*Presented at:*

**OpenVSP Workshop 2016**

# *ALPINE*

## *Automated Layout with a Python Integrated NDARC Environment*

Distribution Statement A – Approved for Public Release – Distribution Unlimited. Review Completed by AMRDEC Public Affairs Office on 19August 2016, (PR #2400)

TECHNOLOGY DRIVEN. WARFIGHTER FOCUSED.

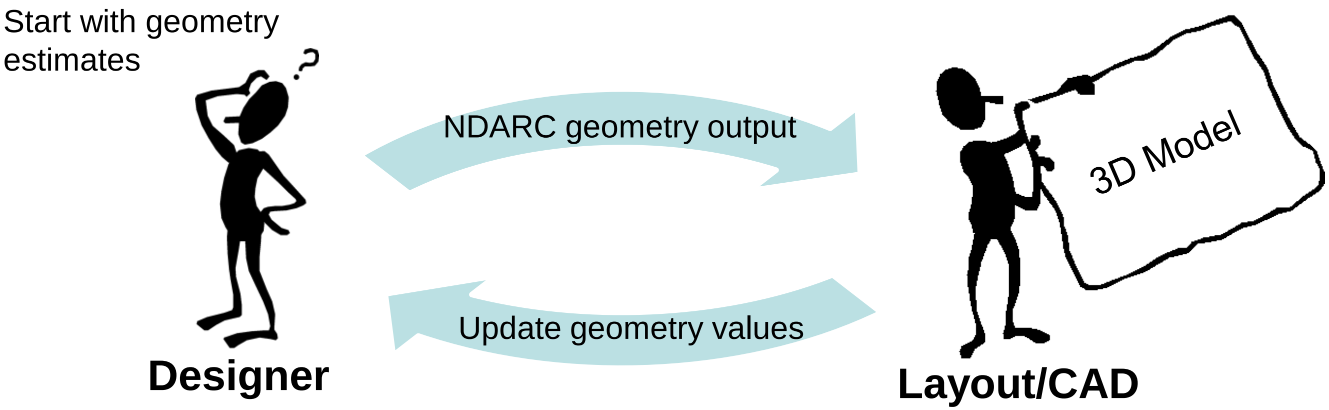
*Presented by:*

**Travis Perry (CTR)**
San Jose State University Research Foundation
Aviation Development Directorate, AMRDEC

***8/25/16***

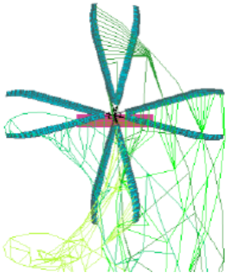
# Our Design Process

- Army Aviation Development Directorate - Concept Design & Assessment Tech Area
- The Army team for conceptual design of rotorcraft
- Our design tool is NDARC (NASA Design and Analysis of Rotorcraft)
- NDARC uses estimates for geometry driven values. In order to close on a design, we iterate with a 3d model
- NDARC does not use a 3d representation to check the values for model consistency
- Use VSP to iterate quickly and reach consistent geometry solution

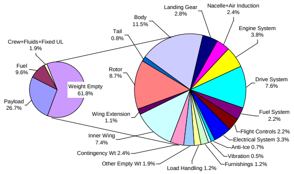


# CD&A Integrated Design Environment

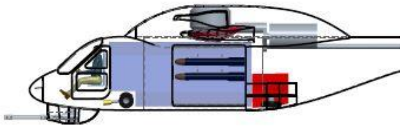
## Aeromechanics



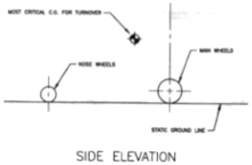
## Mass Properties



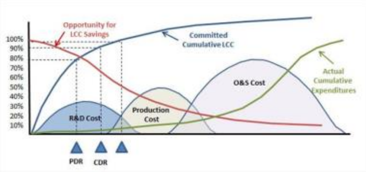
## Internal Layouts



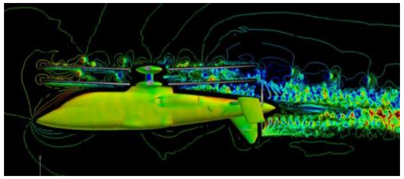
## Landing Gear Calculations



## Cost

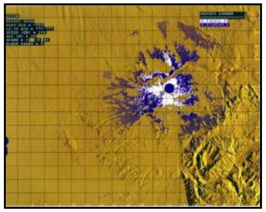


## Aerodynamics

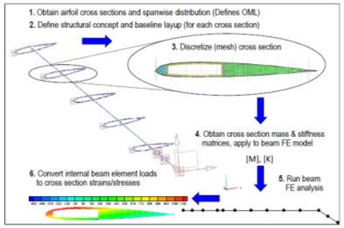


## Model Database / Geometry

## Signatures



## Structures



1. Obtain airfoil cross sections and spanwise distribution (Defines OML)

2. Define structural concept and baseline layup (for each cross section)

3. Discretize (mesh) cross section

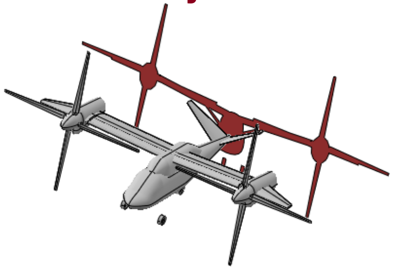
4. Obtain cross section mass & stiffness matrices, apply to beam FE model

[M], [K]

5. Run beam FE analysis

6. Convert internal beam element loads to cross section strains/stresses

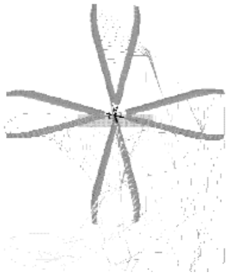
## Wetted/Projected Areas



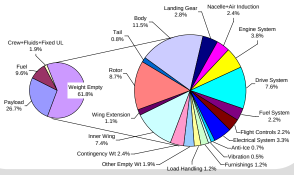
## Presentation Quality Graphics

# CD&A Integrated Design Environment

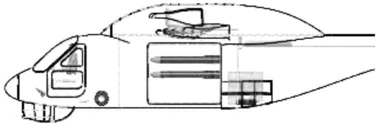
## Aeromechanics



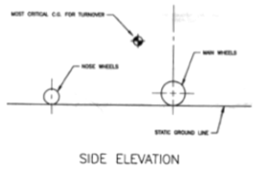
## Mass Properties



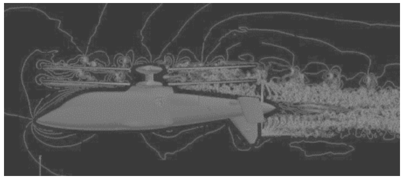
## Internal Layouts



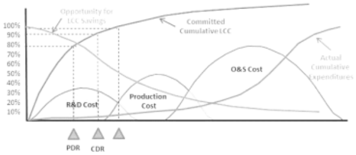
## Landing Gear Calculations



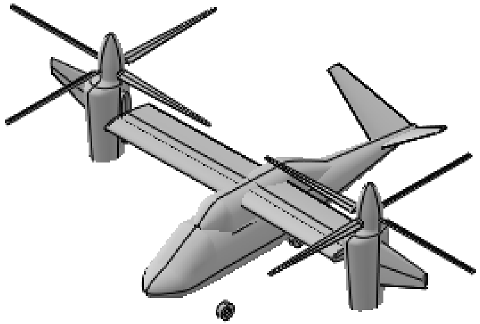
## Aerodynamics



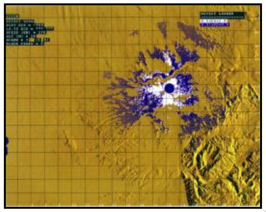
## Cost



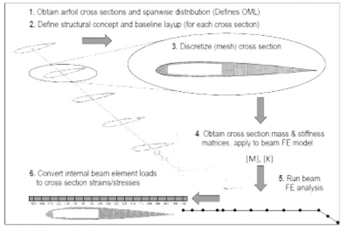
## Model Database / Geometry



## Signatures



## Structures

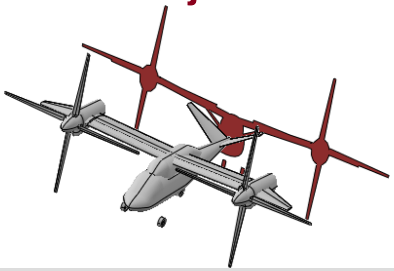


1. Obtain airfoil cross sections and spanwise distribution (Defines OML)
2. Define structural concept and baseline layup (for each cross section)
3. Discretize (mesh) cross section
4. Obtain cross section mass & stiffness matrices, apply to beam FE model

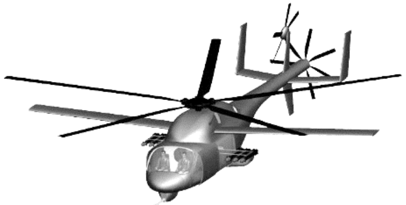
[M], [K]

5. Run beam FE analysis
6. Convert internal beam element loads to cross section strains/stresses

## Wetted/Projected Areas



## Presentation Quality Graphics



# ALPINE

- Automated Layout with a Python Integrated NDARC Environment (ALPINE)
- ALPINE is a python based toolset to generate rotorcraft geometry from NDARC output
- Allows designers to generate geometry rapidly for visual feedback
- Provides parameter feedback for model updates and optimization

# VSP Python API

- The python wrapper is included in the source code but not in the packaged program
- Provides access to all API calls in python
- One to one translation from Angelscript
- Must be built with VSP on the platform that it will be used on

```
def BuildFuse(x, y, z, L, noseL, CckptL, W, H, boomW, boomH, boomL, cargoL):

    fuse_id = vsp.AddGeom('HeliFuse')

    # Set Geometric Values
    vsp.SetParmVal(vsp.GetParm(fuse_id, "cckLength", "Design"), CckptL)
    vsp.SetParmVal(vsp.GetParm(fuse_id, "FuseHeight", "Design"), H)
    vsp.SetParmVal(vsp.GetParm(fuse_id, "FuseWidth", "Design"), W)
    vsp.SetParmVal(vsp.GetParm(fuse_id, "NoseLength", "Design"), noseL)
    vsp.SetParmVal(vsp.GetParm(fuse_id, "FuseLength", "Design"), L)
    vsp.SetParmVal(vsp.GetParm(fuse_id, "CabinLength", "Design"), cargoL * L)
    vsp.SetParmVal(vsp.GetParm(fuse_id, "BoomDiameter", "Design"), boomW)
    vsp.SetParmVal(vsp.GetParm(fuse_id, "BoomHeight", "Design"), boomH)
    vsp.SetParmVal(vsp.GetParm(fuse_id, "BoomLength", "Design"), boomL)

    # Set Positions of each Fuselage Section
    vsp.SetParmVal(vsp.GetParm(fuse_id, "X_Rel_Location", "XForm"), x)
    vsp.SetParmVal(vsp.GetParm(fuse_id, "Y_Rel_Location", "XForm"), y)
    vsp.SetParmVal(vsp.GetParm(fuse_id, "Z_Rel_Location", "XForm"), z)

    # Set Part Density to Default Zero
    vsp.SetParmVal(vsp.GetParm(fuse_id, "Density", "Mass_Props"), 0.0)

    vsp.SetSetFlag(fuse_id,3,True)

    return fuse_id
```

**The python API allows us to use VSP alongside a python NDARC wrapper and packages such as OpenMDAO**

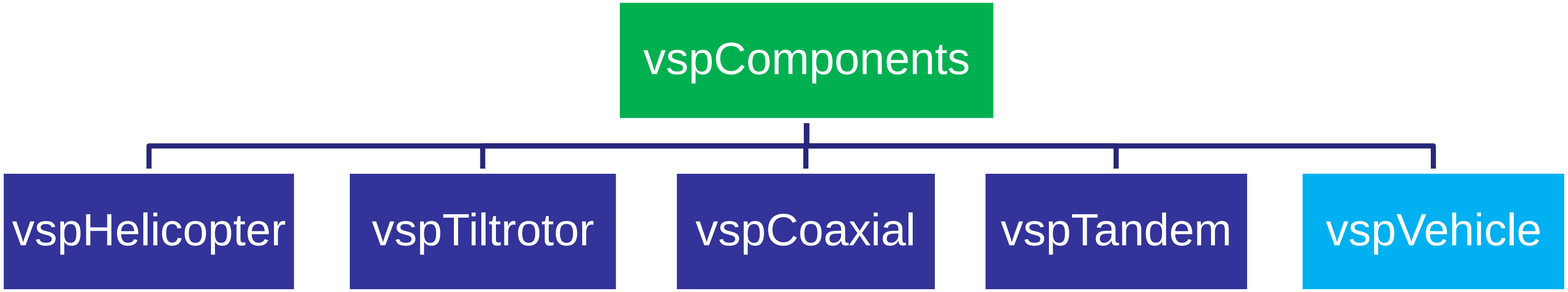
# VSP and NDARC Interface

- The interface chosen to have NDARC transfer to VSP is the optional .geom file output
- The .geom file is a simple text file that contains basic geometric information on all components
- This includes everything from wing and rotor specifics, overall dimensions, component locations, etc
- This geometry file is what is used as input for building a new model in our tool by parsing it into a python dictionary

```
/* Tail 1
  KIND_tail_t1                      = "horizontal"
  area_t1                           =   81.90000
  span_t1                           =   20.97000
  chord_t1                          =   3.905580
  AspectRatio_t1                    =   5.369241
  TailVol_t1                        =  0.7743096
  taper_t1                          =  0.2800000
  sweep_t1                          =   0.000000
  thick_t1                          =  0.1700000
  dihedral_t1                       =   0.00000
  cant_t1                           =   0.000000
  fchord_cont_t1                    =  0.3000000
  fspan_cont_t1                     =  0.8200000
```

# Configurations

- Configurations are separated into classes that build up the components in the .geom file
- Current configurations set up are: SMR, Coaxial, Tandem, and Tiltrotor
- Expandable to any configuration
- Each configuration is built from a library of custom components that chooses the proper components for the configuration
- Configurations complete the geometry information needed to take the NDARC output to a full 3d model

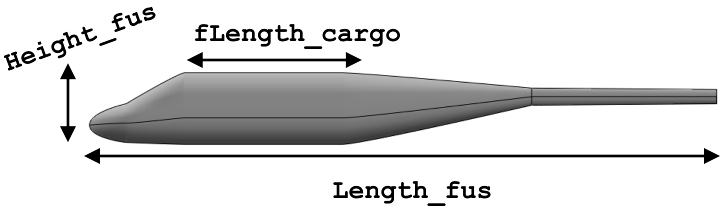


# Component Buildup

- The components used are nearly all custom components.
- We can have components that change based on the parameters given in the geom file.
- This also means that we can create new custom components for unorthodox designs that use the same parameters and they will plug into the code immediately

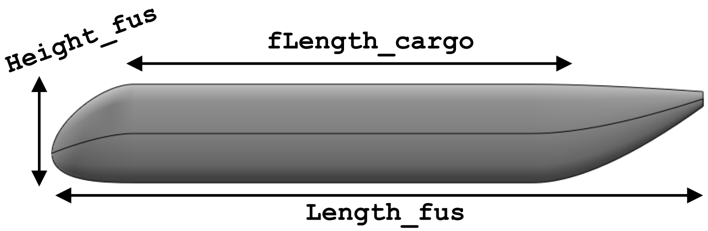
**Built Components**

- Cargo Fuselage
- Utility Fuselage
- Cowling
- Landing Gear Wheels
- Nacelles
- Rotor
- Rotor Hub
- Tilt wing



**Utility Fuselage**

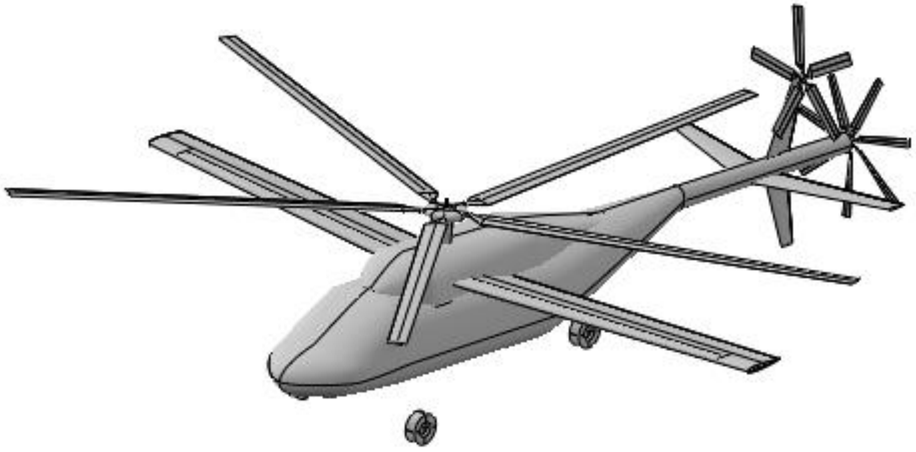
```
/* Fuselage
  Length_fus
  Length_nose
  Length_aft
  Width_fus
  Height_fus
  Swet_fus
  Sproj_fus
  Circum_boom
  Width_boom
  Height_ramp
  fLength_cargo
  KIND_ramp
```



**Cargo Fuselage**

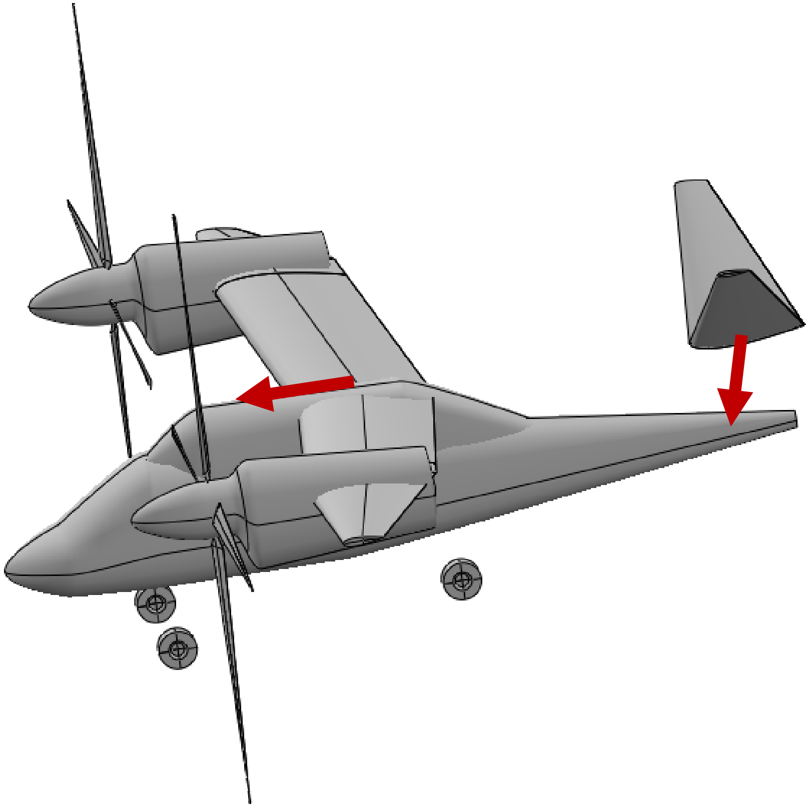
# Model Output

- **Outputs a .vsp3 file**
- **This is an example of a large wing compound made with the tool**
- **The model can now be queried for various values**
  - Wetted area
  - Projected area
  - Wing tank fuel volume
  - Run a geometry update
  - Mass Properties
  - Landing gear sizing and Optimization



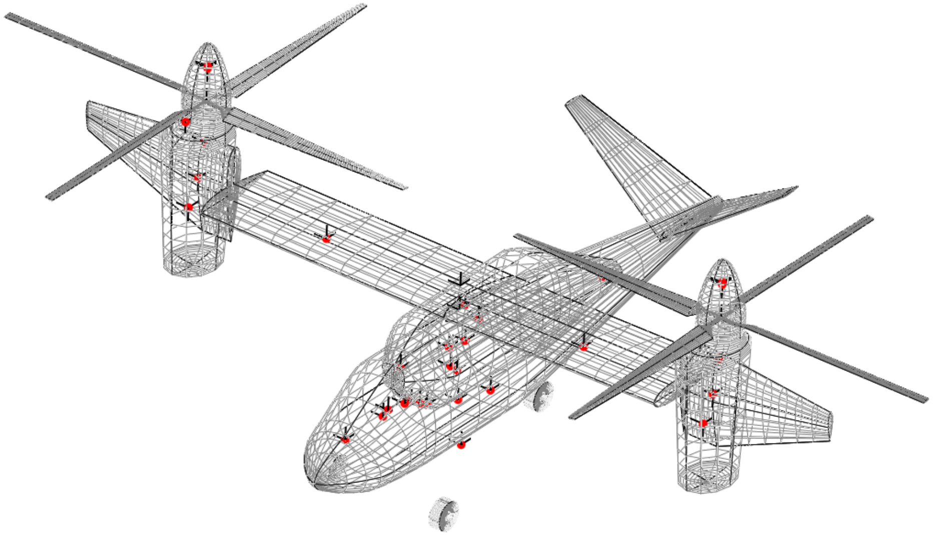
# Geometry Update

- Geometric inconsistencies occur due to NDARC using scaled estimates for geometric placements
- We can run a routine that checks the model versus our own geometry rules
- The routine adjusts placements to fix the inconsistencies of the model to be passed back for iteration



# Mass Properties Analysis

- NDARC design file lists the weight breakdown
- All surfaces are given weight over their areas
- Internal components are represented by ‘BLANKS’ and are assigned corresponding masses
- VSP’s Mass Prop Analysis is run to compute the inertial properties
- You can then use the mass prop output to size and place landing gear



# Demo

# Status and Next Steps

- Current features
  - Reads NDARC geom file and builds from parts library
  - Aircraft: SMR, Tiltrotor, Coaxial, Tandem
  - Mass properties for flight dynamics and tipover
  - Landing gear sizing and layout
  - Tested on Windows, with 32-bit Python 2.7
- Future work:
  - Close loop with NDARC and OpenMDAO
  - Aircraft: UAS (multiple configurations), non-conventional designs
- We are working to release the software as open source to the general public

**AMRDEC Web Site**
**www.amrdec.army.mil**

**Facebook**
**www.facebook.com/rdecom.amrdec**

**YouTube**
**www.youtube.com/user/AMRDEC**

**Public Affairs**
**AMRDEC-PAO@amrdec.army.mil**