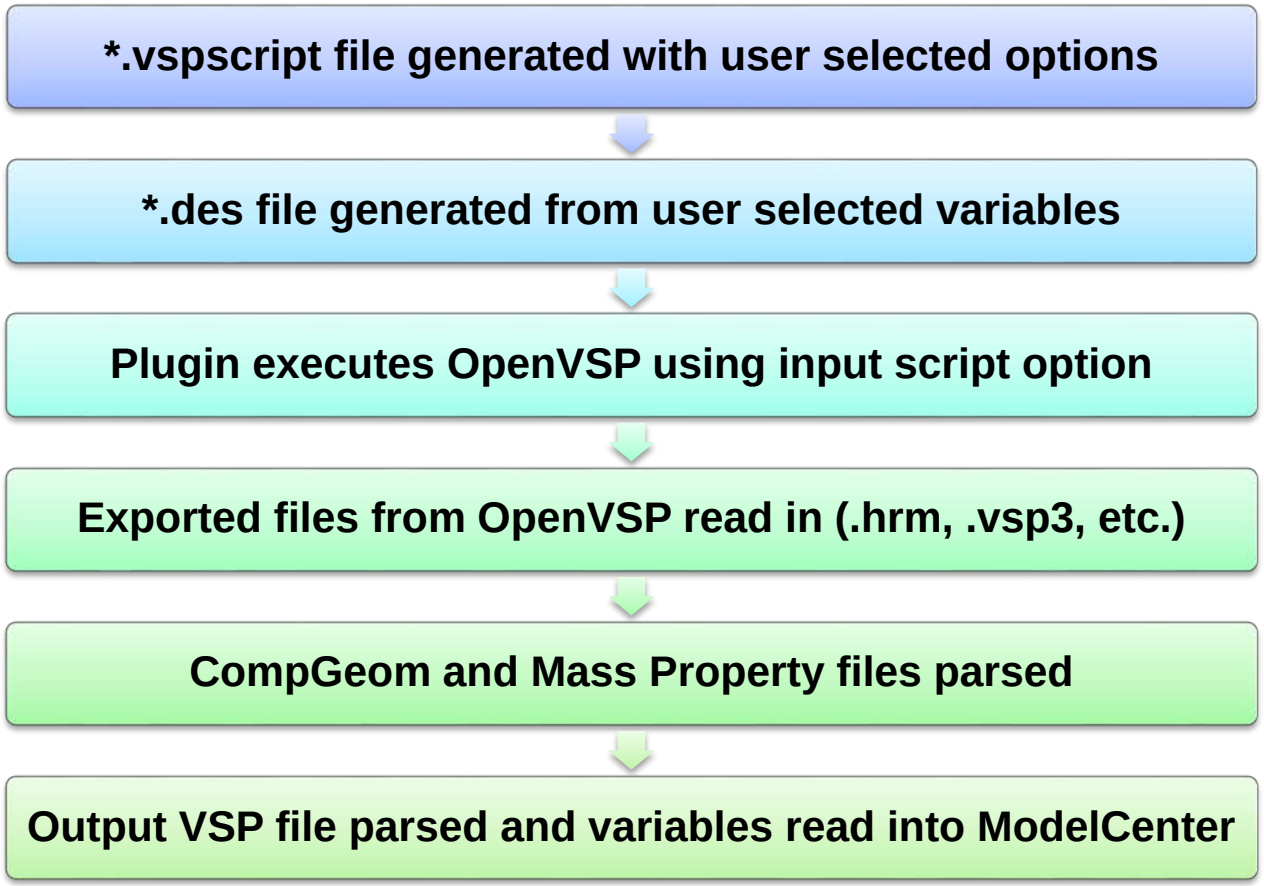
# How It Works – Startup

- State file passed as a command line argument or uses default value (State.xml)
- If state file exists:
  - Load state into plugin
  - Check OpenVSP and plugin versions and alert user if different from previously saved state
  - Identify selected variables in GUI
- If state file does not exist:
  - Select OpenVSP file dialogue box opens for user to select and load a model
  - VSPscript generated to execute compgeom to fully define all variables (related to subsurface areas)
- ModelCenter: state file saved with model and managed through ModelCenter processes
  - Supports ModelCenter multiple copies functionality (DoE, Optimization, etc.) by creating temporary directories for file management and simultaneous execution, but the GUI will not spawn to reduce resources

```
<?xml version="1.0"?>
<State Version="2.0.3" ID="C:\Users\jwelstea\Documents\aatt\sai\starcabl\starcabl_vb\starcabl_vb.vsp3"
       NamingCode="000" SetID="1" NApplyDes="1" VSPVersion="3.13.0" Epsilon="null" LogLevel="OFF">
  <Variable ID="RefWing:XSec_1:Area" Value="6.467000000000000500e+002" VSPID="WQRFJQMBWQN" STATE="Input"
            XPATH="/Vsp_Geometry/Vehicle/Geom[5]/WingGeom/XSecSurf/XSec[2]/ParmContainer/XSec/Area"/>
  <Variable ID="RefWing:XSec_1:Aspect" Value="5.072184024965862300e+000" VSPID="VYYEKFFTQHF" STATE="Output"
            XPATH="/Vsp_Geometry/Vehicle/Geom[5]/WingGeom/XSecSurf/XSec[2]/ParmContainer/XSec/Aspect"/>
  <Variable ID="RefWing:XSec_1:Span" Value="5.727286799999999500e+001" VSPID="DKMFNAVFBGM" STATE="Input"
            XPATH="/Vsp_Geometry/Vehicle/Geom[5]/WingGeom/XSecSurf/XSec[2]/ParmContainer/XSec/Span"/>
</State>
```

# How it Works – Execution

- OpenMDAO Environment
  - Plugin execution complete after startup and files generated (*.vspscript, State.xml)
  - OpenMDAO in control of execution and creation of Design File
- ModelCenter Environment

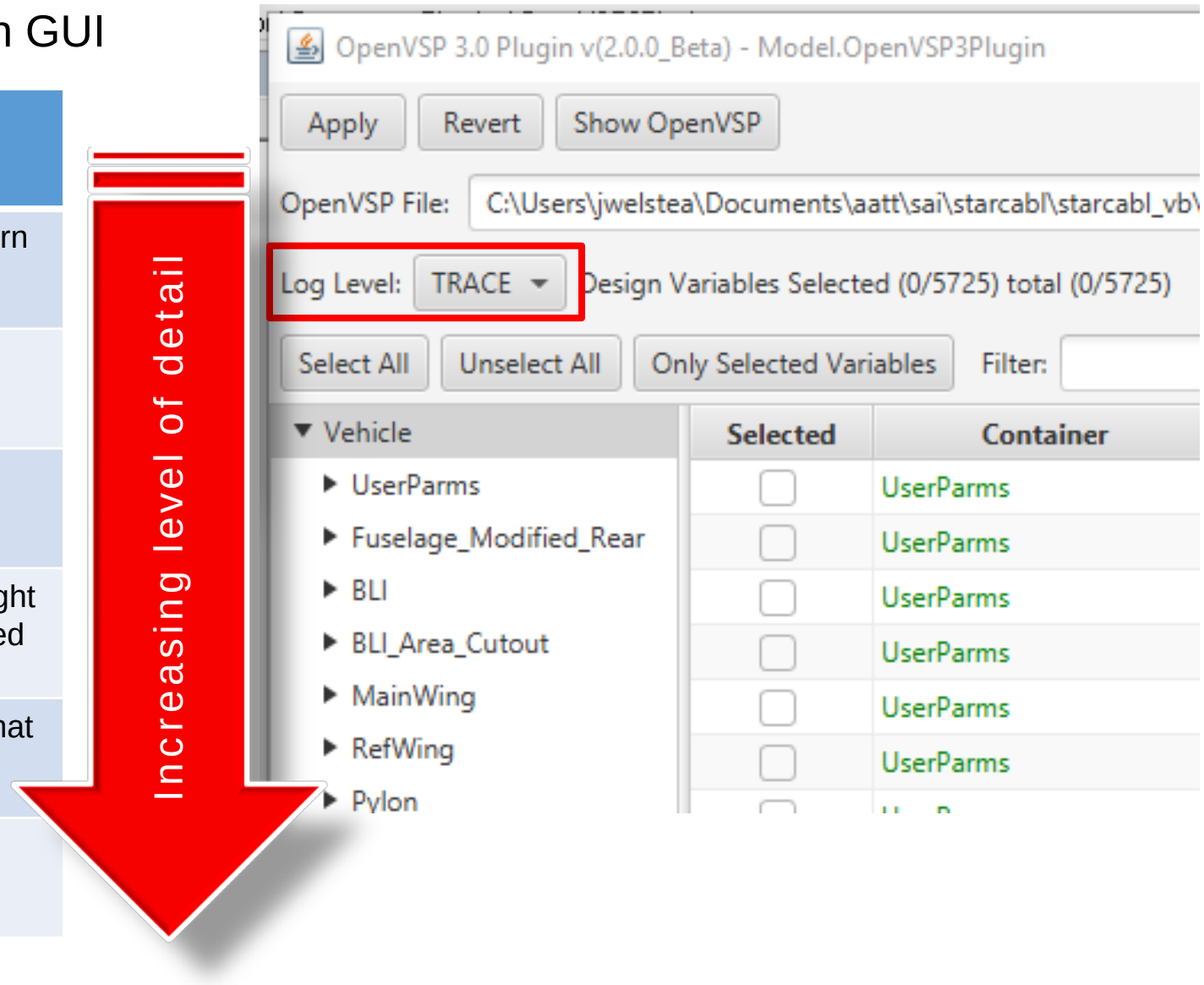



```
void main()
{
  array<string> meshgeoms;
  ReadApplyDESFile("OpenVSP3Plugin.des")
  ReadApplyDESFile("OpenVSP3Plugin.des")
  ReadApplyDESFile("OpenVSP3Plugin.des")
  WriteVSPFile("OpenVSP3Plugin.vsp3", 0)
  SetComputationFileName(COMP_GEOM_TXT_T
  SetComputationFileName(COMP_GEOM_CSV_T
  ComputeCompGeom(1, false, COMP_GEOM_CS
  meshgeoms = FindGeomsWithName("MeshGeo
  CutGeomToClipboard(meshgeoms[meshgeoms
  while ( GetNumTotalErrors() > 0 )
  {
    ErrorObj err = PopLastError();
    Print( err.GetErrorString() );
  }
}
```

# How Does it Work – Data Logging

- Logging level can be controlled by user in GUI

| Log Level | Description |
|---|---|
| OFF | The highest possible rank and is intended to turn off logging |
| FATAL | Designates very severe error events that will presumably lead the application to abort. |
| WARN | Designates potentially harmful situations. |
| INFO | Designates informational messages that highlight the progress of the application at coarse-grained level. |
| DEBUG | Designates fine-grained informational events that are most useful to debug an application. |
| TRACE | Designates finer-grained informational events than the DEBUG. |



# OpenVSP v.3 Plugin Installation Summary

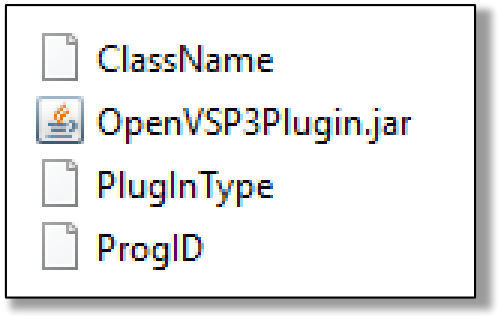
## OpenMDAO

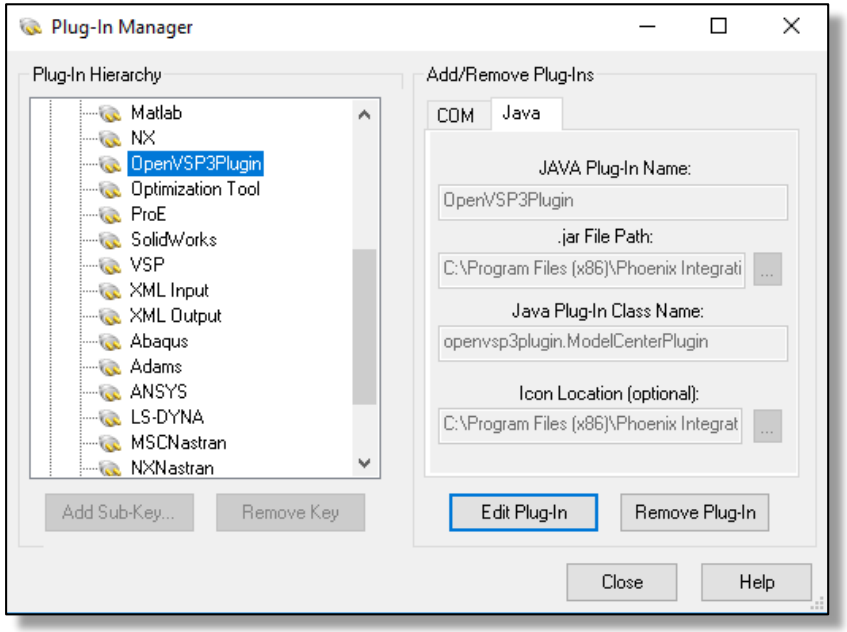
- No installation required
- Behaves as a pre-processor
- Does not control execution of OpenVSP
- GUI executed by running the .JAR file
- Same .JAR file as ModelCenter environment

Must set environment variable OPENVSP_EXE for both processes

## ModelCenter Environment

- Registered Plugin
  - Administer privileges required
  - Allows flexibility in placement of .JAR
  - Ease of installation through GUI
- Registrationless Plugin
  - Requires same information in four separate files
  - Specific file location required
  - No elevated privileges needed



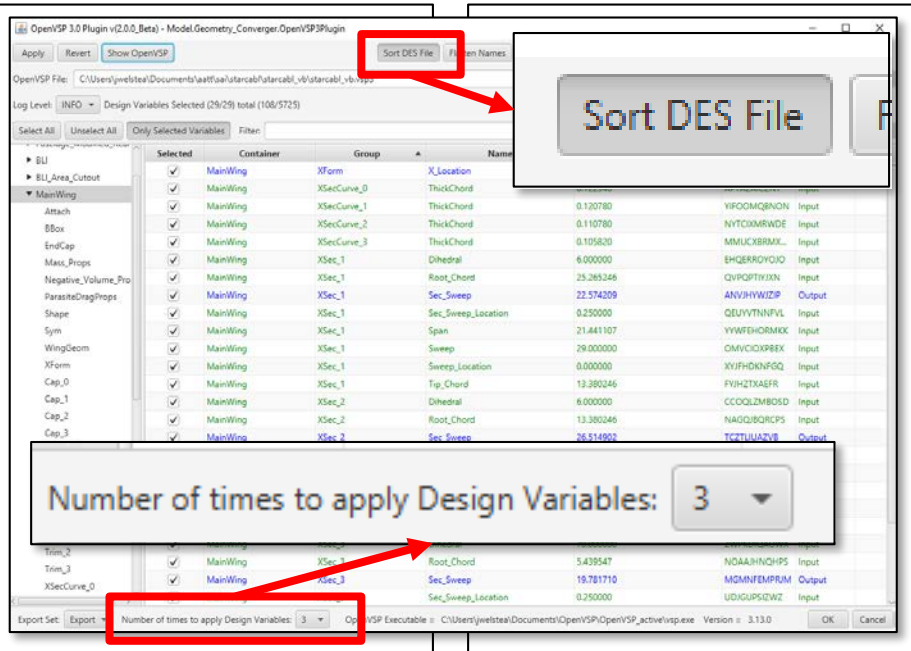

# Identified Best Practices

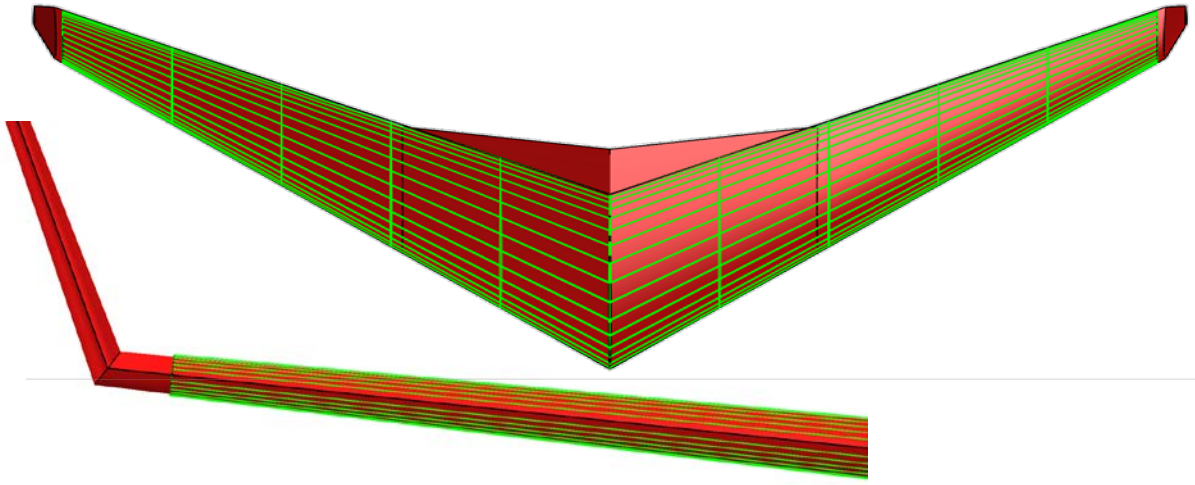
- Take time in developing your VSP model as changing the components can result in issues
  - Variables tracked through parameter ID's, if they are regenerated then connection is lost
  - Nearly all variables can be an input, but not all inputs are active and can be applied
    - Absolute location disabled when component is a child with U/V attach
    - OpenVSP user interface will gray out inactive variables, not available in plugin GUI
- Ensure model is updating as desired, examples of pitfalls to avoid include:
  - Updating wing sweep, but untracked wing sweep location is not at correct percent chord location
  - Using conflicting variables such as planform area and wing section variables
  - Inputting a variable that is an output of a simple or advanced link
- Use extreme care required when defining a multi-section wing component
  - Select root chord, tip chord, and span for the input variables for each wing section even if not changing
  - Select the sort design file option (set by default)
  - Apply the design variables to the model the same number of times as there are wing sections
  - Extreme flexibility in wing definition requires these steps to ensure desired geometry is achieved
- Always use epsilon functionality to check that input variables have not been unintentionally overridden

# Best Practices – Multi-Section Wing Example

## Undesired Geometry

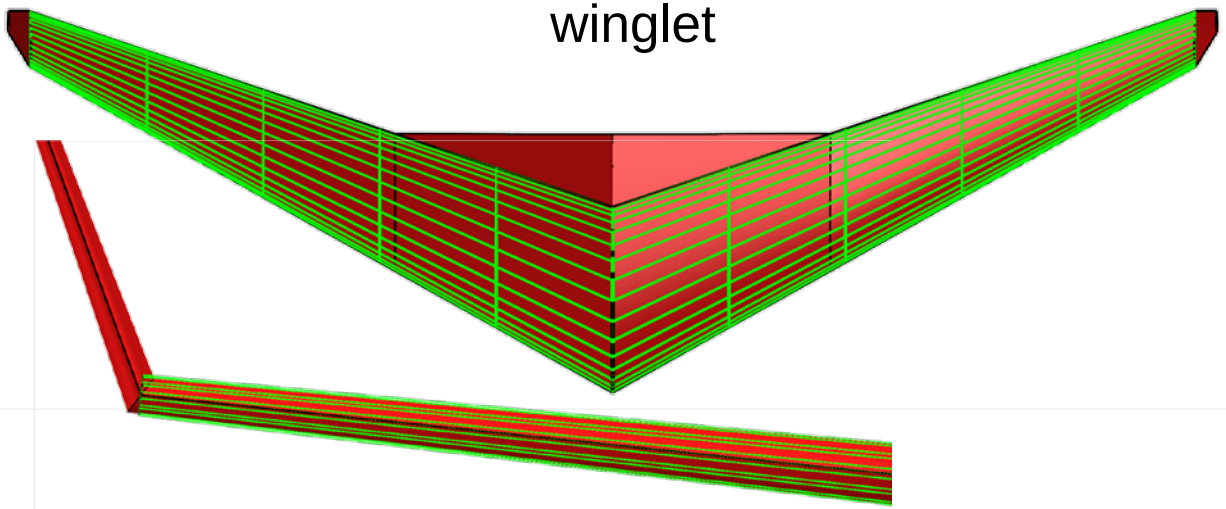
- Only one design variable application
- Design file not sorted
- Non-zero yehudi trailing edge
- Chord at break does not match trapezoidal wing
- Span less winglet does not match trapezoidal wing



## Desired Geometry

- Applied design three times for three sections
- Sorted design file
- Zero degree trailing edge sweep for yehudi
- Chord break matches
- Span matches the trapezoidal wing less winglet



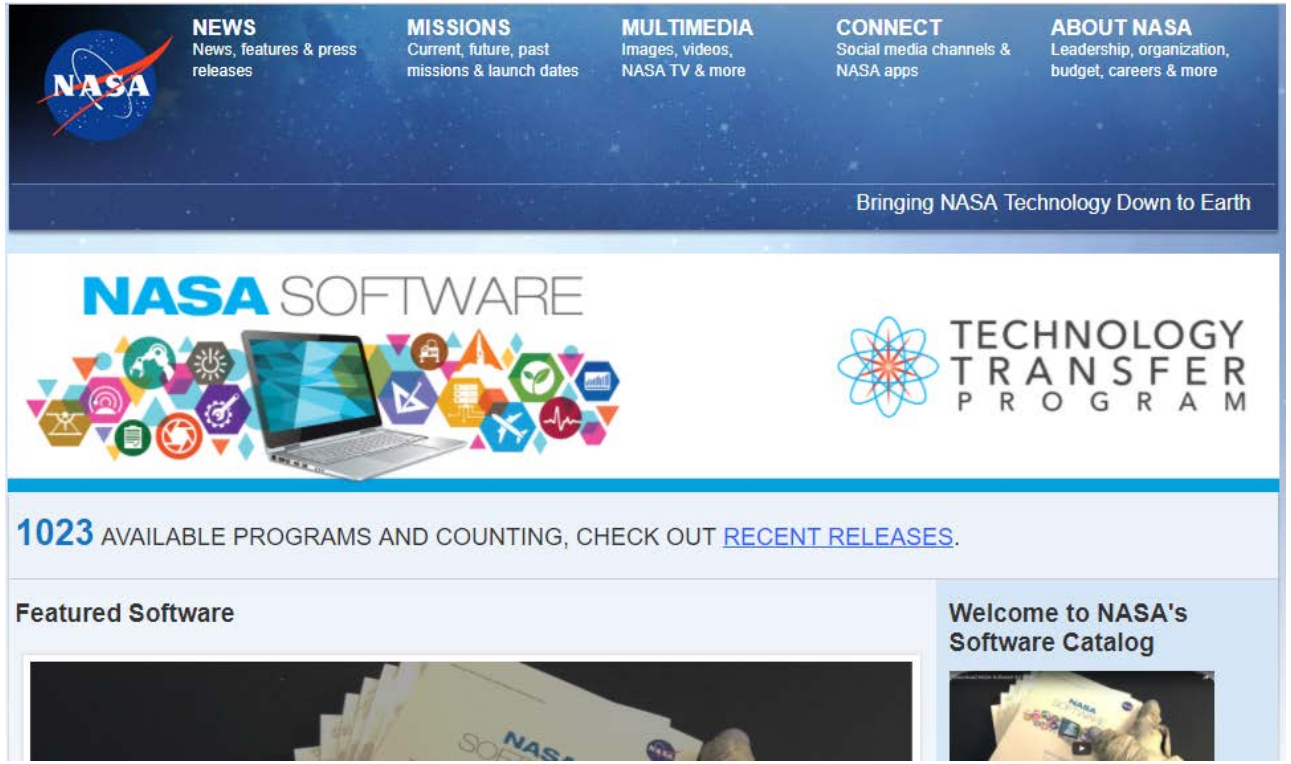
# Lessons Learned

- Extreme flexibility calls for extreme caution
- Use the epsilon feature, it will help catch the errors of a complex model
- Be sure to check which set is being exported when using Comp Geom/Degen Geom/Mass Prop
- Use root chord, tip chord, section span for all sections for a multi-section wing
- When using parent/child relations and attach, ONLY use Xform relative location inputs
- Apply the design variables the same number of times as wing sections
- Avoid duplicate component names and invalid characters such as special characters and spaces

# Next Steps

- Plugin development will continue through in-house NASA efforts
- Exploration of tighter integration with LEAPS and available frameworks (ModelCenter, OpenMDAO, etc.)
- OpenVSP v.3 Plugin in process for release as open source software (nearing completion)

  https://software.nasa.gov/



# Acknowledgements

Development of the OpenVSP v.3 Plugin was funded by the Commercial Supersonic Technology (CST) Project, the Advanced Air Transport Technology (AATT) Project, both of the Advanced Air Vehicles Program (AAVP), and the Transformational Tools and Technologies (TTT) Project under the Transformative Aeronautics Concepts Program (TACP).

I would like the thank the following people for their help and support:

- Jim Fenbert, AMA, NASA Langley Research Center
- Karl Geiselhart, NASA Langley Research Center
- Michael Patterson, NASA Langley Research Center
- Kevin Antcliff, NASA Langley Research Center