Lean Development with the Morpheus Simulation Software

Aaron C. Brogley[1]
L-3 STRATIS Division/NASA JSC Engineering Directorate, Houston, Texas 77058

The Morpheus project is an autonomous robotic testbed currently in development at NASA's Johnson Space Center (JSC) with support from other centers. Its primary objectives are to test new 'green' fuel propulsion systems and to demonstrate the capability of the Autonomous Lander Hazard Avoidance Technology (ALHAT) sensor, provided by the Jet Propulsion Laboratory (JPL) on a lunar landing trajectory. If successful, these technologies and lessons learned from the Morpheus testing cycle may be incorporated into a landing descent vehicle used on the moon, an asteroid, or Mars. In an effort to reduce development costs and cycle time, the project employs lean development engineering practices in its development of flight and simulation software. The Morpheus simulation makes use of existing software packages where possible to reduce the development time. The development and testing of flight software occurs primarily through the frequent test operation of the vehicle and incrementally increasing the scope of the test. With rapid development cycles, risk of loss of the vehicle and loss of the mission are possible, but efficient progress in development would not be possible without that risk.

I. Introduction

THE Morpheus project is one of twenty Advanced Exploration Systems projects in NASA's Human Exploration and Operations Missions Directorate being developed at the NASA Johnson Space Center (JSC). Morpheus was formed in its current state by a reduction in scope of Project M, which was a project designed to land a vehicle containing a humanoid robot, or Robonaut, on the moon in 1000 days[1]. In 2011, Project M was de-scoped into an Earth-based test vehicle now called Morpheus. The Morpheus vehicle is an autonomous robotic testbed that includes four fuel/oxidizer tanks, a gimbaled main engine, a reaction control system (RCS), and vehicle sensors such as Inertial Measurement Units (IMUs). Its primary objective is to test new green fuel propulsion systems and the ALHAT sensor package. If successful, these technologies may be incorporated into future landing descent vehicle used on the moon, an asteroid, or Mars.

In an effort to reduce development costs and cycle time, the project employs lean development engineering practices in its development of flight, ground, and simulation software. Lean development is a shift in paradigm from the traditional way of top-down systems engineering that NASA is more traditionally known for. The project relies on a feverish pace of testing, high tolerance of risk, and iterative development to achieve its mission objectives.

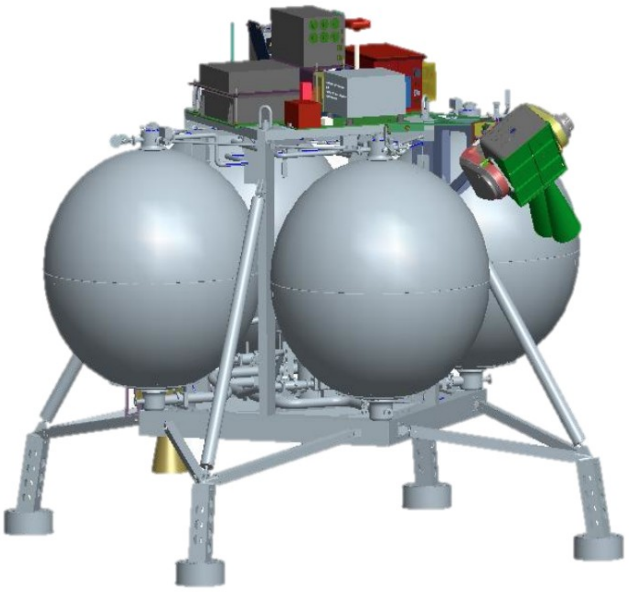

Figure 1. Depiction of the Morpheus flight vehicle with tanks, IMUs, sensors, and engines.

[1] Aerospace Engineer, Software Robotics and Simulation Divison, Spacecraft Software Systems Engineering Branch, 1002 Gemini, Suite 200, Houston, TX 77058, AIAA Regular Member.

II. Software Packages

One of the key cost-saving measures used on Morpheus is the reuse of existing software packages. The Morpheus team makes use of existing generic software packages that have already been tested and validated. These include the JSC-developed Trick© Simulation Environment (Trick), the JSC Engineering Orbital Dynamics (JEOD) package, the shuttle-era Valkyrie package, and Core Flight Software (CFS), developed at NASA Goddard Space Flight Center. In addition to these off-the-shelf packages, Morpheus-specific models and input scripts are stored in a local repository of our simulation.

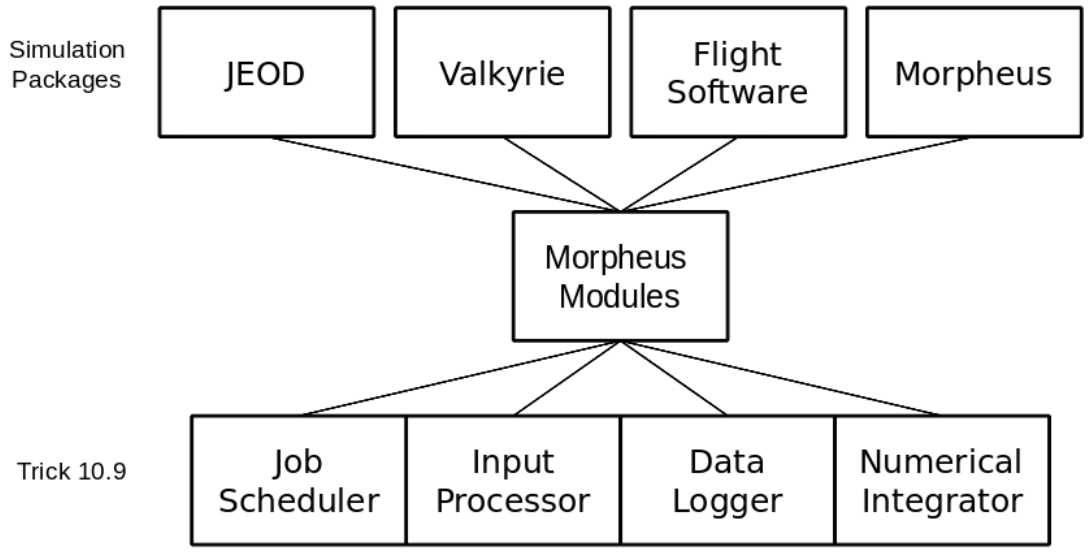

Figure 2. Morpheus simulation architecture and interface with the Trick Simulation Environment.

The Morpheus modules are a set of object-oriented C and C++ structures and classes that either inherit from models in other simulation packages or were written specifically for Morpheus. The simulation tool used for running the Morpheus simulation is Trick©. Trick© is a powerful simulation environment developed at JSC that completes many of the tedious tasks involved in simulation development, allowing a user to quickly set up a simulation. For Morpheus, it is used to set up the vehicle and its systems, collect gravitational and non-gravitational forces and torques, and numerically integrate the state of the vehicle and its actuators. Instances of the classes and structures that represent the Morpheus models are made known to Trick© and its job scheduler via declarations in S_define. The Trick© data recording utility and Monte-Carlo functionality make it incredibly useful for comparing many iterations of the Morpheus simulation. A representation of this ability is shown in Fig. 3, where 25 monte-carlo simulations of a tethered-flight simulation were run while varying the location of the initial center of mass. Trick© also has data port handling to allow third party applications to access live and recorded data. While its main function is to develop space vehicle simulations, Trick© may be used for any system that can be represented with differential equations[2].

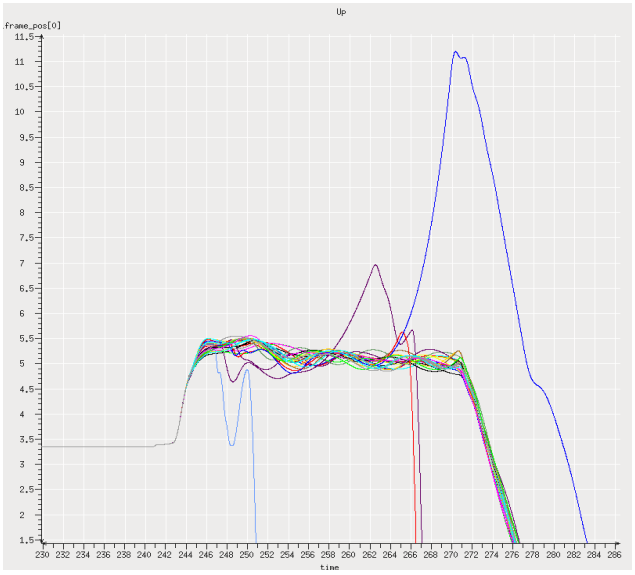

Figure 3. Monte-Carlo runs of Morpheus altitude in meters above sea level for 25 tethered-flight simulations.

The other existing packages are loaded from repository on a Morpheus simulation user's system. One of the main backbones of the Morpheus simulation is JEOD, which provides models for most non-vehicle environmental effects. JEOD contains a set of orbital dynamics models such as generic planetary bodies,

gravitation models, surface models, and atmospheric and other non-gravitational force and torque models. In addition, JEOD has its own numerical integration system, apart from those of Trick©. JEOD has been developed alongside Trick© since the 1990s and for many NASA simulations the two go hand-in-hand.

The Valkyrie repository provides generic vehicle models to use in the simulation. Its legacy comes from the Space Shuttle, but the code is generic enough to apply to Morpheus for many parts of the vehicle. The models include digital and analog sensors, fuel tanks, data interface and channelization, signal filters, interpolation methods, and effectors. These generic models function as they are or can be parent classes for several of the Morpheus specific models, where more specific sensor and effector performance can be obtained.

CFS is being used as the backbone of the Morpheus flight software (FSW). The CFS package is both a mission-independent and platform-independent FSW environment with a reusable Core Flight Executive (CFE). CFS has been used successfully on other unmanned missions. It contains a job scheduler as well as other utilities common to FSW. It serves as the orchestrator for the various FSW applications. Like the other software packages, CFS enabled the Morpheus team to focus on mission-specific development rather than coding from scratch. This saved the team valuable time and resources developing the FSW while also reducing risk by using tested code.

III. Design of Software

Morpheus-specific simulation models include analog and digital cards, the vehicle main engine, RCS jets, electro-mechanical actuators (EMAs), fuel slosh, tether effects, gimbal geometry, motion of the main engine due to gimballing (tail-wag-dog effect), ground contact, and sensors. The simulation is constructed of simulation modules, trajectory-specific input files, and common input deck files. The system is set up so that the user can run an embedded or hardware-in-the-loop (HWIL) simulation with a simple flag change.

The interface between the Morpheus simulation and FSW is handled in one of two ways. The primary way that the Morpheus simulation has been used is to share data across memory buffers. In this scenario both the simulation software and FSW are built on the same machine. The alternate method is using User Datagram Protocol (UDP) in which data packages are sent between the FSW on the vehicle computer and the simulation on another machine.

When running an embedded simulation, the FSW and simulation software run on the same machine. In a HWIL simulation, the FSW runs on the actual flight hardware and is connected to the simulation via UDP. The simulation modules contain instances of objects from the model libraries that Trick© will initialize with data from the input files use in its integration. The input files contain flags which dictate whether certain error sources are turned on or off in the simulation. It also has the ability to send commands to Guidance, Navigation and Control (GN&C) and propulsion in FSW. The common input deck files specify model data common across all test simulations. Different trajectories are selected for simulation by sourcing specific data files in the FSW.

Lean development on Morpheus has enabled telemetry data from flight tests to be compared to simulation models for tuning purposes. An example of this has been the parameter tuning of the EMAs. The actuators had at first been modeled in a way that did not accurately factor the loading that occurs during operation of the engine. This was picked up during the first tethered flights. Particle swarm optimization of the EMA parameters allowed the simulation team to compare telemetry data to the existing models. The EMAs were further tuned and refined with each subsequent tether flight.

Figure 4. Hot Fire Test 5 engine firing[3].

IV. Testing Cycle

The lean development process is apparent in the iterative way that Morpheus team tests the vehicle and refines software. The 2012 testing cycle saw an aggressive campaign that yielded many successes and many learning experiences. In general, the test operations proceed from hot fire engine testing to tethered flight testing to untethered or 'free' flight testing. The advantage of this method is it allows rapid test cycles, often only a few days between flights, while only adding on incremental risk with each subsequent test. In hot fire tests, the vehicle is strapped to the ground, allowing the main engine and the RCS

jets to burn. Testing the propulsion in this manner bolsters confidence in that subsystem before requiring GN&C to be involved. Then as the GN&C and propulsion are tested together with the rest of the vehicle subsystems in tethered flight, confidence in the system as a whole is improved. Figs. 4 and 5 are examples of the hot fire testing and tethered testing that occurs at JSC.

A brief summary of the 2012 Morpheus test campaign is shown in Table 1. The initial hot fire tests were performed in-house at JSC. Due to safety concerns with nearby buildings, free flight tests were performed at the Space Shuttle Landing Facility at Kennedy Space Center (KSC) in Florida.

Test	Date	Notes
Hot Fire Test 5	2/14/12	First M1.5 vehicle engine test
Tether Test 8	3/12/12	First tether at JSC
Tether Test 9	3/15/12	
hot fire Test 6	4/2/12	
Tether Test 10	4/5/12	
Tether Test 11	4/11/12	
Tether Test 12	4/17/12	
Tether Test 13	4/24/12	
Tether Test 14	5/4/12	
Tether Test 15	5/8/12	
Tether Test 16	6/11/12	With ALHAT on board
Tether Test 17	6/17/12	With ALHAT active
Tether Test 18	7/6/12	
Tether Test 19	7/16/12	Methane RCS test
Tether Test 20	7/31/12	Dry run at KSC
Tether Test 20	8/3/12	Wet run at KSC
Free Flight 1	8/6/12	Soft Abort shortly after ignition
Free Flight 2	8/9/12	Loss of IMU data causing loss of vehicle

Table 1. Summary of the 2012 test campaign.[4]

Figure 5. Tether Test 8 vehicle with plume[3].

The loss of the vehicle happened during Free Flight 2. The team was able to learn from data collected from telemetry, video, and the on-board computer that the cause of the vehicle loss was stale data coming from the SIGI IMU leading to a failure to update the inertial navigation state[5]. Immediately after liftoff, the vehicle experienced a modest pitch rate with 1.17g acceleration. That instant was also when the IMU data became stale, and the control subsystem maintained positive pitch correction based on the false vehicle state, flipping the vehicle. Loss of vehicle was recreated in the simulation by setting a termination of IMU data at 0.6 seconds after ignition, which is when the stale data flag was thrown by the SIGI sensor.

Figure 6. Remnants[3] of the Morpheus test vehicle after Free Flight 2.

While the loss of the vehicle is never a desired outcome of testing, it was understood to be an acceptable risk of the aggressive testing schedule. After the failure investigation, several requirements were added to FSW, hardware, operations, and simulation software. New Fault Detection Isolation and Recovery (FDIR) bits were added to the FSW applications. A backup IMU was added to the new Morpheus 1.5B vehicle. Plans for ground tether testing and the use of a flame trench are intended to both test the

ground effect and to mitigate vibration. For the simulation, the team planned a complete code scrub for the new vehicle. As the simulation environment was updated from Trick© 7 to Trick© 10, the simulation team was able to verify models and clean up less rigorous code structure.

V. On-going and Future Work

The lean development approach to testing is unaffected by the new requirements and is still in use going forward with the 2013 campaign. On-going work with the Morpheus simulation includes an automated process for validation of software that tracks changes in parameter sensitivity through the development of the both the simulation and FSW. A prototype now exists as the Simulation Automation Tool and this is in further development. There are also plans for improvements to the terrain contact model and reorganization of the propulsion models in the simulation. The 2013 flight test campaign with Morpheus vehicle 1.5B will match the pace and incremental flight envelope expansion that was planned for the vehicle 1.5.

Test	Date	Notes
Cryocart Test	3/28/13	
Wet Run & RCS check	4/4/13	
Hot Fire 7	4/23/13	First engine test of new M1.5B vehicle
Hot Fire 8	5/1/13	
Tether Test 21	5/24/13	First tether test of new M1.5B vehicle
Tether Test 22	6/6/13	
Tether Test 23	6/11/13	Abort due to loss of telemetry
Tether Test 24	6/14/13	Abort, then successful re-attempt of flight

Table 2. The 2013 Test campaign which is currently in progress.[4]

VI. Conclusion

The Morpheus vehicle has undergone a rapid development cycle made possible by lean development methods. This, along with the aid of the simulation software, has allowed production of FSW to be produced, tested, and validated quickly and cheaply. With the rapid development cycle comes risk of the loss of vehicle and loss of the mission. The team fully understands that some risk to the mission is acceptable in lean development. Morpheus has undergone a test campaign in the spring and summer of 2012 that would not have been nearly as productive had the project's goal been to minimize that risk. The aggressive testing schedule did result in the loss of the vehicle in its second free flight, but two additional vehicles were approved after the investigation. Adaptations in FSW, hardware, and simulation software have been made to prevent a similar vehicle loss in the 2013 test campaign cycle.

Appendix

A. Acronyms

ALHAT – Autonomous Lander Hazard Avoidance Technology
CFE – Core Flight Executive
CFS – Core Flight Software
EMA – Electro-Mechanical Actuator
FDIR – Fault Detection, Isolation, and Recovery
FSW – Flight Software
GN&C – Guidance, Navigation & Control
HWIL – Hardware-in-the-Loop
IMU – Inertial Measurement Unit
JEOD – JSC Engineering Orbital Dynamics
JPL – Jet Propulsion Laboratories
JSC – Johnson Space Center
RCS – Reaction Control System
Trick – Trick© Simulation Environment
UDP – User Datagram Protocol

References

[1]"Landing a Humanoid Robot on the Moon in a 1000 Days", 2010, URL: http://robonaut.jsc.nasa.gov/future/whitepaper/ [cited 17 Jun 2013].

[2]Vetter, K., Panter, D., "Trick Simulation Environment User Training Materials Trick 2010.0", NASA Johnson Space Center, 2010.

[3]Project Morpheus Home, URL: http://morpheuslander.jsc.nasa.gov/ [cited 14 Jun 2013].

[4]Davis, J., Crain, T., "Test Folder", FltDyn SE&I Wiki - Test Folders, NASA Johnson Space Center, 2013.

[5]Davis, J., Crain, T., "Morpheus Free Flight 2 Failure Investigation Page", FltDyn SE&I Wiki, NASA Johnson Space Center, 2012.