Improving the Constellation Mission Control Center System Design using Integrated Executable Architectures and Visualization

Shawn E. Gano*, Raymond K. Hunnicutt†, and John D. Nelson‡
Lockheed Martin, Information Systems and Global Services, Houston, TX, 77058 USA

and

Ronald J. Dockal§
National Aeronautics and Space Administration, Houston, TX, 77058 USA

A systems engineering approach, called CONOPS Simulation, which combines integrated architecture development, discrete event simulation, and graphical visualization into a fully traceable simulation is presented in this paper. This approach allows analysis of architectural alternatives and is also an effective communication tool for describing concepts of operations to an audience of various backgrounds and discipline expertise. Details of the process for developing a CONOPS Simulation are discussed, including the architectural diagrams required, translation of the architecture to a discrete event simulation engine, and different types of real-time visualizations that can be incorporated into the simulation. To further help in understanding the approach, a case study is given demonstrating the use and benefits of CONOPS Simulation to the NASA Mission Operations Directorate as a tool to explore new concepts and architectures for ground operations in support of human spaceflight.

I. Introduction

ON January 14, 2004 U.S. President George W. Bush unveiled a new space policy for the nation entitled the Vision for Space Exploration[1]. This vision established many notable tenets for the space program, including: completion of the International Space Station; retirement of the Space Transportation System (STS), better known as the Space Shuttle, by the end of 2010; implement a sustained and affordable human and robotic exploration program to explore the solar system and beyond; development of a new spacecraft that will initially service the space station by 2014, return astronauts to the moon by 2020, and allow the exploration of Mars and beyond. The announcement of this vision prompted the National Aeronautics and Space Administration (NASA) to initiate a large-scale, system level study of how best to meet these new national goals. This study, now officially referred to as the Exploration Systems Architecture Study[2] (ESAS), formed the basis for the Constellation program. At the end of 2005 the U.S. Congress, by passing the National Aeronautics and Space Administration Authorization Act of 2005[3], formally established the legislative basis for NASA to conduct the programs necessary to attain the goals set forth in the Vision for Space Exploration.

"Today, NASA is moving forward with a new focus for the manned space program: to go out beyond Earth orbit for purposes of human exploration and scientific discovery.[4]" This statement, made by NASA Administrator Michael Griffin, encapsulates the requirement that a new fleet of spacecraft will need to be designed, developed, and operated to support the next generation of manned missions along with a redesign of the Mission Control Center System (MCCS) to support the new vehicles. The design of this new MCCS is a very challenging problem due to

* Ph.D., Senior Systems Engineer, e-mail: shawn.gano@lmco.com, AIAA Member.
† Senior Staff Systems Engineer, e-mail: ray.hunnicutt@lmco.com.
‡ Systems Engineering Senior Manager, e-mail: john.d9.nelson@lmco.com.
§ MCCS Chief Engineer, Lyndon B. Johnson Space Center, Mission Operations Directorate, e-mail: ron.dockal-1@nasa.gov.

not only the complexity of the new spacecraft, but also the complexity of the missions that it will be expected to perform and the wide range of operations which it must support. Traditional systems engineering approaches used to design the MCCS include the development of a set of requirements followed by the development of an architecture which is typically depicted as a set of static drawings defining system interfaces, operational and system functions, hardware, software, and data flow diagrams. This architecture is critical to the long term success of the program and must meet the design requirements in addition to being practical so that it can be implemented to ensure mission success. The value of doing quality systems engineering on programs has been studied in detail by both NASA[5] and the International Council on Systems Engineering[6] (INCOSE). Their results, summarized in Fig. 1, clearly demonstrate the cost savings and schedule overrun dependence that doing or not doing good systems engineering has on a program. In light of this analysis it is imperative that the Constellation program, which is under both a tight schedule to minimize the United States' human spaceflight gap and budgetary pressures, adopt good systems engineering practices. In this paper a pilot systems engineering methodology for helping to analyze the Constellation Program's static architecture for the MCCS is described along with the benefits of improving the systems engineering process as well as other lessons learned in its development. This method, which we call CONOPS Simulation, promotes an integrated architecture design process that enhances documentation and allows for greater communication of the concept of operations and the system requirements. CONOPS Simulation is an interconnected, interactive, simulation of the architecture to help identify potential system interface issues and requirements gaps early in the program life cycle. In a sense this method allows you to *fly before you buy* at the system design level.

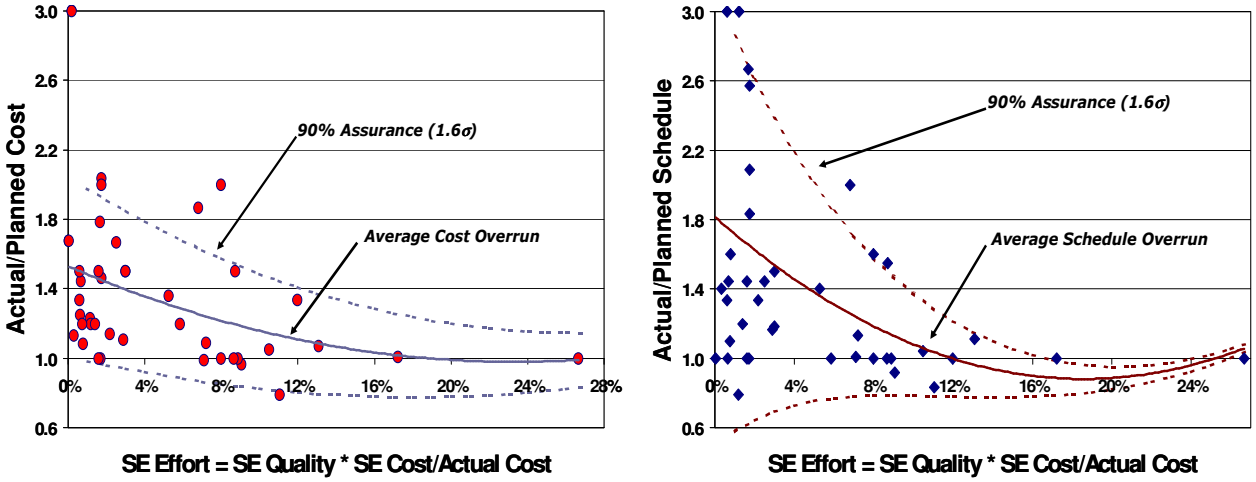

Figure 1. Quality Systems Engineering (SE) Effort reduces the probability of cost and schedule overruns.
Source: SECOE 01-03 INCOSE2003

In this paper an overview of what CONOPS Simulation is and how one is developed is described first. This is followed by a more detailed description of the various aspects of the development process including the integrated architecture development, dynamic simulation of the architecture, visualization of the concept of operations (CONOPS), and analysis of the architecture and its various alternatives. Finally, a case study using CONOPS Simulation involving the contingency voice communication link for the Orion vehicle during launch is given.

II. CONOPS Simulation Overview and Development Process

CONOPS Simulation provides a multi-perspective cooperative and dynamic simulation environment to evaluate architecture alternatives. It also has the unique capability to directly execute the static integrated architecture diagrams, giving users a clear understanding of the candidate architecture's dynamic behavior. While evaluating the static architecture designs and operational plans, trade-offs between proposed systems are also examined and assessed, which may be used to support cost benefit analyses and quantitative acquisition decisions. CONOPS Simulation provides a direct trace to the static architectural model, thus avoiding human interpretation (and misinterpretations) of those models in the simulation. This facilitates simulation of the complex architectural constructs in the MCCS since those simulations are directly tied to the architectural definition model and provides

for unique insight into the dynamic behavior and interoperability of the architecture, which allows for in-depth engineering analyses. The CONOPS Simulation can also be connected to physics based analysis tools, operator displays, performance views, and three-dimensional real world visualization engines to help communicate the concept of operations to a wide range of audiences.

The main motivations in using CONOPS Simulation are for aid in architecture design involving many different interconnected and interdependent systems and for its ability to help communicate the different CONOPS with direct traceability back to the architecture. It also allows for trade study analysis of candidate concepts and architectural alternatives, which can be both from a high level perspective, or can also be very detailed depending on the amount of information available about the various systems and subsystems being modeled.

There are many other technical and non-technical benefits to a program that are derived from developing a CONOPS Simulation. The formulation of the architecture can help to drive out requirements for interfaces, system functionality, timelines, and resource utilization. The modeling of the various systems for the simulation can help in evaluating different algorithms, as well as requirements for operational hardware and software. From a program management standpoint the demonstrations and visualizations of the CONOPS can help differentiate new architecture concepts to various stakeholders and customers. Later in the program lifecycle as the CONOPS simulation is matured it can be used to demonstrate and communicate roles and responsibilities from the user perspective as well as provide aid in user training with human in the loop emulation.

The development of a CONOPS Simulation can be broken down into three main steps and is depicted in Fig. 2. The first, building an integrated architecture and identifying the scenario of interest to be simulated, is described in more detail in Section III. One of the outputs from the integrated architecture development is a consistent set of diagram documentation of the system; in our work this has been in the form of interactive HTML pages. The advantage of the electronic documentation is that it allows other engineers to access the architecture easily for both their own reference and for getting feedback from subject matter experts. The second step, detailed in Section IV, involves translating the architecture into a discrete event simulator for execution and populating the subsystems with algorithmic models[7]. The translation is best done using an automated tool to ensure traceability between the simulation and the architecture. The final development step integrates physical world views, engineering performance views, operational displays, and/or user interfaces as required. The various views developed in this step help to convey what is physically happening during the simulation and to provide the capability for user-in-the-loop decisions to be made. Different options for this step are described in Section V.

III. Integrated Architecture Development

As a testament to the value of an integrated architecture for technically advanced programs the U.S. Congress passed the Klinger-Cohen Act of 1996 (40 U.S.C. 1401(3)), also known as the Information Technology Management Reform Act. The purpose of the act was to reform acquisition laws and information technology (IT) management of the Federal Government. It mandated that new IT acquisitions should be managed as an efficient business, requiring the use of a standardized architecture framework in all of its future acquisitions. In response to this new legislation, the U.S. Department of Defense (DoD) developed what is today referred to as the Department of Defense Architecture Framework[8], or DoDAF, and is based on the Zachman framework[9]. CONOPS Simulation leverages the advantages of using an architectural standard like DoDAF; while NASA by law doesn't have to comply with the DoDAF Standards there are similarities between most architectural styles that may be leveraged for simulation development. To clarify what is meant by an *architecture*, the DoD Integrated Architecture Panel in 1995, based on the IEEE standard 610.12, gave this definition[10], "Architecture: the structure of components, their relationships, and the principles and guidelines governing their design and evolution over time."

An architecture describing a given enterprise or system is typically developed by a team of systems engineers and subject matter experts. This team may not always have good insight into how a complex, interdependent system of systems will behave a priori and this is where a method such as CONOPS Simulation can help in analyzing architectural trade-offs. Key advantages in developing architecture products early in the development lifecycle include the ability to systematically study internal and external interfaces, helping to drive requirement development, and capability gaps can be identified in the system before it is built. Consistency and clarity are critical to developing a productive set of architecture views. Both of these attributes are enhanced by using an integrated set of common definitions and data repositories. The development of an integrated architecture is

typically achieved through the use of an underlying database that contains system definitions, which may be reused throughout all of the different products. This promotes a consistent set of interfaces and nomenclature while providing a traceable means for extracting the architectural information for performing simulations. One other important note about integrated architectures is the fact that they may be used to generate other types of data because of the way that their data is stored in a centralized database. For instance, this can be exploited to output architectural documentation in an easily accessible format such as HTML and can include interactive diagrams that are cross linked between different diagrams and even to requirements. This type of documentation can be a very useful alternative to paper-based static diagrams which can fill many large binders for large systems. This type of documentation is currently being evaluated by the NASA Johnson Space Center (JSC) Missions Operations Directorate (MOD) in the development of the Constellation MCCS architecture.

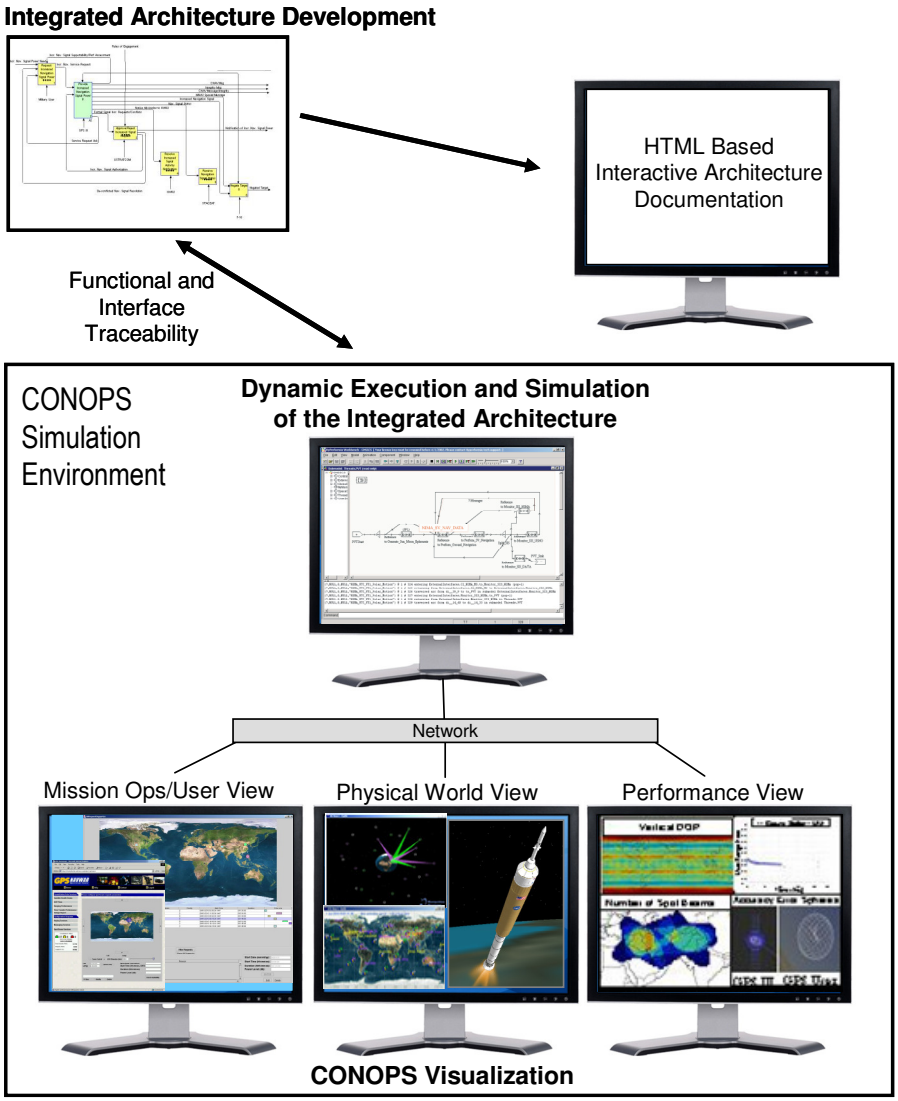

Figure 2. CONOPS Simulation provides a multi-perspective co-simulation that is directly traceable to the architecture.

A CONOPS Simulation can be derived from either an operational perspective or a systems perspective and requires at a minimum two separate types of architecture views. The first of the two required views is a connectivity diagram. In DoDAF this is either the OV-2 (operational view) or the SV-1 (system view). An example of an SV-1 connectivity diagram is shown in Fig. 3, and it graphically shows need-lines or communications links between either operational nodes or systems indicating information exchange.

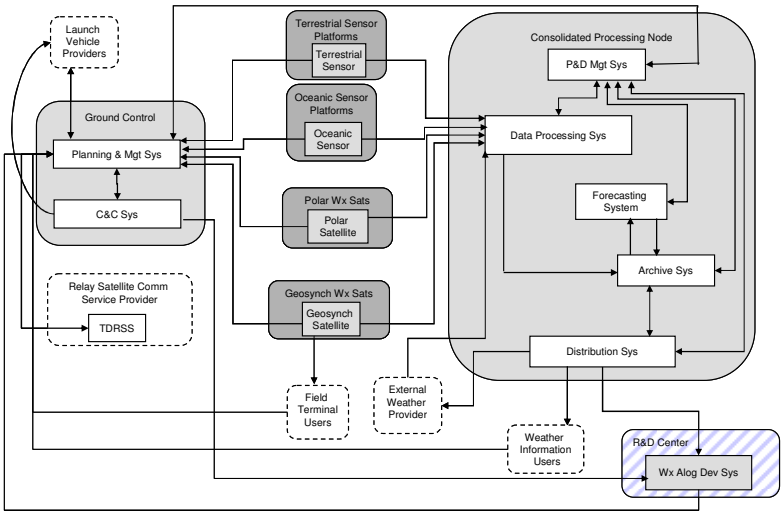

Figure 3. The System Connectivity Diagram identifies links and interfaces between nodes or systems.

The second type of view is an event trace diagram, represented by either an OV-6c or SV-10c in DoDAF. An example of a system event trace diagram is shown in Fig. 4 and contains *swimlanes* (drawn vertically in the example) for each operational node or system, and a time ordered depiction of information exchanges across the various swimlanes. The development of event trace diagrams usually depends on a specific scenario or operating conditions, and is the basis for the next step of the CONOPS Simulation development process. Furthermore, many different event trace diagrams may be required to handle different types of contingencies.

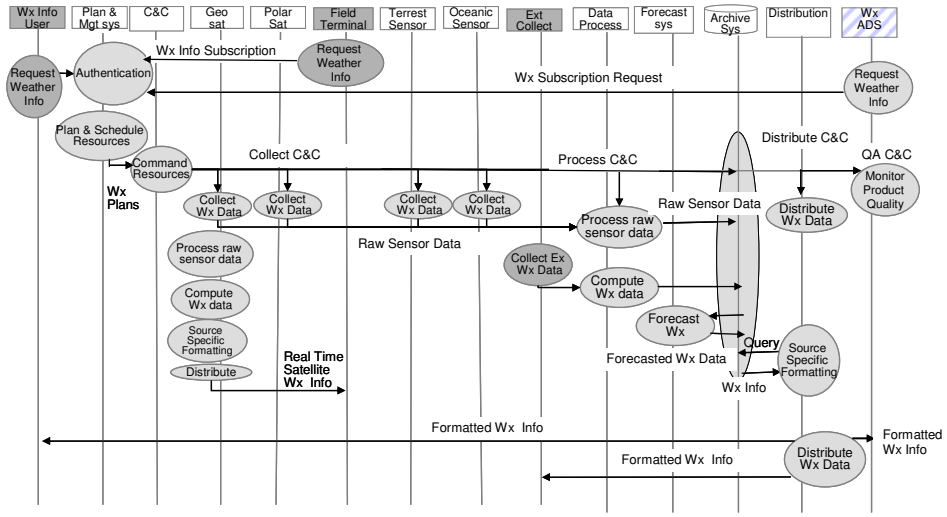

Figure 4. The System Event Trace Diagram provides a sequence of events for the specific scenario.

There are a multitude of commercial off-the-shelf (COTS) options for integrated architecture development tool suites. Deciding which one to use is greatly dependent on each project's preferences and requirements. A few of the more popular choices include: Telelogic's Rhapsody®, Telelogic's System Architect®, 3SL's Cradle®, and Vitech's CORE®.

IV. Dynamic Execution and Simulation of the Integrated Architecture

This step is one of the key differentiators of CONOPS Simulation from standard systems engineering practices and generally represents the bulk of the development effort depending on the maturity of the architecture. This step consists of three main phases: translation of the architecture into a simulation engine, population of the algorithms in the simulation engine, and developing co-simulation interfaces as appropriate. Typically, the simulation engine is a discrete-event simulator[11] with the ability to perform many events in parallel. This type of simulation is used because of the event driven nature of most concepts of operations.

To ensure traceability to the architecture, a means of automating the translation of the architecture from the application in which it was developed or from its corresponding database to a simulator is highly recommended. This automation can be either a commercial product, a custom developed piece of software, or can be done using an architecture tool with an integrated simulation engine. It should be noted that current architecture applications that have internal simulation capabilities are generally much less powerful and provide less flexibility than dedicated simulation applications. The translation should at least include the required operational nodes or system entities, their logical connectivity, node and entity hierarchies. The translation could also include parameters or model data to be used in the simulation for each node or entity. Once the translation is complete, the simulation engine should contain a shell representation of the architecture that is ready to be populated with the underlying models and algorithms.

After the translation from architecture is completed, the initial level of fidelity of the simulation should be determined. The level of fidelity can range between simple timing delays for a function to using real operational code and hardware in the loop co-simulation. The fidelity of the simulation can be easily increased as more data becomes available during the different design phases. The main effort of this step is in implementing the functionality of each element in the simulation and the timing of when events are to occur. In modeling the simulation elements, architectural variables can be included for doing trade study analysis and optimization studies. The event sequence diagrams can be very helpful in this step for both development and validation of the simulation. Not all of the data processing or algorithmic details need to be included in the discrete event simulator, but it is often very beneficial to co-simulate with other models or analytic applications to improve the model fidelity, and custom interfaces may be developed to include human decision making in the loop. Another addition to the simulation in this step is a reoccurring event to update any visualization displays that are to be included; this event is generally independent of the rest of the simulation, but is useful for assuring that the time variable in the discrete event simulation is updated at desired intervals. This is needed because the discrete event simulator skips through time based on when events are scheduled to occur, which can lead to large jumps in time, causing the visualization displays to appear choppy and difficult to follow.

Once the simulation model has been completed, it is considered to be an executable architecture because of its traceability back to the original architecture. The executable architecture can be used in different ways. It can be networked with the visualization and co-simulation tools for demonstrations, or it can be run in a batch mode for more in depth analytical studies. The analytical studies can be statistical in nature such as Monte Carlo methods or for trade space exploration depending on the type of variables that are implemented in the simulation. A few of the different visualization options are further examined in the next section.

V. CONOPS Visualization

Including visualization is the final step in developing CONOPS Simulation and can be a powerful way to communicate the concept of operations and key architectural components to a wide audience. The reason it is so powerful is because it essentially brings a static architecture to *life*. Coupling the animated data flows in the discrete event simulator to a real world graphical representation can bring together engineers, management, and customers of various backgrounds and disciplines who may be comfortable looking at an architecture, a simulation, or maybe just one of the subsystems. With CONOPS Simulation, they now have the ability to understand where those disciplines interact within the enterprise in a graphical representation. This synergy can spark creative multidisciplinary discussions resulting in even better designs or help communicate particular problems; it is this context that the slogan, *fly before you buy*, is realized.

The most common type of visualization used in CONOPS Simulation is a two or three-dimensional model of the various systems and their environment. This modeling can be done using a custom or COTS analysis application

and provides a time synched physical representation of what is occurring as the architecture is executed. Typically, the different visualizations are connected to the discrete event simulation through a network so that they may be displayed simultaneously. Many other types of visualizations can also be included as highlighted in Fig. 2. Event checklists that are updated to reflect completion of major events in the scenario are useful for increasing understanding of what events occur at a given time. Dynamically updated analytical graphs can also help in comprehending the performance of the architecture through time.

VI. Case Study: Launch/Ascent Contingency Voice

During the launch of a manned spacecraft, communications between the crew and ground controllers at mission control are very important. Communications are especially important in the case of a vehicle failure requiring the ground to relay to the crew that they need to execute an abort. Standard S-band communication links can experience loss of connectivity during launch due to many factors including the rocket exhaust plume and maneuvers that affect the spacecraft's attitude. Therefore, NASA requires a contingency voice mode during the launch and ascent flight regimes. The contingency voice uses a different frequency than the S-band communication link and has to be time-synched with the S-band link such that they both arrive at the crew's headset at the same time. If the communication links are out of synch, it causes an echoing effect which can hinder the crew's understanding of the communication transmissions. For Shuttle operations a UHF broadcast from ground stations along the East Coast of the United States is used for the dissimilar contingency voice. These ground stations are set to be decommissioned by the time of the first manned Constellation program launch. One alternative being looked at in a Constellation architecture trade study as a possible replacement for the contingency voice is the use of the Mobile User Objective System (MUOS), which is a Navy UHF communications satellite that is planned to be operational by 2010. For this case study, a CONOPS Simulation is developed for studying the use of MUOS as the contingency voice for the Orion spacecraft (the systems involved in the scenario have been simplified for clarity).

Following the CONOPS Simulation development process, the first step was to develop an integrated architecture, and the first views developed were the logical connectivity diagrams. The highest level diagram, given in Fig. 5, shows the high-level connections for the scenario between the Mission Control Center System (MCCS) and its external interfaces, which include the MUOS Satellite, Tracking and Data Relay Satellite System (TDRSS), Orion Crew Exploration Vehicle (CEV), and the Aries I Crew Launch Vehicle (CLV). System interface diagrams were also created, where needed, particularly for the MCCS.

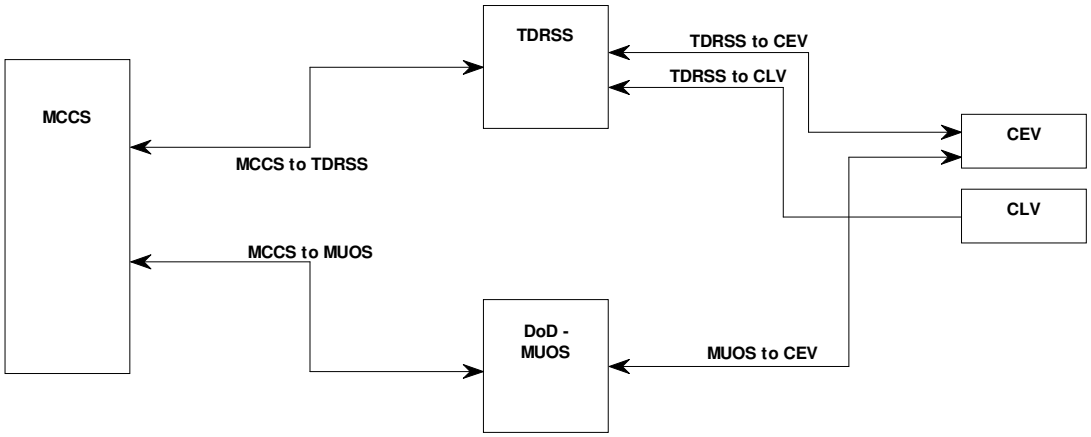

Figure 5. Contingency Voice High-Level Systems Interface Diagram

Next, an event trace diagram was created for the scenario and is shown in Fig. 6. The event sequence runs through an initial loopback of both communication systems to determine the time synch delay parameter followed by a check of the system and then voice transmissions.

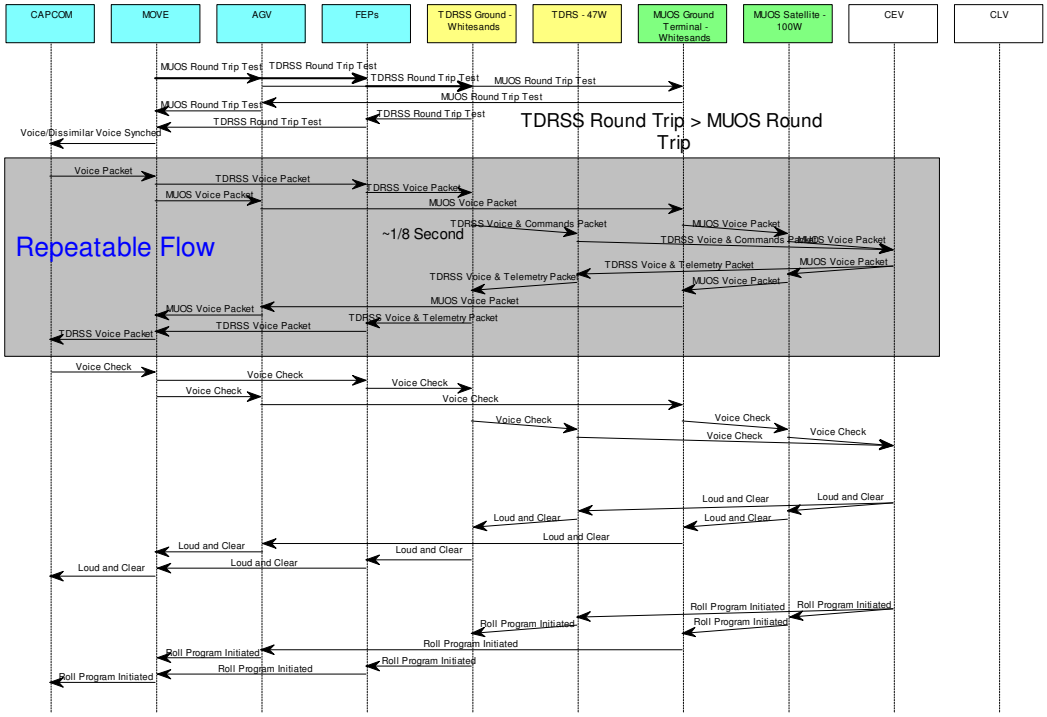

Figure 6. Contingency Voice Event Sequence Diagram

Translation of the architecture to the discrete event simulator was performed using a custom program written to work with our specific toolset choices. This conversion translated the high-level interface diagram, and the child diagrams that included the respective subsystems. Then, the behavior of each system and subsystem were added to the simulation model and consisted mainly of data routing logic and their associated latency timing parameters.

A three-dimensional modeling and analysis tool was used for both visualization and for simulating the communications links. The tool provided information, such as line of sight between the various systems as well as range and range rates. Two specific three-dimensional views were incorporated into the CONOPS Simulation to help the user or observer understand where the different systems are located in space, and how they are interrelated. The first view followed the vehicle while the second view showed the overall communications status.

Execution of the CONOPS Simulation combined the animation of the discrete event simulator with the two three-dimensional views running simultaneously. A rendition of this demonstration showing a data flow between subsystems of the MCCS being executed is given in Fig. 7. The CONOPS Simulation developed in this case study provided means of communicating the MUOS alternative for contingency voice, along with identification of key external interfaces, and the capability of analyzing time synchronization issues with the architected ground systems and dissimilar voice communication links.

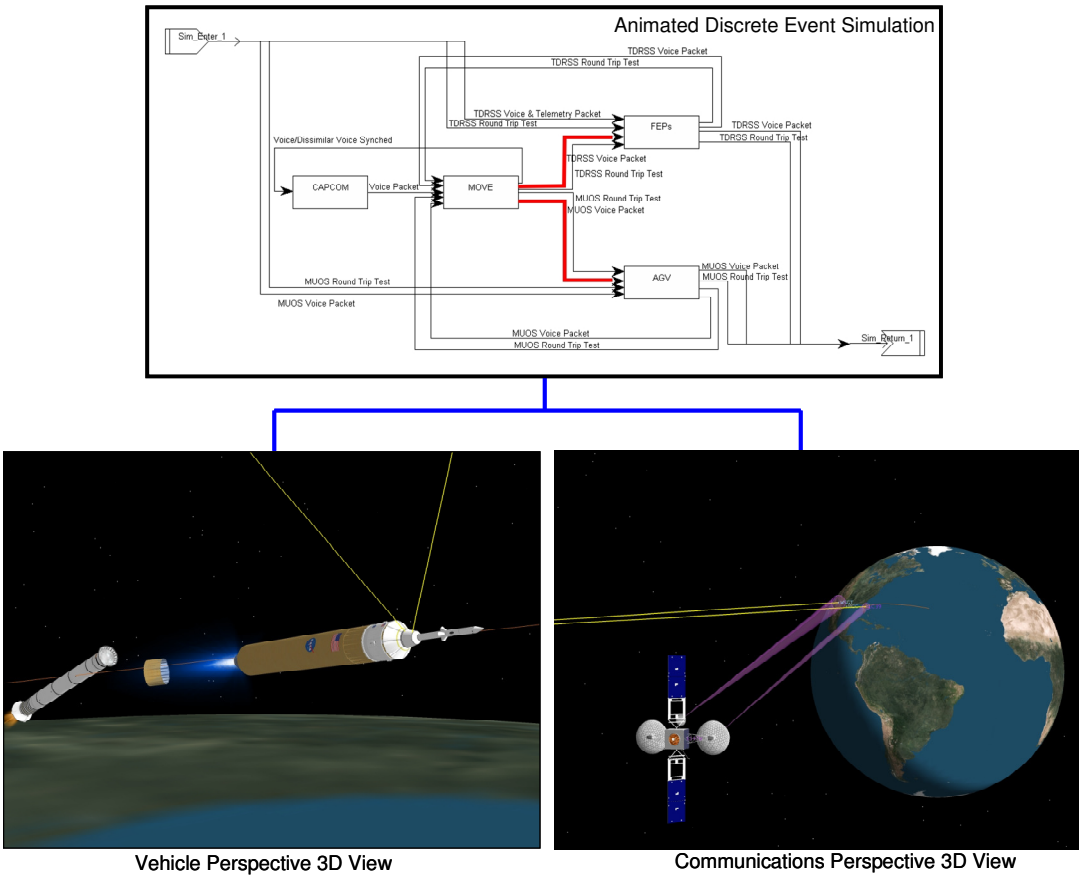

Figure 7. Contingency Voice CONOPS Simulation Demo Displays

VII. Conclusion

The development of a large-scale system is a daunting challenge in the face of time and budget constraints. Historically, it has been shown that when rigorous systems engineering approaches are applied to such programs, cost and schedule overruns can be better controlled. This paper presented a method, CONOPS Simulation, that combines traditional systems engineering approaches like integrated architectures, discrete event simulation, and visualization displays to help meet those challenges. Key CONOPS Simulation benefits include the ability to analyze different architectural alternatives and to communicate concepts of operations to diverse audiences. These benefits were demonstrated using a scenario from work done at NASA on a case study to analyze the contingency voice communications between ground controllers and the crew of the Orion spacecraft during the launch and ascent mission phases. In conclusion, CONOPS Simulation is a unique simulation methodology that is traceable back to an integrated architecture, which can be helpful in credibly solving a variety of technical problems and in defending those solutions.

Acknowledgments

The authors would like to thank the following people for their help in preparing the examples and case study presented in this paper and for their feedback of draft versions of the manuscript: David McGill, Matt Crozier, Craig Coburn, Keith Everett, and Dan Littley.

References

[1] Bush, George W., *President Bush Announces New Vision for Space Exploration Program*, Speech presented at NASA Headquarters, Washington, D.C., 14 January 2004.

[2] National Aeronautics and Space Administration, *NASA's Exploration Systems Architecture Study*, NASA-TM-2005-214062, November 2005.

[3] United States, Cong. Senate. 109th Congress, 1st Session. S. 1281, *National Aeronautics and Space Administration Authorization Act of 2005*. 109th Congress. Congressional Bills, GPO Access. 30 Dec 2005 <http://purl.access.gpo.gov/GPO/LPS68900t>.

[4] Griffin, Michael, *Why Explore Space?*, Speech given 17 January 2007. <http://www1.nasa.gov/pdf/167331main_Griffin_why_explore_0116.pdf>.

[5] Gruhl, W. *Lessons Learned, Cost/Schedule Assessment Guide*, Internal presentation, NASA Controller's Office, 1992.

[6] Honour, E.C. *Understanding the Value of Systems Engineering*, Proceedings of the INCOSE International Symposium, Toulouse, France, 2004.

[7] Gano, S.E., Kim, H., Brown, D.E., *Comparison of Three Surrogate Modeling Techniques: Datascape, Kriging, and Second Order Regression*, Proceedings of the 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, AIAA-2006-7048, Portsmouth, Virginia, Sept 6-8 2006.

[8] U.S. Department of Defense, *DoD Architecture Framework Version 1.5, Vol I: Definitions and Guidelines*, 23 April 2007, < http://www.defenselink.mil/cio-nii/docs/DoDAF_Volume_I.pdf>

[9] Zachman, John A., *A Framework for Information Systems Architecture*, IBM Systems Journal, vol. 26, no. 3, 1987. IBM Publication G321-5298.

[10] U.S. Department of Defense, *DoD Architecture Framework Version 1.5, Vol III: Product Descriptions*, 23 April 2007, <http://www.defenselink.mil/cio-nii/docs/DoDAF_Volume_II.pdf>

[11] Banks, J., Carson, J. S. II, Nelson, B. L., and Nicol, D. M., *Discrete-Event System Simulation*, 3rd ed., Prentice Hall, 2000.