Next Generation Simulation Framework for Robotic and Human Space Missions

Jonathan M. Cameron[1], J. Balaram[2], Abhinandan Jain[3], Calvin Kuo[4], Christopher Lim[4], and Steven Myint[4]
Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91107

The *Dartslab* team at NASA's Jet Propulsion Laboratory (JPL) has a long history of developing physics-based simulations based on the *Darts/Dshell* simulation framework that have been used to simulate many planetary robotic missions, such as the Cassini spacecraft and the rovers that are currently driving on Mars. Recent collaboration efforts between the Dartslab team at JPL and the Mission Operations Directorate (MOD) at NASA Johnson Space Center (JSC) have led to significant enhancements to the Dartslab DSENDS software framework. The new version of DSENDS is now being used for new planetary mission simulations at JPL. JSC is using DSENDS as the foundation for a suite of software known as COMPASS that is the basis for their new human space mission simulations and analysis. In this paper, we will describe the collaborative process with the JPL Dartslab and the JSC MOD team that resulted in the redesign and enhancement of the DSENDS software. We will outline the improvements in DSENDS that simplify creation of new high-fidelity robotic/spacecraft simulations. We will illustrate how DSENDS simulations are assembled and show results from several mission simulations.

I. Introduction and Background

A. Prior History the Dartslab DSENDS simulation frameworks at JPL

DSENDS is a physics-based engineering simulator for space missions developed by the Dartslab[5] team at NASA's Jet Propulsion Laboratory [1]. DSENDS models the spacecraft as a multi-body system where the spacecraft position, attitude, articulation and body flexibility states (and their rates) interact with gravity, atmospheres, terrain, and on-board spacecraft devices in response to ground commands and flight-software directed sensing and control actions. DSENDS is a deployment of the Dshell multi-mission simulation framework [2]. It was originally designed to provide functionality for Entry, Descent and Landing (EDL) problems but has since been generalized to provide capabilities relating to spacecraft ascent, orbit, proximity operations, rendezvous descent and surface operations (e.g. roving).

The DSENDS tool is in use at JPL for technology/concept development – all the way from Pre-Phase A analysis to flight operations. It is used by NASA/JPL missions for performance studies, cross-validation of other simulations and tools, and flight-critical EDL mission operations including lander targeting. It has been used by NASA/JPL Technology Programs, Program Offices, and Mission Analysis teams as a high-fidelity simulator to support proposal development, as an integration platform and test-bed for studies, and as a tool for algorithm and software development.

As part of JPL's end-to-end Mission systems, DSENDS interoperates with JPL's Interplanetary Mission design and navigation software *Monte* [2]. In flight-operations it is used to verify the actions of mission actions (e.g. determine the landing footprint), perform targeting operations (e.g. to design interplanetary trajectory correction maneuvers).

DSENDS development started with the 1991 development of the DARTS dynamics engine. The Dshell framework was initially developed in 1992 for the Cassini mission, with development of the DSENDS deployment commencing in 2000. In 2007 DSENDS was used in the Mars Phoenix EDL Mission Operations and more recently

---

[1] Senior Member of Technical Staff, Mobility and Robotics Systems Section, 4800 Oak Grove Drive, Pasadena, CA 91109, Mail Stop 198-219.
[2] Principal Member of Technical Staff, Mobility and Robotics Systems Section, 4800 Oak Grove Drive, Pasadena, CA 91109, Mail Stop 198-219.
[3] Senior Research Scientist, Mobility and Robotics Systems Section, 4800 Oak Grove Drive, Pasadena, CA 91109, Mail Stop 198-219.
[4] Member of Technical Staff, Mobility and Robotics Systems Section, 4800 Oak Grove Drive, Pasadena, CA 91109, Mail Stop 198-219.
[5] For more information about the Dartslab, please visit http://dartslab.jpl.nasa.gov.

it was used for the Mars Science Laboratory EDL operations. It is currently in use at JPL by the Low Density Supersonic Decelerator (LDSD) project for design of their integrated test campaign. It has also been selected by JSC-MOD as the basis for COMPASS, their next generation flight dynamics tool for all ascent, descent, orbit, proximity-operations, and rendezvous simulations.

To users, DSENDS provides a modern scripting language (Python) to enable data-driven building of applications at run-time, easy configuration setup & initialization, and automated testing. The underlying Dshell framework heavily leverages various Open Source software resources. High computational performance is achieved using C++ implementations of module functions and fast algorithms, with auto-generation of the hooks from the C++ level to Python. Visualization and introspection tool configurations are dynamically generated from the simulation configuration.

B. Motivation for improvements to the DSENDS framework

Due to the production nature of the first users of DSENDS, the software package delivered to them was fairly stable and static. Changes were typically due to bug fixes and mission-specific enhancements or models. In the meantime, the Dartslab team had improved the simulation framework software, including enhancements to multibody modeling (Ndarts [3], [4]), 3D visualization (using Ogre [6]), very large scale terrain modeling (SimScape [6]). Having to maintain an older version of the software while active development is going on the latest version is inefficient and error-prone. A path forward was needed that would allow the DSENDS user base to migrate to a current version of the software.

When the JSC Missions Operations Directorate (MOD) team approached us, they needed a new simulation framework. With the retirement of the Space Shuttle, the MOD team was transitioning into a dynamic role of supporting various vehicles with rapidly changing missions. This required agile new simulation tools that are adaptable to a wide range of human space flight missions. The JSC MOD Flight Dynamics team evaluated a variety of options and completed a 6-DOF simulation benchmarking effort, and determined that DSENDS “offers the most capable and flexible simulation path for the division”.

In the process of training the JSC MOD team in the use of latest version of DSENDS, it became apparent that there were some parts of DSENDS that needed improvements for their use cases. For instance, DSENDS could only support one spacecraft at a time. This led to a collaborative effort to refactor and improve DSENDS.

C. Collaborative development

In order to collaboratively develop a new version of DSENDS, in 2011 we formed a joint JPL-JSC development partnership led by JPL (although most of the development team was from the JSC MOD). Our goal was to refactor much of the simulation software underlying DSENDS so that it would meet the needs of both the JPL and the JSC MOD teams. The primary focus was on creating a new module "DshellCommon" that constitutes a complete re-write of the corresponding functionality of the previous version of DSENDS. There were also extensive changes to the underlying software to incorporate the framework enhancements outlined in the previous section, as well as changes to increase the usefulness, usability, and robustness of the software.

Our development team used Agile methodologies [7] including Scrum principles [8] to define short-term incremental releases on a two-to-three week development cycle. Each cycle typically involved a review of the prior DSENDS functionality and code, planning of how to adapt or rewrite the code, implementing the desired re-coding, and writing related regression tests.

Due to inter-center complications of sharing a software repository, the team in each center maintained a separate software repository (using subversion). The JPL team took responsibility for merging changes from both sides based on JPL-JSC team discussions and consensus building, and committing the changes to both software repositories. Most of the software evolution took place in the DshellCommon module. The JPL team made sure that all changes worked in the JSC environment and the JPL environment (which was evolving independently of the JSC environment). Every several months, the JPL team put together a consolidated package of all the modules and provided it to JSC as a new baseline package. This prevented divergence of the code base over time.

We also conducted a face-to-face Technical Interchange Meeting (TIM) once every 6 months to discuss progress in the software development, talk about technical issues, and plan development strategy for the coming months.

This collaborative process has worked extremely well and has produced a new very useful and high-quality version of DSENDS. The new version of DSENDS was derived from earlier versions of DSENDS but is essentially a new implementation that supports a wider range of spacecraft and mission related scenarios. Not only is the JSC MOD team using the new DSENDS, but the JPL Dartslab team has transitioned to the new DSENDS for all current and new work.

D. The end product: A new DSENDS

The new version of DSENDS is significant step up from earlier versions of DSENDS in terms of flexibility, usability, robustness, and general software quality. DSENDS is made of several parts as shown in this figure on the right. From the bottom up, these components are:

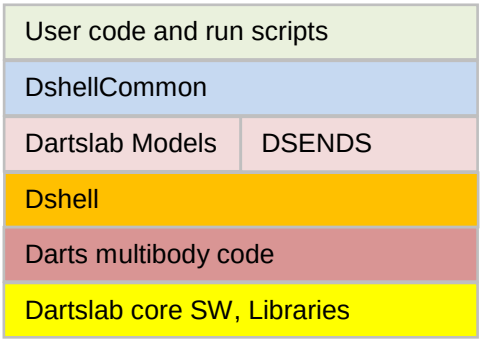

Figure 1 - DSENDS Components

- **Dartslab core SW, libraries** - Underlying software libraries such as vector libraries, terrain modeling libraries, visualization libraries, *etc.*
- **Darts multibody code** – Darts multibody dynamics engine.
- **Dshell** - The Dshell framework of classes for models, data flow management, parameter handling, integration, data logging, *etc.*
- **Dartslab models** - Library of reusable models for system hardware and subsystems.
- **DSENDS models** - Models for simulations involve aerodynamics and other EDL-related capabilities.
- **DshellCommon** - Simulation executives, run-script handling code, and classes for "assemblies" (a higher level model construct which construct and connect sets of related primitive models).
- **User code and run scripts** – Python libraries and run scripts written by the user to make use of all the lower levels to model and simulate complete systems.

The JSC MOD team uses the DSENDS software as the core of their system called COMPASS (Core Operations, Mission Planning, and Analysis Spacecraft Simulation). The block diagram in Figure 2 illustrates how COMPASS uses the underlying DSENDS framework.

This is the same as the previous layout except that JSC has implemented a variety of models and assemblies in the item labeled 'COMPASS Components' related to specific vehicle hardware and simulation scenarios they plan to support as part of their mission.

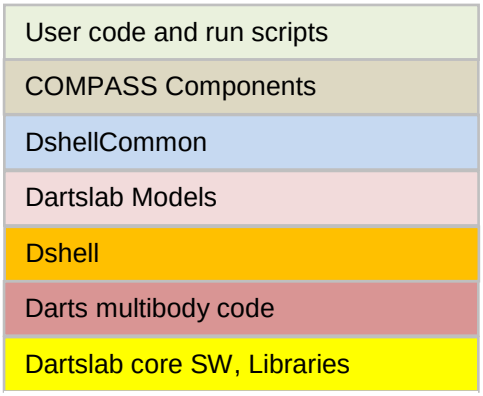

Figure 2 - JSC MOD COMPASS Components

II. Simulations with DSENDS (How does it work?)

E. What kind of systems can DSENDS model?

In short, DSENDS can model *any* type of space, aerial, marine surface, or underwater vehicle. The fidelity of the model is only limited by how detailed the implementers are willing model out the multibody system and the various hardware and software components.

DSENDS can handle arbitrary multibody systems including chain, tree, and closed-chain topologies. The Darts multibody dynamics code, winner of 1997 NASA Software of the Year Award, can handle closed loops with constraint embedding so that a minimum number of coordinates are used [4], [5]. Darts can also model bodies that are flexible using modal descriptions of the flexible-body dynamics.

DSENDS can model actuators that inject forces into the multibody system (such as a thruster) as well as sensors that can read information out of the multibody system (such as joint encoder). DSENDS also supports many different models for specific types of hardware (IMUs, *etc.*). If the user has special subsystems they need to model, they can create new Dshell models in C++ and use them in simulations just like any existing model.

DSENDS includes higher-level models called *Assemblies* that instantiate and set up sets of primitive models, connect their data flows appropriately, and initialize the primitive model parameters as needed. DshellCommon and other modules include a library of Assemblies that model typical spacecraft subsystems (such as a robot arm that includes bodies, joints, joint motors, encoders, *etc.*)

DSENDS can model aerodynamic forces acting on arbitrary bodies in a simulation. For instance, this capability has been used to perform launch vehicle breakup analysis to determine vehicle dispersion on vehicle failure.

DSENDS can handle simulations involving multiple spacecraft and multiple planetary bodies. The spacecraft can be independent or attached to each other. Spacecraft can detach from their parent body and attach to any other body at run time.

DSENDS provides a wide range of planetary bodies (Earth, Moon, Mars, *etc.*) as well as sophisticated atmospheric models based on the latest NASA planetary atmospheric codes. DSENDS can also model small bodies such as asteroids with a polygonal model that captures the variable gravity of oddly shaped bodies very accurately.

DSENDS provides tools for finite state machines that can control the sequencing and behavior of the simulation to construct and execute a very sophisticated mission simulation sequence.

DSENDS can run parameter sweeps or Monte Carlo simulations involving thousands of separate runs to determine dispersions. This capability was used extensively to plan entry trajectories and Entry, Descent, and Landing strategy for the Mars Phoenix and Mars Science Laboratory missions.

DSENDS provides tools for 3D visualization of all parts of a simulation (spacecraft, planets, trajectories, *etc.*) The user can control the view point of the visualization "camera" to better watch and understand the simulation. Once a simulation is developed, the 3D visualization can be captured in a movie.

F. What is "under the hood" of DSENDS?

Underneath the surface of DSENDS, there are many powerful features and capabilities. Simulations based on DSENDS are physics-based. The figure below shows the basic functionality of the DSENDS simulation framework.

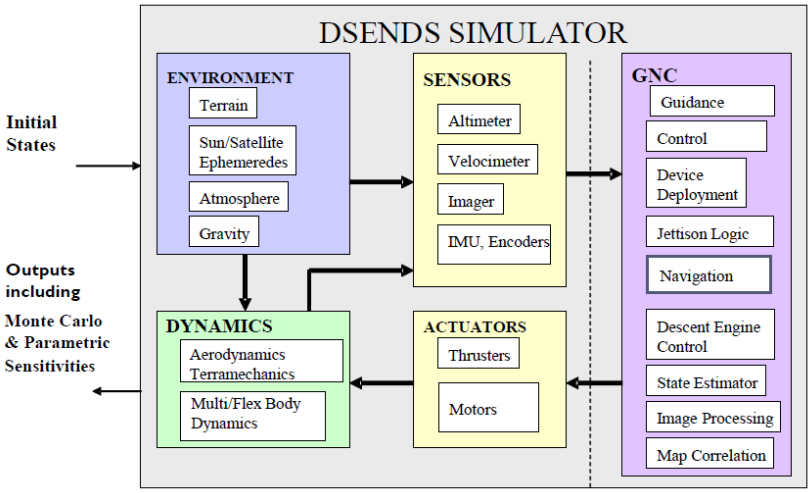

Figure 3 - DSENDS Functional Block Diagram

The high degree of physics fidelity starts with the underlying multibody code, *Darts*. Darts is a multibody dynamics engine based on spatial operator algebra [9] with O(n) performance that scales well to hundreds or thousands of degrees of freedom. Since it uses an inherently efficient algorithm and is implemented in C++, in most situations Darts offers faster than real-time performance. Another key feature is the ability to support a wide range of 'Dshell' models for various hardware devices or subsystems. The models can insert forces or torques into the multibody system (actuators) or read information out of the multibody system (encoders, sensors). The fidelity of the simulation is primarily limited by the fidelity of the models. Dshell models have the advantage of being very efficient since they are written in C++. Connecting models are data flows called 'signals'. Signals not only allow models to communicate with each other, they also define a partial order that allows model sorting (to control order of model execution during simulations).

In order to set up low-level models, data flows, and initialize parameters, we use Python. Each software module is compiled separately and set up so it can be loaded using the Python module import system. This means that simulations can be set up without compiling a large application executable (although this is possible, if desired). The users run script can load the necessary modules for their simulation. Therefore, we can shift away from the "application binaries" paradigm and use Python as the bedrock to glue together appropriate mix libraries at run-time.

Another very useful feature of DSENDS is that it has an underlying data representation called DVar (similar to a URL addressing scheme) that allows the user to access all multibody bodies and joints, model parameters, states, and signals in a simple and consistent way. This makes operations such as changing simulation parameters and logging much simpler and more flexible.

Dshell also includes the typical simulation capabilities such as selecting integrators, controlling simulation step sizes, processing events, and executing the simulations.

Finally, DSENDS includes a wide range of data analysis tools for run time (such as dials, gauges, and strip charts) as well as post-run analysis (such as plotting and data mining).

G. What does a run script look like?

The simulation 'run' scripts are Python files that have the following parts:

1. **Setup** - importing classes for the simulation executive and parameter definition files (not shown).
2. **Create the simulation executive:**

```
sim = SimulationExecutive()
```

This creates the simulation executive that will process and execute the simulation definition in the rest of this user run script.

3. **Configure the simulation configuration and parameterization:**

```
config = {
     'Mars' : {
        'class' : 'TargetAssembly',
        'params' : { 'Target' : targets['Mars'],
                     'Bodies' : bodies['Target']['Bodies'],
                   }
        },
     'SC1' : {
        'class' : 'VehicleAssembly',
        'basename' : 'CapsuleBase',
        'params' : { 'Bodies' : bodies['SC1']['Bodies']
                   },
        'assemblies' : {
            'grav' : {
               'class': 'GeneralGravityActuatorAssembly',
               'context' : { 'body' : 'CapsuleBase' }
               }
            }
        }
     }
```

Definition for target planet (Mars)

Definition for spacecraft 'SC1', including contained gravity model

This section of the user run script extracts parameters from the 'targets' and 'bodies' objects. These are defined by the user in separate Python files (targets.py and bodies.py) that are not shown here.

4. Create the simulation:

```
sim.createAssemblies(config)
sim.bindState()
sim.resetState(0.0)
```

These statements create the simulation objects (such as the spacecraft 'SC1') as well as set up its initial configuration properly.

5. Configure finite state machines to sequence the simulation (not shown).

6. Initialize any simulation-specific initial values (not shown).

7. Execute the simulation:

```
sim.step()
```

Users are free to include loops to execute as many time steps as desired. They may also want to interleave commands (although this can usually be better handled via finite state machine events).

III. Applications of DSENDS (What is it good for?)

H. JSC MOD applications

The JSC MOD team has applied the DSENDS/COMPASS framework to variety of mission scenarios including:

- Ascent scenarios that start fixed on the surface, launch, and reach a goal orbit.
- Rendezvous scenarios that involve two spacecraft in orbit. One spacecraft maneuvers to rendezvous with the other (passive) spacecraft (such as a spacecraft rendezvousing with the International Space Station). See the figure to the right.
- Descent scenarios that start with a spacecraft in orbit, perform a de-orbit thruster firing and track the vehicle downwards through entry, parachute deploy, and surface landing.
- Other scenarios such as orbital proximity operations and vehicle breakup analysis.

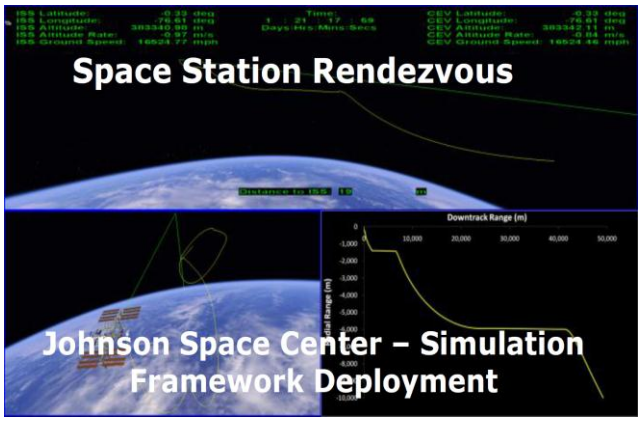

Figure 4 - JSC MOD Simulation

I. MSL Entry Descent and Landing Visualization

DSENDS has been used in a variety of ways in the Mars Science Laboratory mission [11]. Entry, Descent, and Landing was planned out using a prior version of DSENDS [10]. During the landing on August 5, 2012, the MSL Mission Control team used an MSL EDL visualization tool developed by the Dartslab team using the new version of DSENDS [12]. This was visible on of the main screens and on screens of several controller workstations. A screenshot of this visualization tool from the actual landing is shown here. Here the DSENDS models are driven by the real-time telemetry data from the MSL relay satellite (Mars Odyssey) instead of by solving of the underlying dynamics. Note that because of the availability of the dynamics capability, a physics-based extrapolation of the system trajectory and behavior is possible in case of telemetry data drop outs or slow data rates. However in the control room, only the last known status and position (from telemetry) were shown on the screen. In addition DSENDS also uses all the geometrical models (terrain, coordinate frames, etc) to display the telemetry data in the most appropriate mode for a given mission phase (e.g. inertial display versus surface fixed frame relative display). Also, the current positions of the Mars communications satellites (Mars Odyssey and Mars Reconnaissance Orbiter) were shown at all times using the standard DSENDS ephemerides updates. The 3-D views of the spacecraft are overlaid with event data as well as visual displays in the form of dials and text.

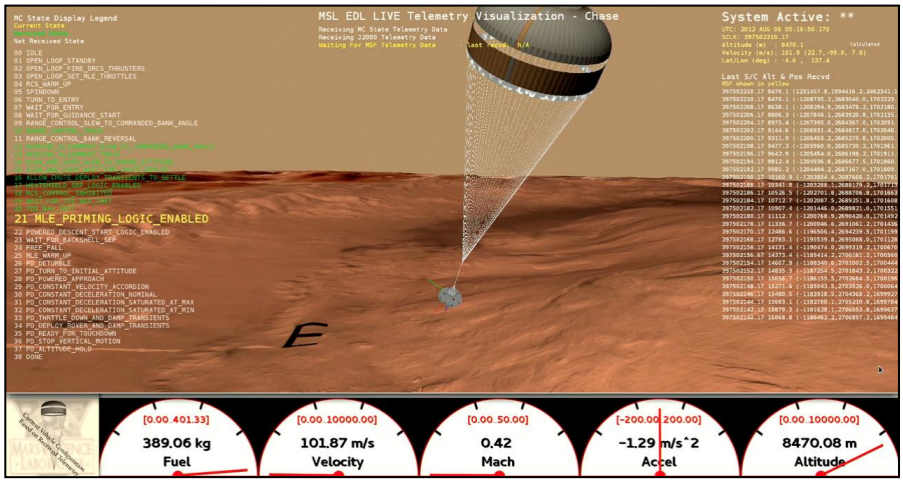

Figure 5 - MSL EDL Visualization Tool based on the new DSENDS used in MSL Mission Control during Mars Entry, Descent, and Landing

J. Low Density Supersonic Decelerator (LDSD) visualization

As the test program for LDSD is developed [10] the various test scenarios are generated in DSENDS and displayed using the visualization capabilities of the tool [13]. This allows the test engineers to design the test system (*e.g.* size the rocket motors used to accelerate the test article), visualize the trajectory and related constraints (*e.g.* land over-flight, communication line-of-sights), camera placements, etc.

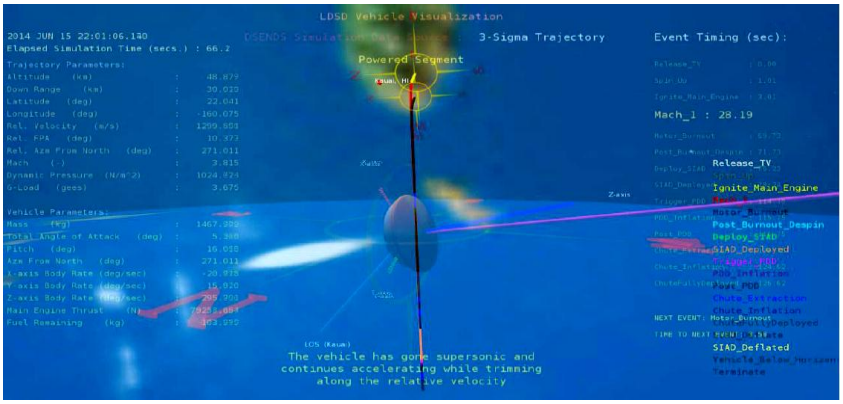

Figure 6 - LDSD Test Visualization

K. SEAS Lunar mission designs

This is an integrated landing and roving simulation with the mission consisting of a descent from lunar orbit, the operation of rover on the Aitken basin of the moon for the duration of a half lunar day, and the subsequent ascent of the spacecraft to lunar orbit with the collected samples. The rover is operated in both autonomous and teleoperated modes (from a crewed facility in halo orbit around the Lagrange L2 point).

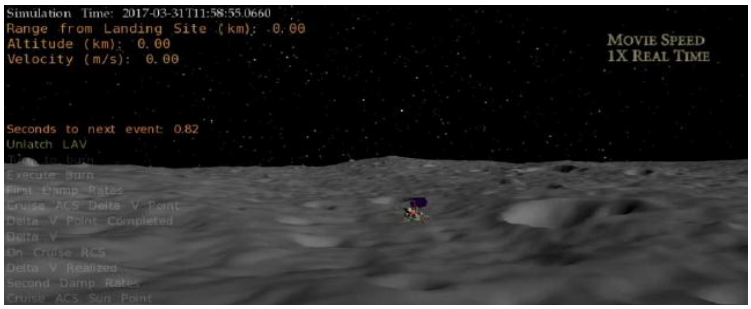

Figure 7 - SEAS Lunar Mission Analysis

L. Potential for University/STEM version

A version of the DSENDS aero-flight simulator system developed at the JPL Dynamics and Real-Time Simulation Laboratory is also being made available for use in undergraduate and graduate programs at universities. Initial users would be students in the Aerospace Engineering department at Georgia Tech.

IV. Conclusions

The new version of DSENDS is a powerful, modular, flight-tested, physics-based modeling/simulation tool for flight mechanics including standalone simulations & test-beds, large-scale parametric sweeps/Monte-Carlos, simulation Services, and visualization. DSENDS uses a Python-based user interface for easy simulation construction and modification. DSENDS has been used a wide variety of missions and is currently in use at JPL and in the Missions Operation Directorate of NASA's Johnson Space Center.

Acknowledgments

We would like to acknowledge our partners from the JSC Missions Operations Directorate who have helped us improve DSENDS including Scott Nemeth, Carl Merry (who now works for Sierra Nevada Corporation), and Ronald Frye, as well as Scott Abadie, Darren Baird, Bryan Evans, Robin Friedrich, Christopher Glenn, John Gouveia, Daniel Ramirez, and Marius Zaczek.

This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References

[1] J. Balaram, R. Austin, P. Banerjee, T. Bently, D. Henriquez, B. Martin, E. McMahon and G. Sohl, "DSENDS - A High-Fidelity Dynamics and Spacecraft Simulator for Entry, Descent and Surface Landing," in *IEEE Aerospace Conference*, Big Sky, Montana , J, Austin, R., Banerjee, P., Bently, T., Henriquez, D., Martin, B., McMahon, E., and Sohl, G, March 2002.

[2] C. Lim and A. Jain, "Component Based, Reusable Space System Simulation Framework," in *Proceedings of the Third IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT 2009)*, Pasadena, CA, July 2009.

[3] NASA, NASA Tech Brief: Mission Operations and Navigation Toolkit Environment, September 2009: http://www.techbriefs.com/component/content/article/5702.

[4] A. Jain, C. Crean, C. Kuo and M. B. Quadrelli, "Efficient Constraint Modeling for Closed-Chain Dynamics," in *The Second Joint International Conference on Multibody System Dynamics*, Stuttgart, Germany Cory Crean, Calvin Kuo, Marco B. Quadrelli, May 2012.

[5] A. Jain, C. Crean, C. Kuo, H. von Bremen and S. Myint, "Minimal Coordinate Formulation of Contact Dynamics in Operational Space," in *2012 Robotics: Science and Systems Conference*, Sydney, Australia, July 2012.

[6] M. Pomerantz, A. Jain and S. Myint, "Dspace: Real-time Visualization System for Spacecraft Dynamics Simulation," in *Proceedings of the Third IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT 2009)*, Pasadena, CA, July 2009.

[7] S. Myint, A. Jain, J. Cameron and C. Lim, "Large Terrain Modeling and Visualization for Planets," in *Fourth IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT 2011)*, San Francisco, California, August 2011.

[8] Wikipedia, Agile Software Development, http://en.wikipedia.org/wiki/Agile_software_development.

[9] Wikipedia, Scrum (development), http://en.wikipedia.org/wiki/Scrum_(development).

[10] A. Jain and G. Rodrigues, "A Spatial Operator Algebra for Computational Multibody Dynamics," in *International Conference on Scientific Computation and Differential Equations (SciCADE'97)*, Grado, Italy.

[11] S. A. Striep, D. W. Way, A. M. Dwyer and B. J., "Mars Science Laboratory Simulations for Entry, Descent, and Landing," *Journal of Spacecraft and Rockets*, pp. 311-323, 2006, vol.43 no.2.

[12] M. I. Pomerantz, C. Lim, S. Myint, G. Woodward, J. Balaram and C. Kuo, "Multi-Mission Simulation and Visualization for Real-Time Telemetry Display, Playback, and EDL Event Reconstruction," in *AIAA Space 2012 Conference*, Pasadena, CA, 2012.

[13] NASA, Low Density Supersonic Decelerator (LDSD), http://www.nasa.gov/mission_pages/tdm/ldsd/index.html.