Lessons Learned from Distributed Virtual Air Refueling (VAR) Integration

Jonica Tramposch
Northrop Grumman Corporation
Orlando, FL
jonica.tramposch@ngc.com

Christian Schwindt
Northrop Grumman Corporation
Orlando, FL
christian.schwindt@ngc.com

ABSTRACT

Aerial refueling over a distributed network requires players across multiple time zones to recreate a virtual world that has stringent temporal and spatial threshold tolerances. Such capability enables a coalition of forces to participate in quality training at a small fraction of the cost and logistics time of a live exercise. Until the Mobility Air Force (MAF) Distributed Mission Operations (DMO) Program, such precision had not been achieved. In 2017, multiple Mobility Air Force platforms embarked upon testing the efforts made thus far in achieving this Distributed Virtual Air Refueling capability. Years of preparatory work and standards development preceded this effort; yet, when theory met practice in 2017 many valuable lessons were learned. This paper discusses the most important lessons learned from the VAR effort. Advice on what to require, avoid, and when to address the following topics is given: physics model fidelity, data self-consistency and accuracy, objective and quantitative test results and analysis to enable correct root cause determination, ways of dealing with legacy code and when to refactor, contract and acquisition strategies, and subject matter expert support required to ensure success. Incorporating these lessons into future efforts will improve the efficiency and success of creating a precise virtual world that serves as a capable battle space for complex, distributed training.

ABOUT THE AUTHORS

Jonica Tramposch works for Northrop Grumman on the Mobility Air Force (MAF) Distributed Mission Operations program. She has been active in testing each MAF platform vying for acceptance for Virtual Air Refueling and has led the analysis that determines root causes for all major issues along the way. Jonica's background is in physics, research with large data sets, business development, and test methods for complex systems. She holds a Bachelor of Science in Physics and Astronomy from the University of Washington, a Master of Science in Space Science and Instrumentation from Université Paul Sabatier, and a Master of Science in Space Technology from Luleå University of Technology.

Chris Schwindt works for Northrop Grumman and is the Contractor Mobility Air Force Distributed Mission Operations Standards Lead and Chief Engineer. He is a physicist by education, with a background in Systems Engineering including virtual distributed training involving high fidelity aircraft simulators. He has worked in the distributed simulation industry for seventeen years. Chris holds a Bachelor of Science in Physics from the University of South Florida, a Master of Science in Engineering and Applied Sciences from the George Washington University specializing in Aeroacoustics, and a Master of Science in Optical Physics from the University of Central Florida.

Lessons Learned from Distributed Virtual Air Refueling (VAR) Integration

Jonica Tramposch
Northrop Grumman Corporation
Orlando, FL
jonica.tramposch@ngc.com

Christian Schwindt
Northrop Grumman Corporation
Orlando, FL
christian.schwindt@ngc.com

INTRODUCTION

The Air Mobility Command (AMC) goal for Mobility Air Force (MAF) Distributed Mission Operations (DMO) Virtual Air Refueling (VAR) is to transfer air refueling training hours from the air to simulators. By moving two of four air refueling training events per student from live to virtual events in simulators, significant cost avoidance can be achieved, estimated at $66M+ per year by AMC (Carey, 2013).

For years, local VAR has been performed by using only one virtual simulation with the other roles being fulfilled by constructive simulations of limited fidelity. This approach cannot achieve the fidelity necessary for training hours to replace time in the air. In local training, simulators at a single site create and interact with constructive entities. For VAR to compete with live AR, the real air crew must interact with each other instead of with computer-generated entities. The training must fully stimulate the participants' sensory systems into believing they are working in a real plane. Great precision in a distributed environment is required to simulate realistic, real-time responses of aircraft, and achieve the AMC business case for VAR.

Due to the boom, tanker, and receiver simulators being distributed across the country, the virtual tanker, virtual boom, and virtual receiver business case of VAR require precision across a distributed network. When in contact, the location of each point of interest must be agreed upon to within visual detection. The Institute of Electrical and Electronic Engineers (IEEE) Standard for Distributed Interactive Simulation Application Protocols (IEEE, 2012) contains the precision, data format, Earth model, and relevant calculations and has proven to be an excellent guide for success in VAR. The challenges have been to correctly implement the standard and to verify that the implementation was successful.

Key to the success of VAR is determining when the simulators are in fact achieving the temporal and kinematic precision specified by IEEE 1278.1, sending self-consistent information, and properly dead reckoning the ownship and othership such that distributed simulators agree on their local view. Considerations when undertaking this task include restrictions of legacy code, knowledge of aerodynamic physics, funding for error analysis and unit testing, and analysis methods for error detection and solution verification. When these potential areas of concern are mediated, the AMC business objective for VAR is achievable.

The VAR lessons learned in this paper will enable future efforts to travel an efficient road to high quality distributed simulations.

LEGACY ENVIRONMENT

As contracts migrate from one company to another, so does the computer code. The code that arrives at a new company is labeled "legacy" and unless there is no other alternative, is never touched again. Editing another company's code is high risk. Try remembering what you had for lunch the first Tuesday this past March – that is analogous to understanding your own company's code. Now try to remember what your boss's boss had for lunch that same Tuesday – that is analogous to understanding legacy code. Instead, companies cater to interfaces. The legacy code ingests data in a certain format and outputs data in another format. Whatever requirements are in the new contract, the data that interacts with this legacy code will need to be transformed into and out of the interface formats legacy requires.

Each transformation in format adds error. In VAR host code, data is often translated between English metrics and Standard International metrics based on which part of the code was part of which contract. Coordinate system conversions can occur a couple dozen times within the data flow. Each time a conversion or transformation is performed, the data becomes less precise – more error is introduced. This accumulation of error is a mathematical principle that is often underappreciated and frequently experienced without understanding why a discrepancy has formed.

In addition to inaccuracies from transformations, another major issue in working around legacy code is the use of "correction factors" or biases. For a given calculation, or system of calculations, a given input should result in a specific output. If that output is not achieved there is a reason why. If that reason why cannot be found, it is common practice, though not good engineering, to add a bias into the system. A bias is a value added to an inaccurate result in order for that incorrect result to become the desired result.

Biases are often found in legacy code, often employed toward the end of a contract when time was restricted. Biases are also frequently employed outside of legacy code to correct for inaccuracies within the legacy code or to offset the accumulation of error from transformations. Biases are a sign that a system is either not understood, or the ability to correct the root cause is unavailable. A stringent code review is advised to locate biases.

When preparing to undertake VAR, have a plan to fully analyze and understand legacy code or be prepared to rewrite it.

Data Flow

The figure below shows the lifecycle of VAR data from a local view. Each platform has a specific detailed flowchart, so the generic method in Figure 1 has been created to encompass the gamut of approaches employed. Legacy code can be in any of the software steps.

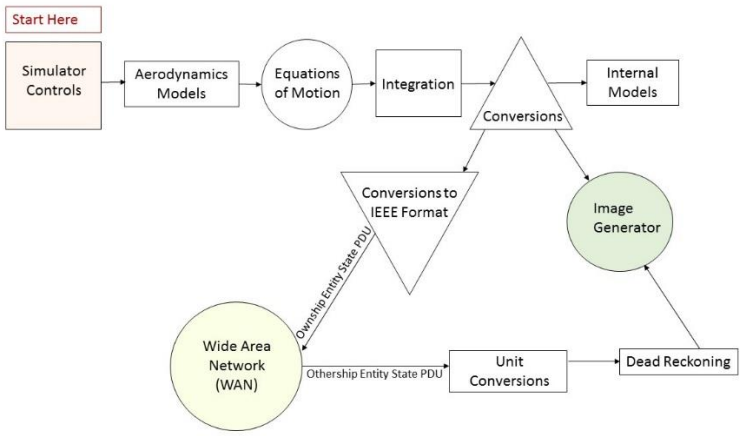

Figure 1. Simulator Information Flow

The information flow presents several opportunities for error. Each conversion provides an opportunity for a miscalculation through loss of precision, a design flaw, or a software bug. Even in a correct, robust implementation, an unavoidable amount of numerical error is introduced. There are essentially four branches of data flow (ownship to IG, ownship to internal models, ownship to world, and world to IG) that must agree to high precision in time in order for the players to interact with each other and receive quality training.

Requesting a host code flow chart will help identify areas of concern, nodes for input/output verification, and a map for troubleshooting.

THE TIME PROBLEM

Section 4.6 of IEEE 1278.1 provides guidelines for time. To paraphrase, the timestamp within each Protocol Data Unit (PDU) must be synchronized to a Stratum 1 time source, such as a Global Positioning Satellites (GPS) receiver, to within a tight threshold (100 microseconds for GPS synchronization) and all data within that PDU must be valid for its timestamp.

While the IEEE 1278.1 standard of time is straight forward, practical implementation is quite difficult. Imagine a trainee operating a simulator. The operator adjusts any one of the controls, sending input from the hardware to the software equations of motion where torques are used to calculate accelerations. The accelerations are then integrated to obtain velocity; then velocity is integrated to determine location. The time stamp must be valid for the torque values, which means that if not all calculations are performed in a single frame, the time must be stamped at the original input frame where the torques are calculated. The system clock may be synchronized to within 100 microseconds of GPS time, but if this precise time stamp is not valid for all of the data in the PDU then the simulator is not synchronized.

The time stamp and its corresponding data is critical on the receiving end – the consumer side. Imagine two different sites: Site A and Site B. Both sites have system time that is synchronized within tolerance to GPS time and a timestamp that is valid for all kinematics (this in itself it a common source of error). Site B receives a PDU from Site A. Time has passed since the PDU was created. More time will pass before the information from the PDU is displayed in "real-time" on Site B's simulator. In order for the out-the-window display to be in real time, the location and orientation in this received PDU must be dead reckoned into the future. This future time must be the time at which the ownship simulator will present the information. Correct handling of the propagation of time is a huge opportunity for error.

The Concept of Now

Time is the most critical requirement and presents the toughest challenge for distributed simulation. There are several clocks in play in a virtual aircraft simulator and it is imperative to keep track of which clocks are being used for the various actions in the virtual simulator software. It is conceptually difficult to discuss multiple clocks and the non-intuitive effects that they can have in a simulation because people are used to there being only one "now," one stream of time. In software implementations, this is not the case. The clocks involved are:

1. The physics time constant. This constant is the time step interval over which the numerical integration algorithms are computed (acceleration to velocity, then velocity to position).
2. The Host Computer System Clock and its related Real-Time Clock, which are part of the host hardware and Operating System. These clocks are required to be synchronized to a Stratum 1 time source, such as a GPS Receiver.
3. Image Generator Top-of-Frame Time. This time is that at which the data to paint the image on the out-the-window display is provided to the Image Generator (IG) System. Note that in most IGs, the image is painted for the observer to see some number of frames later than the dead reckoning calculation due to pipelining in the IG.
4. Software Executive Time. This is the time at which the various models are called into execution.

The goal is to have these four clocks as perfectly synchronized as possible so that it makes no difference which clock is used for any given calculation or time reference. Two fundamental requirements must be met by all clocks in the simulator:

1. One elapsed second of time in the numerical integration calculations, in the host computer clock, and in the IG Frame time must be equivalent to one elapsed second of real-world wall clock time. This is the "one second per second" or "the simulator shall run in real time" requirement.
2. The time difference between the any two simultaneously reported times by any two clocks (accounting for systemic offsets such as time zone or time unit conventions in the time representation) must be a zero-mean, low variance random variable. This "everyone is marching to the same drummer" requirement is what allows valid time calculations to be done between clocks.

If clocks 1, 2, or 4 fail in synchronization or precision, then the kinematic data in an Entity State PDU will not correlate with its time stamp. One way to determine if time stamps match the kinematic data is to back solve the time difference between successive Entity State PDUs via their acceleration, velocity, and position data then compare this delta to the

difference between time stamps. These time deltas will be equivalent if kinematics correlate with their time stamp. When the time deltas are graphed against each other, the points should lie on the identity line of x = y.

At first glance, Figure 2 may look like the delta kinematic time versus the delta time stamp fall along the identity line, where each cluster is at an integer number of frames. However, the horizontal smear at the first set of points shows variation in the DIS time delta of a single frame. The DIS time is varying more than the kinematic time, which suggests that the kinematics are being calculated at regular intervals but this DIS time stamp interval neither exactly matches the kinematics nor are exactly consistent in their periodicity.

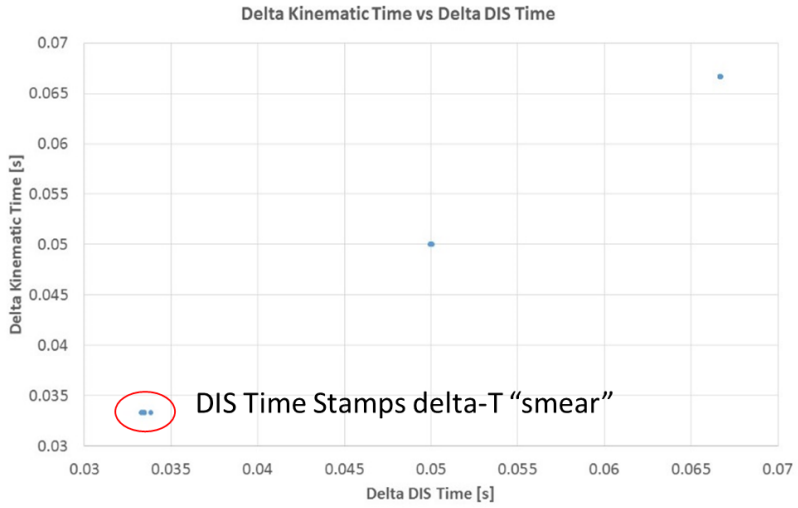

Figure 2. Time smear caused by noisy time stamps

For a more granular look at the DIS time variance seen in Figure 2, time variance is calculated. Time variance is determined by finding the ideal frame time (the frame time with the least standard deviation over a sample) and subtracting the applicable integer number of ideal frame times from the timestamp difference between two Entity State PDUs. This calculation is performed over a large data sample. Figure 3 and Figure 4 below illustrate high variance and low variance.

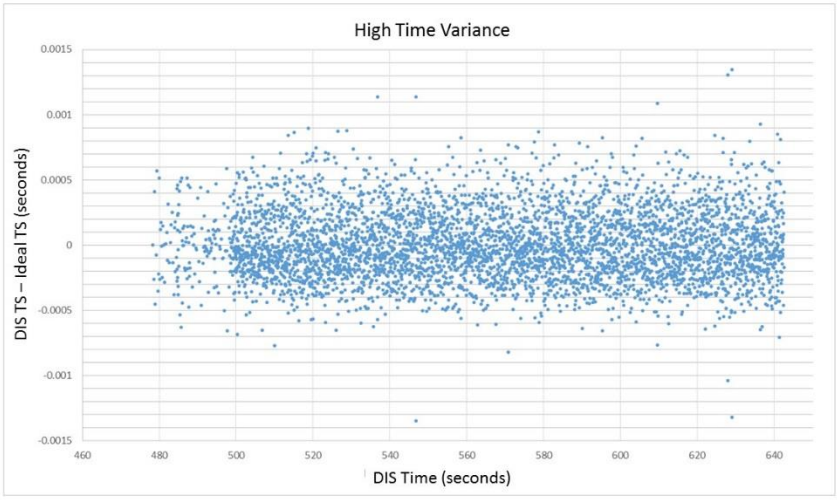

Figure 3. 1000 microseconds of timestamp variance

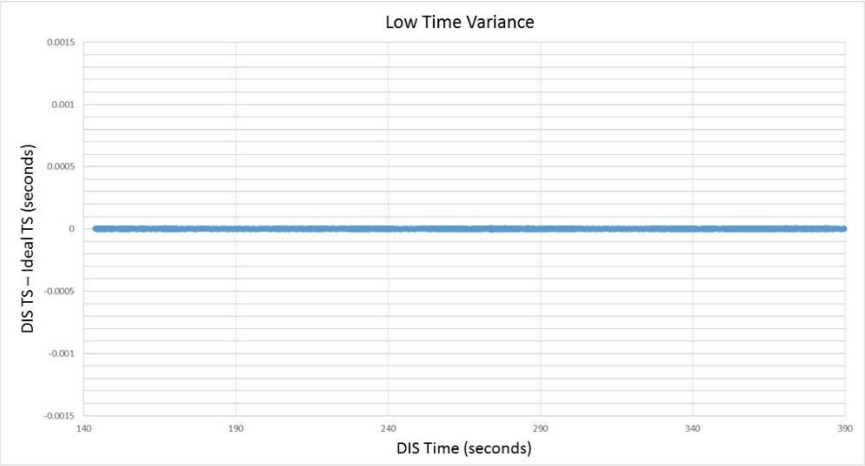

Figure 4. Less than 50 microseconds of timestamp variance

The data in Figure 3 shows a large amount of time variance, which could cause the horizontal smear seen in Figure 2. Figure 4 is the result of precise frame periodicity. The systems in Figure 3 and Figure 4 may both be synchronized to their time source according to standard, but the high variance of the Figure 3 system is a warning sign that its method of assigning a well-synchronized time is not precise. The result is that the kinematics do not correlate to the time stamp to sufficient precision.

DATA SELF CONSISTENCY

All data in an Entity State PDU must be self-consistent. All other simulators in an exercise depend upon this data to recreate the experience the PDU issuer is communicating. If information in this PDU contradicts itself then the receiving simulators will not agree with the issuer on its entity's position and orientation at a given time.

Factors required for self-consistency include precision, units, world model, execution of simple linear calculations, and advanced rotational calculations. Each of these factors is addressed in IEEE 1278.1, but they must be policed. For example, IEEE specifies that the kinematic variables must be reported in double precision; this insinuates that double precision should be used throughout the coordinate transformations and unit conversion calculations. Nevertheless, single precision calculations have been discovered behind double precision reporting. Mathematically, precision cannot be regained after it is lost.

IEEE 1278.1 specifies use of the WGS 84 world model, which is an elliptical model of the Earth. Local training models were built on a spherical Earth model for mathematical simplicity. When replicating the real world to high precision, as is required in VAR, adhering to WGS 84 as a community is necessary. It has been seen that shortcuts to using WGS 84, such as spherical math but elliptical reporting, result in increased position and orientation uncertainty at particular locations on the globe or at particular headings. Simulators must be tested with their entities at a range of latitudes, all Earth quadrants, and a full range of headings in each of these locations in order to verify correct implementation of WGS 84.

All calculations required for self-consistent data exist in IEEE 1278.1. Like the WGS 84 model, the rotation equations are complex and are routinely incorrectly implemented.

Given the multitude of opportunities for error and the fact that errors propagate and accumulate, objective and quantitative tests are required to analyze the compliance with IEEE 1278.1 and to enable correct root cause determination for non-compliance.

Kinematics

Imprecise time stamping can be a root cause of kinematic variable inconsistencies, but kinematic variables can also be inconsistent despite a perfectly executed time system.

Entity State PDUs contain information on the linear and rotational dynamics of the entity, including the time and the associated values for the position vector, the linear velocity vector, the linear acceleration vector**,** the orientation (Euler angles or Quaternion)**,** and angular velocity. Each of these values has an opportunity to contradict another. Figure 5 depicts the relationship between these variables and how they can be cross-checked for consistency.

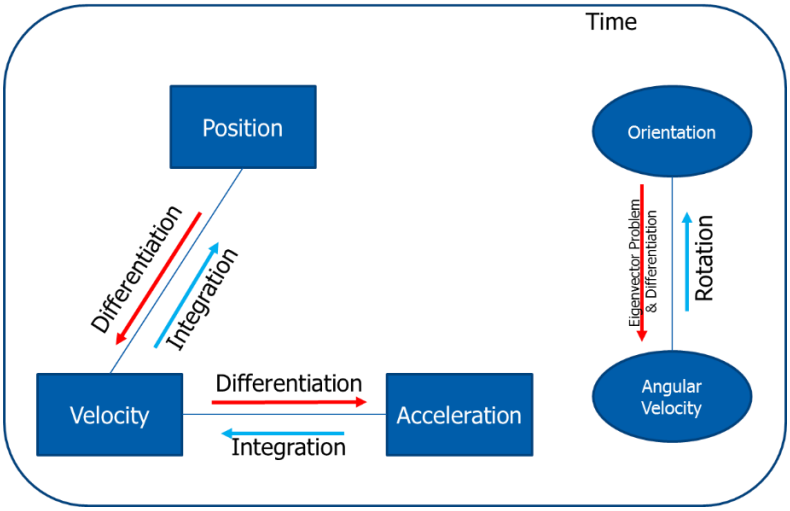

Figure 5. Inter-relationships between Entity State PDU Kinematic Variable Data

The variables named in the blue shapes in Figure 5 are published in an Entity State PDU and are associated with that Entity State PDU's time stamp. Of these blue shapes, the variables in the rectangular boxes represent the linear motion variables; the variables in the ovals represent the rotational variables. Through the calculus operations of integration and differentiation for the linear variables of position, velocity, and acceleration, one can go from any one variable to its counterparts. The same holds for rotation operators for the rotational variables of orientation and angular velocity. This relationship provides the opportunity to cross-check the self-consistency of the kinematic data.

When possible in cross-checking data, integration is employed instead of differentiation to limit noise. Figure 6 shows acceleration and velocity values that contradict each other by a factor of two. If the published acceleration was self-consistent with the published velocity then the two lines would overlay each other.

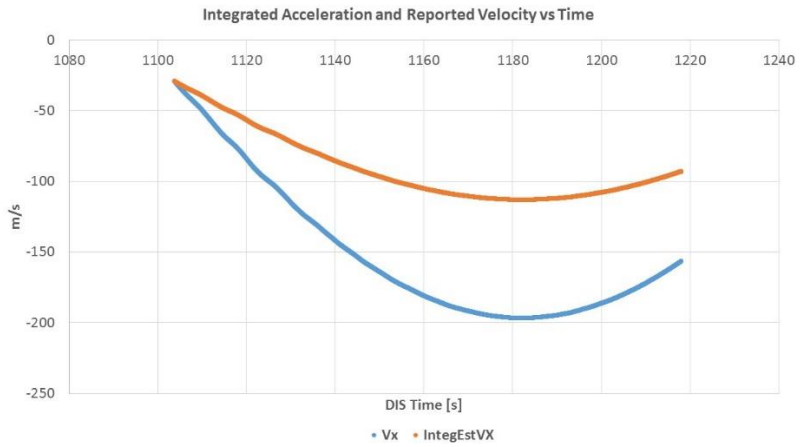

Figure 6. Poor correlation between acceleration, velocity, and time. Factor of 2 calculation error.

Comparing position to velocity is analogous to comparing velocity to acceleration. The velocity is integrated over the time stamp delta to find the position. This calculated position should match the reported position in the Entity State PDU. Figure 7 is an example of velocity not correlating with position. Again, these lines should overlay each other.

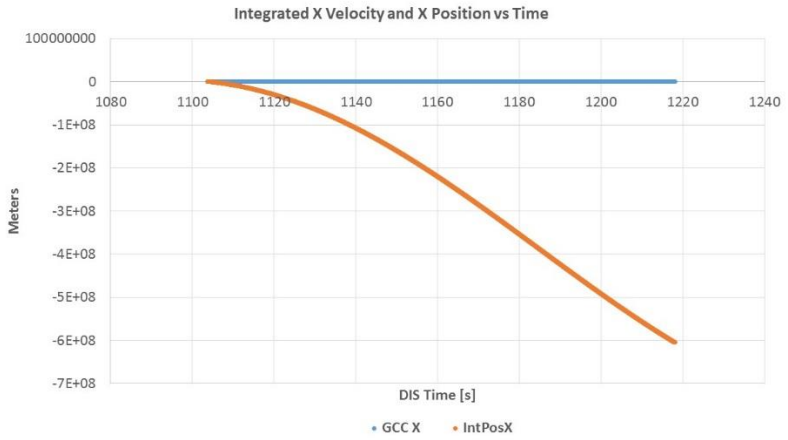

Figure 7. Velocity and position are not self-consistent

There has been speculation on how closely these lines should overlay each other. The answer depends on how fast the entities are traveling, how close they are to each other, and what can be discerned by the crew viewing the simulator's image generator. The faster the entities travel, the more distance is covered in a given time delta, so the greater the discrepancy. If the entities are in contact then the discrepancy tolerance is lower than if they are 50 meters apart. The discrepancy tolerance is also lower for higher definition image generators. For VAR, close proximity and high definition are required, so good enough boils down to human factors. If all inaccuracies equate to a location discrepancy of 2 centimeters, will the crew notice? That may depend on the trainee. Certainly a discrepancy of two feet would be noticed.

Orientation

Orientation is recreated via values in the Entity State PDUs by consuming the angular velocity (omega), and Euler angles. By using successive PDUs, as in the acceleration, velocity, and position technique above, omega can be estimated from the reported Euler angles and delta time. Figure 8 shows decent correlation between the estimated omega and the reported omega. While the points do not exactly overlap, they clearly follow the same pattern.

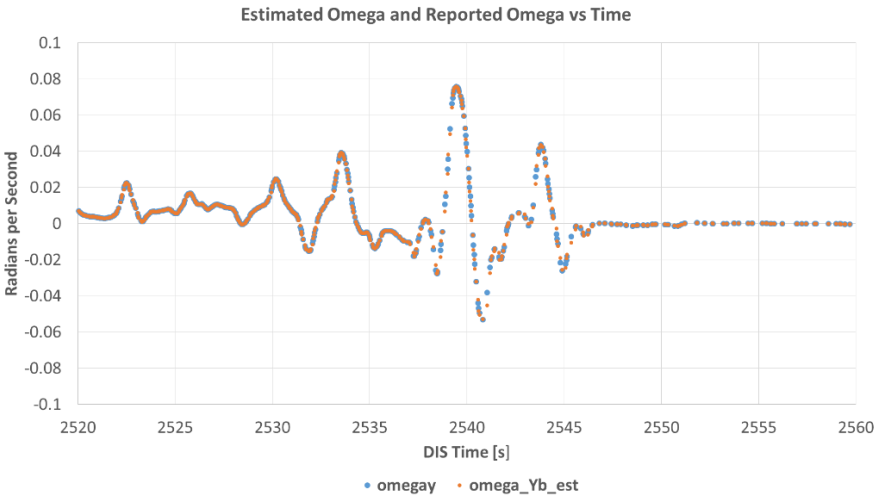

Figure 8. Decent correlation between Reported and Estimated Angular Velocity

Figure 9 displays data where omega was incorrectly calculated and reported. Given that omega is in units of radians per second, sixty radians per second would indicate that the entity was conducting roughly ten revolutions per second; it was not. This graph is an example of how multiple errors in orientation calculations and reporting represent themselves in PDU data. Inaccurate data in a PDU leads to inaccurate out-the-window displays.

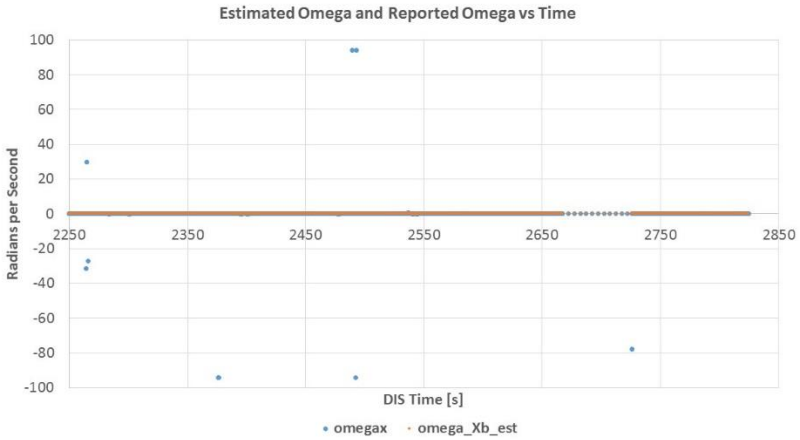

Figure 9. Error in reported Angular Velocity (Omega)

Simply checking the reasonability and self-consistency of data in Entity State PDUs can reveal non-compliance with IEEE 1278.1 that would otherwise be a mysterious anomaly observed in the out-the-window display.

The Consumer Side

The analyses above can only reveal deficiencies in producer self-consistency. Entity State PDUs cannot pinpoint errors or deficiencies in how the consumer processes, or dead reckons, the incoming data for use by its own internal models, including the visual system.

The consuming simulation will dead reckon the state data of the ownship and othership to predict the location of the entities at the consuming simulation's current execution time. This information is then fed to the visual system, which has its own input criteria and possible biases. Errors on the consuming side are internal and therefore not visible on the network.

Both producer and consumer side errors are reflected in the out the window display, so it is nearly impossible to find the root cause of the observed anomaly without stringent analysis throughout the entire data flow depicted in Figure 1. In addition to the time and self-consistency analysis, it is highly advised to have a consumer side data tap that leads into the IG in order for consumer side anomalies to be detected, analyzed, and resolved.

STRATEGY FOR FUTURE CONTRACTS

Ensuring that time is precise, synchronized, and perfectly correlated to the reported kinematics, which in turn must be self-consistent, is a tall challenge for any development team. There are contractual steps that can be taken to both aide and ensure that VAR host code follows IEEE 1278.1.

First, anticipate legacy code that is either imperfect, not understood, or both. Decide whether to invest time in exploring and understanding the existing code then rewriting sections if necessary, or to rewrite it from scratch.

It is suggested to have two subject matter experts (physicists or mathematicians) on the development team to advise and provide software design and code review. Understanding the models that must be implemented, including complex equations that must be evaluated numerically requires advanced training in mathematics and physics. Two specialists will catch each other's inevitable mistakes.

Next, provide ample scope for unit testing. All of the tests presented in this paper could be run as unit tests by developers. Sections of code should be fed known input and its output should be as predicted by the necessary mathematical equations. The code is not ready until the expected outcome is achieved. Inconsistencies can be spotted immediately and resolved while the code is still pliable. Uncovering and remedying such issues in the field is costlier and higher risk.

Once the unit tests pass, error analysis should be conducted to understand the limit of precision. Each value has a finite precision and each operation introduces error. The total error of the system should be known. Errors propagate – a position error of three millimeters in one section could lead to error of one centimeter in a later section. True error analysis is time consuming and challenging. Not performing error analysis is risky.

Finally, be prepared with analysis tools, consumer nodes for data collection, and personnel with experience in large data sets to interpret the results. Third party verification of compliance with IEEE 1278.1 will rely on these tools, data, and expertise.

The toughest challenge for VAR is knowing when and how simulator code is out of compliance with IEEE 1278.1. A contractual commitment to rigorous testing, troubleshooting, and root cause resolution from the unit level to acceptance testing will enable future VAR pursuits to avoid delays and confusion in a quest that will ultimately lower fuel costs while providing high quality training.

ACKNOWLEDGEMENTS

The authors wish to thank AMC A3T/D and AFLCMC/WNS for the opportunity to work on what is so far the most challenging DMO capability to date. The authors also wish to thank Rich Grohs for valuable feedback as this paper matured. Finally, thank you to Northrop Grumman for supporting the production this paper.

REFERENCES

Carey, S. (2013, December 2). Air Mobility Command Distributed Mission Operations. (J. Lamar, Interviewer)

IEEE. (2012). *IEEE Standard for Distributed Interactive Simulation -- Application Protocols.* New York, NY: Institute of Electrical and Electronics Engineers, Inc.