View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by NASA Technical Reports Server

BENEFITS OF A UNIFIED LaSRS++ SIMULATION FOR NAS-WIDE AND HIGH-FIDELITY MODELING

Patricia Glaab, NASA Langley Research Center, Hampton, VA

Michael Madden, NASA Langley Research Center, Hampton, VA

Abstract

The LaSRS++ high-fidelity vehicle simulation was extended in 2012 to support a NAS-wide simulation mode. Since the initial proof-of-concept, the LaSRS++ NAS-wide simulation is maturing into a research-ready tool. A primary benefit of this new capability is the consolidation of the two modeling paradigms under a single framework to save cost, facilitate iterative concept testing between the two tools, and to promote communication and model sharing between user communities at Langley. Specific benefits of each type of modeling are discussed along with the expected benefits of the unified framework.

Current capability details of the LaSRS++ NAS-wide simulations are provided, including the visualization tool, live data interface, trajectory generators, terminal routing for arrivals and departures, maneuvering, re-routing, navigation, winds, and turbulence. The plan for future development is also described.

Acronyms

ADRS	Aeronautical Datalink and Radar Simulation
ARINC	Aeronautical Radio Incorporated
ASAB	Aeronautics Systems Analysis Branch
ASTOR	Aircraft Simulation for Traffic Operations Research
ATM	Air Traffic Management
ATOL	Air Traffic Operations Lab
ATOS	Airspace and Traffic Operations System
CMF	Cockpit Motion Facility
CSAOB	Crew Systems and Aircraft Operations Branch
DOF	Degrees of Freedom
FAA	Federal Aviation Administration
FMS	Flight Management System
HITL	Human in the Loop
JNI	Java Native Interface
JSC	Johnson Space Center
KTG	Kinematic Trajectory Generator
LaRC	Langley Research Center
LaSRS++	Langley Standard Real-time Simulation in C++
MACS	Multi-Aircraft Control System
MAVERIC	Marshall Aerospace Vehicle Representation in C
MSFC	Marshall Space Flight Center
NAS	National Airspace System
SDAB	Simulation Development & Analysis Branch
SID	Standard Instrument Descent
SMART NAS	Shadow Mode Assessment using Realistic Technologies in the NAS
STAR	Standard Terminal Arrival Route
VM	Virtual Machine

Background

In 2012, the Langley Standard Real-time Simulation in C++ (LaSRS++) was extended to allow the framework to support systems-level simulation of the National Airspace System (NAS), also called NAS-wide simulation. The LaSRS++ high-fidelity vehicle simulation actively supports studies involving commercial transport and military aircraft, launch vehicles, and spacecraft. Simulations are hosted as fast-time on the desktop, with humans-in-the-loop (HITL) in research cockpits, and in flight tests using Langley Research Center (LaRC) aircraft. The simulation architecture was already designed to support varying research missions on varied platforms. With the addition of the NAS-wide modeling capability, this range of research was extended to include systems-level analysis of the

airspace, primarily in support of the Next Generation Air Transportation System, or NextGen.

The initial implementation of the LaSRS++ NAS-wide model was performed as a proof-of-concept to verify that the LaSRS++ high-fidelity C++ program was able to handle the very large number of objects and fast processing speed required to support NAS-wide modeling. After successful proof-of-concept testing, work progressed to evolve the LaSRS++ NAS-wide simulation with models needed to support systems-level airspace research.

Creation of a single software framework to support Langley technology concepts offers significant potential cost benefits. This cost savings will result from the use of an existing in-house LaSRS++ software support team that can be tasked on an as-needed basis. Although some models and techniques are unique to the new NAS-wide paradigm, the shared style and architecture of the LaSRS++ NAS-wide and high-fidelity vehicle framework allows software developers to transition quickly to the NAS-wide paradigm. Another important benefit is the greater potential for communication, model sharing, and mission testing that is facilitated by allowing different research groups to operate in a single, unified environment.

A similar set of goals was envisioned in 1997 when the LaSRS++ simulation framework was extended to support the simulation-to-flight mission for the LaRC 757 aircraft. This capability resulted in an estimated savings of $17M in the first 10 years of use [1]. Once the LaSRS++ NAS-wide simulation is ready for research, it is expected to save more money *per year* than the total cost of its development.

The Aeronautics Systems Analysis Branch (ASAB) is providing the expertise for the initial NAS-wide model development. Models hosted in other NAS-wide simulation programs cannot be ported directly to LaSRS++ because of the differences in language and calling structure. Therefore, the Langley Simulation Development and Analysis Branch (SDAB) is providing expertise to stage the models in the LaSRS++ architecture. The successes achieved to date are a result of this collaboration.

Benefits of High-Fidelity Simulation

High-fidelity vehicle simulations are critical to determining the feasibility and acceptability of new technologies to pilots and controllers who must use them. This is facilitated early in the design by human testing through HITL simulation. This style of simulation attempts to present as accurate a depiction of reality as possible to the pilot subject flying the mission for research and evaluation. The simulation scenario is staged from the point of view of the primary aircraft being modeled, also called the "ownship". These simulations are used to evaluate display concepts, pilot workload, handling qualities, and vehicle response. They can also help determine exactly how a concept technology can be staged in consideration of existing hardware systems on an aircraft, which is a key factor in assessing the cost of adoption of the technology to operators.

Benefits of NAS-Wide Simulation

NAS-wide simulation modeling studies became prevalent in the early 2000's, enabled by increasing compute speed and modern software languages. Unlike high-fidelity vehicle simulations which focus on the details of the ownship, NAS-wide simulations focus on the overall picture. Details of how aircraft characteristics are achieved are not as important as succinctly capturing the behavior of the vehicle so it can be tested in the larger flow. These simulations were developed to support cost and benefit assessment of the impact of individual technology concepts deployed at a NAS-wide scope. Improvements that seem minor for a single vehicle can demonstrate substantial system-wide benefit if applied to a large number of flights. Conversely, a seemingly substantial benefit that can only be realized by a few flights or in a limited region may not justify the infrastructure cost of adoption.

A substantial benefit of NAS-wide simulations is the allowance for emergent behavior in the resulting flow dynamics. Emergence is a process whereby larger patterns and regularities arise through interactions of smaller or simpler entities. In a NAS-wide simulation, these emergent behaviors can reveal flow characteristics that are difficult or impossible to predict when the individual model is considered separately or at a limited scale. These

unpredicted consequences can be the "Achilles heel" of technology products making their way into larger NAS testing for the first time. A NAS-wide simulation can uncover many of these issues early in the development process, often remedied with minor changes if realized early enough.

NAS-wide simulations must run their full scenario in a very short amount of time to be useful, preferably in minutes rather than hours. To model this many aircraft concurrently and interactively on a single computer (often a laptop), they use aircraft models that are at lower detail level than their high-fidelity vehicle simulation counterparts. Dynamic actors in NAS-wide simulations are often based on parameterized models, and the output metrics relate to systemic properties (like total delay, average throughput, and overall fuel reduction). The NAS-wide aircraft can also contain action and response algorithms to portray pilot behavior. This allows each of the tens of thousands of flights to be an intelligent actor in the scenario in a repeatable fashion and without the need for support hardware (like pseudo-pilot stations).

Unified Vehicle and NAS-Wide Simulation Approach

The adoption of promising new NextGen technologies into real-world use will require substantial changes to the existing Air Traffic Management (ATM) infrastructure by the FAA and industry stakeholders. The costs associated with these changes must be justified by the NAS-wide benefit and vetted for user acceptability. Technology concept development using an iterative approach with both high-fidelity and NAS-wide modeling can quantify these benefits more completely and demonstrate acceptability, resulting in a product that is commercially viable earlier.

At the same time that NAS ATM stakeholders demand more from the research community, budgets provide less. Efforts to integrate labs or to re-host products into new test beds can be tedious and expensive endeavors when done on a project-by-project basis. This type of integrated capability is only cost effective if it can be shared by many projects and can provide continued return on investment over time. No one project can absorb this cost within their allocated time and budget. Once available, however, many projects can benefit.

This same philosophy is a driver for NASA's current investment in the Shadow Mode Assessment using Realistic Technologies for the National Airspace System (SMART NAS) initiative. SMART NAS is a simulation framework that is expected to accelerate the transformation of the NAS by providing a platform for more comprehensive testing of integrated airspace concepts. [2] NASA is investing in the design and development of the SMART NAS capability up front, expecting that the return on investment will be exponentially greater than the cost of development.

Impediments to a Unified Simulation

An impediment to iterative testing in both a high-fidelity and a system-level environment is the unique tool sets used by each group. Research simulations, regardless of their style, are supported by complex executives and architectures. Transitioning models from one environment to another is much more complicated than simply relocating the software and supplying inputs and outputs. Architectural constraints often require changes to the software to allow it to operate properly within a different calling scheme and using a different set of available state variables.

Differing software language conventions for real-time versus NAS-wide simulations also impede iterative concept testing. Most NAS-wide simulations are in Java. Simulations which require hard-deadline real-time operation (which includes the LaRC HITL simulators) cannot use Java because of the garbage-collection process, which is used by the Java Virtual Machine (VM) to clean up unused memory. Though the user can tune the way Java runs the garbage collection and can set recommended limits on its duration and frequency [3], it cannot be turned off entirely.[1]

[1] Disabling the garbage collection was removed after Java 1.1, although the syntax was supported until Java 1.4.

For a system without real-time operational constraints, the Java garbage-collector is a valuable service that eliminates the tedium of tight memory management. However, the process is inherently non-deterministic in duration. This prevents the level of control required for the tightly managed real-time frame. Simulations that must support hard-deadline real-time operation are written in languages that allow the programmer to control *all* system calls and memory allocation. Legacy real-time simulations were often written in FORTRAN. Modern frameworks often use some derivative of C. For example, the Johnson Space Center (JSC) "Trick" simulation is written in C--, Marshall Space Flight Center's (MSFC) Marshall Aerospace Vehicle Representation in C (MAVERIC) simulation is in C and C++ [4], and LaSRS++ is written in C++. Java and C-based language developers also use different coding styles, which contributes significantly to the difficulty in transitioning between existing real-time and NAS-wide paradigms.

Table 1 captures some of the differences in vehicle versus NAS-wide simulation frameworks.

Trait	Vehicle Simulation	NAS-Wide Simulation
Language	Usually a version of C or FORTRAN	Usually Java
Programming considerations	Memory allocation and system calls must be completed before or after the real-time run. System calls are non-deterministic in length and often cause frame overruns (a simulated second takes more than a second of wall clock time). Frame overruns cause a buildup of error between simulation states and internally computed states of avionics and simulator hardware, such as a control loader or Flight Management System (FMS).	Overall time to execute the run is prioritized, rather than consistency in timing between individual frames. Techniques like distribution of operations and event-driven computations are frequently employed to increase overall run speed.
Clock	Can support real-time or fast-time	Usually only fast-time
Length of run	Simulated runs span minutes, seldom more than a few hours.	Simulated runs can span several days (in fast-time)
Viewpoint	From the ownship (vehicle being simulated)	Bird's eye view (no ownship)
Vehicle Lifespan	Simulated vehicle (ownship) is active for entire duration of run	Flights enter and exit the simulated day[s] at designated departure and landing times
Metrics	Typically relate to the ownship (aircraft state data, environmental states, Cooper-Harper scales for pilot workload, pilot controls movement)	Typically relate to system characteristics (averages/ totals of measured states of all flights)
Traffic intelligence	Traffic (non-ownship) movement is usually predetermined (e.g., based on a previously recorded path) or is assisted by pilot actions with a simplified cockpit (a pseudo-pilot station).	Traffic aircraft have artificial intelligence to respond to simulated situations through a "pilot" model. All flights can be intelligent actors.

Table 1. Characteristic differences between vehicle and NAS-wide simulations

Unified LaSRS++ High-Fidelity and NAS-Wide

While the Langley airspace research community can benefit from an iterative concept development and test approach offered by both vehicle and NAS-wide simulations, this was not a cost-effective option in the past for reasons mentioned in the previous section. The time and effort required to re-host a software model from an event-based Java calling scheme (as typical for NAS-wide simulations) into a time-based C++ calling scheme (for Langley real-time vehicle simulations) is difficult to the point of being impractical, and so rarely occurs.

This situation is being remedied for the Langley research simulation community by the expansion of the existing LaSRS++ vehicle simulation to support NAS-wide modeling. This effort began in 2012 and continued as a grass roots effort supported by one or two developers at a time, but with considerable progress in the two years since it was first proposed.

Langley ATOS and ATOL Facility

The Langley Airspace and Traffic Operations (ATOS) and Air Traffic Operations Lab (ATOL) are development and test systems for new air traffic management concepts and airborne technologies [5]. This lab allows pilots and controllers to assess the usability, feasibility, and acceptability of new flight deck technologies.

Figure 1. Langley ATOS Monitoring System [6]

Within this lab are numerous "Aircraft Simulation for Traffic Operations Research" (ASTOR) stations, which can be configured to support single or dual crew operations. ATOL and ATOS are operated by the Crew Systems and Aviation Operations Branch (CSAOB) at LaRC. The lab can model hundreds of interactive background aircraft to create realistic scenarios as staging for the live pilot test subjects. Though the software is maintained independently by CSAOB, the systems have some commonality with the SDAB HITL simulator lab run using LaSRS++. The ATOL lab is also written in C++ and runs in either fast-time or real-time. The lab can run independently, or in joint simulations with the Cockpit Motion Facility (CMF) simulators linked to ATOL for expanded test missions. ATOL also has access to the LaSRS++ software repositories and configuration management system, and reuses some of the aircraft models from LaSRS++. Therefore, models that are developed under the LaSRS++ framework for the NAS-wide models will also be available to ATOL developers. This may be particularly useful for sharing the SMART NAS interfaces that will be created once that system is available.

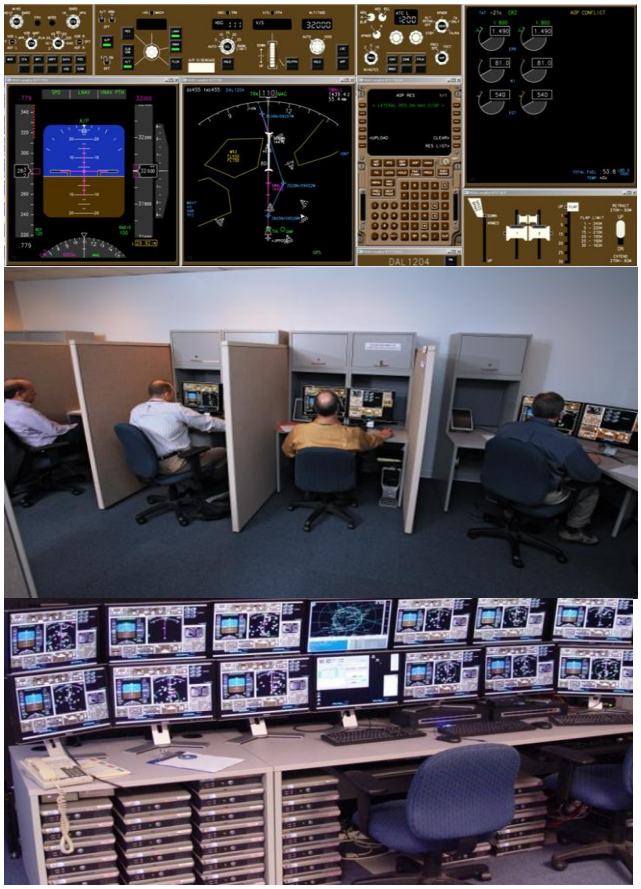

Figure 2. ATOL Displays and Controller Stations [6]

LaRC LaSRS++ Framework

Extension of the LaSRS++ simulation to support NAS-wide operation was simplified by the software architecture, which was designed to provide flexibility for varied operational scenarios and was extended several times in its lifespan to that end [7]. The simulation was originally adopted in 1995 for use in the any of the simulator cockpits maintained by SDAB, including Langley's Cockpit Motion Facility (CMF), or with fast-time operation on the desktop. In 1998, the mission for LaSRS++ was extended to additionally provide the research system for the Langley 757 aircraft, and eventually for all Langley aircraft running flight test software. In 2005 and 2008, Mars and Moon environment models were added, respectively, to support the space science research.

In 2011, LaSRS++ was extended to add a distributed simulation capability for traffic modeling. This allows LaSRS++ vehicles to send and receive situational data required to model ADSB antenna communication using traffic states supplied by ASTOR models of the LaRC ATOS. It also provides a one-way gateway to Ames' Aeronautical Datalink and Radar Simulation (ADRS) which allows LaSRS++ vehicle simulations to use the Ames' Multi-Aircraft Control System (MACS) as an ATC station. For the NAS-wide model, a new simulation entity was created to manage the life cycle of NAS traffic objects within the local simulation. This "TrafficFlowManager" component builds models into the simulation as simple aircraft that have type-specific parameter-based trajectories, pilot decision-making, and which can respond to airspace management requests. TrafficFlowManager also controls the lifecycles of the airspace management components, which provide scheduling and routing directives to initiate and adjust the flow of traffic.

Figure 1 presents the high-level class architecture of the three model styles within LaSRS++. Any of the three model types can run independently or together in a simulation scenario. In this diagram, the ATOS is the supplier of ADSB information, but this data can alternately come from the Ames MACS system or from playback data from a previous run.

SMART NAS Shared Development

The timing of this effort will allow both the vehicle and NAS-wide LaSRS++ research communities to share interface tools to SMART NAS as they are added to the framework. The new SMART NAS interface system will have the benefit of two distinctly different operational paradigms contributing to the design from the outset. LaSRS++ is expected to be one of the early-adopters of SMART NAS technologies. By developing an interface to serve both the NAS-wide and HITL simulators from the outset, overall cost of transitioning to SMART NAS will also be minimized.

Current Status of LaSRS++ NAS-Wide

The majority of the simulation architectural and infrastructure changes needed to support NAS-wide simulation in LaSRS++ were completed during the feasibility study in late 2012 and early 2013. Once the NAS-wide framework changes were completed, models running within it were extended and matured to begin to provide functionality to ready them for research.

Progress Monitor Visualization Tool

Emphasis was put on early development of a visualization monitor. One of the lessons shared by other NAS-wide development teams was the need to have visualization of trajectories available as soon as possible. Without a visualization tool, verification and validation relies on inspection of data values which is error-prone. Problems that are obvious with a visualization display running can go undetected for years without one. Therefore, a display called the Progress Monitor was created early in the development process. This display uses a birds-eye viewpoint that can be zoomed, slewed, or rotated. When the Progress Monitor runs, it artificially slows down the simulation speed to force the traffic to move slowly enough to be captured for display. When run speed is critical and the monitor is not needed, the user can run the simulation without it.

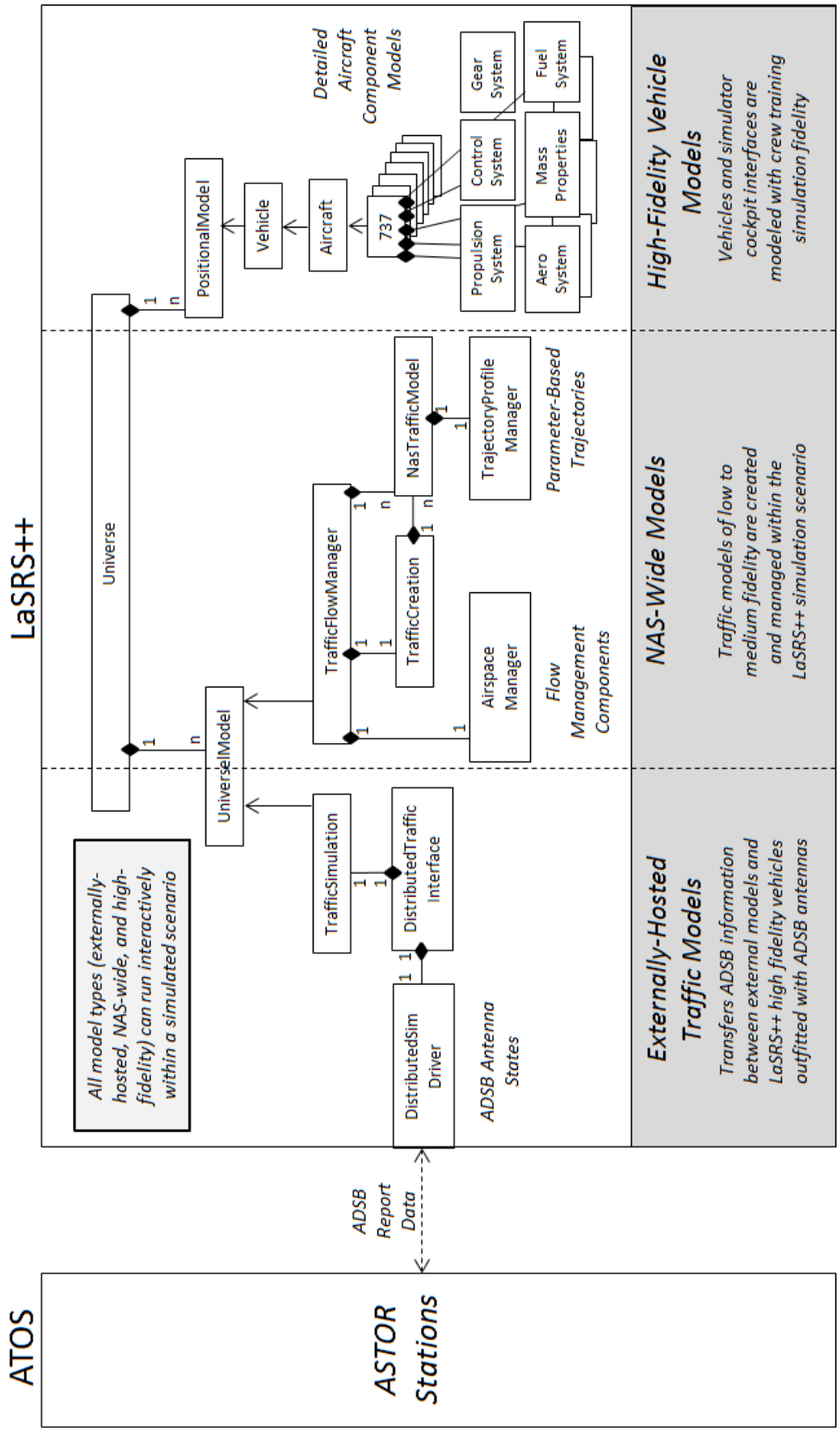

Figure 3. LaSRS++ Class Architecture for NAS-Wide, External Traffic, and High-Fidelity Operation

The Progress Monitor is a good example of software reuse (and associated cost savings) possible by using a common framework. The map background for the display was created to support an earlier LaSRS++ real-time project and was resurrected for the NAS-wide simulation. The interface architecture between the simulation program and the display program (which runs as a separate process) is also reused and is standard for display communication with LaSRS++ real-time simulator projects. Since the new Progress Monitor display uses LaSRS++ standard methods, this display can be reused in the future for real-time projects and is already being considered for cockpit display of weather. The icons that show traffic were also reused from an existing simulator cockpit navigation display.

A feature was recently added to this display to allow a spacing disk to be optionally enabled around any or all aircraft. The spacing disk radius and thickness are sent as individual aircraft parameters and are used to monitor loss of separation events for testing or for demonstration.

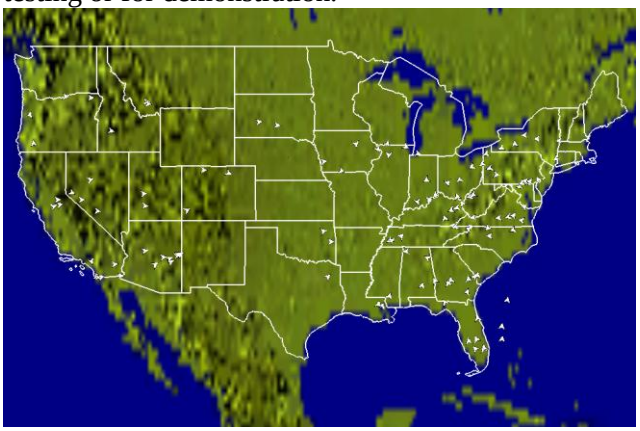

Figure 4. LaSRS++ NAS-Wide Progress Monitor

Live Data Interface

An interface was added to the simulation to process live data using a web-based blended source data service. Prior to the creation of this interface, live data was never used for LaSRS++. The capability was added as a feasibility test and to provide insight into the benefits and challenges of using this type of data in a simulation. FlightAware was selected because their service provides data from a compilation of sources through a single protocol. This allows a user to easily experiment with different data types through the same server. The LaSRS++ NAS-wide simulation currently only takes advantage of aircraft state data, which is used to locate traffic for interactive modeling and for display on the Progress Monitor. However, the API already exists in the simulation to access weather data, flight plans, and all other information offered by the service.

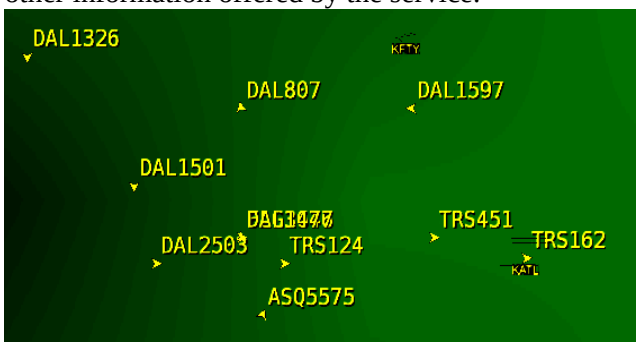

Figure 5. Live Traffic Portrayed in LaSRS++ NAS-Wide

Trajectory Generators

The term "trajectory generator" is used in a different context by the flight vehicle modeling community versus the NAS-wide modeling community. In the vehicle modeling world, a trajectory generator is the component of a flight management system (FMS) which predicts a path through space for the host aircraft to follow to navigate efficiently along the 3-D route selected by the pilot. In this case, the trajectory is not the state of the aircraft, but rather the target state. In the NAS-wide modeling community, a trajectory generator is the component of the simulation that provides the actual state for a modeled aircraft at all points along its simulated path based on performance database criteria. The trajectories are constructed from estimated performance data for specific aircraft types. The resulting paths are point-mass models with 3 degrees of freedom (3 DOF), as opposed to the 6 DOF trajectories of the high-fidelity aircraft models. The lower fidelity state models are appropriate for the systems-level studies that use them, and they run very quickly. This allows the NAS-wide simulation to run tens of thousands of aircraft on a single processor.

The LaSRS++ vehicle simulation already contained many high-fidelity performance models and can model FMS trajectories, but the type of trajectory generators needed by the NAS-wide framework had to be added. Three trajectory types

are currently available. A nodal model was created during the initial development to exercise the framework features (like data logging, file processing, and timing features). A second option called the Kinematic Trajectory Generator (KTG) [8] from Intelligent Automation, Inc. (IAI) was added which uses aircraft-specific performance characteristics and provides an integrated path. The KTG trajectory is also used in ACES, and provided a common thread between the two simulations that was useful for comparison testing during the proof-of-concept phase. The downside of this version of KTG for the LaSRS++ simulation is that it is written in Java, requiring it to be interfaced to LaSRS++ through the Java Native Interface (JNI). Communication between the C++ host and the Java KTG code through the JNI is very difficult to debug and impeded the progress of route modification modeling between LaSRS++ and KTG.

Two solutions were initiated to remove the Java code from the LaSRS++ NAS-wide model. One is the replacement of the Java KTG with a new version now available from IAI that is entirely in C++. This work is scheduled for completion by the end of 2014. The second solution is a new in-house C++ trajectory that is an evolution of the original nodal model and provides an integrated path using aircraft-specific performance data. The in-house C++ trajectory was completed in spring of 2014. The KTG trajectory is more mature and provides a richer set of features and higher fidelity, but the integrated C++ trajectory is also a useful model for many applications. The in-house C++ trajectory has the additional benefit of accessibility of the source code, since the C++ version of KTG will only be available as a linked executable library.

Arrival and Departure Routes

The ability for traffic to use Standard Instrument Departures (SIDs) or Standard Terminal Arrival Routes (STARs) is available for LaSRS++ NAS-wide. At startup, the program reads a text file that uses the same format as the FAA's 56 Day NASR "stardp.txt" data file. This file is available via the web from the FAA for US government use through subscription.

The 56-Day NASR STAR and SID routes file commonly contains multiple versions of routes, in which case the first version is used by the simulation. This file contains the superset of all route options, but does not determine which subset is used for any given simulation run. A separate initialization file contains the list of airports that will use arrival and departure routes, and which routes are active at startup. Only active routes are used by simulated traffic. Airports not specifically designated for SID/STAR arrivals default to a nodal terminal airspace model for run speed efficiency.

Navigation Database

The NAS-wide simulation reuses the Navigation Database system already available in LaSRS++ to determine the location of arrival and departure airports, named waypoints in arrival and departure routes, and runway parameters. This data resides in an ARINC[2] 424-formatted text file for the continental US which can be used as-is or tailored for research use. This navigation database information is used by the NAS-wide simulation to determine the airport centers for the default terminal airspace regions and for the locations of the runway thresholds at each airport.

Maneuvering and Rerouting

After the initial path for a flight is created, the flight may have to alter its path to avoid a conflict or to change its arrival time to interim waypoints or to the arrival runway. A maneuver can be added to the current route to issue a short-term divergence from the original path. Once the purpose of the maneuver is accomplished, the flight reacquires the original route as soon as possible. With rerouting, the latitude and longitude points that contribute to the aircraft's route are changed permanently and the flight is not expected to reacquire the original route. This might be done, for example, to avoid a large weather system.

[2] Aeronautical Radio, Incorporated (ARINC) maintains formatting standards for communication protocols for the aeronautics industry, including the 424 standard which pertains to navigation databases.

The KTG trajectory generator option within the NAS-wide simulation supports maneuvering with a user-friendly interface to request changes. The simulation currently only uses the path stretch and the speed change maneuvers to alter aircraft paths, but options are also available for altitude and course changes, and for combinations of several of these options linked in tandem. Maneuvering is not yet available for the C++ trajectory, but is a planned future feature. Rerouting is available for either trajectory and is handled by modifying the reference points for a flight's path and requesting a new trajectory prediction.

Wind and Turbulence

The NAS-wide simulation makes use of the existing wind and turbulence models from LaSRS++. A variety of models are available, including constant and 4-D location wind models and several options for wind turbulence and wind shears. Numerous other environmental models are available in LaSRS++ that are frequently used by the vehicle simulations but are not yet used in the NAS-wide simulation. These include cloud layers, fog, and sun rise and set timing which could be useful in the future for localized visibility constraints on spacing, for example, while the wind information could be useful in assessments of noise impact in the vicinity of airports.

Future Development

Additional functionality is in progress, with optional new capabilities envisioned for farther term development. The near-term features center on arrival and departure scheduling models. Though aircraft can already follow SID or STAR routes, there is no system monitoring or advising them for specific waypoint crossings or runway threshold touchdown times to target safe separation. The addition of a system to provide this is a near-term research need. Properly implementing such a scheduler will rely on an expanded maneuvering and rerouting capability, which is also targeted for near-term development.

As discussed in the *Trajectory Generator* section, a new version of the KTG trajectory generator is currently being interfaced to the NAS-wide simulation. This trajectory will be available as a fourth option and will allow the more mature KTG features and fidelity to be used in the simulation without Java. As part of the addition of KTG in C++, the trajectory generator manager is being reworked to allow any trajectory generator to be added to the simulation in the future as a plug-in.

A long term goal is to allow the LaSRS++ HITL simulator cockpits to use the NAS-wide aircraft as intelligent traffic models. Use of this type of intelligent traffic model in the current HITL simulations requires connection to and support of the ATOL lab. Once the NAS-wide models are integrated to the real-time simulation, intelligent high-volume traffic will also be available in LaSRS++-only operations and also with fast-time vehicle simulations on the desktop.

Adding this capability will require some minor changes to the vehicle simulation executive to create and manage the NAS-wide flights. Once integrated, this will also allow the HITL simulations to run in scenarios that use actual live traffic in progress. The implementation of this set of features is conditional on funding and need from the research clients that will use that service.

References

1. Madden, M., and Glaab, P., "The Langley Standard Real-time Simulation in C++; 2005 Software of the Year Presentation", NASA IV&V Facility, Fairmont WV, June 22, 2005.
2. SMART NAS NNA13446416L, Attachment A: Statement of Work, Nov 2, 2012, http://prod.nais.nasa.gov/eps/eps_data/154428-DRAFT-001-001.docx.
3. Oracle, "Tuning Garbage Collection with the 5.0 Java Virtual Machine", Sun Microsystems, http://www.oracle.com/technetwork/java/gc-tuning-5-138395.html.
4. NASA, "Marshall Aerospace Vehicle Representation in C (MAVERIC-II) Computer Program", Tom Knight, June 12, 2014, http://techtran.msfc.nasa.gov/software-catalog/MFS-31989-1-MAVERIC-II.php.
5. NASA, "NextGen Takes Flight with Air Traffic Operations Lab Upgrades", Denise Lineberry, July 22, 2013, http://www.nasa.gov/larc/nextgen-

takes-flight-with-air-traffic-operations-lab-upgrades/.
6. Lewis, T., "Airspace and Traffic Operations Simulation (ATOS) and the Air Traffic Operations Laboratory (ATOL)", Crew Systems and Aviation Operations Peer Review, Feb 29, 2012, NASA, Hampton, VA.
7. Madden, M., "Architecting a Simulation Framework for Model Rehosting", AIAA 2004-4924, AIAA Modeling and Simulation Technologies Conference, Providence, RI, Aug. 2004.
8. Zhang, Y., Satapathy, G., Manikonda, V., and Nigam, N., "KTG: A Fast-time Kinematic Trajectory Generator for Modeling and Simulation of ATM Automation Concepts and NAS-wide System Level Analysis", AIAA 2010-8365, AIAA Modeling and Simulation Technologies Conference, Toronto, Canada, Aug. 2010.