RAPID PROTOTYPING OF AN AIRCRAFT MODEL IN AN OBJECT-ORIENTED SIMULATION

P. Sean Kenney*

Systems Development Branch
NASA Langley Research Center
Mail Stop 125B
Hampton VA 23681

Abstract

A team was created to participate in the Mars Scout Opportunity. Trade studies determined that an aircraft provided the best opportunity to complete the science objectives of the team. A high fidelity six degree of freedom flight simulation was required to provide credible evidence that the aircraft design fulfilled mission objectives and to support the aircraft design process by providing performance evaluations. The team created the simulation using the Langley Standard Real-Time Simulation in C++ (LaSRS++) application framework. A rapid prototyping approach was necessary because the team had only three months to both develop the aircraft simulation model and evaluate aircraft performance as the design and mission parameters matured. The design of LaSRS++ enabled rapid-prototyping in several ways. First, the framework allowed component models to be designed, implemented, unit-tested, and integrated quickly. Next, the framework provides a highly reusable infrastructure that allowed developers to maximize code reuse while concentrating on aircraft and mission specific features. Finally, the framework reduces risk by providing reusable components that allow developers to build a quality product with a compressed testing cycle that relies heavily on unit testing of new components.

Introduction

NASA's Mars Scout Opportunity was created to enlist proposals for innovative investigations that complement NASA's core Mars Exploration Program. NASA Langley Research Center teamed with the Jet Propulsion Laboratory (JPL), NASA Goddard Research Center, Lockheed Martin Astronautics, Aurora Flight Sciences, Charles Stark Draper Laboratory, Malin Space Science Systems and several prominent academic researchers to participate in the opportunity. The team established science goals for the project that required a regional survey of Mars. Trade studies determined that an aircraft provided the best opportunity to complete the science objects of the team. This led the team to propose a project where an aircraft would be released into the atmosphere of Mars to perform an Aerial Regional-scale Environmental Survey (ARES).

To create a credible proposal, the team needed to provide a detailed aircraft design and demonstrate that it could complete the mission objectives. As the design evolved, aircraft capability needed to be evaluated. Specifically, could the aircraft pullout before striking the ground after being released from a spacecraft? Could the aircraft fly long enough to meet the science objectives? Could the aircraft provide a stable platform that would allow instrumentation to make usable measurements? A high fidelity six degree of freedom flight simulation was required to provide answers to these questions.

* Aerospace Engineer, Member AIAA.

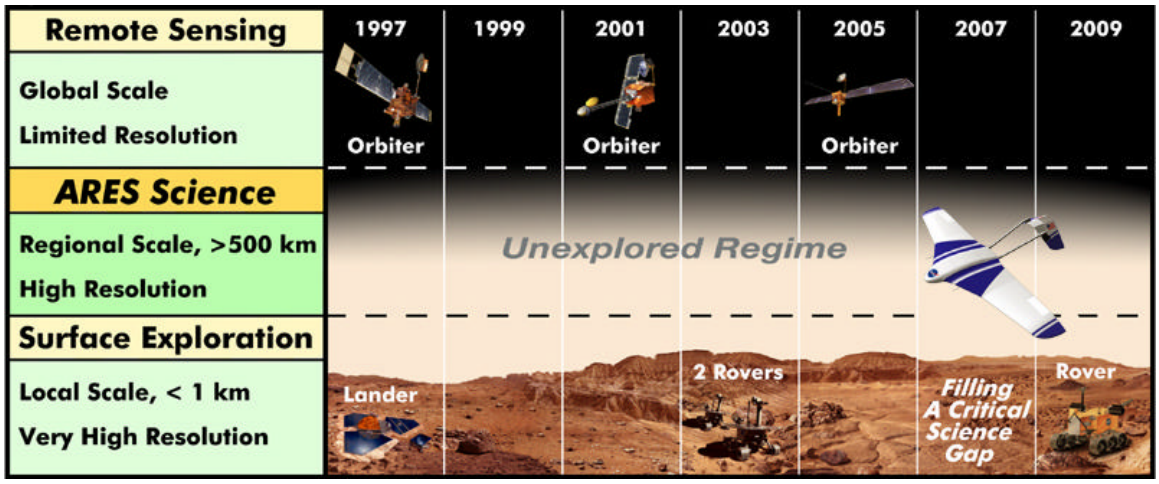

Figure 1. ARES Science related to Mars Exploration Program

Project Timeline

Due to the very nature of the Mars Scout Announcement of Opportunity, the ARES project had a very short timeline. The major milestones were as follows:

- September 2001 - Team members identified. Defined science objectives and perform trade studies of platforms to carry out mission.
- December 2001 - Aircraft selected as best platform to meet science objectives.
- March 2002 - Aircraft outer mold lines established.
- June 1 2002 – Design Freeze
- August 1 2002 - Proposal turned in.
- December 2002 – Four proposal winners announced.
- February 2003 – Revised aircraft outer mold lines defined.
- May 1 2002 – Design Freeze
- May 2003 – Phase A Concept Study Report turned in.
- August 2003 – Winning team announced.

It can be seen from this timeline that in both phases of development, the actual time available for simulation development and analysis was roughly three months.

The Langley Standard Real-Time Simulation in C++ (LaSRS++) application framework was used to support the aircraft design process. The framework allowed rapid prototyping of the aircraft as the design and mission parameters were modified.

Simulation Requirements

The primary science objectives defined by the science team were intended to bridge critical scale and resolution measurement gaps in the Mars Exploration Program (Figure 1). The objectives required the aircraft to autonomously fly a pre-planned aerial survey approximately 1.5 km above the surface of Mars in the southern highlands while carrying several scientific instruments (Figure 2).

To accomplish its mission objectives the aircraft must be inserted into the atmosphere of Mars. An aircraft design was selected that allowed the aircraft to be folded up and fit inside the aeroshell of an entry vehicle. The deployment sequence is illustrated in Figure 3. The sequence begins with the spacecraft releasing the entry vehicle into the atmosphere. The entry vehicle then deploys its parachute and begins to decelerate. The heatshield is released after the entry vehicle has slowed sufficiently. Shortly thereafter, the folded aircraft is released from the aeroshell. As the aircraft falls

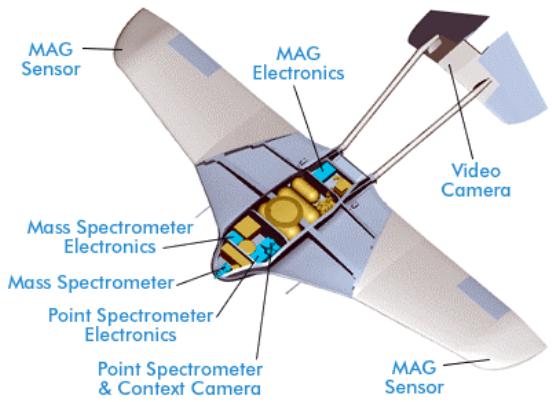

Figure 2. ARES Instrumentation

away from the entry vehicle, the tails and wings unfold and the aircraft begins its flight. Its first task is to arrest its descent and to pullout into horizontal flight. Once the pullout is complete the aircraft can begin its pre-planned aerial survey.

The LaSRS++ simulation framework was selected to evaluate the flight of the aircraft starting from where the aircraft was fully deployed, i.e. free from the entry vehicle and unfolded. Other simulation tools were selected to model the aircraft/entry body dynamics and aircraft unfolding. The LaSRS++ based simulation was therefore required to initialize the aircraft from the outputs of the other simulation tools. Then the LaSRS++ based simulation would perform the pullout maneuver and then navigate following the planned flight profile. These requirements necessitated the development of two items: a Mars environment and an ARES aircraft model.

LaSRS++ supports different world models and required little modification to add the Mars environment[1]. Classes were developed to model the Martian atmosphere, winds, and surface topology. These classes were implementations of the existing LaSRS++ design patterns for environmental models. The new classes were designed to use the data and equations found in the Mars-GRAM 2001 distribution and were unit-tested against the Mars-GRAM application.

This paper focuses on the rapid-prototyping of the ARES aircraft and will not go into further detail on the Mars model.

The ARES aircraft model was initially constructed with a minimal set of components. Aerodynamic, propulsion, and control system components were leveraged from other simulation projects and configured to ARES specifications. Reused simulation components were substituted with ARES specific models as the ARES models became available. Later, each ARES model would be replaced with progressively more detailed models. This allowed performance evaluations to be performed as the design was maturing.

Aircraft Systems

The LaSRS++ simulation framework was designed to encourage software reuse. It provides a large number of generic components that developers can leverage when building a new simulation model. Figure 4 is a Unified Modeling Language (UML) diagram that illustrates a simplified depiction of a typical aircraft model in LaSRS++ [5]. In this diagram, the Ares class derives from the class Aircraft. The Ares class is a composite class that aggregates a number of 'system' classes like AresAeroSystem, and AresPropulsionSystem. Each system class contains a subsystem model. The system classes are examples of the mediator pattern [3,4]. The system classes decouple the subsystem models from other subsystem models and the composite aircraft model.

Figure 5 is a UML diagram that provides a detailed look at the Ares aerodynamic components. The Ares class contains an AresAeroSystem which in turn contains an AresAero object. The AresAeroSystem class provides AresAero with all of its inputs and instructs the class when to perform its aerodynamic computations. The AresAero class is responsible for computing the aerodynamic coefficients and the total aerodynamic forces and moments of the aircraft when its *update*() method is called. The class is not dependent upon other simulation models and may therefore be unit-tested prior to integration into the simulation framework. The AresAero class is broken into longitudinal and lateral components. Each of these component classes is also unit-testable.

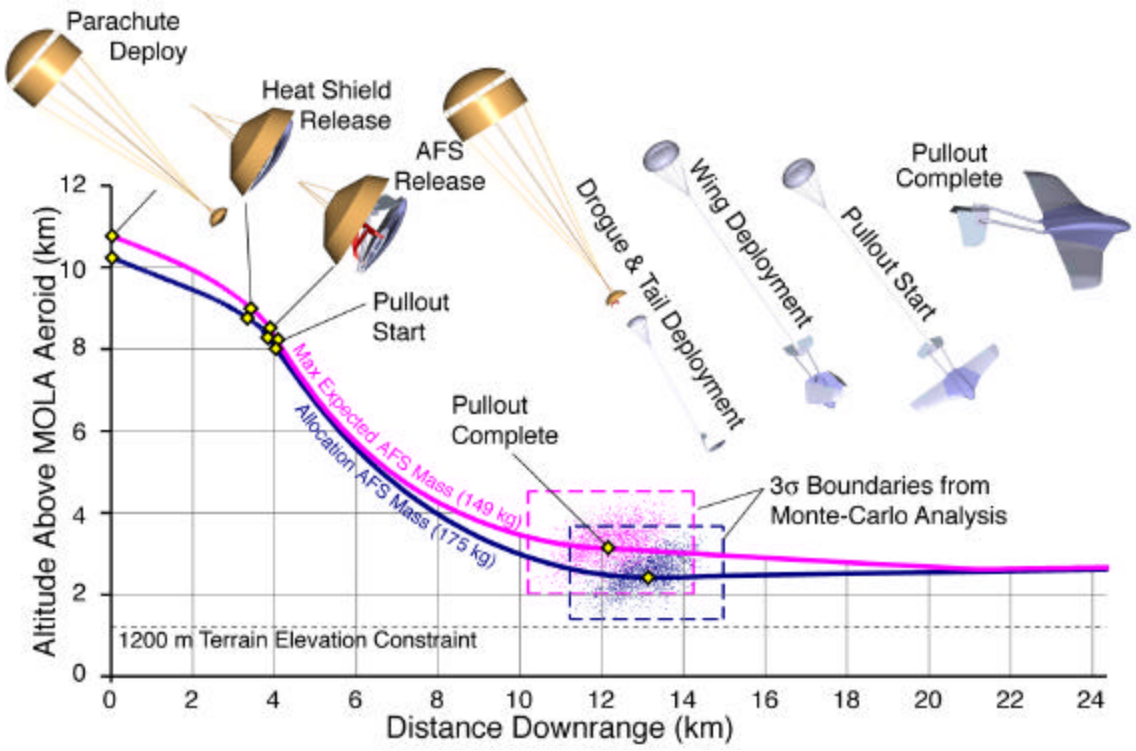

Figure 3. ARES Deployment Sequence

A typical frame of computation begins with the aircraft gathering external inputs and then instructing all of its system classes to update themselves. Next the aircraft sums the forces and moments computed by the system classes and then integrates its states. Figure 6 demonstrates a sequence diagram for the Ares aero classes. The diagram shows that when the Ares aircraft requests the AresAeroSystem to update, AresAeroSystem gets state parameters from the aircraft, provides them to the AresAero object, and then calls its update method. The AresAero class then uses its components to compute aerodynamic coefficients and the total aerodynamic forces and moments acting on the aircraft. The aircraft then requests the newly computed force and moment vectors prior to aircraft state integration.

Rapid Prototyping of the Aerodynamic Model

When the first outer mold lines were established there was no data available with which to build an aerodynamic model. The only data was estimated design parameters established during the selection of the aircraft's shape. While several computational tools were generating aerodynamic data, the simulation needed some type of aerodynamic model to begin performance studies of the aircraft in different atmospheric effects (winds, turbulence). With this in mind, an aerodynamic model from a different aircraft was used to simulate the ARES aircraft. A general aviation linearized aerodynamic model was selected, and its parameters were set to ARES-like properties. The linearized model consists of two components, a longitudinal model and a lateral model. Both models used constant aerodynamic coefficients to build up the total lift and drag coefficients. Although this was a very simple aerodynamic model, it allowed the aircraft to be used for initial mission analysis.

As the computational tools began to produce results, the linearized aerodynamic models were refined to use the new data. Changes to the parameters of the linearized models only required a few minutes and thereby allowed the modifications to be evaluated very quickly. Eventually one of the more complex tools had produced enough data to create an aerodynamic coefficient database. The longitudinal linearized model was then replaced

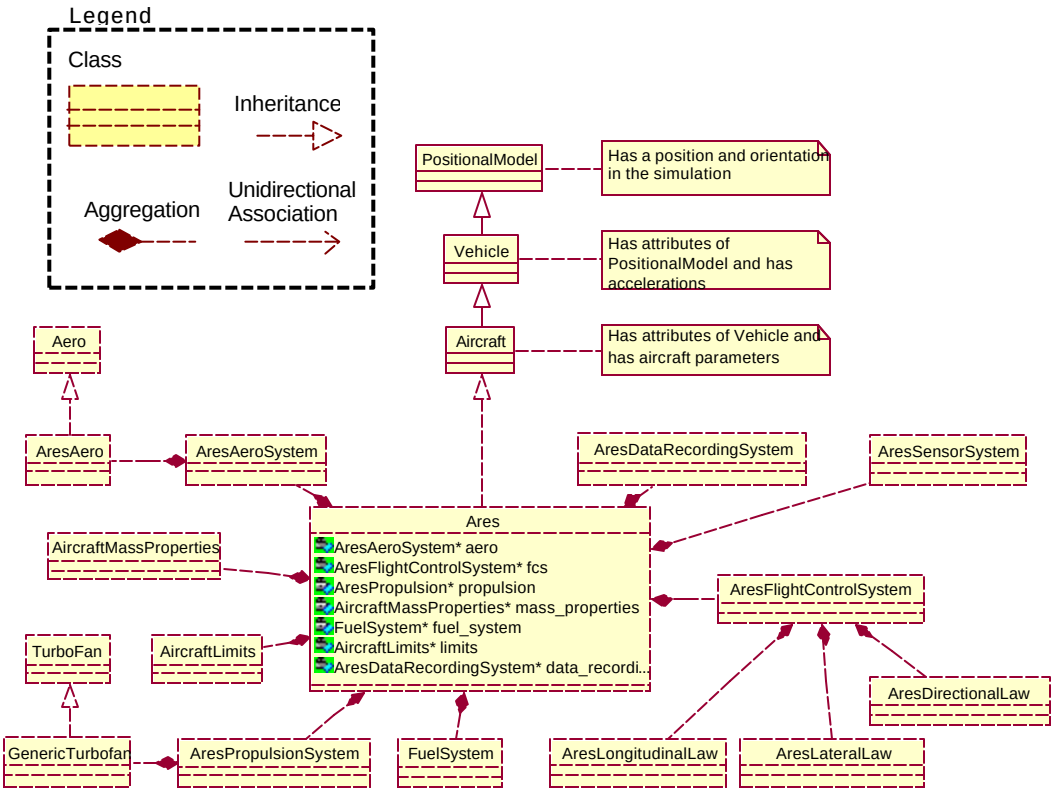

Figure 4. ARES Components

with a new model using four-dimensional lookups of the aerodynamic coefficients. Because the longitudinal and lateral models were contained in the AresAero class, it was the only class modified to accommodate the changes. This allowed the simulation to use the new model within minutes of when the model had completed unit-testing. Shortly thereafter, the linerized lateral aerodynamic model was also replaced with a model using two-dimensional lookups.

As more data became available, the longitudinal and lateral models were continually updated. Many different aerodynamic databases were evaluated to study the effects of different types of control surfaces, the placement and sizes of control surfaces, the use of a drogue chute during pullout, and other aircraft design options. By the end of the first proposal the integration process only needed thirty minutes to take a new set of aerodynamic data tables, integrate the new data into the aero classes, unit-test, and begin evaluation of the aircraft's performance with the modifications.

Rapid Prototyping of Flight Control System

The flight control system began in much the same way as the aero system. Initially the control system was leveraged from the general aviation project for two reasons. First, the control laws were very simple and allowed for easy modification. Secondly, the laws matched the initial aerodynamic database and provided a stable platform. As the aerodynamic parameters were modified to reflect ARES performance, the gains of the control laws were tweaked to maintain a stable aircraft.

A high level view of the control system can be seen in Figure 4. The AresFlightControlSystem is composed of three distinct laws, a longitudinal, a lateral and a directional. As the project controls group created control laws for the ARES aircraft, the general aviation laws were replaced. The longitudinal law was designed as an angle-of-attack command system, and the lateral law was designed as a bank angle command system.

In order to begin detailed mission analysis, the aircraft needed additional navigational capability.

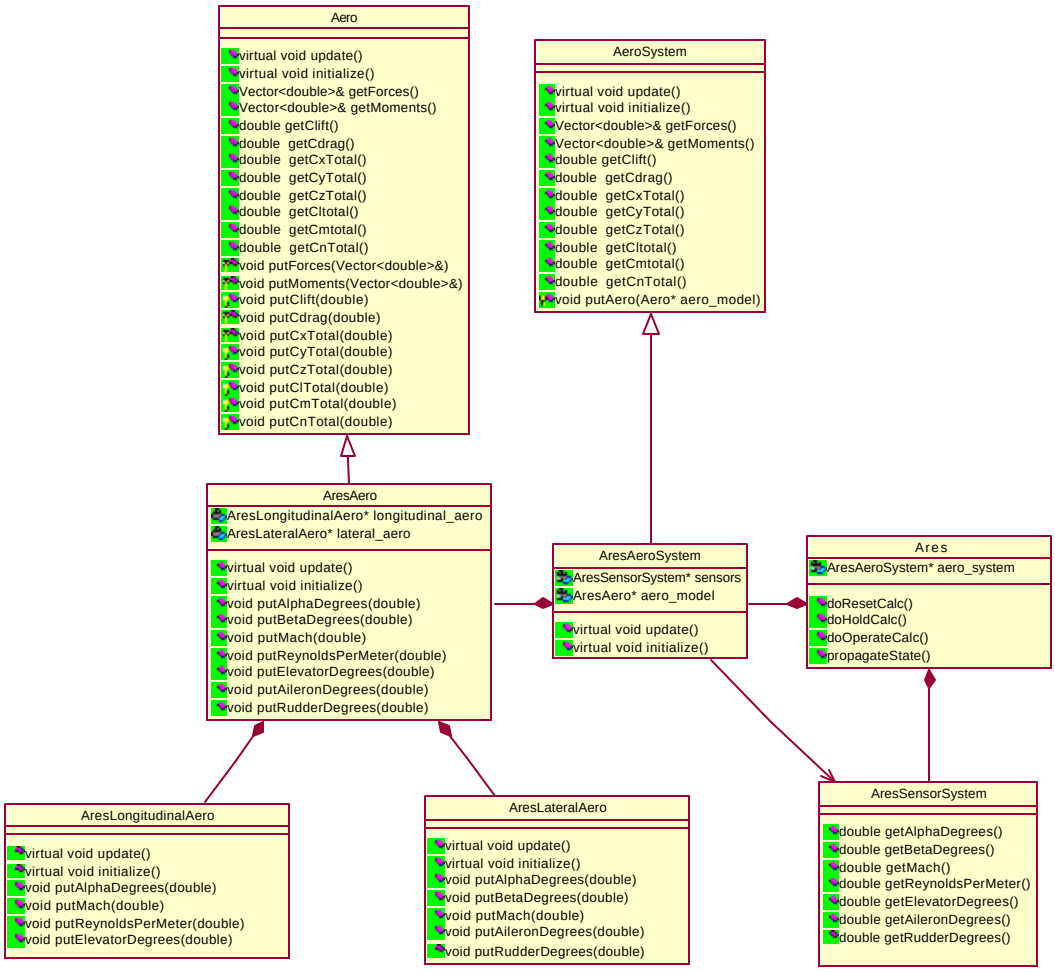

Figure 5. ARES Aero Components

With that in mind, two autopilot modes were leveraged from the GenericTransport aircraft project. An altitude-hold mode was incorporated into the flight control system to allow the aircraft to maintain a constant altitude above the terrain. While the original altitude hold code was based on an N_z(vertical acceleration) command control system, it was easily modified to work with an angle-of-attack command system. Most of the changes involved setting the gains of the altitude-hold mode to work well with the Ares aircraft.

A track hold mode was also incorporated into the flight control system. This mode allowed the aircraft to follow a pre-determined path via a sequence of heading commands. The incorporation of the two autopilot modes allowed mission profiles to be analyzed for both different aircraft configurations and environmental conditions.

In the second phase of the proposal a second longitudinal law was implemented using an Nz command system. The flight control system was modified to be able to switch between the two control laws. The new law was created to evaluate the performance of the aircraft under a different control algorithm.

Rapid Prototyping of Other Models

Several other components were required to simulate ARES. The propulsion system initially used a GenericTurboFan object to model a simple linear thruster with lag. Initially the thruster could be throttled to allow the aircraft to be placed in a

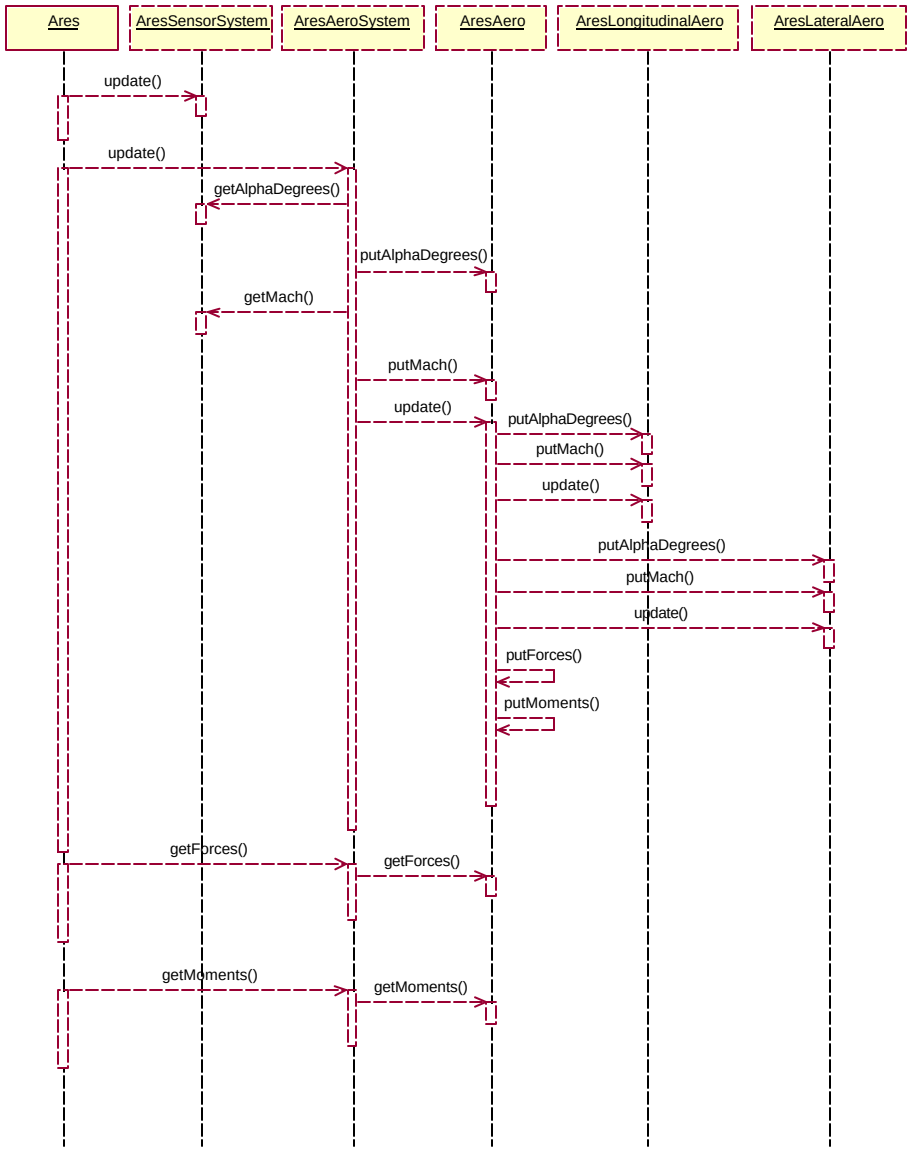

Figure 6. ARES Aero Sequence Diagram

trimmed state. Autothrottle logic was incorporated to hold the aircraft's speed while flying a mission profile. As the design of ARES progressed, a particular thruster was selected, and the thruster modeled in the simulation was modified to reflect its performance. The selected thruster cannot be throttled meaning that it is commanded either on or off. The autothrottle law was also modified to accommodate the pulsing mode of the thruster.

The mass properties system, the fuel system, aircraft limits, and the sensor system were either reused framework components or contained framework components. These systems required virtually no modifications other than setting the parameters associated with the Ares aircraft. Because all of these components have been heavily reused for many years, they required no testing beyond confirmation of correct initialization.

Reuse Metrics

A common problem associated with a rapid prototyping project is that there is an increased risk of coding errors due to developing at such a fast pace. Software reuse can mitigate this risk by

Reuse Metrics of ARES Aircraft Project								
Aircraft Name	Model File Count	LaSRS++ File Count	Model LOC	LaSRS++ LOC	Model Class Count	LaSRS++ Class Count	Amount of Reuse (AOR)	External Reuse Level (ERL)
Ares Phase 1	53	2208	6197 (750)	262859	25	1046	97.7%	.97
Ares Phase 2	83	2215	8862 (1488)	257952	40	1051	96.7%	.96

Table 1. ARES Reuse Metrics

reducing the amount of code that needs to be tested.

There are many different ways to categorize reuse. This study will use the dependency chain estimation method described by Madden [2]. The dependency chain estimation method identifies all of the source files that are required to build a simulation containing an aircraft model. The method provides an upper bound on the amount of reuse because it falsely counts some components as reused even though they are conditionally created. However, for LaSRS++ simulations the dependency chain method was found to be a simple and accurate assessment of reuse.

Table 1 illustrates the file counts, lines of code (LOC), number of classes (NOC) and reuse statistics for the Ares aircraft at the end of each development phase. The two reuse metrics presented are the amount of reuse (AOR) metric and the external reuse level (ERL) metric. The amount of reuse value is defined as the ratio of LOC reused over the total LOC of the aircraft simulation.

$$AoR = \frac{LOC_{framework}}{\left(LOC_{framework} + LOC_{aircraft}\right)}$$

Similarly, the external reuse level is defined as the ratio NOC reused over the total NOC of the aircraft simulation.

$$ERL = \frac{NOC_{framework}}{\left(NOC_{framework} + NOC_{aircraft}\right)}$$

As the table illustrates, the Ares aircraft exhibited very high levels of reuse. The framework statistics only include reused LaSRS++ code, not code from other aircraft. Considering that some of the autopilot features found in the flight control system were leveraged from a different aircraft model and copied into Ares source code, it could be argued that the values might be even higher.

Because the project was able to attain such high levels of reuse, team members were able to rely heavily on unit testing of new components. LaSRS++ components have been through more than 180 testing cycles and are considered very mature. The few defects found were due to integration errors. No defects were discovered in unit-testable components.

Concluding Remarks

The design of LaSRS++ proved to be both flexible and extendible. The design patterns used in the framework allows a developer to maximize code reuse while concentrating on aircraft and mission specific features. The high levels of re-use attained by this project also helped mitigate the risk associated with rapid prototyping. Achieving 96% AoR and ERL reduced the testing requirements of the ARES simulation without compromising quality. The framework also allowed for component models to be designed, implemented, unit-tested, and integrated quickly, thereby facilitating the rapid prototyping required by this project. Aerodynamic data modifications were incorporated and tested in less than 30 minutes. As a result of this, the simulation was used to asses the capabilities of ARES as its design evolved over a

three month period. The final proposals included credible pullout, time-of-flight, range, and stability results generated by the simulation. The ARES team was selected as one of the four first round proposal winners and hopes to win the final round of the design competition.

Bibliography

[1] Richard A. Leslie, et al. *LaSRS++ An Object-Oriented Framework for Real-Time Simulation of Aircraft*. Modeling & Simulation Technologies Conference, Paper Number AIAA-98-4529, August, 1998.

[2] M. Madden, *Examining Reuse In LaSRS++ Based Projects*, Modeling & Simulation Technologies Conference, Paper Number AIAA-2001-4119, August, 2001.

[3] Gamma E., Helm R., Johnson R., Vlissides J. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading, Massachusetts, 1995.

[4] Cunningham, K. *Use of the Mediator Design Pattern in the LaSRS++ Framework*. AIAA Modeling & Simulation Technologies Conference, Paper Number AIAA-99-4336, August, 1999.

[5] Douglas, B. *Real-Time UML*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1999.