

Generating Distributed Interactive Simulation (DIS) Codebases using `opendis7-source-generator`

Don Brutzman, Rick Lentz, Terry D. Norbraten, Christian Fitzpatrick, Curtis L. Blais

Modeling Virtual Environments Simulation (MOVES) Institute
Naval Postgraduate School (NPS), Monterey California USA

brutzman@nps.edu rlentz@gmail.com tdnorbra@nps.edu christian.fitzpatrick@nps.edu clblais@nps.edu

Keywords: Distributed Interactive Simulation (DIS), Streaming, Command and Control (C2), Live Virtual Constructive (LVC), Unit testing, Validation Verification Accreditation (VV+A), Testing Development Operations (TestDevOps), X3D Graphics, XML EXI JSON

ABSTRACT:

The `opendis7-java` library is a major upgrade now providing 100% coverage of all 72 PDU types in IEEE DIS version 7, as well as full coverage of over 22,000 SISO Enumeration data structures identifying diverse entities, sensors, weapons, domains, nationalities, etc. as Plain Old Java Objects (POJOs). A singleton threaded network interface class facilitates integration of DIS reading and writing together with diverse Java applications. Special emphasis has been placed on recording of PDU streams in multiple encodings (native binary, plaintext, base64, XML, EXI, JSON). Following the principle “a stream is a stream” we are beginning to show support for repeatable unit testing with expected benefits for sustainable Live Virtual Constructive (LVC) interoperability, Validation Verification Accreditation (VV+A), and Testing Development Operations (TestDevOps). A growing set of examples, course projects, and tutorial assets demonstrate effective `opendis7-java` library usage.

This paper also describes DIS design rationales and interoperability standards for streaming, diverse file encodings of simulation data, Rich Semantic Track (RST), C2SIM bridging and a Data Strategy for Unmanned Systems. Production of multiple examples demonstrates the broad utility of DIS-based track analysis, replay and visualization.

1. Overview

A distinguishing hallmark of the IEEE Distributed Interactive Simulation (DIS) Protocol [1] is that it provides data-centric representations for carefully defined data messages. Message structures, semantics and interaction patterns have only been standardized following extensive working-group evaluation. Implementing software that produces and parses the strict data messages conforming to this open standard is permitted to vary widely.

The Open-DIS project is a long-running effort to produce open-source software libraries that implement the DIS protocol in a variety of programming languages, unrestricted for any use. Motivations for use include modeling of realistic representations of entities and behaviors, distributed simulation programming, and practical interoperation between applications to promote collaboration, research, and education.

The open-dis project [3.0] has evolved over many years, slowly but steadily increasing in scope and capability. Prior to the current revision, the essential design was successfully developed by Don McGregor at NPS and produced codebases with coverage across multiple programming languages: Java, Python, JavaScript, C++, C# and ObjectiveC. Each codebase worked satisfactorily and remains available on the public version-control website. Sadly, due to his untimely passing, Don’s work was not fully completed and only partial coverage of the many Protocol Data Units (PDUs) defined by the DIS Protocol was accomplished. The current generation of this code remedies these gaps, applying Don’s design across the full range of the IEEE DIS version 7 specification [1] and associated SISO Enumerations definitions [2].



Figure 1. OpenDIS Surfer Dude, with thanks to Don McGregor.

An innovative design expressed data structures for all IEEE DIS Protocol Data Units (PDUs) [1] in XML, permitting source-code generators to produce a variety of functionally similar codebase libraries. The original project provides support for approximately half of the DIS vocabulary using multiple programming languages (Java, Python, JavaScript, C++/C#) and file encodings (XML, JSON, EXI). This earlier generation of code is online and remains useful.

2. opendis7-source-generator Architecture

The original innovative design representing data structures for all DIS Protocol Data Units (PDUs) in XML proved to be an excellent basis for opendis7-source-generator [3.2] codebase refinement. The central design takes advantage of custom XML definitions for each DIS family and PDU time, with amplifying annotations explaining each data structure included in the definitions. Greater detail and rigorous cross-checking was added in this major refactoring of the earlier library. Development is ongoing.

Two sources of data are utilized: the formal SISO enumerations and locally produced DIS definitions, both in XML, which are subsequently converted into compiled source code. Autogeneration input and output files are highlighted in blue in Figure 2. Additional utility classes are manually authored to round out all functionality needed in each library.

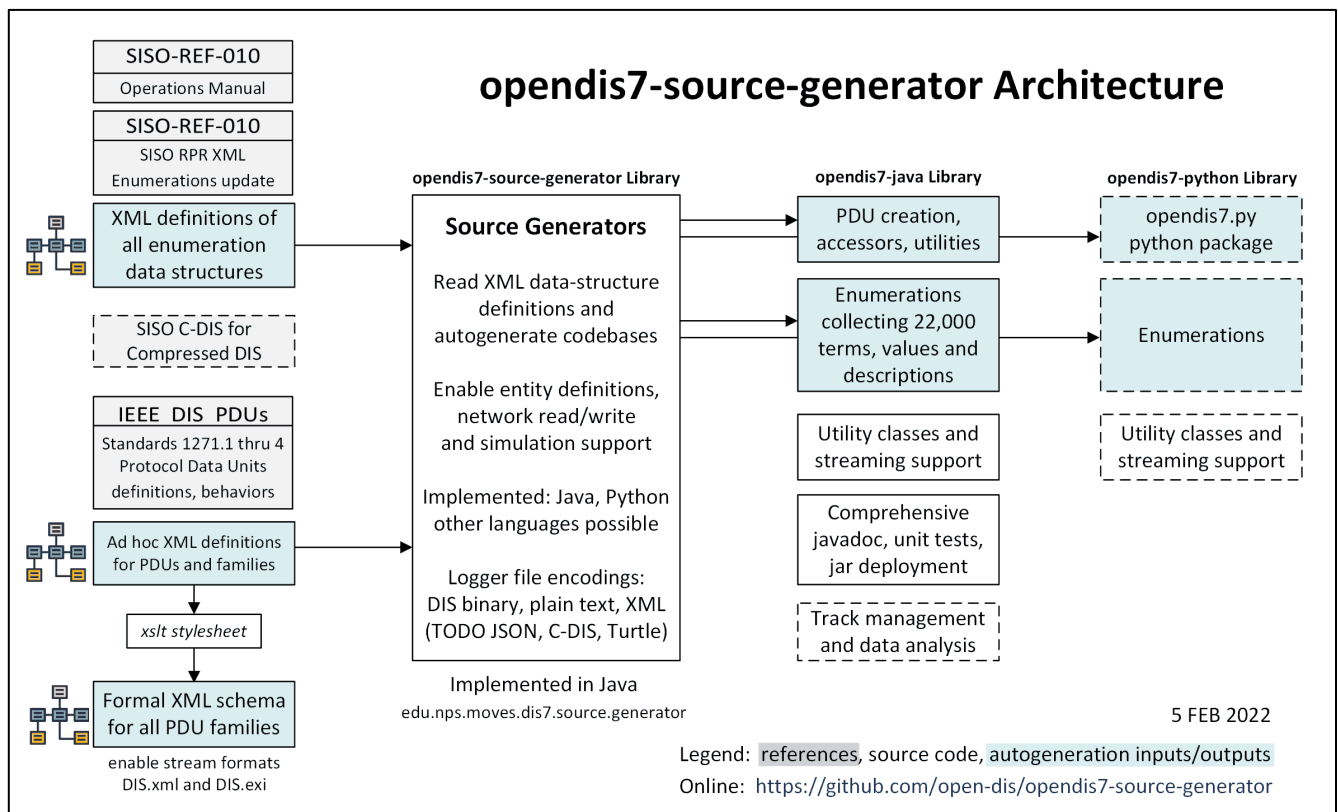


Figure 2. The opendis7-source-generator Architecture emphasizes strict adherence to the IEEE DIS Protocol through definition of corresponding data structures that are used to autogenerate best-practice software libraries in multiple programming languages.

SISO-REF-010 enumerations [2] are updated via formal working-group review, updated weekly, and published annually. They are also extensive: currently 88, 320 lines of XML source are converted into 24,651 well-documented Java classes. Clearly such massive assets provide a major step towards large-scale interoperability for distributed simulations. Multiple output encodings are also possible. The primary form (of course) is the well-defined binary PDU format specified by the IEEE DIS Protocol itself, with an example shown in Figure 3. NPS testing and experimentation has also shown that alternate formats for plain text, XML and EXI are also useful. Future support for JavaScript Object Notation (JSON), Compressed DIS (C-DIS), and Semantic Web relations using Terse Triple Language (Turtle) syntax are also expected.

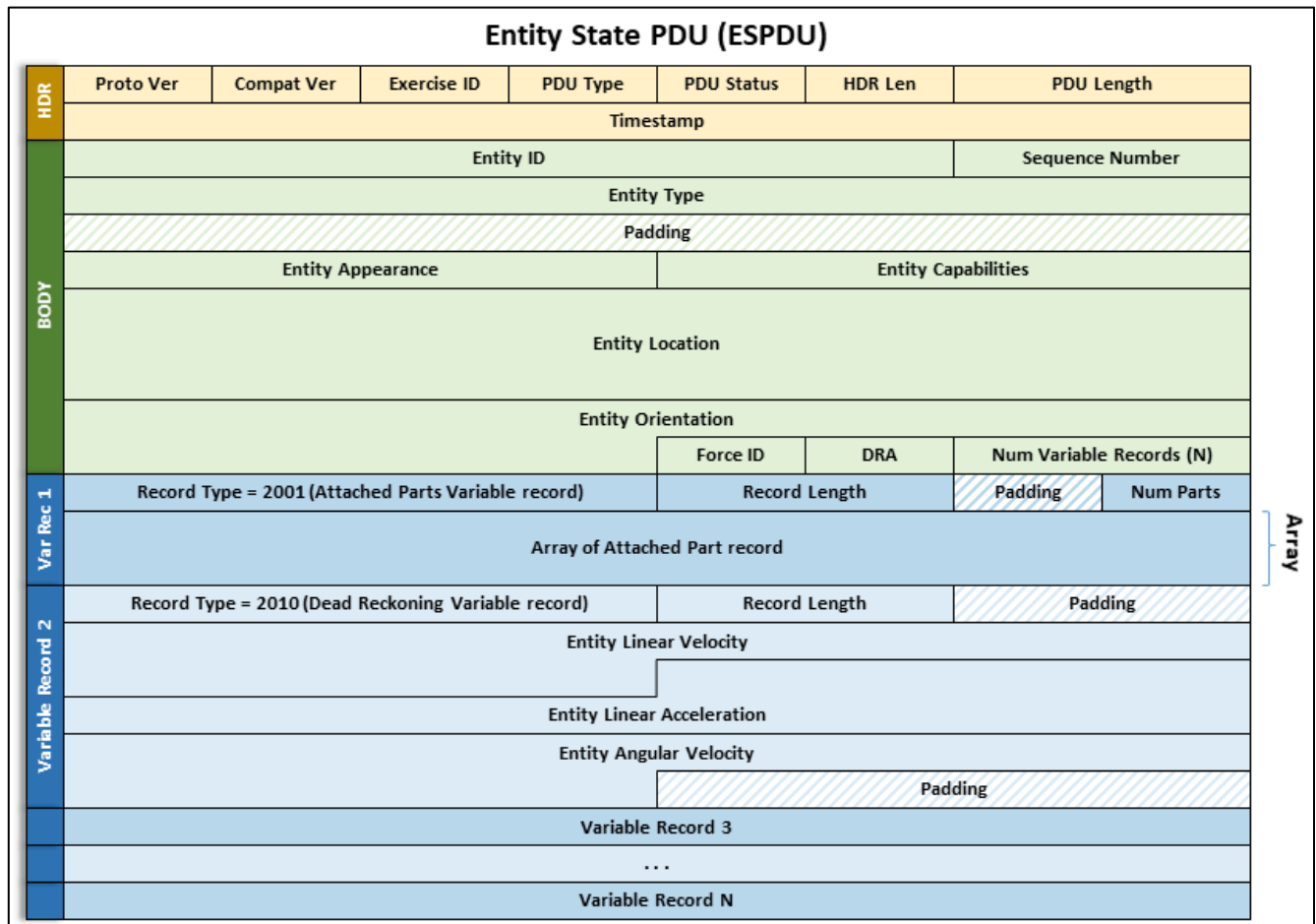


Figure 3. Binary formats are strictly defined for a total of 72 IEEE Distributed Interactive Simulation (DIS) Protocol Data Units (PDUs). As an example, field lengths for the Entity State PDU (ESPDU) are shown here, with each row representing 8 bytes (32 bits).

An XSLT stylesheet is used to re-express local definitions as a strictly designed XML Schema for DIS. This enables DIS packets and streams to be defined as validatable XML documents. XML-based files can be digitally signed and also encrypted using the XML Security standards. XML definitions further enable type-aware compression of XML streams using the Efficient XML Interchange (EXI) standard by World Wide Web Consortium (W3C). EXI provides best-of-breed compaction of data along with significant increases in computational performance and memory consumption which are especially important for power-constrained unmanned systems.

Flexibility in output formats when recording DIS stream assets is expected to facilitate the production of analytic tools and diverse post-processing algorithms for evaluating exercise activities and outcomes. Becoming adept at such practices holds the potential of facilitating exposure of DIS behavior streams using Big Data techniques. Much future work awaits.

2.1 opendis7-java

The opendis7-java library [3.2] is a major upgrade providing 100% coverage of all 72 PDU types in IEEE DIS version 7, with full coverage of over 22,000 SISO Enumeration data structures identifying diverse entities, sensors, weapons, domains, nationalities, etc. as Plain Old Java Objects (POJOs). A singleton threaded network interface class facilitates integration of DIS reading and writing with diverse Java applications. Additional utility classes simplify socket connections, PDU recording and playback, creation of new simulation programs, and other common tasks facing simulation programmers. Course project [4] and thesis testing continue to improve library reliability.

Annotation information from each of the XML definition files is carried forward and further augmented by the Source Generator routines so that both summaries and in-depth descriptions are provided in context when programming with an Integrated Development Environment (IDE) such as Netbeans. Full searchable documentation is also published as Javadoc, both with the `opendis7-full.jar` distribution and online, as shown in Figure 4.

The screenshot shows the 'open-dis7 Javadoc' website. At the top, there's a navigation bar with tabs: OVERVIEW, PACKAGE, CLASS, USE, TREE, DEPRECATED, INDEX, and HELP. A search bar on the right contains the text 'DisTime'. Below the navigation bar, the page title is 'open-dis7 Javadoc'. A paragraph describes the library: 'The open-dis7-java library provides a complete type-safe Java implementation of both the DIS Protocol version 7 (IEEE 1278.1-2012) and SISO-REF-010 Enumerations specifications, interfaces and objects, all as open source.' This is followed by a bulleted list of features:

- Over 22,000 SISO-REF-010 enumeration values are provided.
- `edu.nps.moves.dis7.pdus` package contains IEEE DIS Protocol Data Unit (PDU) packet-definition classes.
- `edu.nps.moves.dis7.utilities` package contains network interface and data-conversion utilities.
- `edu.nps.moves.dis7.utilities.stream` package contains stream utilities for DIS file saving, conversions and playback.
- Experimental DIS XML schema and documentation assist with data-centric processing of DIS streams.

 To the right of the text is a logo featuring a silhouette of a person surfing on a wave, with the text 'OPEN-DIS' below it. Below the list, a paragraph mentions: 'Distribution binary available at `open-dis7-full.jar`. Further examples, presentations and projects are provided in Networked Graphics MV3500 course archive by the Modeling, Virtual Environments, Simulation (MOVES) Institute of the Naval Postgraduate School (NPS)'. Below this is a section titled 'Packages' containing a table:

Package	Description
<code>edu.nps.moves.dis7.entities</code>	The entities packages provide a large number of autogenerated utility classes for world entities of interest.
<code>edu.nps.moves.dis7.entities.afg.lifeform.land</code>	Afghanistan (AFG) LIFE_FORM LAND typed classes for world entities defined by SISO-REF-010-v30-DRAFT-20220129-d11 (2022-01-29) enumerations.
<code>edu.nps.moves.dis7.entities.afg.platform.land</code>	Afghanistan (AFG) PLATFORM LAND typed classes for world entities defined by SISO-REF-010-v30-DRAFT-20220129-d11 (2022-01-29) enumerations.
<code>edu.nps.moves.dis7.entities.alb.platform.surface</code>	Albania (ALB) PLATFORM SURFACE typed classes for world entities defined by SISO-REF-010-v30-DRAFT-20220129-d11 (2022-01-29) enumerations.
<code>edu.nps.moves.dis7.entities.are.platform.air</code>	United Arab Emirates (ARE) PLATFORM AIR typed classes for world entities defined by SISO-REF-010-v30-DRAFT-20220129-d11 (2022-01-29) enumerations.

Figure 4. *opendis7-java* products include full Javadoc documentation for all enumeration and PDU classes.

2.2 `opendis7-python`

To prepare for upgrading the previously published `opendis python` library, a trial project was performed by Rick Lentz that mapped aircraft track information to DIS [3.3] [4]. Automatic Dependent Surveillance-Broadcast (ADS-B) is an international standard for reporting aircraft position updates, supporting shared track for safe navigation and efficient operations. Data fields for ADS-B traffic provide position via latitude and longitude, course, speed, altitude, and vertical airspeed in feet per minute. Data from a nearby airport was captured in September 2021 using an open-source software-defined radio (SDR) that monitored tower control traffic, recorded ADS-B reports, and then used the original `open-dis-python` library to share and save corresponding PDU streams. Over 10 hours of recorded position-report traffic from aircraft within a 200 nautical-mile radius resulted in 277K PDUs recorded for thousands of aircraft. Preliminary aircraft identification via EntityID, SiteID, and ApplicationID were extracted from ADS-B messages to populate each ESPDU. Once corresponding aviation-database information and DIS enumeration values are available, similar approaches can enrich such values uniquely for historical ADS-B recordings as well. Figure 5 illustrates this process.

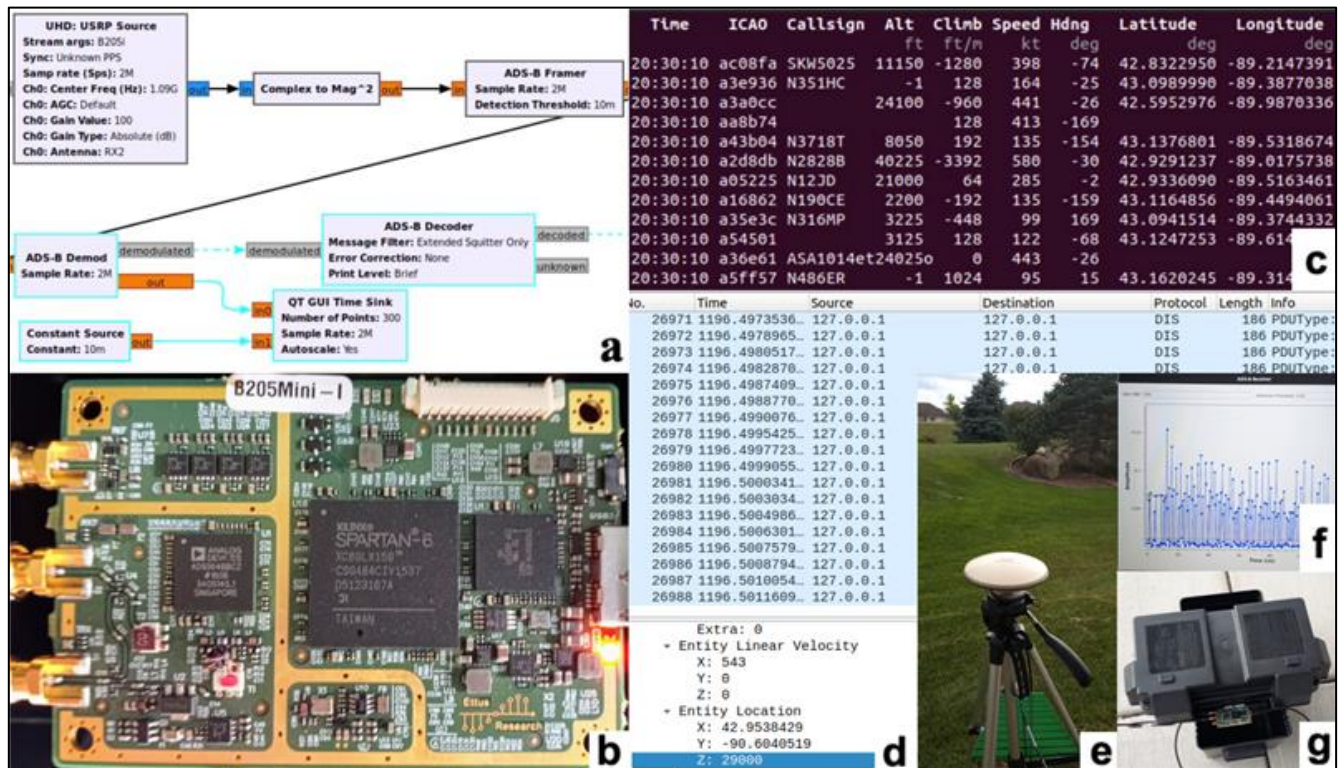


Figure 5. Testing legacy python DIS using a software-defined radio. Image mosaic shows (a) GNU Radio block diagram, (b) Ettus Research USRP B205mini-i Software Defined Transceiver, (c) ADS-B PDU Encoder console output, (d) Wireshark UDP network capture of ADS-B messages, (e) backyard antenna position, (f) demodulated signal response, and (g) radio transceiver box with USB connection.

The opendis7-python source generator is again operational, restoring prior python-generation capabilities for DIS. A full package is now undergoing testing and refinement, with corresponding design and autogeneration of enumeration classes underway. Publication is planned for Python Package Index (PyPi) to facilitate wide availability and update capabilities for Python programmers. Subsequent projects will demonstrate use of the Jupyter library to combine opendis.py and x3d.py programming and visualization within Web browsers, facilitating even broader re-use and new applications.

When working across multiple codebases, we have found that it is not particularly difficult to adapt diverse functionality, enumerations, algorithms and heuristics from one programming language to another. We expect that the functionality of each programming-language variant in the overall opendis7 project will evolve consistently and compatibly.

3. Examples and Testing

Since the opendis7 library is produced for use with other applications, the development of examples is essential. A growing set of examples, course projects, and tutorial assets demonstrate effective opendis7-java library usage. A short set of simple examples is also provided as part of the official distribution. Given the richness and breadth of the DIS Protocol, in-depth NPS examples are typically produced and published in concert with project work.

NPS course MV3500 Networked Graphics and Simulation provides an in-depth look at DIS programming to MOVES graduate students. This course is project based and four years of student assignments and projects are available online. The opendis7 library has correspondingly matured during this interval as usage revealed issues and opportunities. The new library has also been used in tutorials [5] and graduate theses [6].

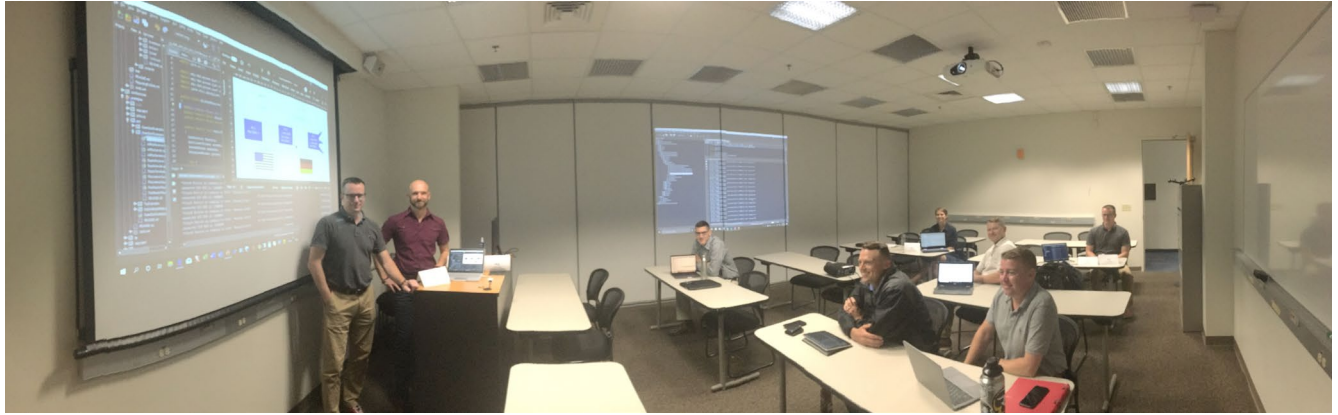


Figure 6. NPS graduate students use the opendis7 libraries as part of the MOVES MV3500 course [4] and thesis work.

A unit-testing suite in the opendis7-java distribution performs regression tests to confirm that ongoing codebase development does not break existing capabilities. While the current set of tests is not fully comprehensive across the entire library, such checking is definitely useful and also provides an additional set of source-code examples. Since each opendis7 library is itself autogenerated, we see no reason blocking the corresponding autogeneration of entire test suites. This is an important area of future work. If repeated across multiple programming languages, such a test suite for Quality Assurance (QA) might also be used to further increase confidence regarding interoperability across multiple programming languages and application tools.

Serialization and deserialization of PDUs is the most sensitive and critical conformance task. It can also be the most difficult to get exactly right. Since routines are consistently produced for each data type whenever it might occur, having autogenerated routines definitely helps support implementation of correct PDU creation and parsing (known as unmarshalling and marshalling in Java). When testing, round-trip creation/sending and receipt/parsing provides unambiguous results for the entire production chain. Successful test results are committed into version control so that subsequent round-trip stream collections can be compared as regression tests, effectively confirming success or else isolating emergent issues. Open-source publication of tests and results further helps demonstrate compatibility across a wide range of application versions, programming languages, and operating systems.

To date, heavy-duty testing has only been performed using the opendis7-java library, which now appears ready for broad use. Similar testing is expected to be performed this year using the opendis7-python library.

Following lengthy development on a large architecture with two phases of autogeneration, we are transitioning development, issue tracking and code improvements back into public forums on GitHub. Work in progress includes design refinement, autogeneration of an open-dis7-python library, and track distillation to create both X3D animation interpolators [9] and corresponding KML waypoints [8] for visualization of entity tracks.

As shown in Figure 6, we have found that the popular Wireshark open-source network-monitoring tool includes excellent built-in DIS support, facilitating comparison testing of proper DIS PDU serialization/deserialization.

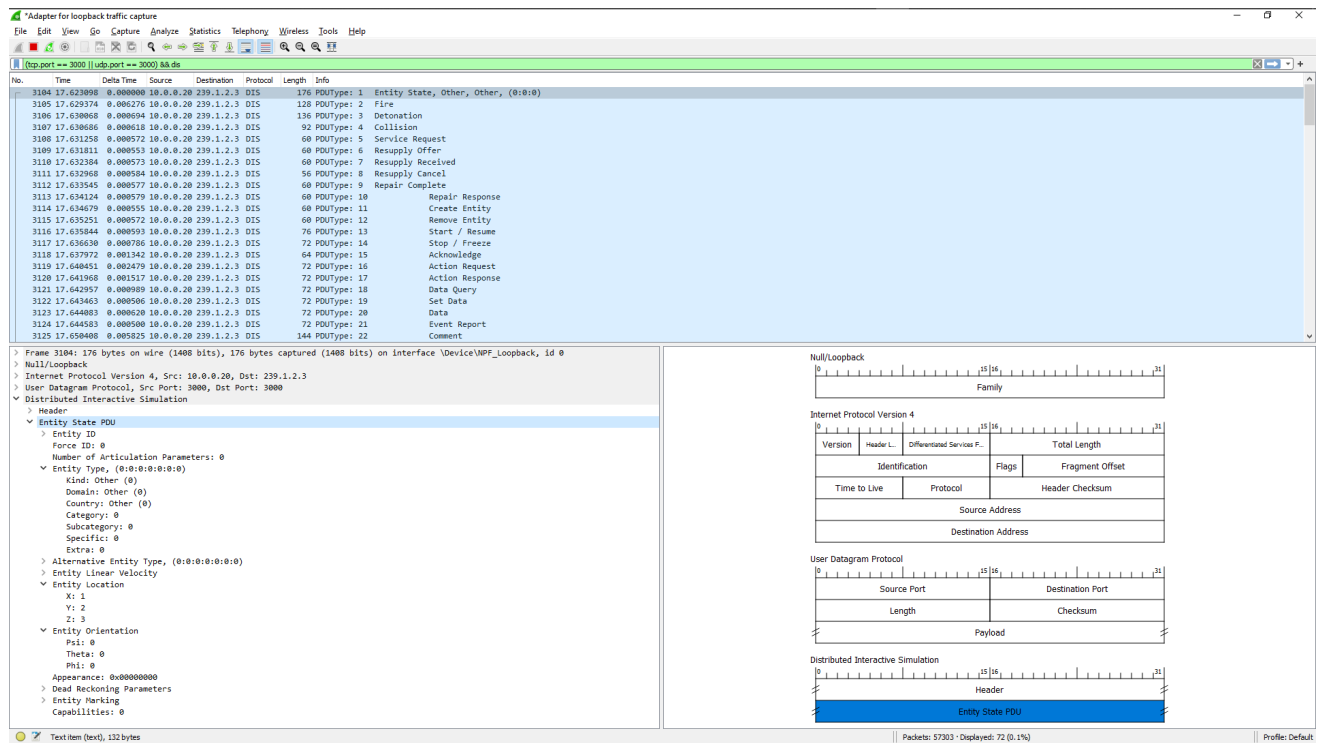


Figure 7. Wireshark support for the IEEE DIS Protocol is excellent, facilitating packet testing and debugging.

4. DIS Streaming, Logging and File Encodings

A guiding heuristic for this project is “A Stream is a Stream is a Stream.” This is derived from Unix stream design which unified methods for file access and network access, a design principle that has continued as common practice in most modern programming languages and operating systems. Live streams flowing across a network for a distributed simulation are received by each application one PDU at a time, forming a sequenced list of PDUs, which can be further sorted by timestamp heading found in each PDU that was produced by each participating application. If sent to a file (commonly called log files) then the information is identical. It is possible to consider such streams as “data in motion” or “data at rest.” A recent NPS thesis explored these principles in detail [4].

- REPEATABLE UNIT TESTING OF DISTRIBUTED INTERACTIVE SIMULATION (DIS) PROTOCOL BEHAVIOR STREAMS USING WEB STANDARDS
- Brennenstuhl, Tobias, Master’s Thesis, Naval Postgraduate School (NPS), Monterey California, June 2020.
- *Abstract.* The IEEE Distributed Interactive Simulation (DIS) protocol is used for high-fidelity real-time information sharing among simulations and trainers across the entire international Modeling and Simulation (M&S) community. If archivally saved and replayed, DIS streams have the potential to become a valuable source of Big Data. The availability of archived prerecorded behavior streams for replay, adaptation, and analysis can benefit an immense variety of application areas. The computer science principle “a stream is a stream” indicates that data in motion is equivalent to data at rest. This characteristic can enable powerful capabilities for DIS. This thesis presents prototypes to demonstrate how various forms of repeatability are key to gaining improved benefits from DIS stream analysis. Unit testing of DIS behavior streams allows confirmation of both repeatability and correctness when testing all manner of applications, exercises, simulations, and training sessions. A related use case is automated after-action review (AAR) from recorded DIS streams. This thesis also shows how a DIS stream is converted into autogenerated code that can animate an X3D Graphics model. Many obstacles were overcome during this work, and so various best practices are provided. Of note is that unit testing might even become a contract requirement for incrementally developing and stably maintaining Live Virtual Constructive (LVC) code bases. This progress provides many opportunities for future work.

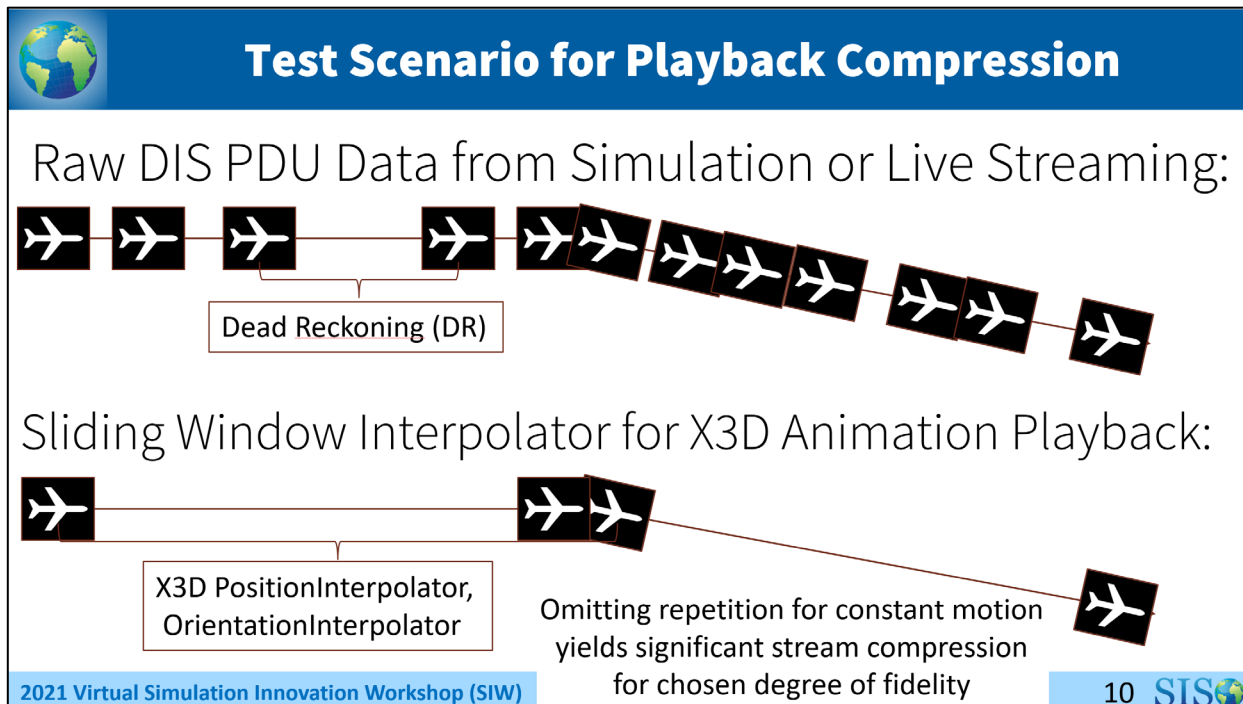


Figure 8. OpenDIS recording of data streams with dead-reckoning (DR) linear-regression data reduction and corresponding production of X3D interpolators for animation playback in 3D virtual environments.

The IEEE DIS specifications deliberately define only a single set of binary representations for application interoperability. Strict adherence to over-the-wire transmission of PDUs is fundamentally important, allowing completely dissimilar applications to cooperatively share state without restrictions on either software applications, programming languages or operating systems. Conducting distributed simulations in this manner means that DIS PDUs can essentially be considered “data in motion” that is effectively shared with consistent, specification-defined syntax and semantics. Recorded log files are “data at rest.” Certain special cases (such as mobile robots with both internal and external stream participation) can be considered simultaneously in motion (externally) or at rest (internally).

Special emphasis has been placed on recording of PDU streams in multiple encodings (native binary, plaintext, base64 text compression, XML, EXI, and JSON). The availability of multiple file encodings facilitates quality assurance (QA) testing, the important of DIS data into other tools, and a wide range of analytic re-use.

Our simulation examples are now generously issuing COMMENT PDUs as part of simulation conduct, especially to report significant changes in state or simulation phases. Taken together, comments naturally build a narrative of simulation conduct that assists after-action review and subsequent analysis, highlighting interactions of special interest.

5. Live Virtual Constructive (LVC) Archiving

For a number of years this work focused on real-time streaming to and from Web browsers. While this remains a viable path for progress, evolving blockers related to network security make such a path difficult. We have instead begun to “dive deeper” in exploring mechanisms for LVC archiving and sharable re-use. While many DIS application libraries include the ability to perform logging, re-use appears to be limited and rarely performed in broader contexts. Several common difficulties likely contribute to the paucity of LVC archives getting replayed and “mashed up” together.

- Clock synchronization can be difficult in real time, and recorded timestamps do not match replay clock restarts.
- Network configuration of distributed simulations is difficult and typically requires special configuration of firewalls between domains.
- Privacy, exercise sensitivity, and scenario classification can discourage sharing.

Some attributes of DIS simulations lend themselves to shared state without prior preparation.

- Simulation Management rules require announcing new entities that enter and depart a simulation. Thus a unified roster of participants may be constructed as a simulation proceeds, rather than requiring advance coordination.
- Unique identifiers help classify entities according to network hosts.
- Precise identifiers regarding entity type help notify all participants of the physical nature of an entity.
- Common rules for shared distributed physics allow meaningful exchange of cause-and-effect activity.

The widespread use of Network Time Protocol (NTP) on modern computers has greatly improved historical difficulties with synchronization between DIS applications. Nevertheless, a primary blocker inhibiting broad re-use of DIS streams appears to be both numerical fidelity and shared rules associated with timestamp determination. The DIS version 7 timestamp is 32 bits wide, providing half the data length used by most operating systems and programming languages. Consequently, the DIS version 7 protocol dictates that the simulation clock “roll over” to zero at the top of each hour. This makes any stream recording including timestamp rollover difficult (if not impossible) to consistently sort.

Current work is exploring two approaches. First, we start the timestamp clock at zero upon receipt (or delivery) of the first PDU in a session. This provides sixty minutes of uninterrupted recording behavior which appears to be sufficient for a large number of scenarios. When starting, posting a COMMENT PDU is a good way to report actual local clock time corresponding to “zero time” for the simulation. Second, once a PDU list segment of interest is identified, establishing methods that routinely re-zero time to a new epoch, and possibly add time for synchronization with the LVC exercise in progress, is an approach that appears to offer a path for much broader LVC integration.

Following the principle “a stream is a stream” we are beginning to show support for repeatable unit testing with expected benefits for sustainable Live Virtual Constructive (LVC) interoperability, Validation Verification Accreditation (VV+A), and Testing Development Operations (TestDevOps).

An in-depth project using unmanned air vehicle (UAV) exercise telemetry explored decoding, parsing, conversion and DIS retransmission in detail.

- a. Obtained the data stream (wirelessly transmitted UAS data captured into PCAP Next Generation (PCAPNG) format) from a Joint Interagency Field Experimentation (JIFX) event.
- b. Transformation of MPEG-2 Transport container wireless data capture in PCAPNG format to human readable and other binary formats via Apache Data Format Description Language (DFDL).
- c. Extraction and transformation of Key Length Value (KLV) Local Sets (LS) embedded within the MPEG-2 Transport container into human readable and other binary formats via Apache DFDL library.
- d. Decoding of UAS KLV, then populating Distributed Interactive Simulation Protocol Data Units (PDUs) for re-streaming, thereby enabling mission playback in 3D simulation environments via iHarder KLV, West Ridge Systems jMisb, OpenDIS, and Xj3D.
- e. Mission playback, analysis, archiving and potential injection of decoded data streams into LVC environments.

Despite a lengthy process with several novel achievements and multiple opportunities for difficulty, datasets that previously were only parsable using proprietary software were nevertheless “unlocked” for DIS replay and downstream data analysis. Figure 9 provides a replay screenshot showing telemetry playback via the opendis7-java library into the Xj3D open-source player for X3D graphics [10]. The full video, available at the following url, shows that this unmodified reconstructed motion is smooth and continuous.

- KLV motion data extraction and replay using DIS streaming into an X3D scene
- <https://savage.nps.edu/videos/ScanEagle-KLV-3D-2021-10-13.mov>

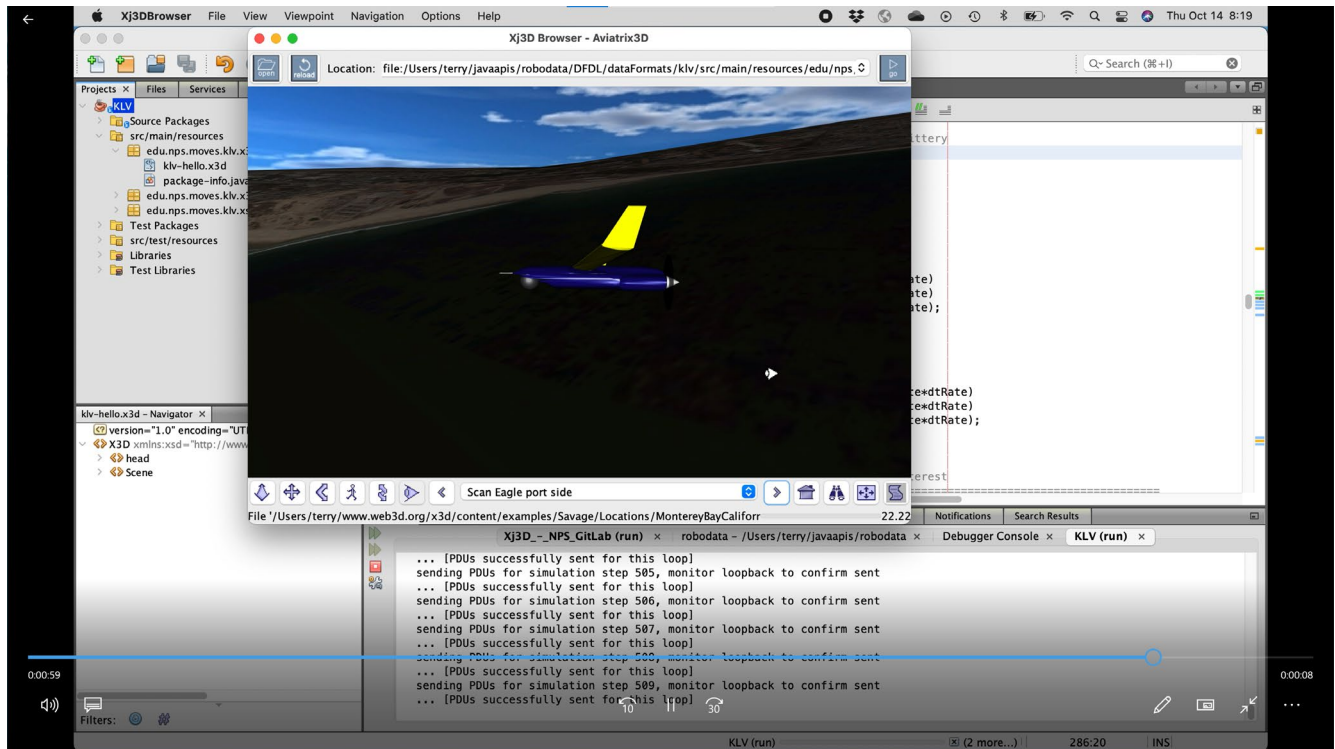


Figure 9. Recent NPS work unlocked binary unmanned air vehicle telemetry using Data Format Definition Language (DFDL) and then converted results into DIS for playback with an X3D Graphics browser. DIS is well suited for interoperability analysis.

6. Track

Research work by Blais on Rich Semantic Track [9] examined numerous track data models in use across C2 systems, M&S systems, and Robotics and Automation Systems (RAS) to synthesize the essential concepts that are communicated. Despite decades of effort, success is elusive and interoperability is lacking: warfighters are unable to “fight as they train” and typically synthesize tracks through human interpretation. The research developed a formal semantic representation of track data to create a basis for unification of track data semantics and pragmatics. The research also examined use of the ontology in several operationally relevant use cases to provide a foundation for community adoption and implementation.

That work focuses on achieving semantic and pragmatic interoperability across C2 systems, M&S systems, and RAS through a shared semantic model of track data (for semantic interoperability) and shared operations on track data (i.e., for pragmatic interoperability). Multiple track dialects from diverse systems were considered for exploring data interchange issues and for demonstrating transformation to the formal semantic model. All had a large “common denominator” of interoperability that closely corresponds to the IEEE DIS Entity State PDU and associated interaction PDUs. Rather than continue with synthesizing an artificial vocabulary, our work is currently focused on DIS as the basis for interoperability.

Current work is applying techniques demonstrated by Brennenstuhl [x] to utilize linear-regression techniques for curve-fit distillation of line segments characterizing track motion over time. Arbitrary precision is possible using various bounds for accuracy, as needed by a given simulation or analysis task. Composing, animating, and visualizing entity track via KML placemarks [9] or X3D interpolators [10] improves situational awareness and analytic insight. Further progress is expected to generalize to a wide range of test scenarios for unmanned systems, both actual and simulated. For further detail, see Data Strategy for Unmanned Systems Field Experimentation (FX), Simulation and Analysis [11].

7. C2SIM

Command and Control (C2) systems typically provide operators with situational awareness (SA) of the surrounding world via maps, data streams, imagery, etc. Surprising as it may seem, Modeling and Simulation (M&S) systems rarely communicate with C2 systems for military operations. Instead M&S systems are often relegated to training environments, notwithstanding widely held LVC motivations. The Command and Control Systems – Simulation Systems Interoperation (C2SIM) international standard for information interchange across C2 systems, simulation systems, and robotic and autonomous systems (RAS) holds great promise [8]. The rehearsal and evaluation of unmanned system tracks are fundamentally important tasks for each of these domains. Unclassified work to make data streams comprehensible and analyzable might have broad benefit. As illustrated in Figure 9, endeavors in all three arenas of activity are complementary and share overlapping mutual benefits.

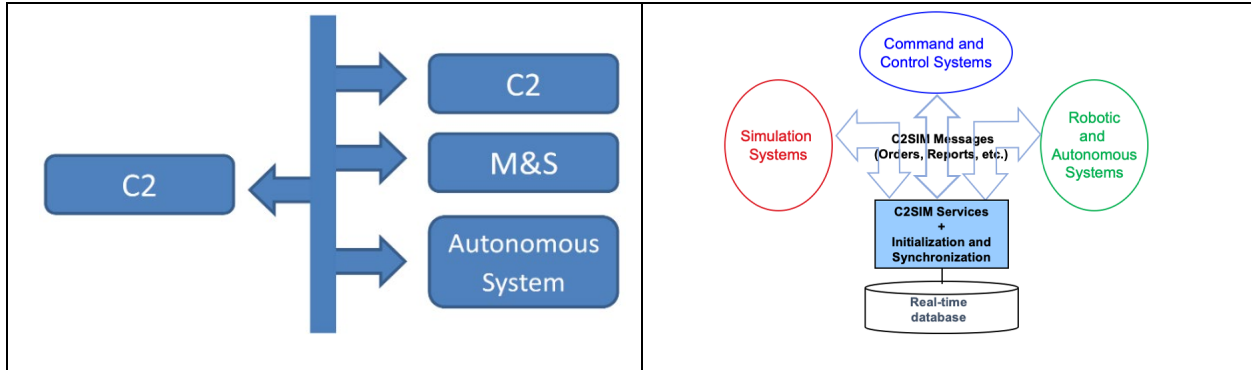


Figure 10. Command and Control (C2), Modeling and Simulation (M&S) and Autonomous Systems might functionally interoperate via shared information exchange using the C2SIM standard (SISO 2021).

8. Planned Future Work

We have begun to transition academic development, issue tracking and code improvements back into public forums on GitHub. Work in progress includes design refinement and autogeneration of an open-dis7-python library, as well as track distillation to create both X3D animation interpolators plus corresponding KML waypoints for track visualization. Planned work for 2022 includes future support for Compressed-DIS (C-DIS) and DISv8 protocols, plus continued adaptation of Rich Semantic Track (RST) principles as part of an Unmanned Systems Data Strategy. A unified approach to bridging DIS streams across C2SIM, TENA, and HLA RPR FOM environments has potential to further demonstrate broad LVC interoperability. Such repetition of possible, planned, rehearsed, actual, replayed, and analyzed streams can provide the missing links needed for effectively informing real-world operations. We expect that continued exploration of real, virtual and hybrid exercise streaming can establish a common path for Modeling and Simulation (M+S), Command and Control (C2), and unmanned-systems experimentation to be actionable as repeatable Big Data across all domains.

9. Conclusions and Recommendations

The opendis7-source-generator is a proven approach that can work across multiple programming languages. Working on open-source implementations has significantly advanced the sophistication and capabilities of autogenerated libraries. Further work is recommended.

The opendis7-codebase is now mature and appears ready for broad re-use. Since data-structure serialization and deserialization patterns are consistent for individual data types across the entire DIS vocabulary, we have only performed in-depth stress testing of perhaps 10 of 72 PDUs. A group approach to reporting issues and implementing refinements is expected to quickly advance the maturity of the remaining PDUs.

10. Acknowledgements

Much work by many students and collaborators have improved library capabilities over many years. We especially thank J. Michael Bailey for numerous library improvements, and again acknowledge Don McGregor's essential achievements. We look forward to further collaboration through renewed public efforts in the opendis community.

11. References

- [1] IEEE Standard for Distributed Interactive Simulation (DIS) - Application Protocols, IEEE 1278.1-2012, Piscataway NJ. <https://ieeexplore.ieee.org/document/6387564>
- [2] SISO-REF-010-2020: Reference for Enumerations for Simulation Interoperability, version 30 draft, January 2022. <https://www.sisostds.org/ProductsPublications/ReferenceDocuments.aspx> (linked at end)
- [3.0] opendis Project, <https://github.com/open-dis>
- [3.1] opendis7-source-generator Project, <https://github.com/open-dis/opendis7-source-generator>
- [3.2] opendis7-java Project, <https://github.com/open-dis/opendis7-java>
- [3.3] opendis7-python Project, <https://github.com/open-dis/opendis7-python>
- [4] MV3500 Networked Graphics Simulation, 2020, coursework at the Modeling Virtual Environments and Simulation (MOVES) Institute, Naval Postgraduate School (NPS), <https://gitlab.nps.edu/Savage/NetworkedGraphicsMV3500>
- [5] Brutzman, Don, Distributed Interactive Simulation (DIS) 101 Tutorial: The Basics, Interservice Industry Training Simulation Education Conference (IITSEC), 29 November – 3 December 2021, Orlando Florida. <https://gitlab.nps.edu/Savage/NetworkedGraphicsMV3500/-/tree/master/conferences/IITSEC2021>
- [6] Brutzman, Don, Tobias Brennenstuhl, and Terry Norbraten, Repeatable Unit Testing of Distributed Interactive Simulation (DIS) Protocol Behavior Streams Using Web Standards, presentation at SISO Simulation Interoperability Workshop (SIW) 2021-SIW-028, February 2021. <https://gitlab.nps.edu/Savage/NetworkedGraphicsMV3500/-/raw/master/conferences/SIW2021/BrutzmanBrennenstuhlOpenDIS7UnitTestingSiwFebruary2021Official.pdf>
- [7] Blais, Curtis L., *Rich Semantic Track (RST) Ontology: Unified Semantics and Pragmatics for Track Data Interchange*, Ph.D. dissertation, Naval Postgraduate School, Monterey, California USA, 2018. Dissertation available on request. Description and presentation online at <https://wiki.nps.edu/pages/viewpage.action?pageId=1082228797>
- [8] C2SIM Command and Control Systems-Simulation Interoperation, Simulation Interoperability Standards Organization (SISO), 2021. <https://www.sisostds.org/StandardsActivities/DevelopmentGroups/C2SIMPDGPGS-CommandandControlSystems.aspx> with open-source implementation at <https://openc2sim.github.io>
- [9] KML International Standard, Open Geospatial Consortium, 2008. Available via <https://www.ogc.org/standards/kml>
- [10] Brutzman, Don, and Daly, Leonard, *X3D: Extensible 3D Graphics for Web Authors*, Morgan Kaufmann Publishing, 2007. 468 pages, book website is <http://x3dGraphics.com>
- [11] Brutzman, Don and Curt Blais, *Data Strategy for Unmanned Systems Field Experimentation (FX), Simulation and Analysis*, Technical Memorandum, January 2022. <https://wiki.nps.edu/display/NOW/Data+Strategy+for+Unmanned+Systems>

Author Biographies

DON BRUTZMAN is a computer scientist and Associate Professor of Applied Science working at NPS in the Modeling Virtual Environments Simulation (MOVES) Institute, Undersea Warfare Academic Group, and Information Sciences Department. Dr. Brutzman leads the Network-Optional Warfare (NOW) project exploring fleet stealth using efficient messaging, optical signaling, semantic coherence and ethical control of unmanned systems. He is cochair for the Extensible 3D (X3D) Graphics Working Group for the non-profit Web3D Consortium. His research interests include underwater robotics, real-time 3D, artificial intelligence (AI) and networking.

RICK LENTZ is a Machine Learning Engineer at the Joint Artificial Intelligence Center (JAIC). He follows the frontier of model architecture, methodology, and benchmarks to guide DoD implementations for unstructured data solutions. His recent projects have focused on applications of intelligent agent systems, human-machine knowledge graphs, large language models for rationale-based information retrieval, and single-click multimodal targeting. His current research focuses on spatial understanding models that fuse modeling and simulation capabilities to current state environments

TERRY NORBRATEN's Naval career began in July of 1981 where he enlisted and served in various Naval Aviation billets as a Naval Aircrewman and Avionics Technician. He earned an A.S. in Digital Technology in 1993 from San Diego City College and was subsequently selected for the Enlisted Commissioning Program. In 1994 he graduated with a B.S. in Interdisciplinary Studies from Norfolk State University and was also commissioned an Ensign in the Regular Navy and served as a Surface Warfare Officer. In 2004, Terry earned a Masters of Science in Modeling, Virtual Environments and Simulation (MOVES) at the Naval Postgraduate School. Terry retired from active duty in 2005 and has served as a Research Associate with the MOVES Institute by instructing in the Java and JavaScript programming languages and working on various software projects involving stand alone and web-based scenario generation, discrete event simulation, training and analysis for fleet and field requirements.

CHRISTIAN FITZPATRICK received an M.S. in Modeling and Simulation from the Naval Postgraduate School in 2009. Within the Department, Mr. Fitzpatrick teaches Advanced Simulation Networking. Prior to joining the Faculty at NPS, Mr. Fitzpatrick served in the Marine Corps as a KC-130 pilot and Air Support Control Officer. After receiving his degree from NPS in 2009, he served at Marine Corps Combat Development Command where he developed scenarios for combat simulations to analyze the Expeditionary Fighting Vehicle (EFV) and Joint Light Tactical Vehicle (JLTV) using DoD-approved tools including COMBATXXI. He also spent 3 years at the Office of Naval Research (ONR) where he served as a Program Manager and established a tactical cyberspace/electronic warfare S&T development program. As a Faculty Research Associate in the MOVES Institute, his research thrusts include networking, LVC simulations, and system interoperability for TT&E.

CURTIS L. BLAIS is a member of the research faculty of the Naval Postgraduate School's Modeling, Virtual Environments, and Simulation (MOVES) Institute. He has over 47 years of experience in modeling and simulation development, management, education, and research. Dr. Blais is active in the Simulation Interoperability Standards Organization, serving in organizational committees and several working groups, including the Command and Control Systems – Simulation Systems Interoperation (C2SIM) combined Product Development Group/Product Support Group. He earned his doctorate at the Naval Postgraduate School in MOVES and holds Bachelor of Science and Master of Science degrees in Mathematics from the University of Notre Dame.