A Generic Multibody Parachute Simulation Model

Jason Richard Neuhaus*
Unisys Corporation, Hampton, VA, 23681

and

Patrick Sean Kenney†
NASA Langley Research Center, Hampton, VA, 23681

Flight simulation of dynamic atmospheric vehicles with parachute systems is a complex task that is not easily modeled in many simulation frameworks. In the past, the performance of vehicles with parachutes was analyzed by simulations dedicated to parachute operations and were generally not used for any other portion of the vehicle flight trajectory. This approach required multiple simulation resources to completely analyze the performance of the vehicle. Recently, improved software engineering practices and increased computational power have allowed a single simulation to model the entire flight profile of a vehicle employing a parachute.

Nomenclature

α	=	angle of attack
β	=	angle of sideslip
b	=	parachute span
$C_{d\alpha 2}$	=	change in drag coefficient proportional to angle of attack squared
C_{d0}	=	parasite drag coefficient
C_{drag}	=	total coefficient of drag
C_{lift}	=	total coefficient of lift
I_{xx}	=	parachute roll product of inertia about the center of mass
I_{yy}	=	parachute pitch product of inertia about the center of mass
I_{zz}	=	parachute yaw product of inertia about the center of mass
l	=	parachute height
m	=	parachute mass
$\overline{R}_x$	=	non-dimensional x radius of gyration
$\overline{R}_y$	=	non-dimensional y radius of gyration
u	=	air relative velocity in the body x direction
v	=	air relative velocity in the body y direction
w	=	air relative velocity in the body z direction
x_{bridle}	=	parachute bridle body x offset from the reference center of mass

* Aerospace/Software Engineer, NASA Langley Research Center, Mail Stop 169, Senior Member.
† Aerospace/Software Engineer, Flight Simulation and Software Branch, NASA Langley Research Center, Mail Stop 125B, Senior Member.

I. Introduction

LaSRS++ is the Langley Standard Real-time Simulation in C++ framework used at the NASA Langley Research Center to support activities at the Flight Simulation and Software Branch (FSSB). LaSRS++ supports a wide range of simulation activities, ranging from batch tests, to real-time pilot and hardware in the loop simulations. Current simulation projects include transports, general aviation, and fighter aircraft, as well as a Mars airplane, and spacecraft. The framework includes a suite of classes that allow new vehicles to be simulated with a maximum amount of reuse of existing code. Among these classes is a generic parachute model. Any LaSRS++ simulation model can easily instantiate a generic parachute and examine the dynamics of the parachute/simulation model combination. The generic parachute model provides a high degree of flexibility in the implementation of the dynamics, and is easily extended through object-oriented design techniques.

The LaSRS++ framework allows multiple simulation models to be run simultaneously on a host computer. Typically, these simulation models propagate independently of each other. However, for the parachute and vehicle pair, the two simulation models interact with each other through a riser line. Each model has a system that injects forces due to the separation of the parachute and vehicle. The forces are determined based on the change in length and rate of change of length in the riser line. The forces calculated are applied to the equations of motion of both the vehicle and parachute. After all of the simulation models have recalculated their internal states, the inertial states are integrated, including the effects of the riser forces and moments, and the new orientations and positions are set. The calculations then proceed to the next simulation frame.

II. Math Model

The LaSRS++ parachute model was created to simulate the first order effects of a parachute on a vehicle. The equations of motion for the parachute model are governed by the same six Degrees of Freedom (DOF) used by all other LaSRS++ simulation models. The aerodynamic terms have been simplified down to a set of key parameters, including lift, drag, and pitching moment coefficients. This representation allows the parachute to be modeled quickly for a first order look at the dynamics of the vehicle/parachute system.

The parachute model includes modeling of the riser line that connects between the parachute and the vehicle. The riser line is modeled as a mass-less spring and damper, and applies equal and opposite forces to each simulation model, vehicle and parachute, based on the dynamics of the riser line.

A. Physical Representation

The parachute model uses a body coordinate system where the body x axis is aligned with the bridle, and the body y and z axes lie in the plane of symmetry of the canopy, as shown in Figure 1.

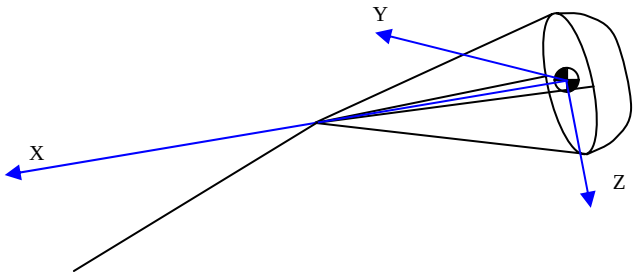

Figure 1. Parachute Body Coordinate System.

The following assumptions were made when implementing the 6 DOF parachute model.

1. The parachute is symmetric in the body y-z plane (Iyy = Izz)
2. The parachute is fully inflated (no opening forces)
3. No interference effects are calculated
4. The bridle position is fixed with respect to the chute center of mass (rigid body)
5. The riser line is assumed to be rigid in tension (no slack/droop)
6. The riser line is mass-less

B. Aerodynamics

The aerodynamics for this parachute model were created to allow for coefficients to be easily specified to a first order degree and were based on observed trends[1]. In this simplified parachute model, the lift is assumed to be zero, and the drag is assumed to vary with the square of the angle of attack.

$$C_{lift} = 0$$
$$C_{drag} = C_{d_0} + C_{d_{\alpha 2}} \cdot \alpha_{equiv}^2 \tag{1}$$

In order to simplify the input parameters of the parachute model, only pitching moment coefficients are considered. Yaw effects of the parachute are implemented such that they mirror the pitching moment effects. The parachute pitching moment coefficients are implemented under the assumption that the parachute is symmetric, and therefore the yaw and pitch terms are similar, except for the small contribution due to the angle of attack on the yaw coefficients, Cn_β and Cn_r. In order to calculate the pitching and yawing moments due to angle of attack and sideslip, the parachute model calculates an equivalent angle of attack. This equivalent angle of attack is computed as the angle between the body x axis and the velocity vector, wind x axis. (as opposed to between the body x axis and the stability x axis).

A roll angle (ϕ_{wind}) is computed to transform the moments from the intermediary, equivalent angle of attack, frame to the body frame. Note that positive alpha and beta produce a negative wind roll angle.

$$\alpha_{equiv} = a\cos\left(\frac{u}{V_{Total}}\right)$$
$$\phi_{wind} = -a\tan\left(\frac{v}{w}\right) \tag{2}$$

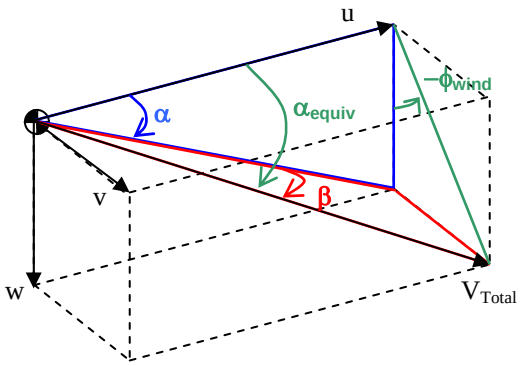

Figure 2. Parachute Air Relative Values.

A similar calculation is done to convert body pitch and yaw rates into an equivalent pitch rate (q_{equiv}). A new intermediary roll angle ($\phi_{body\ rate}$) is computed to convert the equivalent pitching moment back into body pitch and yaw coefficients.

Using the effective angle of attack as the independent variable, the pitching moment coefficient, Cm_α, is calculated. The current model supports either zero or one angle of attack breakpoint. The pitching moment coefficients are specified by the slope before and after the breakpoint, in addition to the angle of attack at which the breakpoint occurs. Negative angles of attack mirror the results of the positive angle of attack lookups. The pitching moment due

to pitch rate coefficient, Cm_q, is specified as a constant in the initialization file for the parachute. Figure 3 shows an example Cm plot with one breakpoint.

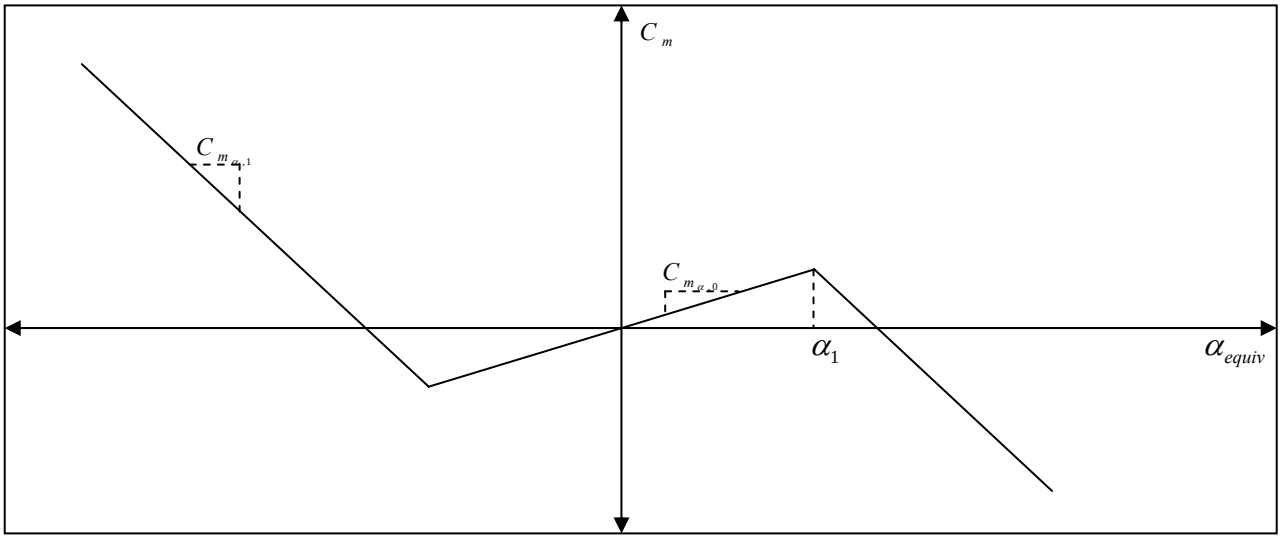

Figure 3. Pitching Moment Coefficient (Cm) versus Equivalent Angle of Attack.

C. Mass Properties

The mass properties of the parachute may either be specified directly, or estimated based on the dimensions of the parachute. If the mass properties are specified directly, the mass, and x and y moments of inertia in body coordinates (Ixx, Iyy) must be supplied. The z moment of inertia (Izz) is assumed equal to the y inertia. If the mass properties are not specified directly, a radii of gyration method is used to determine the approximate mass properties of the parachute. When the radii of gyration method is used, the user may either specify the radii of gyration, or use the default values[2].

When the radii of gyration method is used, the mass properties of the parachute are calculated based on values supplied for the overall height, span, and bridal location. Additionally, the user may override the two radii of gyration values. If the overall length of the parachute is not specified, it is estimated based on the distance between the bridle and the parachute. If the mass was not specified, it is estimated based on the span. And lastly, if a moment of inertia is not specified, it is calculated based on the characteristic length, mass, and corresponding radius of gyration value. The radii of gyration values and mass estimation method were calculated based on data given in Reference 3.

In addition to the specification of the mass and inertias, the LaSRS++ parachute model allows the center of mass to be offset from the reference center of mass (aerodynamic center). Use of this offset allows for the effects of the center of mass offset from the aerodynamic center to be included.

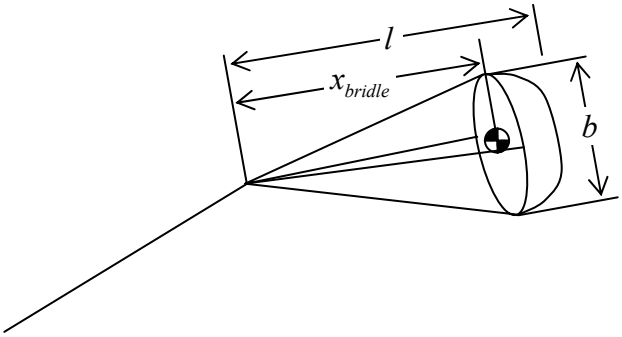

Figure 4. Parachute Characteristic Lengths.

$$
\begin{aligned}
l &= 1.5 \cdot x_{bridle}\left(ft\right) \\
m &= \frac{81.1 \cdot b}{85.3 \cdot g_{ref,earth}}\left(slug\right) \\
I_{xx} &= \frac{b^2 \cdot m \cdot \overline{R}_x}{4}\left(slug - ft^2\right) \\
I_{yy} &= \frac{l^2 \cdot m \cdot \overline{R}_y}{4}\left(slug - ft^2\right) \\
I_{zz} &= I_{yy} \\
\overline{R}_x &= 0.786 \\
\overline{R}_y &= 0.498 \\
g_{ref,earth} &= 32.174\left(\frac{ft}{s^2}\right)
\end{aligned}
\tag{3}
$$

D. Riser Dynamics

The riser line is modeled as a mass-less spring damper, and allows for the transmission of forces between the vehicle and the associated parachute. The riser attachment points on the vehicle and the parachute are assumed to rotate freely, and therefore the riser does not transmit torques. The riser also does not transmit any type of compressive force. This prevents the riser line from transmitting a force when the line is slack. The parachute and vehicle models each specify the location of the attachment point for the riser line. This attachment point is used to determine the moments due to the offset of the riser force application location from the center of mass.

The force on the riser line, T (as shown in Figure 5), is calculated based on the spring constants and damping ratio specified. The force applied on the riser is directly proportional to the length of the riser and the rate of change of the length. Note that if the calculated riser force is negative, indicating slack in the line, the force is set to zero. The riser length takes into account the difference in location between the center of mass and the attachment points on the vehicle and the parachute[4].

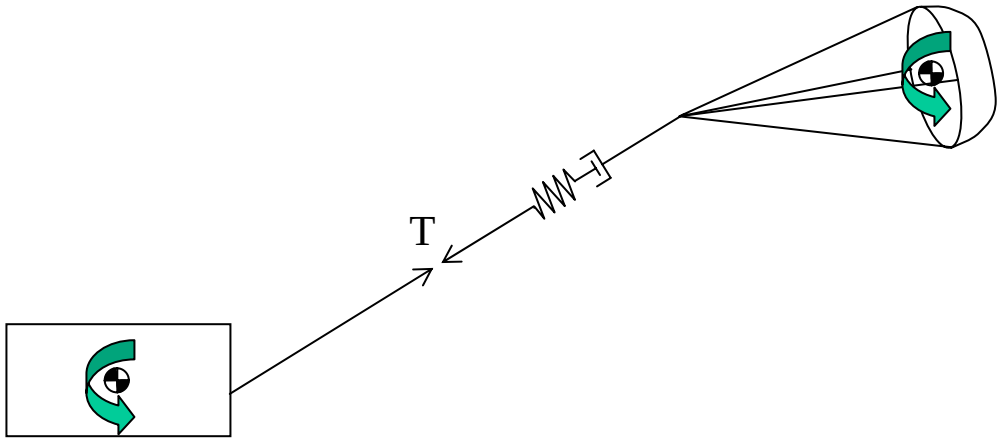

Figure 5. Riser Line Tension.

$$T = k1 \cdot \Delta L + k2 \cdot \Delta L^2 + c \cdot \frac{d\Delta L}{dt}$$

$k1 =$ Linear spring constant

$k2 =$ Second order spring constant (4)

$c =$ damping ratio

$\Delta L =$ change in riser line length

III. Software Model

Before the LaSRS++ parachute model software design can be examined, a brief description of the LaSRS++ framework is required. The framework is composed of six distinct components: Simulation Control, Real-time Services, the Virtual Environment, Hardware Communication, Hardware Interfaces, and the User Interface[5]. Of these components, the virtual environment is the only one relevant to this discussion.

A. The Virtual Environment

The virtual environment is composed all of the simulation models and the environmental models with which a simulation model might interact with. Figure 6 illustrates a high level perspective of the virtual environment.

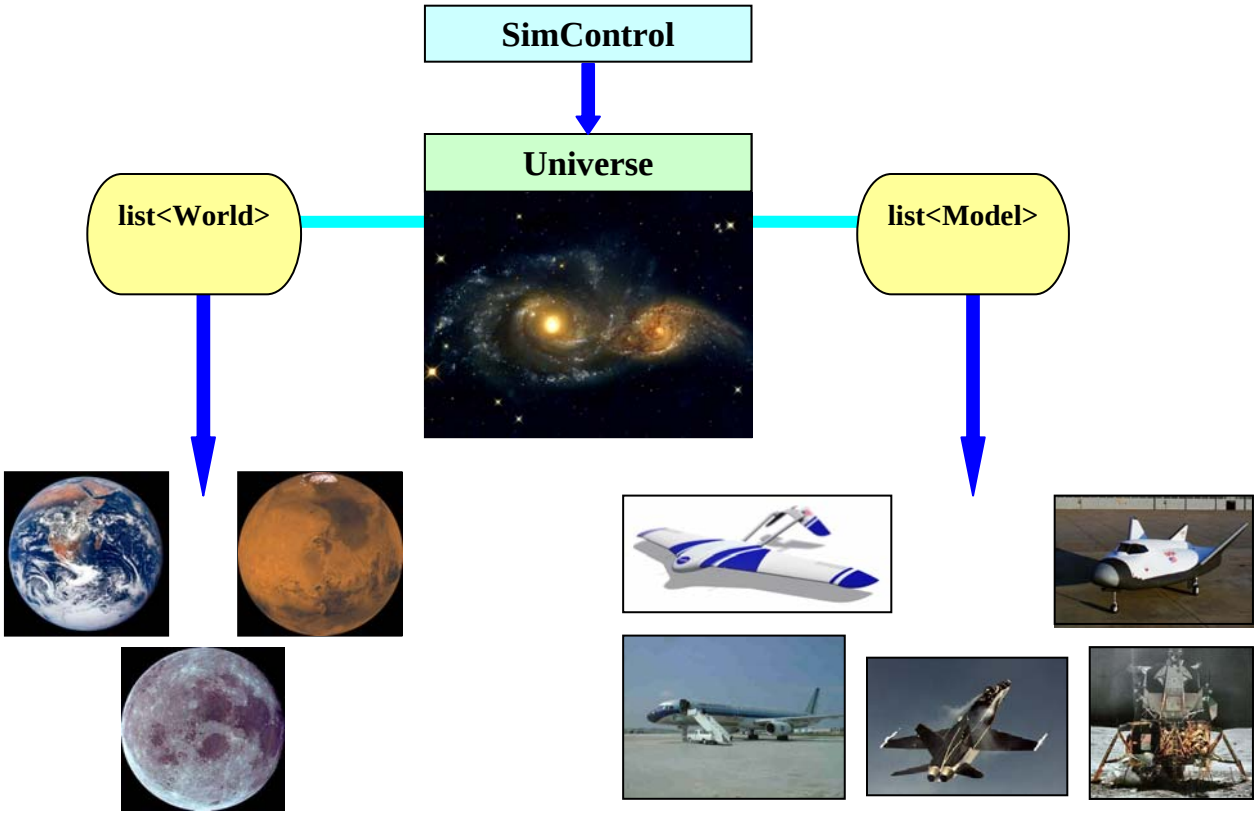

Figure 6. LaSRS++ Virtual Environment.

The Universe class encapsulates the entire simulated reality. It contains a list of all the simulation models in the virtual environment and all of the relative geometry between them. At the highest level, a model is an object that a pilot or other models interact with. Universe also contains a list of Worlds that represent different environments for the models. A World is any celestial body and has a gravity model and it may also have an atmospheric model and navigation related models as well. SimControl is a class utility that supplies global access to the mode, time, time step, and multi-CPU synchronization symbols to Universe and other framework classes. Universe examines the

current simulation mode and instructs the simulation models and worlds to perform tasks corresponding to that mode.

B. Models

Figure 7 is a UML diagram that illustrates the hierarchy of classes used by LaSRS++ simulation models. The base class PositionalModel, occupies a location within the virtual reality and can exhibit movement. A PositionalModel has a position and orientation, and can dynamically update these states. A Vehicle is a PositionalModel that reacts to external forces and moments. A Vehicle therefore has accelerations and integrates it states. An Aircraft is a type of Vehicle whose external forces and moments are significantly influenced by the atmosphere. An Aircraft has additional states like angle-of-attack, sideslip, etc… The Vehicle and Aircraft classes are abstract and cannot be instantiated. A simulation model must derive from one of these classes to obtain the base class behaviors and attributes. In this illustration, an F18a aircraft derives from the class Aircraft.

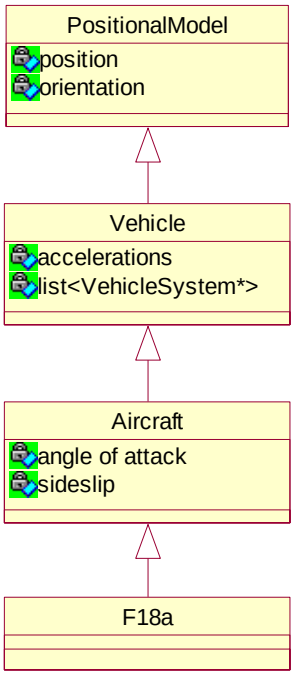

Figure 7. Simulation Model Hierarchy.

C. Vehicle System

All simulation models developed with LaSRS++ derive from either the Aircraft class or the Vehicle class. Figure 8 provides a simplified representation of the architecture employed by LaSRS++ simulation models. A simulation model is composed of three major parts[6]:

1. Subsystem models descended from SimulationModel. Examples of subsystem models are aerodynamic, propulsion, and control system models.
2. Mediator classes descended from VehicleSystem. VehicleSystems follow the mediator design pattern to decouple subsystem models from other parts of the aircraft. VehicleSystems construct the subsystem model and handle I/O for the subsystem model. In Figure 8, the F18aAeroSystem descends from VehicleSystem. It creates the F18aAero object. It retrieves from various sources the inputs to the F18aAero object and instructs it to perform its computations. Finally, the F18aAeroSystem makes the F18aAero model's outputs available to other objects (e.g. other VehicleSystems).
3. The simulation model of interest descends from Vehicle. In this example, the F18a model descends from Vehicle and instantiates an F18aAeroSystem model and places this on the list of VehicleSystems. The F18a represents an F18a aircraft by creating all of the systems that define the aircraft's behavior and placing them on the list of VehicleSystems as well. While not shown in this diagram, the F18a also creates an F18aPropulsionSystem, F18aContolSystem, F18aLandingGearSystem, and so on.

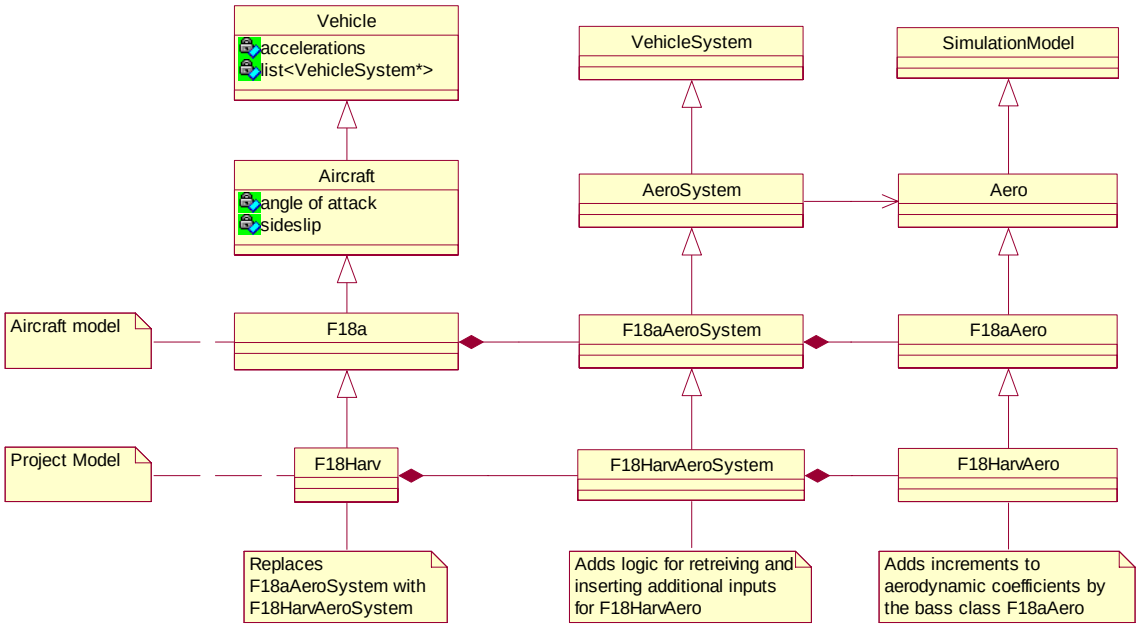

Figure 8. Vehicle Hierarchy.

D. Parachute

Figure 9 illustrates the parachute model and its components. The Parachute class derives from Aircraft and is composed of four major systems: ParachuteAeroSystem, ParachuteRiserSystem, MassPropertiesSystem (not shown), and AircraftLimits (also not shown). The ParachuteAeroSystem class acts as a mediator between the ParachuteAero model and the rest of the Parachute systems. The ParachuteAeroSystem and ParachuteAero models derive from the AeroSystem and Aero classes respectively. The ParachuteAero model produces the total force and moment vectors acting on the parachute about the center of mass.

The MassPropertiesSystem[7] and AircraftLimits classes are existing LaSRS++ framework classes that also derive from VehicleSystem. The MassPropertiesSystem class contains the current mass and inertias of the parachute model. The parachute model does not use any consumables and therefore has constant mass and inertia values. The AircraftLimits class is used as a mechanism to restrict the performance of the model within a given set of parameters. Example limits are maximum G's in each body axis, maximum rotational rates, altitude limits, etc… If a limit is exceeded then the model effectively freezes at that moment in time, and is not restarted until the simulation is reset.

The ParachuteRiserSystem also derives from VehicleSystem. The riser system is composed of several vectors used to keep track of the bridle position, riser line force and moments, and the inertial positions of the riser connections. The total riser tension is computed by the ParachuteSystem class, which is instantiated by the host aircraft model. During operation, the host aircraft is responsible for making use of a ParachuteSystem object and updating the riser line tension within the ParachuteRiserSystem. This allows the ParachuteRiserSystem to compute the forces and moments acting on the parachute due to the riser line tension.

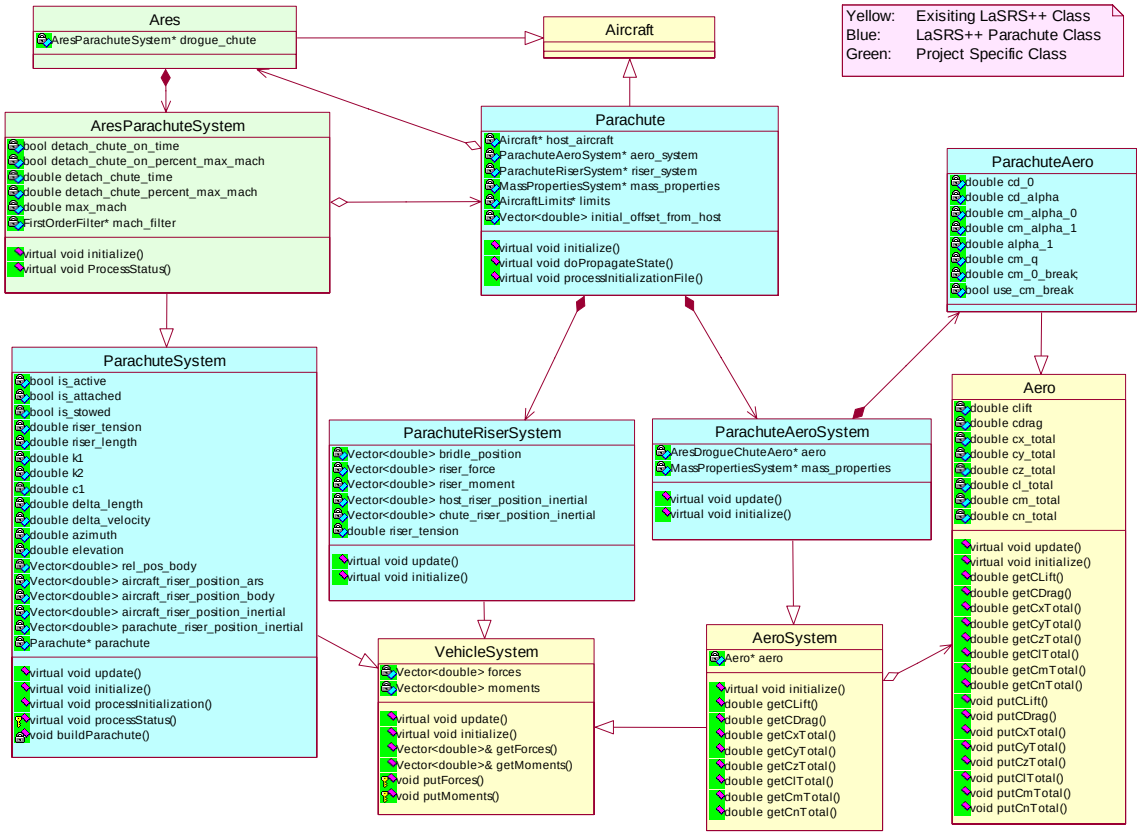

Figure 9. Parachute Hierarchy.

E. Parachute System

The ParachuteSystem class derives from VehicleSystem and is the host aircraft's interface to the parachute model. The ParachuteSystem class contains all of the performance parameters that define the interaction between the host aircraft and the parachute model. The class also contains several vectors regarding the riser position and the parachute position and velocity. From this data, the class computes the riser tension in the riser line and provides the results to the ParachuteRiserSystem instantiated by the Parachute object. One final feature of the class is that it requests the framework to instantiate a Parachute object, thereby simplifying the process needed to create and use a Parachute model with any aircraft.

The design of the parachute model makes it extremely simple to add a parachute to a vehicle, as only three tasks are required. First, the design requires that the host vehicle create a new class that inherits from ParachuteSystem that defines how the parachute will be deployed and/or released. Next, the constructor of the host model must add the newly created class to the list of VehicleSystems stored in Vehicle. Finally, an initialization file that sets the performance parameters of the aircraft's parachute must be created. Because the ParachuteSystem class requests that the Parachute model be constructed and internally handles communication of the riser line tension to the parachute model, the host aircraft only has to add two lines of code to its constructor to make use of a parachute.

F. ARES Drogue Chute

The ARES aircraft implemented the parachute model as a drogue chute[8]. The ARES simulation model was used to assess pullout and cruise performance of the aircraft design[9]. For all of the pullout scenarios, a simulation run began with the drogue chute fully deployed and with tension in the riser line. As part of the many trade studies performed, the AresParachuteSystem implemented several parachute cutting strategies. The AresParachuteSystem class diagram shown in Figure 9 illustrates two of the strategies, namely to cut the parachute with a timer, or to cut the parachute as a percentage of the maximum Mach number achieved during the pullout. Figure 10 illustrates the ARES Drogue Chute model.

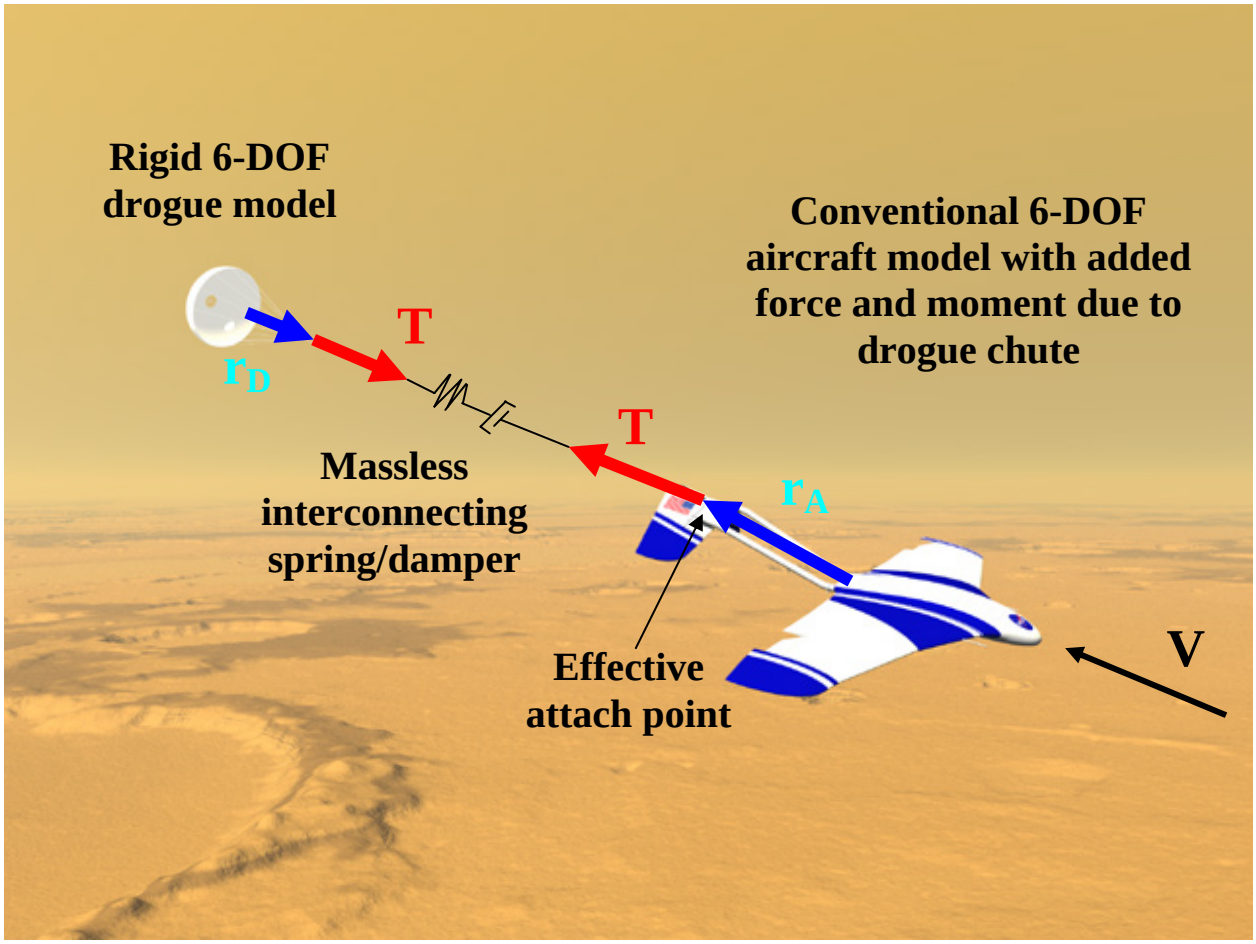

Figure 10. ARES Drogue Chute Model.

G. Fighter Spin Chute

To illustrate the flexibility of the design, the F18a model was outfitted with a spin recovery parachute. Figure 11 illustrates the class diagram for this configuration. Again, note that only one new class was created, and two additional lines of code were required to add the spin recovery parachute to the F18a model. The FighterSpinChuteSystem uses a cockpit switch to command both the deployment of the parachute and its release. The class was implemented as a framework class, thereby allowing any fighter aircraft model to make use of a spin chute if needed.

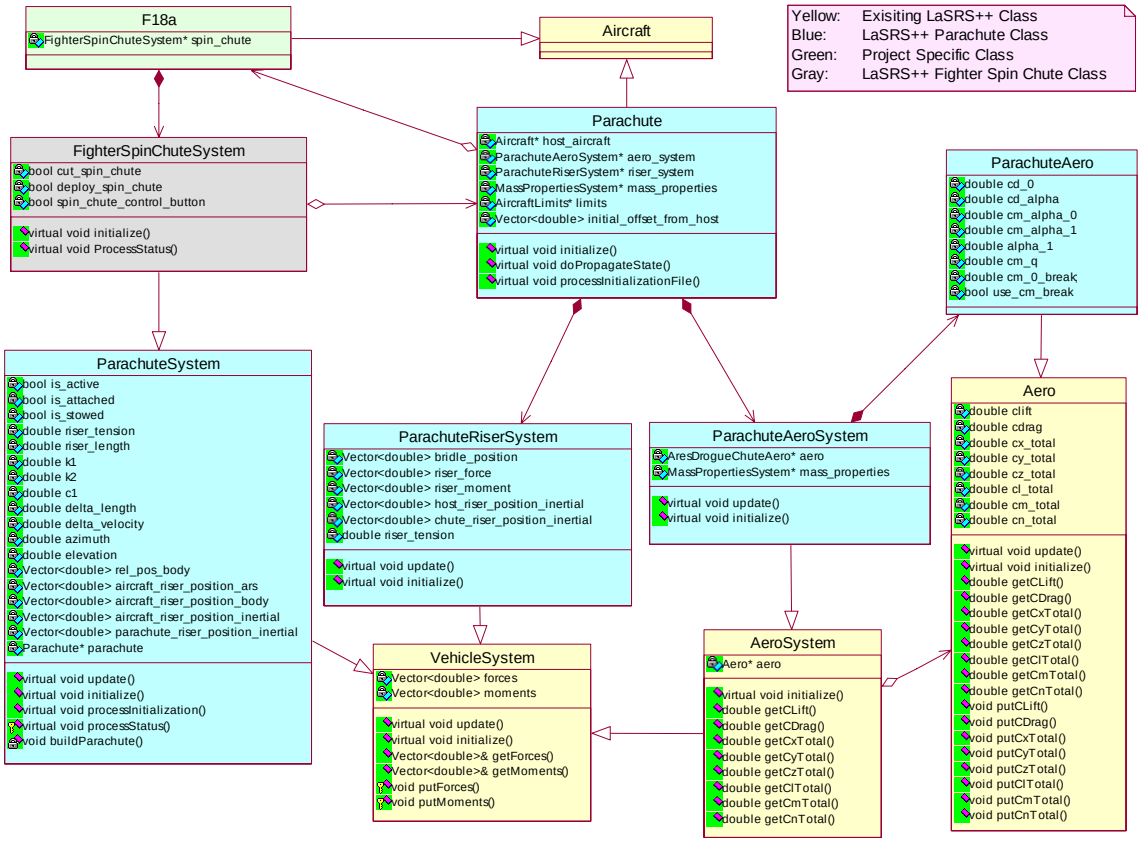

Figure 11. Fighter Spin Chute Hierarchy

IV. Future Plans

Future plans for the generic parachute model include adding additional aerodynamic options to allow for more flexibility, modeling the parachute opening forces, and adding porosity effects. Although this parachute model is simplified, LaSRS++ can very easily support higher fidelity models by replacing any of parachute's models.

The ARES aircraft model is currently being used in trade studies for a proposal to be submitted to the 2012 MarsScout opportunity. Enhancements to the ARES model and its systems may be discussed in a future paper.

The FighterSpinChuteSystem and spin chute configurations are currently undergoing testing. The model will be tested in FSSB's Differential Maneuvering System (DMS) cockpits and will be validated against results obtained from testing performed in NASA Langley's Spin Tunnel for the F18a and F16a spin chutes. Modeling of the parachute opening forces may be required for proper spin chute performance.

Additionally, an atmospheric re-entry body simulation is currently being planned that will also allow the parachute model to be validated against flight trajectories generated by other simulation tools and existing flight data. This will require the aerodynamic model to be enhanced to simulate the drag rise associated with transitioning through Mach 1.

V. Conclusion

The LaSRS++ design of a generic parachute model has proven to be both flexible and extendible. The design patterns used in the LaSRS++ framework allow a developer to maximize code reuse while concentrating on aircraft and mission specific features. The generic Parachute model can easily be extended with higher fidelity models if needed by future projects. The performance characteristics of the model are configured from parameters read from configuration files, which allows developers to customize the model without changing any code. The design promotes re-use by only requiring a new project to create a single new class that defines how a parachute is deployed and/or released. The LaSRS++ framework also allowed for the component models to be designed,

implemented, unit-tested, and integrated quickly, thereby facilitating the rapid prototyping required by most projects.

References

[1]Knacke, T.W., *Parachute Recovery Systems Design Manual*, 1st ed., Para Publishing, Santa Barbara, CA, 1992.

[2]Roskam, J., *Airplane Design: Part V: Component Weight Estimation*, 2nd ed., DARcorporation, Lawrence, KS, 1989, pp. 17-23.

[3]Whitlock, C.H., Bendura, R.J., and Coltrane, L.C., "Performance of a 26-Meter-Diameter Ringsail Parachute in a Simulated Martian Environment," NASA TM X-1356, 1967.

[4]Raiszadeh, B., and Queen, E.M., "Partial Validation of Multibody Program to Optimize Simulated Trajectories II (POST II) Parachute Simulation With Interacting Forces," NASA TM-2002-211634, 2002.

[5]Leslie,R.; et al. "LaSRS++: An Object-Oriented Framework for Real-Time Simulation of Aircraft," AIAA Modeling & Simulation Technologies Conference, Boston, August 1998, AIAA-98-4529.

[6]Madden, Michael M., and Neuhaus, Jason R., "A Design For Composing And Extending Vehicle Models," AIAA Modeling & Simulation Technologies Conference, Austin, August 2003, AIAA-2003-5458.

[7]Neuhaus, Jason R., "Modeling Mass Properties in an Object-Oriented Simulation," AIAA Modeling & Simulation Technologies Conference, Providence, Rhode Island, August 2004, AIAA-2004-5166.

[8]Kenney, P Sean, "Rapid Prototyping of an Aircraft Model in an Object-Oriented Simulation," AIAA Modeling & Simulation Technologies Conference, Austin, August 2003, AIAA-2003-5816

[9]Kenney, P Sean, "Simulating the ARES Aircraft in the Mars Environment," 2nd AIAA "Unmanned Unlimited" Systems, Technologies, and Operations -Aerospace, Land, and Sea Conference, San Diego, September 2003, AIAA-2003-6579