Integrated System Modeling for Nuclear Thermal Propulsion (NTP)

Stephen W. Ryan, Stanley K. Borowski
NASA Glenn Research Center, Cleveland, OH, 44135

Nuclear thermal propulsion (NTP) has long been identified as a key enabling technology for space exploration beyond LEO. From Wernher Von Braun's early concepts for crewed missions to the Moon and Mars to the current Mars Design Reference Architecture (DRA) 5.0 and recent lunar and asteroid mission studies, the high thrust and specific impulse of NTP opens up possibilities such as reusability that are just not feasible with competing approaches. Although NTP technology was proven in the Rover / NERVA projects in the early days of the space program, an integrated spacecraft using NTP has never been developed. Such a spacecraft presents a challenging multidisciplinary systems integration problem. The disciplines that must come together include not only nuclear propulsion and power, but also thermal management, power, structures, orbital dynamics, etc. Some of this integration logic was incorporated into a vehicle sizing code developed at NASA's Glenn Research Center (GRC) in the early 1990s called MOMMA, and later into an Excel-based tool called SIZER. Recently, a team at GRC has developed an open source framework for solving Multidisciplinary Design, Analysis and Optimization (MDAO) problems called OpenMDAO. A modeling approach is presented that builds on previous work in NTP vehicle sizing and mission analysis by making use of the OpenMDAO framework to enable modular and reconfigurable representations of various NTP vehicle configurations and mission scenarios. This approach is currently applied to vehicle sizing, but is extensible to optimization of vehicle and mission designs. The key features of the code will be discussed and examples of NTP transfer vehicles and candidate missions will be presented.

Nomenclature

ASV	=	Asteroid Survey Vehicle
CEV	=	Crew Exploration Vehicle
DRA	=	Design Reference Architecture
FTD	=	Flight Technology Demonstrator
IMLEO	=	Initial Mass in Low Earth Orbit
Isp	=	Specific Impulse (seconds)
K	=	temperature (degrees Kelvin)
klb_f	=	thrust (1000's of pounds force)
LEO	=	Low Earth Orbit (= 407 km circular)
LH_2	=	Liquid Hydrogen
MET	=	Mission Elapsed Time
NERVA	=	Nuclear Engine for Rocket Vehicle Applications
NTP/NTR	=	Nuclear Thermal Propulsion/Nuclear Thermal Rocket
RCS	=	Reaction Control System
SLS	=	Space Launch System
t	=	metric ton (1 t = 1000 kg)
ΔV	=	velocity change increment (km/s)

I. Introduction and Background

IN recent work, a number of scenarios for NTP enabled missions beyond LEO have been presented. Among these are the Mars Design Reference Architecture (DRA 5.0), crewed missions to a near Earth asteroid (NEA), the Earth-Moon Lagrange point (EML2) and lunar cargo and crew missions. [1, 2, 3, 4, 5, 6]

---

[1]LSC Branch, 21000 Brookpark Road, MS: 86-1, AIAA Member
[2]LTR Branch and Technical Lead, NTP Systems, 21000 Brookpark Road, MS: 86-4, AIAA Associate Fellow

In the early 1990s, a Fortran code was developed at NASA's then Lewis Research Center (GRC) to provide mass estimation for NTP vehicles. The code (designated MOMMA by its author, for Mark's Optimizing Mission Mass Allocator) organizes a mission as a series of phases where each phase has a duration and potentially a ΔV and/or a change in mass due to jettison or pick up of payload. The initial mass is provided as a guess based on vehicle structural and payload requirements and an estimated fuel load. Sizing rules are used for fuel tanks, cooling and radiators. The program then solves the rocket equation for each of the ΔV maneuvers in sequence to determine a final mass. Hydrogen fuel boil off and jettison/pickup are also accounted for at the appropriate times in the sequence. The mission is repeated by iterating the initial guess until the final fuel load is zero, thus minimizing the vehicle mass. See Figure 1 for a flowchart of the MOMMA code.

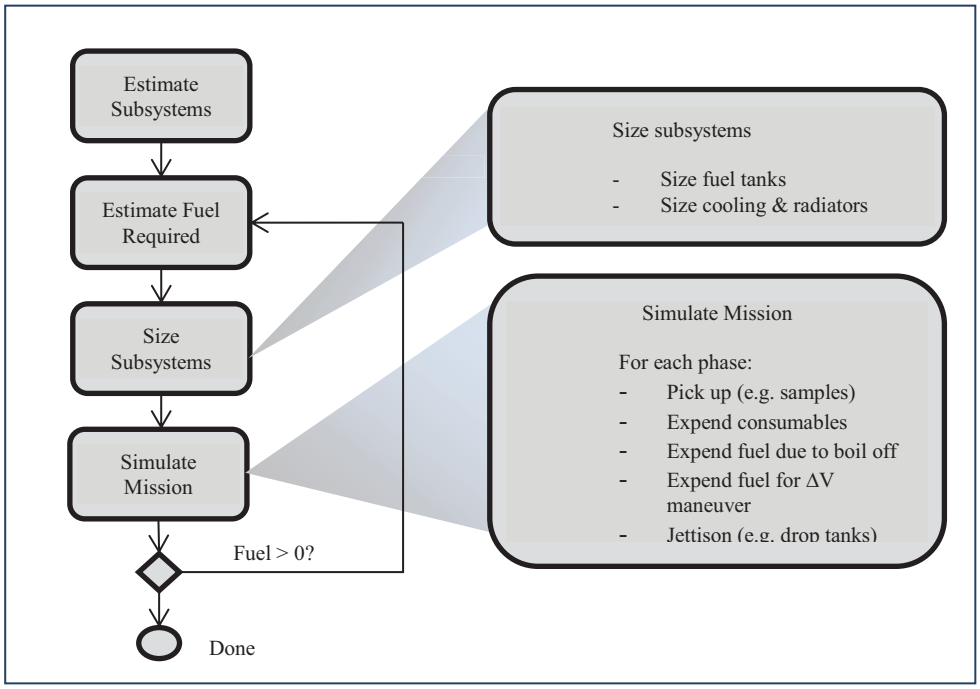

Figure 1 - MOMMA Flowchart

OpenMDAO is a software framework under development primarily at NASA GRC. It was conceived under the Fundamental Aeronautics Program as a tool for the application of state of the art computational techniques for the design of unconventional aircraft, but is architected such that it is applicable to the wide range of MDAO problems. [7, 8] The key features of OpenMDAO are that it is a) open source b) object oriented and c) scriptable. The virtue of being open source is that it is freely available and extensible. One can easily modify or extend aspects of the framework to meet the needs of the problem. Contributions and plug-in extensions are also accepted from the user community to further enhance its capabilities. Being object oriented further strengthens the extensibility of the framework by enabling drop-in replacement for key modules as well as providing a flexible motif for model development. Black box encapsulation and clearly defined interfaces are key to identifying the interactions between subsystems and controlling complexity. Finally, the fact that OpenMDAO is implemented on top of the powerful scripting language Python means not only that it is relatively quick and easy to develop with, but also that it is easy to read, which is important for ensuring transparency and traceability in a model. Python is a popular tool for scientific and engineering applications and is supported by a community that contributes powerful extensions like NumPy and SciPy and a myriad of other open source modules. [9, 10, 11] Together, these attributes make OpenMDAO an excellent platform on which to build a multidisciplinary system model.

The goal of this work is to build upon the mission mass optimization work done previously and to leverage the power of the OpenMDAO framework to create a flexible and powerful tool for integrated system modeling of NTP spacecraft and missions beyond low Earth orbit.

II. NTP Vehicle Configurations

For human missions beyond low Earth orbit, it is important to minimize flight time in order to protect the health of the crew against the long term effects of weightlessness as well as the radiation environment of deep space. NASA's Mars Design Reference Architecture (DRA) 5.0 study in 2009 evaluated multiple approaches to a manned mission to the Mars surface and concluded that the optimal strategy in terms of a number of figures of merit including crew safety, scientific return and affordability is a series of "fast-conjunction" mission in which the transit times to and from Mars are minimized in exchange for long surface stay times. [1] The high thrust and a high specific impulse of NTP along with its proven technology, flexibility and growth potential, made it the propulsion system of choice for Mars DRA 5.0 and similarly make it well suited to a variety of missions from cis-lunar to asteroid rendezvous.

The Mars DRA 5.0 mission is built on two vehicle configurations. The first, a cargo vehicle, is sent to Mars ahead of the crew so as to provide a ready supply of provisions when they arrive. Two of these are employed to deliver supplies to the Mars surface and a lander vehicle to Mars orbit. The cargo configuration is composed of a core propulsion stage with a cluster of three 25klbf "Pewee" class engines, an in-line liquid hydrogen (LH_2) fuel tank and the aero-braked cargo module. The second configuration that carries the crew is based on the same core propulsion stage as the cargo vehicle but has a larger in-line LH_2 tank, an additional drop tank, a habitat module with power and life support systems, and an Orion crew exploration vehicle (CEV). The cargo and crewed vehicle configurations are both shown in Figure 2. This architecture is referred to as the "9-launch" architecture based on the number of heavy lift launches required to deliver the components to LEO. Subsequent refinement has yielded a "7-launch" architecture, eliminating the in-line LH_2 tank. [2] In this scenario, the cargo vehicles are delivered in two launches each (one for the core NTP stage and one for the cargo), while the crewed vehicle requires three launches (one each for the core stage, the saddle truss and drop tank assembly, and the crewed payload). An illustration of these configurations from the referenced report is shown in Figure 3. Variations on these basic configurations have been envisioned for lunar and asteroid missions as well. [4, 5, 6]

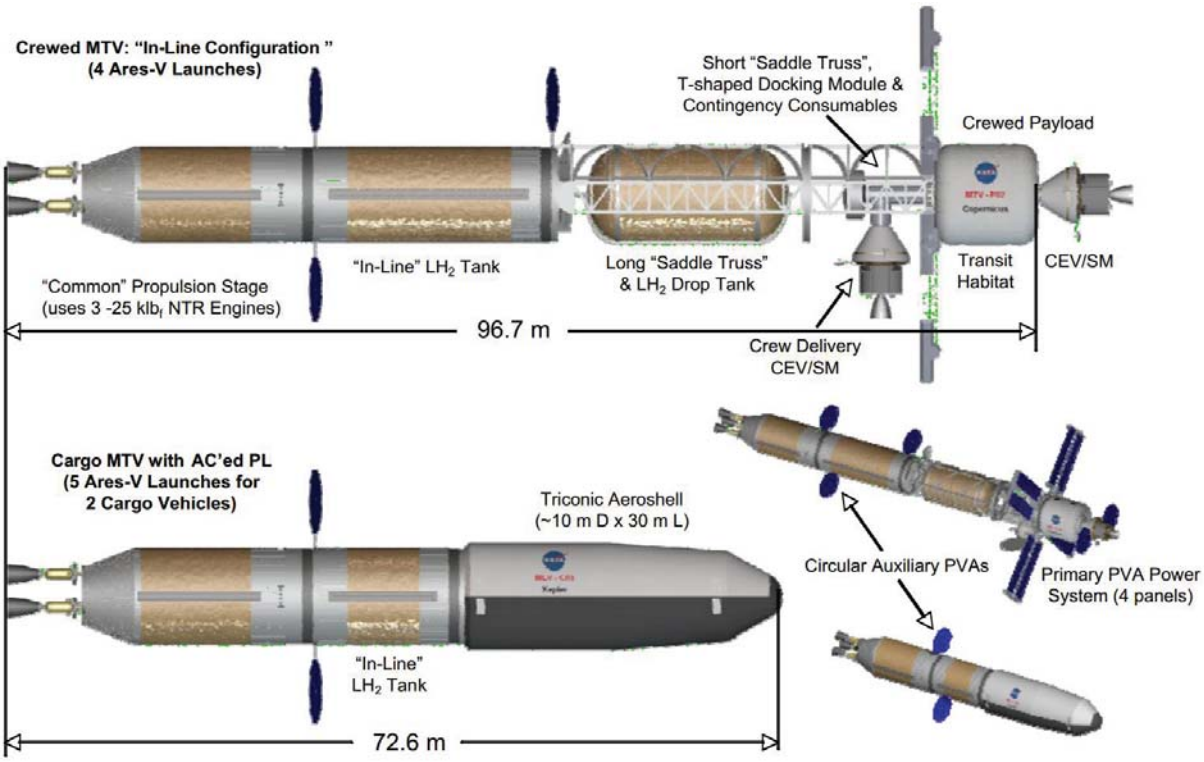

Figure 2 - Mars DRA 5.0 NTP Vehicle Configurations

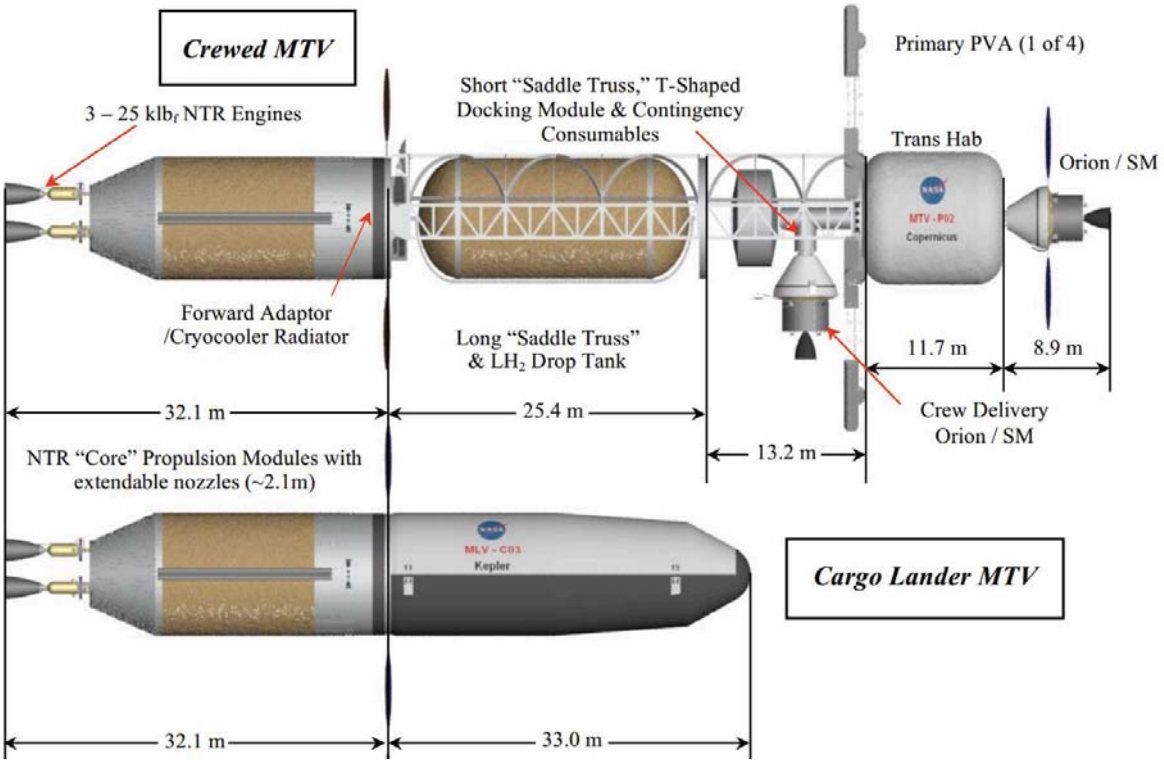

Figure 3 - "7-Launch" Mars NTP Vehicle Configurations

In Reference 4, the crewed MTV vehicle, called "Copernicus", is adapted to function as a standalone Asteroid Survey Vehicle (ASV) for a crewed expedition to a near Earth asteroid (NEA). For this configuration, dubbed "Searcher", a small 2-person pressurized excursion vehicle (MMSEV) is attached to the crewed payload element in place of the contingency consumables canister in the Mars configuration. This excursion vehicle is used to approach the asteroid and do proximity operations such as sample collection or even extravehicular activity (EVA). Variations of the Searcher configuration were explored, starting with "Search Lite", a fully reusable vehicle scaled down with 16 klb_f engines and 7.6m diameter fuel tanks used for the low energy mission to the asteroid 2000 SG344. Expendable and reusable options capable either of high energy missions to the asteroid Apophis or 1991 JW were enabled using the baseline 25 klb_f engines and 8.4 m or 10 m diameter tanks, and re-introducing the in-line tank from Mars DRA 5.0. Finally, a small robotic flight technology demonstrator (FTD) spacecraft is envisioned based on the Delta Cryogenic Second Stage (DCSS) but with an ~7.5 klb_f NTP engine. Such a spacecraft could deliver a science payload to an asteroid such as 2000 SG344 while serving as a stepping stone to development of the full scale NTP vehicles.

In Reference 5, variations of the NTP vehicle are explored to enable fast-transit Mars missions and reusability by adding a fourth engine and/or using oxygen "afterburner" nozzles to the basic NTR engine for augmented thrust. A 4-sided "Star Truss" design is also introduced; replacing the single drop tank with four laterally mounted tanks for more fuel capacity (see Figure 4). Configurations were also developed for reusable lunar cargo and crew transport missions. The crewed lunar transfer vehicle carries a lunar lander and Orion capsule in place of the short saddle truss and TransHab module. The lander is attached via a common docking element at the front of the long saddle truss that allows the Orion capsule to dock with it once the drop tank has been jettisoned, after which they can transfer to the lander and descend to the lunar surface.

Also in Reference 5, some short round trip/short stay Mars mission options are outlined, including the innovative approach of swapping the crewed payload element over to a previously positioned propulsion and drop stage for the return trip. This mission would not land on Mars, but would enable a survey from orbit for up to 60 days.

Table 1 provides a summary of some of the various vehicle configurations under consideration. Abbreviations in the table include STA for Short Truss Assembly, THAB for TransHab, TDM for T-Docking Module, LDAV for Lunar Descent/Ascent Vehicle and MMSEV for Multi-Mission Space Excursion Vehicle.

Table 1 - Spacecraft Configurations

Vehicle Configuration	Diameter	Engines	Inline	Drop	Payload
7 launch DRA5 Cargo	10m	3x25k @ 900s	no	no	cargo lander/habitat module
7 launch DRA5 Crew	10m	3x25k @ 900s	no	saddle	STA, THAB, TDM, Canister
Mars Split Mission (ERV)	10m	3x25k @ 906s	no	saddle	Earth Return Vehicle
Mars Split Mission (MSV)	10m	3x25k @ 906s	no	saddle	STA, THAB, TDM, Canister
Mars "All Up"	10m	3x25k @ 906s	yes	saddle	STA, THAB, TDM, Canister
Reusable DRA5 #1	10m	3x25k LANTR @ 726/920s	yes	saddle	STA, THAB, TDM, Canister
Reusable DRA5 #2	10m	4x25k @ 906s	yes	star with 4 tanks	STA, THAB, TDM, Canister
Fast Transit	10m	4x25k @ 906s	stretched	star with 2 tanks	STA, THAB, TDM, Canister
ASV1 Reusable SG344 "Search Lite #1"	7.6m	3x15k @ 906s	no	saddle	STA, THAB, TDM, MMSEV, 4 crew
ASV2 Reusable SG344 "Search Lite #2"	8.4m	3x25k @ 906s	no	saddle	STA, THAB, TDM, MMSEV, 4 crew
ASV3 Reusable SG344	8.4m	3x25k @ 906s	no	saddle	STA, THAB, TDM, MMSEV, 6 crew
ASV4 Expendable Apophis	8.4m	3x25k @ 906s	no	saddle	STA, THAB, TDM, MMSEV, 4 crew
ASV5 Reusable Apophis	8.4m	3x25k @ 906s	yes	saddle	STA, THAB, TDM, MMSEV, 4 crew
ASV6 Reusable Apophis "Searcher"	10m	3x25k @ 906s	no	saddle	STA, THAB, TDM, MMSEV, 6 crew
Robotic FTD, 2000 SG344	4.8m	1x7k @ 905s	no	no	Science
NTP Lunar transport Class-I	7.6m	3x15k @ 906s	yes	no	STA, LDAV or THAB, MMSEV
NTP Lunar transport Class-II	8.4m	3x25k @ 906s	yes	no	STA, LDAV or THAB, MMSEV
Prospector EML2	7.6m	3x16.7k @ 900s	yes	no	THAB, MMSEV, lifting body
Commercial CPT	7.6m	3x16.7k @ 900s	yes	no	THAB, lifting body
Reusable Lunar Cargo	7.6m	3x16.7k @ 900s	no	saddle, docking port	lunar habitat module
Reusable Lunar Crew	7.6m	3x16.7k @ 900s	no	saddle, docking port	lunar lander & Orion
Reusable Lunar Cargo	8.4m	3x25k @ 900s	no	saddle, docking port	2x lunar habitat modules
Reusable Lunar Crew	8.4m	3x25k @ 900s	no	saddle, docking port	lunar lander & Orion

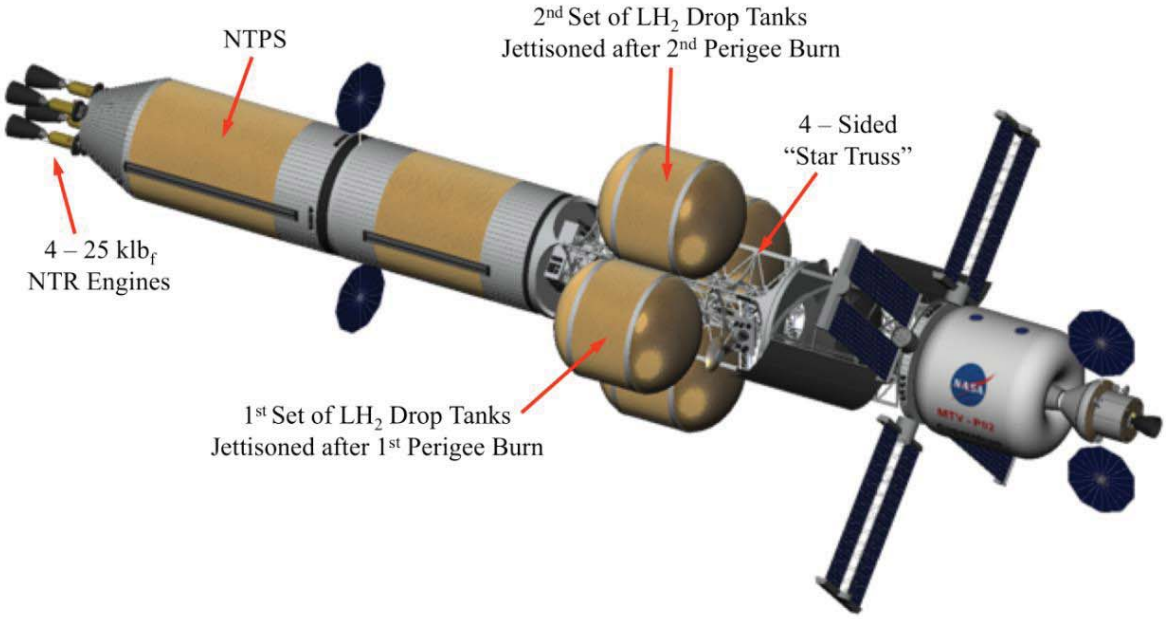

Figure 4 – Four Engine Configuration with Extra Fuel Tanks

III. Mission Descriptions and ΔV Budgets

As discussed in the previous section, a variety of NTP vehicle configurations have been described, customized for different missions to the Moon, cis-lunar space, NEAs, Mars and its moons. In this section, we provide a brief summary of the mission profiles for these missions for the purpose of sizing the spacecraft and validating the feasibility of the design for accomplishing the mission.

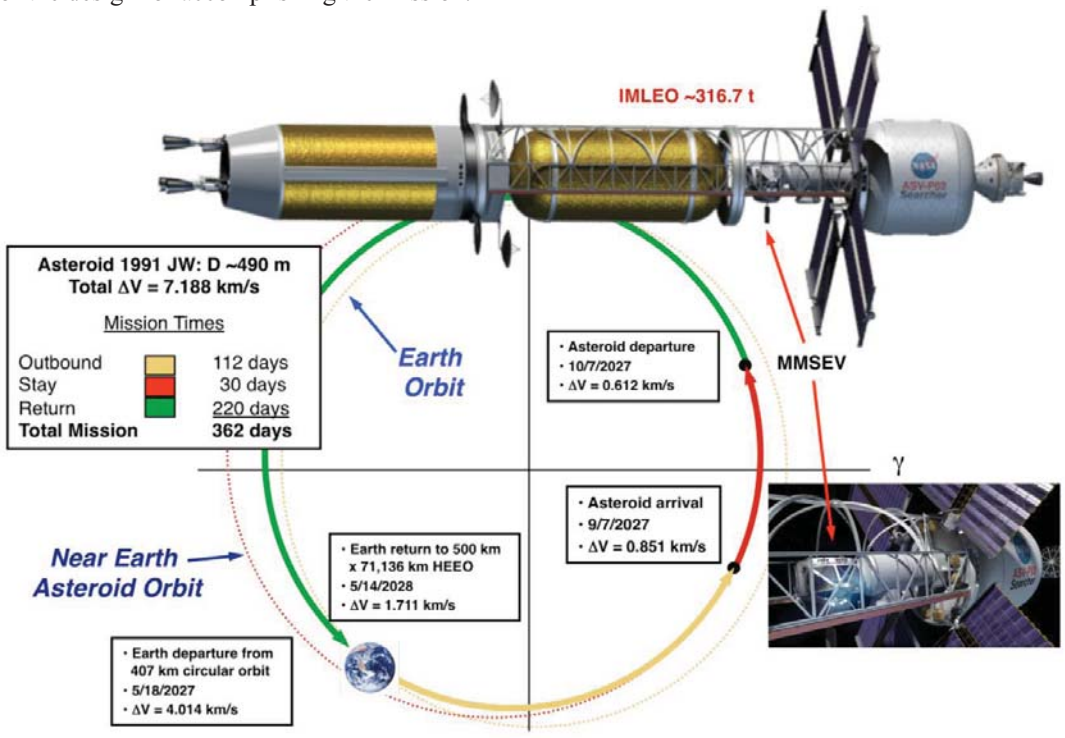

Figure 5 - Trajectory Details for Mission to 1991 JW

The most significant factor in sizing the vehicle for missions such as these is always the amount of fuel required; therefore the missions are described in terms of the major maneuvers required of the main propulsion system. These are described in terms of the change in velocity (ΔV) effected by the maneuver. Given this ΔV and the mass of the spacecraft at the time of the maneuver, the fuel burn can be calculated via the rocket equation. Figure 5 is an illustration of the major maneuvers for one such mission, to the asteroid 1991 JW.

See Table 2 for a sampling of some of the missions referenced in the previous section in terms of their major ΔV maneuvers. The TDI and DOC columns indicate the ΔV required for the trans-destination injection maneuver and destination orbit capture, respectively. The TEI and EOC columns similarly indicate the ΔV required for the trans-Earth injection and Earth orbit capture maneuvers. In some cases, the TDI maneuver is split into two burns. Note that the TEI and EOC maneuvers are not required for robotic science missions, and the EOC maneuver is not required for non-reusable human missions.

Table 2 - Mission Description by Major Maneuvers

Mission	TDI1 (km/s)	TDI2 (km/s)	DOC (km/s)	TEI (km/s)	EOC (km/s)	Total ΔV (km/s)
7-Launch DRA5 Cargo, Propulsive Capture	3.662		1.341			5.003
7-Launch DRA5 Cargo, Aero-capture	3.839					3.839
7-Launch DRA5 Crew	3.992		1.771	1.562		7.325
Mars Crewed	3.212	1.07	1.806	1.593		7.681
Mars 600day	2.9734	0.9166	1.47	3.08		8.44
Mars 900day	2.321	2.00	1.0393	1.4898		6.8501
2009 HC, Robotic	3.419		1.924			5.343
2009 HC, Crew	3.678		0.338	0.54		4.556
2000 SG344, Robotic	3.26		1.655			4.915
2000 SG344, Crew, 4/27/28	3.254		0.144	0.392		3.79
2000 SG344 Crew, 4/28/28	3.254		0.142	0.399		3.795
2000 SG344 Crew, 4/28/28	3.252		0.154	0.408		3.814
2000 SG344 Crew, 4/28/28	3.252		0.169	0.417		3.838
2000 SG344 Crew Reusable, 327d	1.693	1.627	0.144	0.392	1.711	5.567
2000 SG344 Crew Reusable, 319d	2.293	1.084	0.169	0.417	1.711	5.674
2008 EV5 Robotic, 12/25/21	4.127		2.354			6.481
2008 EV5 Crew, 6/22/24	3.81		1.203	2.766		7.779
Apophis Robotic, 10/17/21	4.429		0.356			4.785
Apophis Crewed, 5/8/28	3.783		1.542	0.342		5.667
Apophis Expendable, 344d	2.6593	1.2367	1.542	0.342		5.78
Apophis Reusable, 344d	2.799	1.237	1.542	0.342	1.711	7.631
1991 JW 5/27	4.014		0.851	0.612	1.711	7.188
Mars Split Mission (ERV) 12/30	3.662		1.34			5.002
Mars Split Mission (MSV) 5/33	3.83		1.53	3.12		8.48
Mars "All-Up" Mission 5/33	3.83		1.53	3.12		8.48
Reusable Lunar Cargo	3.3		0.915	0.915	1.7	6.83
Reusable Lunar Crew	3.3		0.915	0.915	1.7	6.83

In addition to the major maneuvers, each mission will also require several smaller maneuvers such as mid-course corrections, rendezvous and docking, station keeping, etc. These would normally be performed using the vehicle's reaction control system (RCS) rather than the main propulsion system, but must be included in the mission scenario to fully validate the mission.

There are a number of other constraints that must be satisfied to successfully complete the mission besides the main propulsion and RCS fuel requirements. For example the burn time and number of restarts on the NTP engines must be kept within the limits demonstrated in testing. The NERVA experimental engine (NRX-XE) was tested to approximately two hours of burn time and twenty seven restarts in 1969. [12] Another set of constraints is levied by the launch system which must deliver the spacecraft stages to low Earth orbit. Most of the missions under consideration assume some variant of the Space Launch System (SLS) currently under development by NASA. The Block-I version of this vehicle is expected to lift approximately 70 metric tons. [13] This is sufficient to launch the smaller scale "Search Lite" class NTP vehicles, such as those described for some of the 2000 SG344 missions and for the lunar missions. An upgraded Block-IA version of SLS with a lift capability of approximately 110 tons enables the more challenging NEA missions, including those targeting the asteroid Apophis. The ultimate Block-II SLS, which is planned to be able to lift approximately 130 tons to low Earth orbit, supports the larger vehicles, including the 10 meter tank variations, for the Mars missions. Note that the available volume of the launch vehicle's payload shroud is also a key constraint, in relation to both the diameter and the length of the stages of the NTP spacecraft.

IV. Spacecraft Model Architecture

In OpenMDAO, a model is composed of Components and Assemblies. A Component defines a simple object that has a set of input variables, a set of output variables and an 'execute' function that calculates the values of the output variables based on the values of the input variables. An Assembly is a special type of Component that can contain one or more child Components or Assemblies. The input and output variables of an Assembly are typically connected to those of its children. In addition, there may be connections between the child components, such that the output of one component's calculation will be the input to another. Instead of an execute function, the Assembly has another special type of Component called a Driver. The Driver is responsible for invoking the execution of the assembly's child components in the proper sequence, which is called the Workflow. There are many types of drivers available, but most commonly a solver or an optimizer is used. With a solver, you can set parameters based on the components input variables and constraints based on their outputs and the solver will drive the model to a valid solution. An optimizing driver adds the ability to specify an expression (objective) to be minimized. See Figure 6 for an illustration from the OpenMDAO documentation.

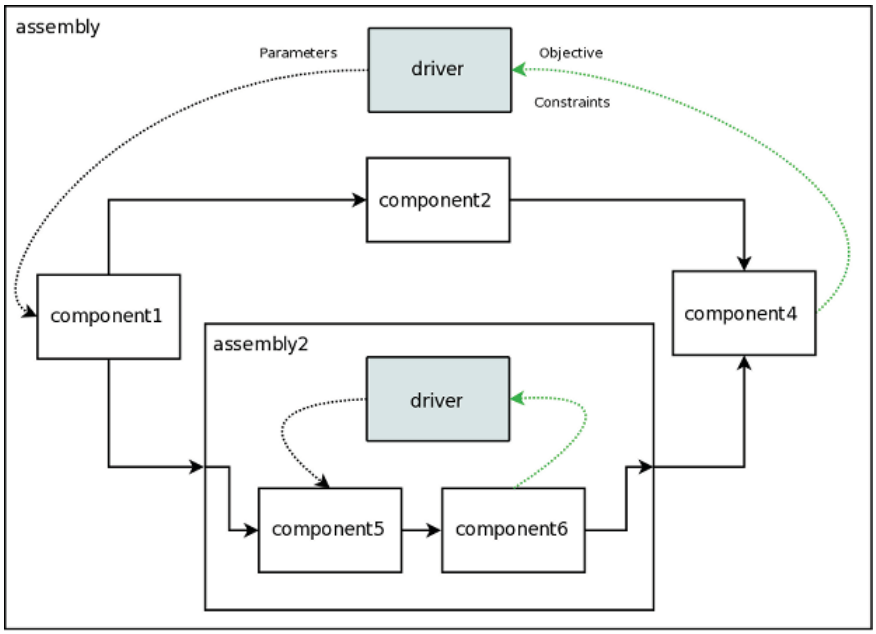

Figure 6 - OpenMDAO Assembly Structure

This hierarchical, object-oriented structure of OpenMDAO lends itself to systematic break down of a spacecraft into its constituent subsystems. To that end, a base class called Subsystem has been created for assemblies used to build the spacecraft model. A basic Subsystem assembly contains the logic to calculate the dry mass and the wet mass of the subsystem it represents. A MassItem component has been defined to represent the mass properties of the individual pieces of equipment that makes up a subsystem. In addition, as described above, a Subsystem may have additional child subsystems. The base functionality of a Subsystem is simply to roll up the mass properties of its children.

To implement specific types of subsystems, additional attributes and behaviors are added to the Subsystem component via Interface specifications. For example, a propulsion subsystem has two additional outputs: thrust and specific impulse (Isp). These attributes are specified by the IPropulsion interface. A rocket engine component can then be created as a Subsystem component that implements the IPropulsion interface. The implementation of RocketEngine must then provide or calculate the thrust and Isp attributes in addition to its mass.

Many subsystems have attributes that are dependent on other subsystems. For example, many of the structural components, including the saddle truss, are sized around the diameter of the fuel tanks. As mentioned previously, this is generally a constraint of the launch vehicle and available fairing size. Here again we can make use of the capabilities of the OpenMDAO framework to link attributes of one subsystem to another. Looking at a FuelSubsystem as an example, we know that the mass of the MMOD and TPS subsystems both depend of the surface area of the fuel tank. We can make connections between those attributes as depicted in Figure 7.

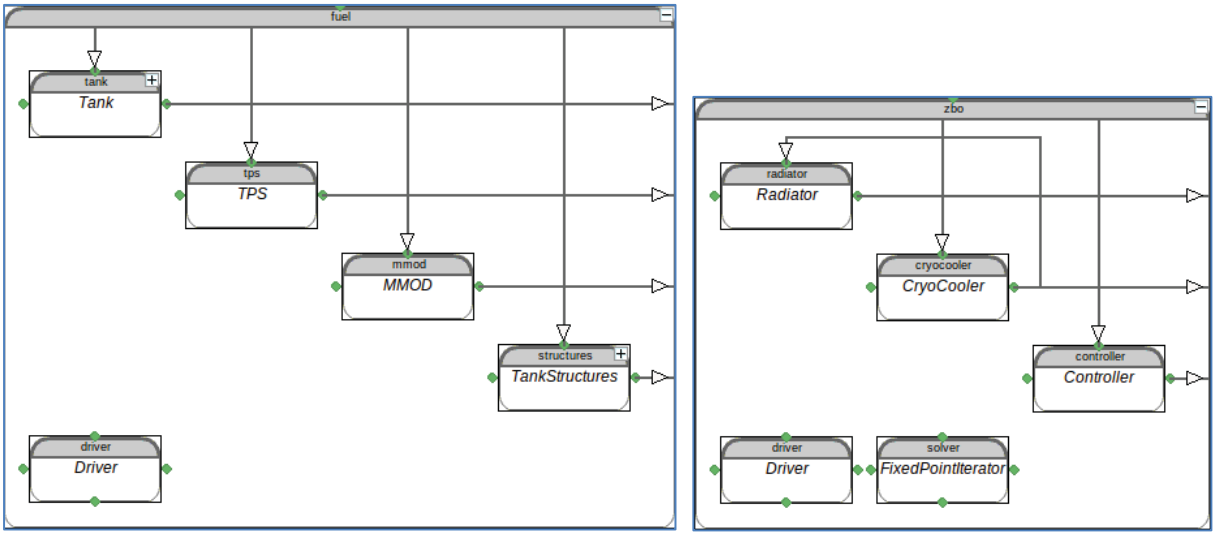

Figure 7 - Fuel Subsystem and ZBO Subsystem

A more complex example can be found in the Zero Boil-off Subsystem, also shown in Figure 7. This component represents the subsystem that maintains the liquid Hydrogen (LH_2) propellant at cryogenic temperatures and is based on previous sizing analysis at NASA GRC. There are three components in this subsystem: the cryocooler, the thermal radiator and the controller. The inputs to the CryoCooler component come from FuelSubsystem and include the volume and surface area of the tank as well as the fluid temperature. These are used to estimate the heat leaks that need to be addressed and thus the size and mass of the cryocooler. The electrical power to run the cryocooler is added to the heat leak to calculate the total heat that needs to be rejected by the radiator, which is another output value. The radiator component takes the total heat as an input and calculates the cooled temperature in addition to the radiator mass. As can be seen in the figure, this creates a circular dependency between the cryocooler and the radiator that must be solved, so a solver (FixedPointIterator) is included in the assembly to accomplish this. The relevant portion of the code showing the structure of the ZBOSubsystem module is shown in Figure 8.

```
class CryoCooler(Equipment):
    """ calculate total heat generated and mass of cryocoolers
    """

   # inputs
   n = Int(2, iotype='in', desc='number of cryocoolers')

   temperature = Float(302.0, iotype='in', units='K', desc='temperature')
   tank_area   = Float(0.0, iotype='in', units='m**2',desc='surface area of tank')
   tank_volume = Float(0.0, iotype='in', units='m**3', desc='volume of tank')
   redundancy  = Float(0.25, iotype='in',desc='redundancy factor')

   # outputs
   total_heat  = Float(iotype='out', units='W',desc='total heat')
   power       = Float(iotype='out', units='kg',desc='electrical power required')
   mass        = Float(iotype='out', units='kg',desc='cryocooler mass')


class Radiator(Equipment):
   """ calculate the temperature and mass of a radiator given the area and
       the total heat to be radiated
   """

   # inputs
   area        = Float(49.84, iotype='in', units='m**2', desc='radiator area')
   total_heat  = Float(0.0,   iotype='in', units='W',desc='heat to be radiated')
   Tenv        = Float(270.0, iotype='in', units='K', desc='equilibrium temperature')
   emmissivity = Float(0.85,  iotype='in', desc='emmissivity')

   # outputs
   mass        = Float(iotype='out', units='kg', desc='radiator mass')
   temperature = Float(iotype='out', units='K', desc='radiator temperature')


class ZBOSubsystem(Subsystem):
    """ calculate mass of Zero Boil-Off Cryocooler subsystem
    """

    # inputs
    tank_area   = Float(0.0,  iotype='in', units='m**2', desc='surface area of tank')
    tank_volume = Float(0.0,  iotype='in', units='m**3', desc='volume of tank')
    redundancy  = Float(0.25, iotype='in', desc='redundancy factor')

    def configure(self):
        self.add('cryocooler', CryoCooler())
        self.add('radiator',   Radiator())
        self.add('controller', Controller())

        self.connect('tank_area',   'cryocooler.tank_area')
        self.connect('tank_volume', 'cryocooler.tank_volume')
        self.connect('redundancy',  'cryocooler.redundancy')

        self.connect('cryocooler.total_heat',  'radiator.total_heat')
        self.connect('radiator.temperature',   'cryocooler.temperature')

        self.connect('cryocooler.power',  'controller.power')
        self.connect('redundancy',        'controller.redundancy')

        self.add('solver', FixedPointIterator())
        self.solver.workflow.add('cryocooler')
        self.solver.workflow.add('radiator')
        self.solver.workflow.add('controller')

        self.driver.workflow.add('solver')
```

Figure 8 - Zero Boil-Off Subsystem Definition

In some cases the mass of a subsystem is dependent on the mission it is intended to be used for. A simple example would be the TransHab module that provides living quarters and life support for crew members of a manned mission. Clearly the size and mass of this module will depend on how many crew members need to be accommodated and for how long. In this case ‘crew size’, ‘duration’ and ‘consumable rate’ would all be inputs to the TransHab Subsystem component and are used to calculate its mass. Solar power systems are also a prime example of being sized based on the destination, with missions to distances farther from the sun requiring larger photovoltaic arrays.

Typically for the missions under study, spacecraft are assembled in stages. A Copernicus-type crew transport vehicle, for example, is composed of a core propulsion stage, a saddle truss and drop tank assembly stage, and a crew/payload stage. The core stage is composed of the NTP propulsion system, fuel tanks and supporting subsystems. The drop stage is predominately a jettisonable fuel tank with supporting subsystems. The crew stage is a collection of crew support systems. Spacecraft and Stage components have been defined based on the Subsystem component to model this structure. As an example, the three stage Asteroid Survey Vehicle (ASV) has been modeled with the structure shown in Table 3.

Table 3 - Subsystem Structure for Asteroid Survey Vehicle (ASV)

ASV (Spacecraft)		
core_stage (Stage)	drop_stage (Stage)	crew_stage (Stage)

- ntp (NTPSubsystem)
  ○ engine (NERVA_25klbf)
  ○ tvc (TVCSubsystem)
- rcs (RCSSubsystem)
- fuel_sys (FuelSubsystem)
  ○ tank (Tank)
  ○ mmod (MMOD)
  ○ tps (TPS)
  ○ structures (Subsystem)
- mps (MPSSubsystem)
- zbo (ZBOSubsystem)
- hds (HeliumSubsytem)
- avc (Subsystem)
  ○ avionics (Equipment)
  ○ comm (Equipment)
- eps (Subsystem)
  ○ pva (SolarArraySubsystem)
  ○ batteries (BatterySubsystem)
  ○ pmad (Equipment)
- docking (DockingSubsystem)
	- truss (Subsystem)
	- fuel_sys (FuelSubsystem)
	  ○ tank (Tank)
	  ○ mmod (MMOD)
	  ○ tps (TPS)
	  ○ structures (Subsystem)
	- mps (MPSSubsystem)
	- hds (HeliumSubsystem)
	- eps (Subsystem)
	  ○ batteries (BatterySubsystem)
	  ○ pmad (Equipment)
	- avc (Subsystem)
	  ○ avionics (Equipment)
	  ○ comm (Equipment)
	- gyros (Subsystem)
	- docking (DockingSubsystem)
		- mmsev (Subsystem)
		- TransHab (Subsystem)
		- cev (Subsystem)
		- food (Subsystem)
		- crew (Subsystem)
		- truss (Subsystem)
		- tdm (Subsystem)
		- rcs (RCSSubsystem)
		- eps (ElectricalPowerSubsystem)
		- avionics (Subsystem)

There are several advantages to the modular structure of this spacecraft model. The first is that it provides the ability to easily set up a model to capture the various vehicle configurations discussed previously. It also allows for a component to be implemented as simple equipment item with a rough mass estimate, as a basic Subsystem component that calculates its mass based on a scaling law, or as a multi-component assembly that does a more complex solution or optimization. Given this flexibility, a full spacecraft model can be constructed quickly with fixed mass estimates for each subsystem. The details for each subsystem can then be filled in as higher fidelity models become available by simply replacing that component with an updated version.

V. Mission Simulation

Given a spacecraft model such as the ASV shown above, the goal is to size the vehicle to accomplish a given mission. To this end, a structure has been devised that follows the example of the MOMMA code in specifying a mission in terms of a sequence of phases. Each phase encapsulates a part of the mission that impacts the spacecraft in some way.

The most significant phases involve a maneuver, which imparts a change in velocity (ΔV) to the spacecraft via the burning of fuel. The ASV, for example will perform several major maneuvers as shown in Table 2: Earth departure (TDI1/2), asteroid orbit capture (DOC) and trans-Earth injection (TEI1/2). These large maneuvers are performed using the core propulsion system. In addition, there are numerous smaller 'burns' for purposes such as rendezvous and docking, attitude control, station keeping and propellant settling. These are accomplished via reaction control systems in the core and crew stages. To encapsulate the concept of a maneuver, a Maneuver class has been implemented. An instance of this class specifies one of a number of known maneuver types. If a ΔV is specified, the maneuver will simply request the spacecraft to perform a burn sufficient to affect that change in velocity. As an example, the asteroid orbit capture burn is captured as a mission phase with the syntax shown in Figure 9.

```
# orbit capture at asteroid using core propulsion
phase = Phase()
phase.description = 'Asteroid Orbit Capture'
phase.duration = 0
maneuver = Maneuver()
maneuver.dV = 1.542
maneuver.other_reserve = 0.03
phase.add maneuver(maneuver)
```

Figure 9 - Phase Definition for a Maneuver

Alternatively, there are a number of orbital maneuvers that can be specified for which the ΔV will be calculated based on the initial and desired orbit. The following orbital maneuvers are available, given a solar system body and the current orbit: 'Departure from Apoapsis', 'Departure from Periapsis', 'Capture at Apoapsis', 'Capture at Periapsis', 'Circularize at Apoapsis', 'Circularize at Periapsis' and 'Plane Change'.

In addition to a maneuver, a mission phase may also have a number of other possible impacts. If a phase has a duration greater than zero, this will trigger the expending of crew consumables and boil off of cryogenic fuels. It is also possible to add or remove mass to the vehicle by the dropping of subsystems (e.g. the fuel tank from the drop stage of the ASV) or by picking up scientific samples. Figure 10 illustrates the mission phase definition for jettisoning the drop stage fuel tank:

```
# drop the fuel tank from the drop stage
phase = Phase()
phase.description = 'Tank Drop & Residuals'
phase.duration = 0
phase.drop subsystem = 'drop stage.fuel sys'
```

Figure 10 - Phase Definition for a Jettison Event

Once all the phases of the mission have been specified, a complete mission profile is then implemented via the Mission class. A Mission is an Assembly with a workflow that first executes the spacecraft and then each of mission phases. The initial execution of the Spacecraft performs all of the sizing and mass calculations, resulting in a model of the fully fueled vehicle at the beginning of the mission. As each subsequent Phase is executed, the spacecraft is called upon to perform the specified action. For example, if the mission phase includes a maneuver the spacecraft will be called on to perform the necessary burn to implement the specified or calculated ΔV. The spacecraft will determine the appropriate propulsion system and calculate the burn time and fuel consumed via the rocket equation, using its current mass and applying the appropriate margins and cool-down factor. The mass of the expended fuel is then removed. For jettison events, the mass of an entire subsystem can be removed. The mission is configured such that the spacecraft mass and fuel levels as well as the Mission Elapsed Time (MET) at the end of each phase are provided as inputs to the next phase as shown in Figure 11.

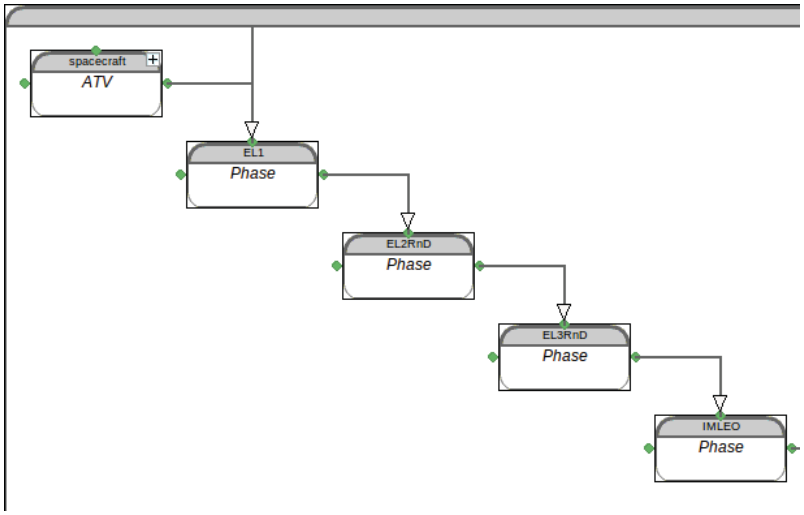

Figure 11 – Subset of Apophis Mission with ASV

To close a mission design, the Mission assembly is put under the control of a solver, which has the ability to vary the fuel capacities (for both the core stage propulsion and RCS) in order to size the spacecraft such that it has just enough fuel to accomplish the mission. For example, the Apophis mission problem statement shown in Figure 12.

```
        # Apophis mission simulation
        self.add('driver', BroydenSolver())
        self.add('mission', Apophis())
        self.driver.workflow.add('mission')

        # with ATV spacecraft
        self.mission.replace('spacecraft', ATV())

        # parameters:
        #     H2 fuel tank capacity (assuming common H2 tank size)
        #     RCS propellant capacity for core and crew stages
        self.driver.add_parameter('mission.spacecraft.fuel_capacity',
                                  low=10000., high=100000.)
        self.driver.add_parameter('mission.spacecraft.core_rcs_capacity',
                                  low=10000., high=100000., scaler=.10)
        self.driver.add_parameter('mission.spacecraft.crew_rcs_capacity',
                                  low=10000., high=100000., scaler=.10)

        # initial guesses
        self.mission.spacecraft.fuel_capacity = 50000
        self.mission.spacecraft.core_rcs_capacity = 5000
        self.mission.spacecraft.crew_rcs_capacity = 5000

        # constraints:
        #     no fuel or RCS propellant left at end of mission
        self.driver.add_constraint('mission.end_fuel    = 0.')
        self.driver.add_constraint('mission.end_prop[0] = 0.')
        self.driver.add_constraint('mission.end_prop[2] = 0.')
```

Figure 12 - Apophis Mission Problem Statement

In this example, we are assuming a common fuel tank capacity for the ASV's core and drop stages. We then provide initial guesses for the fuel and RCS propellant requirements and constrain the end of mission to have zero remaining fuel and propellant. Upon execution, the simulation provides a snapshot of the state of the spacecraft at every phase of the mission, updating the propellant masses and adjusting for the jettisoning of subsystems and pick up of cargo. A simple plot of the mission mass profile, generated from one such execution of the model, is shown in Figure 13. Upon successful solution, the initial state of the spacecraft reflects the total launch mass (IMLEO). The IMLEO breakdown of the ASV for the Apophis mission solution is shown in Figure 14.

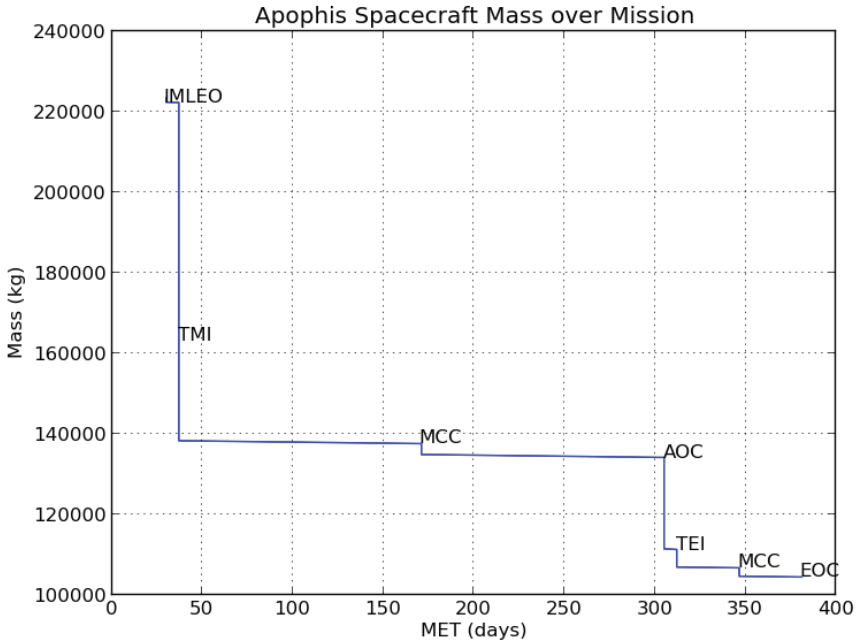

Figure 13 - Apophis Spacecraft Mass during Mission Simulation

```
spacecraft        dry: 106365.19   wet: 223594.77
  core_stage     dry:  33779.58    wet:  90190.64    (  56411.07)
    mps                    524.38
    zbo              dry:    871.34   wet:    871.34
      cryocooler             513.98
      radiator               274.12
      controller              83.24
    docking          dry:    649.94   wet:    649.94
      docking_adapter        649.94
    rcs              dry:   1009.21   wet:   1926.40   (    917.19)
      reactant               917.19
      hardware              1009.21
    fuel_sys         dry:   9968.05   wet:  65366.85   (  55398.80)
      mmod                  1388.47
      tps                    777.54
      tank              dry:   7583.83   wet:  62982.63   (  55398.80)
        fuel               55398.80
        shell               7583.83
      structures         dry:    218.20   wet:    218.20
        aft_ring               57.70
        forward_ring           57.70
        fwd_adap_skirt        102.80
    avc              dry:    434.24   wet:    434.24
      communications           42.09
      avionics                392.15
    ntp              dry:  17950.40   wet:  17950.40
      thrust_structure        878.70
      EMA_control             620.10
      base_shield            1492.06
      tvc                     338.10
      engine_cntlrs           232.28
      ext_rad_shield         6037.50
      engine1                2670.15
      engine3                2670.15
      engine2                2670.15
      turbopump               341.20
    eps              dry:   1536.08   wet:   1536.08
      pva                    565.80
      pmad                   287.44
      batteries              682.84
    hds              dry:    835.94   wet:    931.01   (     95.08)
      helium                  95.08
      hardware               835.94
  drop_stage     dry:  19184.74   wet:  74678.62   (  55493.87)
    truss                 5820.96
    gyros                  625.60
    mps                    524.38
    docking          dry:      0.15   wet:      0.15
      docking_adapter          0.15
    fuel_sys         dry:   9968.05   wet:  65366.85   (  55398.80)
      mmod                  1388.47
      tps                    777.54
      tank              dry:   7583.83   wet:  62982.63   (  55398.80)
        fuel               55398.80
        shell               7583.83
      structures         dry:    218.20   wet:    218.20
        aft_ring               57.70
        forward_ring           57.70
        fwd_adap_skirt        102.80
    avc              dry:    837.20   wet:    837.20
      communications           732.55
      avionics                 104.65
    eps              dry:    572.47   wet:    572.47
      pmad                     261.97
      batteries                310.50
    hds              dry:    835.94   wet:    931.01   (     95.08)
      helium                  95.08
      hardware               835.94
  crew_stage     dry:  53400.87   wet:  58725.51   (   5324.64)
    crew                   400.00
    truss                 3554.52
    transhab             22700.00
    communications         241.04
    arm_Tdock             1759.50
    mmsev                 6700.00
    food                  3440.00
    cev                  13500.00
    eps              dry:     96.60   wet:     96.60
      pmad                    96.60
    rcs              dry:   1009.21   wet:   6333.85   (   5324.64)
      reactant               5324.64
      hardware               1009.21
```

Figure 14 - IMLEO Solution for Apophis Asteroid Survey Vehicle (ASV)

VI. Summary

We have described the development of a tool for integrated system modeling of spacecraft and missions. The tool has been used to develop models of a number of different NTP spacecraft configurations and mission scenarios, which have been used for validation of NTP vehicle studies. The modular nature of the tool, built on the OpenMDAO Framework, has enabled development of the spacecraft models in an incremental manner, with the ability to increase the fidelity of each subsystem analysis independently and asynchronously. The ability to link the key parameters of the spacecraft subsystems enables a truly integrated system model. In the future we expect to more fully take advantage of the optimization capabilities of the OpenMDAO Framework to perform more sophisticated analysis.

VII. Acknowledgments

The authors express their thanks to John Warren and Chris Moore (NASA/HQ), Mike Houts (MSFC), also John Taylor, Joe Roche, Dennis Rohn and Mark Klem (GRC) for their interest in this work and funding support through the Advanced Exploration Systems' Nuclear Cryogenic Propulsion Stage (NCPS) project.

VIII. References

[1] Human Exploration of Mars Design Reference Architecture 5.0, Drake, Bret G., ed., National Aeronautics and Space Administration, NASA-SP-2009-566, Washington, DC, July 2009.
[2] Borowski, S. K., McCurdy, D. R., and Packard, T. W., "7-Launch NTR Space Transportation System for NASA's Mars Design Reference Architecture (DRA) 5.0", AIAA-2009-5308, August 2009.
[3] Borowski, S. K., McCurdy, D. R., and Packard, T. W., "Nuclear Thermal Propulsion (NTP): A Proven, Growth Technology for "Fast Transit" Human Missions to Mars", AIAA-2013-5354, September 2013
[4] Borowski, S. K., McCurdy, D. R., and Packard, T. W., "Near Earth Asteroid Human Missions Possibilities Using Nuclear Thermal Rocket (NTR) Propulsion", AIAA-2012-4209, July 2012.
[5] Borowski, S. K., McCurdy, D. R., and Packard, T. W., "Modular Growth NTR Space Transportation System for Future NASA Human Lunar, NEA and Mars Exploration Missions", AIAA-2012-5144, September 2012.
[6] Borowski, S.K., McCurdy, D.R and Burke, L. B., "The Nuclear Thermal Propulsion Stage (NTPS): A Key Space Asset for Human Exploration and Commercial Missions to the Moon", AIAA-2013-5465, September 2013
[7] Moore, K. T, Naylor, B. A. and Gray, J. T., "The Development of an Open Source Framework for Multidisciplinary Analysis and Optimization", AIAA-2008-6069, September 2008
[8] http://openmdao.org
[9] https://www.python.org/
[10] http://www.numpy.org/
[11] http://scipy.org/
[12] Koeing, D. R., "Experience Gained from the Space Nuclear Rocket Programs (Rover / NERVA)," Los Alamos National Laboratory, Report LA-10062-H, Los Alamos, NM, May 1986.
[13] http://www.nasa.gov/pdf/664158main_sls_fs_master.pdf, Space Launch System Fact Sheet