ANTARES: Spacecraft Simulation for Multiple User Communities and Facilities

Amanda Acevedo
NASA – Johnson Space Center
amanda.acevedo@nasa.gov

Jason Arnold
NASA – Johnson Space Center
jason.arnold@nasa.gov

Jon Berndt
Jacobs Engineering
jon.berndt@escg.com

Robert Gay
NASA – Johnson Space Center
robert.gay@nasa.gov

William Othon
NASA – Johnson Space Center
bill.othon@nasa.gov

ABSTRACT
The Advanced NASA Technology Architecture for Exploration Studies (ANTARES) simulation is the primary tool being used for requirements assessment of the NASA Orion spacecraft by the Guidance Navigation and Control (GN&C) teams at Johnson Space Center (JSC). ANTARES is a collection of packages and model libraries that are assembled and executed by the Trick simulation environment. Currently, ANTARES is being used for spacecraft design assessment, performance analysis, requirements validation, Hardware In the Loop (HWIL) and Human In the Loop (HIL) testing.

Keywords
Modeling and Simulation, Vision for Space Exploration, Crew Exploration Vehicle, Project Orion

1 Introduction

With the Constellation program ramping up, NASA is well into building up a simulation and analysis capability to support a broad set of needs. At the lowest level, engineers at various centers need to be able to execute short, single vehicle runs (such as a Crew Launch Vehicle (CLV) ascent run, or for a Launch Abort System (LAS) run) on a desktop workstation. On a broader scale, distributed simulations will be implemented, tying together the efforts of various centers. Underlying the development of these capabilities is a desire to avoid duplication of efforts, and adhering to agreed-upon conventions and established standards to maximize compatibility.

2 Simulation Development In A Distributed Environment

The Constellation Program represents an interesting challenge for engineers and simulation developers. The Program must integrate the activities of a variety of groups, including: 1) different NASA centers, 2) various government contractors and subcontractors, and 3) spacecraft element designers and builders. The various areas of development include vehicle hardware systems, ground and mission operations, mission support infrastructure, and information technology. The coordination and integration of these diverse groups is a challenge that must be addressed early in the Program lifecycle.

From a modeling and simulation perspective, there are several core areas that bear special attention in this environment. These include the development of model interface standards, the development of data exchange mechanisms, and the establishment of integration techniques to support multi-system testing.

Model Libraries

Each development group will be creating models of important elements of the design process. These models can include vehicle subsystems, environmental effects and other mission environment models, displays and controls interfaces, etc. Ideally, groups would prefer to leverage off *authoritative models* of the elements being analyzed. In the absence of cooperation, individual projects are often left to build interface simulations that emulate the performance of the elements being integrated. But this is often based on the Interface Requirements Documents (IRDs), not necessarily on how the elements are actually being built. One approach is to develop *model interface standards*, to allow groups to share models of their systems with others. Once these standards are created, a *common model library*, can be established to support collection and distribution of these models. By using authoritative models that conform to an interface standard, integrated vehicle analysis can be done with higher fidelity. Subsequent errors or problems with system interfaces can be identified much sooner, leading to reduced cost.

Configuration Management and Issue Tracking

Due to the multi-project, multi-center user base it is important to adhere to a well-defined Configuration Management (CM) process. This aids in leveraging off others' work, efficient integration of many users' development and testing efforts (both local and remote users), preservation of the simulation baseline, easy recreation of previous baselines, and delivery of internal ANTARES releases as well as external deliveries to remote users.

The requirement for protection of the official baseline and allowing parallel development among users is also critical to efficient development and building trusted deliveries to users internal and external.

An issue-tracking tool that allows us to monitor status of work, track change requests and gather project metrics on a periodic basis is necessary to our build process. The tool used for issue tracking must be accessible to all users and contributors to submit, track and prioritize development as it progresses thru the software life cycle and is integrated and built into an official

ANTARES delivery.

These methods that the ANTARES community employs assist greatly in achieving the goal of sharing model libraries and individual models.

Integration Methods

The diverse development communities will be creating M&S tools to evaluate and test component designs, within a particular system in the Constellation Program. These tools represent the best knowledge of the performance of the system. To leverage off this knowledge, it is possible to create *integrated simulation architectures* to coordinate and interface the various tools to create an *integrated system analysis tool*. This integration can take one of two forms: *tight integration* and *loose integration*.

Tight integration represents the ability to absorb models of external systems directly within the M&S tools themselves. The common model libraries described before help facilitate this type of model exchange. Tight integration may be required within systems that demand the highest performance possible. For instance applications with hardware in the loop must reduce response latency as much as possible. Engineers performing Monte Carlo and other computationally-intensive tests require that the simulation run as quickly as possible. So tight integration may be a requirement for these teams.

Loose integration represents the coordination of M&S tools across possibly geographically dispersed groups. This type of test requires the application of distributed simulation technologies such as the High Level Architecture (HLA) or the Common Object Broker Request Architecture (CORBA). This type of distributed testing allows for early integration of simulation elements and early interface testing among systems. Instead of waiting for all software components to be co-resident a distributed capability can be used with software resident in its native development environment. This has the added benefit of engaging distributed systems developers together much earlier in a program lifecycle. So errors can be capture more quickly, and relationships among the project teams can be formed sooner.

And these loose and tight methods are not mutually exclusive. In some cases loose integration can be applied to tight integration problems. Loose integration techniques refined across distributed groups can be utilized in close proximity, reducing the latency and bandwidth problems associated with large, widely distributed simulation architectures. So for instance two processes that were initially separated by thousands of miles can be co-located within the same facility or even the same computer but continue to use the same interfaces.

Data Interoperability

When dealing with multiple different organizations distributed geographically around the country, the issue of data interoperability becomes important. Each project or group may have different ways of defining or identifying data elements. Local project databases will contain parameter attributes that will be useful to other projects. Another problem is how the data is physically exchanged, either as communication between database systems, file exchanges, or data streams shared by real-time processes.

Within Constellation there are two related activities that are addressing the issues of Data Interoperability. The Command, Control, Communication, and Information (C3I) team is defining specifications to support communication among Constellation projects during mission operations. These specs will be used to help define telemetry stream formats, exchange of voice and video data, and file transfer. C3I specs can also be used for ground infrastructure issues associated with distributed elements of facilities.

The NASA Exploration Initiatives Ontology Model (NExIOM) team is creating a specification for a data exchange mechanism. A NExIOM schema and data architecture model is being created to help support data exchange among the different projects. This effort focuses on how the data is defined and stored, what the important data attributes are that support project operations, management, and control. An NExIOM-compliant Extensible Markup Language (XML) format is being defined to implement the exchange mechanisms among the various groups.

3 Models/Libraries/Externally Delivered Models

The ANTARES simulation comprises of various different models, libraries and externally delivered packages and must integrate them all together to provide for flight dynamics analysis and development while at the same time maintaining

modularity to allow for rapid model/library/package upgrade and enhancements.

GN&C Simulation Architecture

The core user for ANTARES is flight dynamics analysis and testing and integration of core GN&C software. In order to facilitate rapid prototyping, the GN&C interface inside of ANTARES was developed to allow for exchange of software prototype models and to also facilitate highly desirable analysis capabilities like variable fidelity versions of GN&C software running in tandem. In addition, ANTARES is to support flight dynamics work across flight phases so the GN&C architecture must also support the capability of an "end-to-end" simulation functionality. To emulate the eventual reality of the flight software and vehicle systems, the GN&C simulation architecture contains several high level functions. At the highest level are the vehicle and GN&C executives, the vehicle executive handling vehicle specific functionality and the gnc executive handling the overall GN&C functionality. In addition there are several managers for each subset of GN&C to manage and interface with each other and the GN&C executive. Each executive and manager has a clear set of division lines based on responsibility and this allows for an easy exchange of models as a developer/analyst has to be concerned primarily with I/O versus internal functionality. The GN&C simulation architecture ties together most of the internal models and many of the eternal packages inside of ANTARES . In addition to the core GN&C software ANTARES is comprised of some core model sets that have been developed by multiple groups and combined to produce this high fidelity six degree of freedom simulation.

The Common Model Library

There is more than one way in which simulators can reuse or share models. Historically, simulators used at one location might be tied very tightly to a specific piece of hardware (such as a cockpit mockup), to a local operating approach, and may involve the use of different programming languages. Sharing of simulation software components can be tedious. Increasingly, however, simulators are being built that use a more generic software architecture, and are more data-driven. Some aspects of a flight simulation should be very similar for all simulations: the equations of motion (EOM), data logging, real-time scheduling, ephemeris data, etc. So, the logical question follows: can a broadly capable, multi-purpose, flight simulation be entirely data-driven? There are steps in that direction, in some quarters. For example, the open source flight simulator, FlightGear [Ref. …], and its various flight models JSBSim [Ref. …], YASim, and a modified version of LaRCSim [Ref. …], are all generic and driven by data stored in an XML format. Everything from instrument panels, to navigation instruments, to flight control systems, aerodynamics, and so on are all described in XML configuration files – effectively offloading the task of creating a specific simulation model from a programming one, to one of populating a configuration file with appropriate data. That works well within the aircraft world of FlightGear, and with relatively simple GNC systems. But, it's still not really portable to other simulation installations to the point that aircraft models can be shared.

Recognizing that model sharing can foster an accelerated schedule and help to lower costs, an effort was begun in 2002 [Ref. Jackson and Hildreth, AIAA-2002-4482] to move forward a standard for flight model exchange using a file format built on XML. That standard has been written and is in the approval process at this time (Summer 2007). The idea with DAVE-ML [Ref. http://daveml.nasa.gov] is to provide a common data format as a medium of exchange between simulations. DAVE-ML is analogous in a way to a memory stick, which is formatted and can hold data in a particular way, thus allowing the memory stick to be plugged into one machine, loaded with data, and carried to another machine, with the data being dumped at the target machine. Currently, one part of DAVE-ML is being proposed as a standard: AERO-ML. AERO-ML codifies a mechanism to store aerodynamic data, tables, functions, check cases, etc. Support mechanisms for AERO-ML have been appearing that make it an increasingly attractive technology. One of those is the open source Janus library, which reads an AERO-ML file [Ref. Janus paper/site].

Another way that models can be shared is to write specific models so that data is not stored by the function, but is only passed in. That way, models that are needed can be assembled together and handled by a larger simulation manager.

For ANTARES, the desire to lower development and testing costs has been addressed by creating a Common Model Library (CML). The CML provides a central repository where developers can contribute software models that conform to a lightweight coding and interface standard, where current and future projects at any center can draw from to incorporate tested and documented models.

The CML is a collection of generic models and prototype flight software of varying fidelity, allowing the system to be matured gradually through the lifecycle of the project. It incorporates models from a variety of sources and authors. Note that the resources and responsibility for support of any given model in the CML resides with the author. Appropriate and complete documentation must accompany each model, as well as complete unit tests with comparison data. NASA has at its disposal a long list of legacy software models. A number of these models that are useful in meeting current needs have been

brought into the CML. As an example, for early Crew Launch Vehicle simulation a Solid Rocket Booster model was taken from an existing, well tested, space shuttle engineering simulator used at the Johnson Space Center after being slightly modified to conform to function calling and data storage constraints. Likewise, a space shuttle main engine model has been ported over and modified to model the J-2 rocket engine.

The task to design the next generation of NASA spacecraft will engage engineers from across the United States, including multiple NASA centers and contractors. Designing a library to incorporate elements from multiple different authors and disciplines will enable this distributed group to get the most out of the modeling and simulation elements being produced.

Aero CAP

Some functionality in ANTARES is delivered via a package which is developed by domain experts and then inserted into the simulation. The NASA Orion team will be providing the official aerodynamics/aerothermal products for the Orion project. The aerodynamics data base will be provided by the CAP (CEV Aerosciences Project) team and have an API that can interface with the simulations which need to utilize aero data to produce analysis for design and requirements decisions. The CAP product is GFE and will be used across the project for all Aerosciences needs. In ANTARES the CAP api provide a means by which the simulations can get access to any Aeroscience related data necessary. Via the API data exchange is straightforward and most importantly enables rapid updates to the data base as new versions are released since the only method of interaction between core ANTARES and the Aero CAP API is data exchenge.

JEOD

JEOD is the JSC Engineering Orbital Dynamics package and is the core dynamics capability within ANTARES. It is a package developed and maintained by the Automation, Simulation and Robotics division at JSC and has gone through a rigorous V&V activity in recent years. Like the CAP product, JEOD will be used across the project for in various simulations for core dynamics functionality. It's core capability consists of gravity models, planetary ephemeris, orbital mechanics, and other core dynamics functionality. Much of JEOD is in the form of generic dynamics equations, data driven gravity models and other simulation independent models. The advantage to that is that on it's own it is more simple to verify and validate the models as the basic functionality has a long heritage, so there is flight data available for a lot of the validation and domain expertise to provide verification.

4 Process

The ANTARES project has chosen tools from the IBM Rational tool suite to handle Configuration Management and issue tracking. ClearCase is the tool used for CM and Distributed Defect Tracking System (DDTS) is in place for issue tracking. Both tools have a long and successful history in the Aeroscience and Flight Mechanics Division (AFMD) at JSC for many projects. The tools readily meet the needs of the ANTAERS user base described in the "Configuration Management and Issue tracking" sections above.

CLEARCASE

ClearCase provides and easy interface to code bodies for many users without the users impacting each other's work (parallel development). Additionally, thru the use of a configuration specification file users may easily configure their environment to view and work with any version of a file, simulation, testing scenario etc.. ClearCase readily supports early integration and collaboration of developers thru the use of "Labels". Labeled files, directory structures, data, executables and any other CSCI may be shared among developers by simply adding a line to their configuration specification file.

ClearCase also provides a GUI interface to the Version Object Base (VOB) which contains information on all versions of any element in the CM system. This tool allows for a clear visual picture of the complete history of any element in the VOB. All ClearCase actions/commands that are available via command line interface are also available to the user thru the GUI. This capability aids greatly in understanding the evolution of a file as well as comparing versions, and resolving development and integration issues.

DDTS

DDTS provides access to the issue-tracking database for all ANTARES users both local and remote. To ensure that development is scheduled, planned, prioritized and coordinated there must be a central database of all proposed and ongoing development and tasks within the project.

DDTS meets these needs and has proven to be an efficient tool for the ANTARES project. Developers and users locally and at other centers have adopted our use of this tool and often contribute their development or request modifications or enhancements thru the use of this tool. Combined with a weekly tag up where we coordinate activities, schedule and priorities we are able to keep our process running smoothly and efficiently. Integration of so many users' work can often be challenging but with the combination of selected tools and a regular planning meeting we have been able to perform

successful integrations.

CEV, CML, TRICK, ETC…

As described, ANTARES is compromised of many different models developed both internally by the Orion team and externally by groups in various organizations at JSC and at NASA centers across the country. In order to provide a clear division between model sets and to facilitate ease of model and package sharing in addition to generic interfaces in ternal to the simulation, ANTARES also utilizes a top level directory structure designed to break up models and packages for interchangeability and quick upgrading from a CM perspective. All Orion/CEV specific models and data reside in a directory at the top level called cev. All CML models from the Common Model Library are stored in the cml directory and many of the models here are heritage models from other programs and projects and are designed to be as generic as possible based on their applications. Models specific to the ISS modeling reside in their own directory maintained in tandem with the cml. Externally delivered packages also have their own directories, Trick the simulation framework, JEOD the core dynamics package, and DOUG graphics all reside in their own directory structures as delivered. The Aero CAP API resides within the CEV VOB however it has it's own directory structure to maximize integration in the simulation while at the same time easing the CM process. As new packages, enhancements etc… are introduced they are put in the proper place within the cev or cml/iss directories or the packages themselves are upgraded in the case of Trick, JEOD and DOUG.

VV&A

With all the advantages of model and simulation sharing comes the responsibility of providing full and accurate information about the Verification and Validation status of a given model or simulation so that teams who would like to leverage off the existing work and authoritative models and simulations can perform an accurate accreditation of the tool for their specific purposes. Many of the models in the ANTARES libraries have varying levels of testing, fidelity, and documentation associated with them. This is to be expected at the early life cycle phases of a project. As the project evolves so will the attributes of the models that comprise the model libraries.

It is necessary to maintain an up to date catalog of the current credentials of all models in a given library and provide this to the users. This will include information and status on testing to include unit tests, integrated tests and code coverage metrics, an assessment of V&V status, listing of documentation, and a description of level of fidelity of the model.

The accreditation of any model or simulation/tool is essential to ensure that the tool is being used as intended and that its attributes are appropriate for the analysis or task being performed. A template has been created for the Constellation Project to aid in identifying and documenting accreditation criteria and system level expert sign-off for each task that uses a given tool.

The accreditation effort must be performed for each tool involved in an analysis task. The template has been designed to be a low impact document that should not add significantly to the workload already being performed for an analysis task.

Doxygen

With ANTARES being comprised of so many individual files and models, and with the data structures being so numerous and nested, keeping track of them all during development (especially for new developers) is difficult. Being able to follow the data flow in the program code is a necessary capability. To help to meet that need, the open source tool, Doxygen, was utilized [Ref. Doxygen web site]. Doxygen processes source code files, reads comments that are formatted in a particular way, and produces documentation files in HTML. Trick already has a similar capability, but Doxygen takes it farther by creating an index of functions and data items and hyper-linking to specific instances of their use in the source code itself – which is also hyperlinked back to structures and other function calls. The result is a massively hyperlinked set of files within which developers can navigate through the code and data structures, rapidly gaining a comprehension of how the program code passes data and calls functions. The key to using Doxygen with ANTARES code lies in the ability of Doxygen to call a script just prior to processing any file. Since the ANTARES code was not originally written with Doxygen in mind, there are no formatted comments in code that Doxygen would normally parse. Files are commented in a way that Trick understands, however. A perl script was written that creates some formatted comments on-the-fly as a file is being parsed by Doxygen. That allows Doxygen to recognize a majority of the function prototypes, variables, function calls, etc.

CONCLUSION

ANTARES addresses the many problems introduced by a project of very large magnitude by leveraging off of existing expertise while at the same time establishing a data driven format for simulation that enables rapid updating, analysis and VV&A. It makes it possible for a developer or analyst to make a quick run at their desk and then use the same simulation to communicate literally across the country allowing for multi-center collaboration. Early efforts have been made to streamline the data input process to make clean interfaces between large externally delivered packages and to collaborate between developers across the country in order to avoid duplication of effort, adhere to agreed upon standards and maximize the compatabiliity of the simulation.

ACKNOWLEDGEMENTS

REFERENCES

1. Author,"Title", reference, date