Scalable and Embeddable Data Logging for Live, Virtual and Constructive Simulation: HLA, Link 16, DIS and more

Björn Möller, Pitch Technologies, Sweden
Fredrik Antelius, Pitch Technologies, Sweden
Tom van den Berg, TNO, The Netherlands
Roger Jansen, TNO, The Netherlands

bjorn.moller@pitch.se
fredrik.antelius@pitch.se
tom.vandenberg@tno.nl
roger.jansen@tno.nl

Keywords:
Simulation, Training, After Action Review, Interoperability, Data logging, HLA, DIS, Link 16, Voice, Viking

ABSTRACT: *One of the most important simulation assets is the data that is collected during executions. Imagine being able to look back, analyze and reuse the data of simulations that have been run during the last decade. However, data logging has a number of challenges, not the least in today's environment where we need to train jointly and combined and mix a number of Live, Virtual and Constructive simulators, using different standards.*

This paper summarizes some requirements for LVC data logging as well as replay. It also describes some early experiences from developing and testing a data logger that can perform fully synchronized, simultaneous data logging of HLA, DIS, Link 16 and other data streams. Some details are given on aspects like embedding, chase play, ownership and import/export. Some challenges and limitations when mixing these different interoperability and data link standards are also covered.

1. Introduction

Data is one of the most important results or outputs of computer-based modeling and simulation. Even though computer-based modeling and simulation is a relatively young discipline, many models have been executed over the years, a lot of data has been produced and most of this output data is forever lost, in many cases since it was not logged.

No matter if a simulation is executed in real-time or using logical time, time-stamped simulation data can be logged for later use, like analysis or after-action review. This data will typically have a closer connection to the original set of simulators that produced the data than what a naïve user may initially think. It will usually be necessary to understand the goal, the assumptions and the limitations of the original simulators and scenarios to be able to play it back and use in a meaningful way. Nevertheless, the simulation data can be highly useful, both for the original purpose and for new purposes, such as input to other simulations, testing, training, and for new types of analysis.

1.1 An LVC perspective on data logging

Taking a Live-Virtual-Constructive [1] perspective on data logging adds a number of additional aspects to the above, for example:

- A challenging mix of simulation standards and protocols may need to be supported, usually together with a set of corresponding information exchange data models ("FOMs") that may be more or less coherent. More widely used types of data to be recorded include HLA [2], often with the RPR FOM [3], DIS [4] and voice (both as part of the HLA/DIS communication and using other ways of communication). Additional types of data may include Link 16 [5], TENA [6], streaming video, and proprietary protocols, for example for Command and Control systems.
- In a virtual or constructive model the data values in each model may define the ground truth. In a live simulation we can only attempt to capture measurements or perceived truth. One example of this is positions measured using a GPS where the inaccuracy may be measured in meters and will vary over time. The time stamps for data from different live

sources may also need to be adjusted when data from different sources is merged.

- While a constructive or virtual simulation can be re-run, you may only be given one, or a very limited number of opportunities to capture output data from a live simulation. One example of this is the firing of a prototype missile in a test range.
- Live simulations may require wireless data connections to some of the players. This may result in less reliable communication lines, leading to gaps in logged data. Additional precautions may need to be considered to address this problem.

In many cases there may be no major differences between the collection of data from a real life system for LVC simulation purposes and for other purposes.

Still, the ability to combine data from several Live-Virtual-Constructive sources makes the potential of this data even higher. It allows us to understand a bigger picture than before, to train in a more realistic and effective way, and to better analyze the total impact of new concepts.

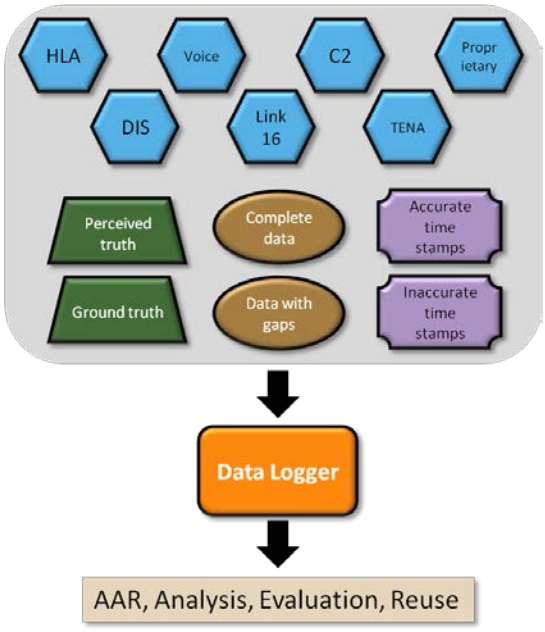

Figure 1: LVC Data Logging Challenges.

1.2 The role of the data logger in a simulation

A data logger is typically a software application that is either built-in into a simulation application or that is standalone. It may connect to one or more applications using a network protocol like DIS or interoperability services, like HLA. A data logger for HLA or DIS is usually more reusable than a built-in proprietary data logger but it may be limited to recording the public data provided in the FOM or the DIS protocol.

The most common functionality of a data logger includes:

- Recording of time stamped data from a data source like HLA or DIS into a file or a database.
- Playing back all or selected parts of the recorded data to a data sink of the same type (HLA or DIS). This may be done at the original speed, scaled to lower or higher speed, or using a completely different time-advance pattern, for example using HLA Time Management and/or event driven time advance.
- Support for human inspection of the data in a user interface.
- Support for automated inspection and analysis of the data through an API.
- Making the recorded data available in other formats like databases and plain text formats.
- Managing the timeline, for example by setting bookmarks or moving the playback time to a bookmark or a specific time value.
- Filtering the data during recording or playback.
- Adapting the data during playback, for example DIS exercise id, DIS entity id or HLA object instance name.

2. Use Cases for LVC Data Logging

There are many ways to benefit from data logging in LVC simulations. Some of the more common applications are described here. The use cases are based on the experiences from a number of simulation systems developed within TNO as well as practical experiences provided by staff at Pitch.

2.1 Simulation for training

Simulation for training is a common application where data logging is used. One particular class of training applications is the virtual, man-in-the-loop simulations, for example for training pilots, drivers, forward air controllers or straddle carrier operators.

Figure 2: TNO Forward Air Controller simulator.

Data logging in these simulations is mainly used to record and playback an exercise in real time, where instructors use VCR type functions to control data logging and playback.

The characteristics of these simulations are:

- The simulation is typically Virtual.
- It is a real time simulation where HLA Time Management is not used.
- Data logging is used for simulation data (ground truth), and sometimes also live voice or video data.
- The execution is controlled via start, stop, pause, and resume management messages.
- Simulation applications may join or leave the simulation execution when they want.
- Real-time replay of the logged data is used for after action review.

Since these simulations have been around for a while, the required functionality for recording and replay is generally well understood.

From a high-level point of view, three major states can be identified for this kind of simulation.

- Preparation. In this state a training scenario is prepared and previously recorded data may be used for the construction of a new scenario. Common simulator functions in this state are: create new scenario, edit scenario, delete scenario, load scenario and save scenario. Editing a scenario involves many different functions which will differ per training application, such as entity placement on a 2D map, route planning and entity behavior configuration. When the scenario is started, the prepared scenario becomes the initial situation at the start of the scenario execution.
- Execution. In this state the scenario is executed over time. Simulation data and other relevant data are recorded for after action review. It is possible to bookmark certain events for use in after action review. When the execution is stopped, the existing situation may become a new scenario in the preparation state.
- After Action Review. In this state a previously recorded exercise can be replayed and visualized in the original simulators or in 3D or 2D viewers. It is possible to view the list of available bookmarks, to jump to a bookmark or to a certain point in time in the recording. It is also possible to pause and resume the replay. When the after action review is stopped, the existing situation may become a new scenario in the preparation state.

2.2 Simulation for analysis

There are many different types of analysis models. Here we have chosen to focus on stochastic simulation (Monte Carlo [7] simulation). Stochastic simulation typically involves thousands or more simulation runs, varying one or more parameters. The simulation runs can be long lasting (in elapsed time), and are executed in non-real time. In most cases these simulations run as-fast-as-possible. Analysis involves processing and aggregating large amounts of data that has been recorded over the various runs. Ad-hoc queries on the recorded data may be needed to zoom in on certain aspects. Analysis is usually performed afterwards when all the data can be aggregated and searched.

Two examples where stochastic simulation is applied are described in earlier papers [8][9]. In [8] the effect of dynamic train management is studied, using small stochastic variations in the train schedule. In [9] a footprint analysis is performed to determine the region that a ship can defend against a missile, using stochastic variations in sensor behavior.

In both examples a large amount of data is collected during the simulation execution and transferred to a dedicated analysis application. Stochastic simulation also requires more extensive simulation states to control simulation execution, such as states for simulation initialization, warm-up, steady state execution, iterations and shutdown. This is quite different from the relatively simple simulation states in the training case.

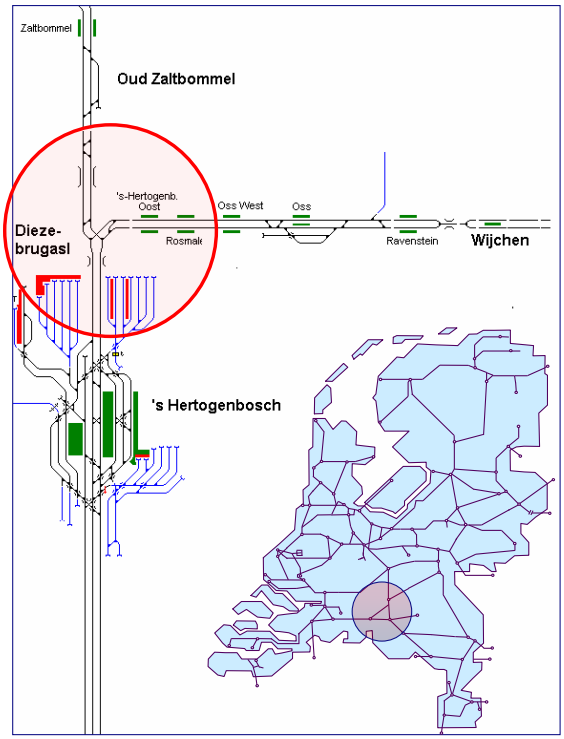

Figure 3: Study area to analyze dynamic train management using Monte Carlo simulation [8].

We can summarize the characteristics of a stochastic simulation as follows:

- The simulation is typically Constructive.
- It is a non-real time simulation where HLA Time Management is used.
- Data logging is used for simulation data (ground truth) as well as simulated operational data, like the Link 16 BOM (perceived truth).
- Execution is controlled via synchronization points and save/restore points
- All applications need to be present throughout the simulation execution.
- Replay of certain runs may be possible, but results may also just be charts such as bar or line charts of aggregated data

2.3 Simulation for test and evaluation of live systems

This use case involves connecting real-time (operational, live) systems to a simulation for test and evaluation. The idea behind this is to test and evaluate a system early in the development cycle and certainly before the system arrives in the target environment. A simulation can provide, for example, stimuli or 'ground truth input' in order to verify if the resulting behavior of the system is correct. Alternatively a data logger may be used to replay previously recorded data to stimulate a system. The resulting system behavior may be the transmittal of certain tactical (operational) messages, which may be fed back in the simulation for additional stimuli. Thus simulation for test and evaluation involves simulation data, operational data and real-time execution.

Analysis involves correlating simulation data with operational data to verify if the right data was generated at the right moment where timing of certain messages may be important. For example, which Link 16 track corresponds to which simulation entity? Are the correct tactical messages generated across all phases of a missile engagement?

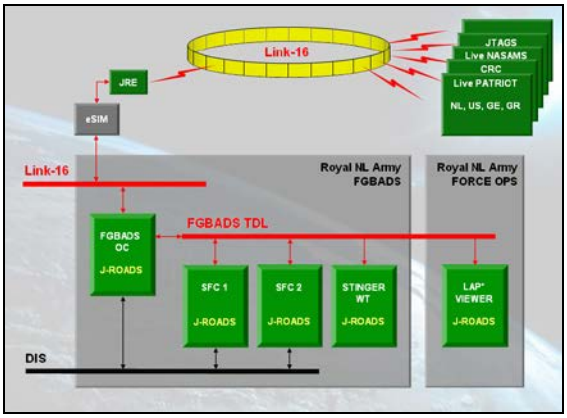

Figure 4: JROADS simulation integrated with live systems via a tactical data link.

Analysis may be performed on-line (during simulation execution) or off-line (after simulation execution). With on-line analysis, both simulation data and operational data are monitored during the simulation execution. It is possible to pause the monitoring in order to look at certain data, while at the same time the recording of data continues. The monitoring can be resumed and fast forwarded to catch up with the ongoing execution, so called chase play, just like modern hard-disk video recorders that can record and play a film at the same time, while jumping back and forth in the film. Bookmarking may be used to jump to certain important points that have been marked earlier in the recorded data.

With off-line analysis the recorded simulation data and operational data is reviewed after the execution has finished. Data may be replayed in real time or faster/slower than real time (n times real time). Important to note is that the timing of messages that are replayed can be important or even critical, due to the correlation between simulation data and operational data over time. Also, data from external sources may need to be combined with the recorded data, such as log files from command and control systems. Data from external sources can be provided in different formats (e.g. comma-

separated value file or xml file). An application for off-line analysis is described in [10].

Again, we can summarize the characteristics of a simulation for test and evaluation as follows:

- The simulation can be regarded as Live.
- It is a (hard) real-time simulation where HLA Time Management is not used.
- Data logging is used for simulation data (ground truth) as well as live/simulated operational data, like Link 16 (perceived truth).
- Execution is controlled via start/stop management messages, monitoring via pause/resume/jump messages
- Depending on the system, all applications in the simulation environment need to be present throughout the simulation execution
- Real time and non-real time replay of data is used for after action review.

2.4 Federation development

Logged simulation data is highly useful to minimize time, cost and risk during the development of simulation software, in particular when adding HLA or DIS interfaces. The output data of a simulator can be logged, inspected and checked against the expected output. Well-known, correct simulation data can be fed into a simulator from a data logger to check stability and correct behavior. You may even exchange logged data between several simulators before you connect them for real. An integration leader may apply a pre-integration methodology where all systems are required to be tested against a well defined set of test data before they are allowed to join the full federation. Data logging for simulator development is applicable to all the above types of simulation. It generally shares all of the above requirements but the requirement to be able to exchange data files is prominent.

3. Requirements and Challenges for Data Loggers

The different use cases all lead to a set of requirements for recording and replay. Ideally we're looking for a multipurpose recording and replay capability that can fulfill all requirements. This section of the paper lists the requirements and maps them to the use cases above that are most relevant.

3.1 Data streams

Requirement 1: The data logger must support several data streams (HLA, DIS, etc, as required by the simulation), or be extendable with new data streams.

Most applicable to: Training, Test and Evaluation

Today's simulation environments are open and all kinds of systems can be connected, generating different types of data. A well known example in the missile and air defense domain is Link 16. Another example is voice. Recording should not just be limited to simulation data.

3.2 Session management

Session management concerns the management of recording sessions: create a recording session with the required (DIS, HLA, etc) simulation connection parameters; destroy a previously created recording session; open a recording session for replay; close a previously opened recording session; start, stop, pause, resume the recording or replay within a session; jump to bookmark or jump to time within a session.

Requirement 2: The data logger must be able to record data streams and store them as a recording session. Recorded data streams (DIS, HLA, etc.) must be stored together in a recording session.

Most applicable to: All use cases

Requirement 3: The data logger must be able to retrieve a recording session and replay all or a subset of the recorded data streams.

Most applicable to: All use cases

Data streams in a recording session should also be replayed together. The precise timing of data stream messages may be important. For example if in data stream A messages are recorded at time 0, 5, 10, ..., and in data stream B at time 2, 5, 8, ..., then these should also be replayed exactly this way. Thus during replay, data streams in a recording session must remain synchronized in time.

Requirement 4: The data logger must be able to replay a data stream in a different format than was recorded.

This requirement implies that the data logger is aware of the data being recorded. For example record a DIS data stream and replay the DIS data stream as an XML formatted data stream.

Most applicable to: Test and Evaluation

Requirement 5: The data logger must be able to pause/resume/fast forward/fast backward a replayed recording session.

Most applicable to: All use cases

Requirement 6: The data logger must support the filtering of data from a data stream on recording and on replay.

Most applicable to: All use cases

Requirement 7: The data logger must support the concurrent recording and replay of a data stream.

Most applicable to: Test and Evaluation, Federation development

Usually replay happens only when recording has finished. But in some cases it must be possible to view and analyze data streams while they are being recorded. Thus data streams are replayed at the same time as they are recorded (concurrently). Also the requirements to pause/resume/fast forward/fast backward, to jump to a bookmark or jump to a point in time, and replay in a different format apply on the replayed data streams. When a replayed data stream lags behind on the recording, it is called ”chase play”.

The following figure shows the principle.

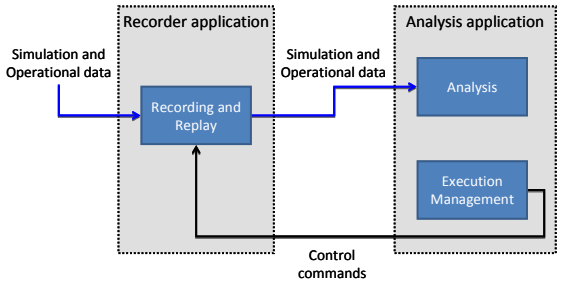

Figure 5: Activity diagram for concurrent recording and replay.

The data streams that come out of the Recording and Replay activity should not be replayed on the same DIS exercise or HLA federation where the data is recorded from. Thus the data streams should be replayed in a different DIS exercise or HLA federation, or even in a different format, for example as XML on a TCP connection.

Requirement 8: The data logger must support the grouping of recording sessions and support the addition of meta-data to each group.

Most applicable to: Analysis

With Monte Carlo simulations, each run results in a recording session. Recording sessions of related runs (for example where only the seed is different) should be grouped and have the variation number and other variable settings added as meta-data.

3.3 Bookmark management

Requirement 9: The data logger must support the management of bookmarks (create, delete, update bookmark; retrieve bookmarks).

Most applicable to: All use cases

Requirement 10: The data logger must be able to jump to a bookmark or jump to a point in time in a replayed recording session.

Most applicable to: All use cases

When jumping to a certain point in time (say time T), it may be necessary to scan the data stream backwards in time to build up a complete picture for time T. For example, with a DIS data stream the data logger may need to scan back up to 13 seconds in order to find all entity state updates for time T.

3.4 Time management

Requirement 11: The data logger must be able to record data in a real-time simulation (which does not use HLA Time Management or similar services).

Most applicable to: Training, Test and Evaluation, Federation development

Requirement 12: The data logger must be able to record data in a (real time or non-real time) HLA time managed simulation.

Most applicable to: Analysis, Federation Development

Time management concerns the use of HLA Time Management services when the simulation is time-managed. With a time-managed simulation a data logger is usually a time constrained federate in recording mode and, depending on the federation, a time regulating federate in replay mode. Recording and Replay needs to support HLA Time Management.

Requirement 13: The data logger must be able to replay a recording session at different speeds (real time or faster/slower than real time).

Most applicable to: Test and Evaluation

Restrictions may apply for certain data streams in certain situations. For example, a DIS data stream may in some cases only be replayed real-time, otherwise dead-reckoning models in applications like viewers may not work correctly.

3.5 Ownership management

Requirement 14: The data logger must be able to transfer ownership of object instances in a certain data stream on a mode change (between recording and replay).

Most applicable to: Training

Ownership management concerns a transfer of ownership of HLA object instances when the mode changes between recording and replay. In some situations an ownership transfer is required, for example when the state of the object instances provided in replay mode is used as the initial state for a new simulation execution. The data logger needs to release ownership and the different applications that model the object instances must acquire ownership.

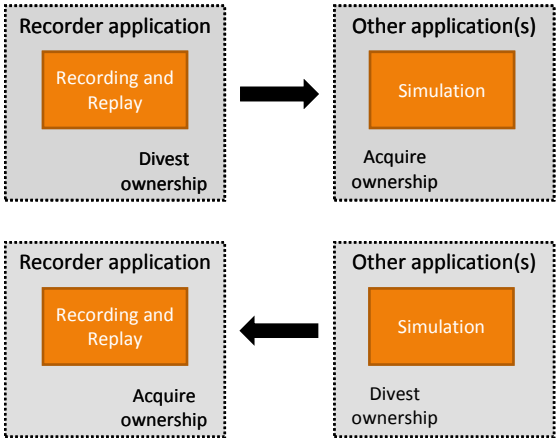

Figure 6: Ownership transfer on mode change: (1) from After Action Review to Execution or Preparation (top) and (2) from Execution or Preparation to After Action Review (bottom).

3.6 Data management

Requirement 15: The data logger must support the exchange (import/export) of recorded data with other applications.

Most applicable to: Analysis, Test and Evaluation, Federation development

For export, there are different options to consider, for example: raw export (export the data streams as they were recorded), structured export (export the data streams to a structured format, e.g. a SQL database where the schema matches the HLA FOM).

3.7 Control and embedding

Requirement 16: The data logger must be able to handle execution management messages that are received via a data stream.

Most applicable to: Analysis

In some cases execution management messages (such as HLA synchronization points, HLA Save/Restore, and user defined simulation management interactions) need to be interpreted by the Recording and Replay activity. This can for example be a certain HLA interaction that identifies the end of a Monte Carlo simulation run. The Recording and Replay activity must provide hooks to handle these execution management messages. A default hook could implement some default behavior, like achieving an HLA synchronization point.

Requirement 17: The data logger must be embeddable in and completely controllable by another application, concerning all of the earlier mentioned requirements.

Most applicable to: Training, Analysis, Test and Evaluation

This is an important requirement and allows recording and replay to be integrated with virtually any simulation application. The following figure shows an example of embedded control.

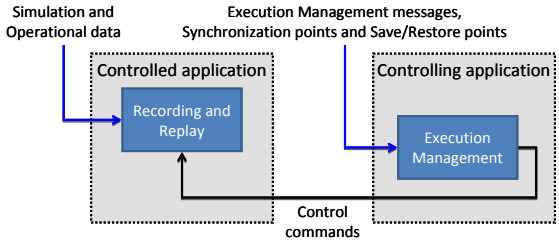

Figure 7: Activity diagram for recording mode.

In this example the controlling application performs the activity execution management. It controls the controlled application (i.e. the data logger) that performs the activity recording and replay. The controlling application handles for example the HLA synchronization points, HLA Save/Restore and Execution Management messages (such as start-resume and stop-freeze DIS PDUs in a DIS exercise) and if needed initiates mode changes on the controlled application. The controlled application (i.e. the data logger) does not interpret any Execution Management messages (these messages are just recorded as any other data) and achieves (by definition) any HLA synchronization or HLA save/restore it is involved in.

Thus, with embedding, recording and replay is dedicated to performing just this activity, while it is part of some application.

3.8 Scalability

Requirement 18: For initial testing, the data logger shall be able to operate on a regular computer without extensive setup. When used with a full federation, the data logger must be able to record/replay many different data streams concurrently and support long lasting and large recording sessions with tens of thousands of recorded events per second.

Most applicable to: All use cases

Note that there are advanced use cases where several data loggers could be used concurrently, for optimum scalability, or in different locations to conserve bandwidth. Merging of the logged data may introduce additional challenges that are not covered in this paper.

4. Practical Experiences

This section summarizes our experiences from extending a COTS data logger with an additional LVC protocol.

4.1 About Pitch Recorder

Pitch Recorder, a COTS product, is a general purpose data logger with a rich set of features [11] targeted at LVC simulations. It provides parallel, synchronized recording of the following data streams:

- HLA data for any FOM with support for HLA 1.3, 1516-2000 and 1516-2010 RTIs.
- DIS version 4, 5 and 6 plus experimental PDUs
- Audio (for example for voice recording).
- User defined data streams, for example national C2 protocols

In addition to the concept of a data stream, the Pitch Recorder introduces the concept of channels. For an HLA data stream it is possible to configure different channels, for example for land, sea and air entities as well as fire, detonation and radio. Pitch Recorder is not locked to any particular FOM and has been used for military, security, space, and civilian federations.

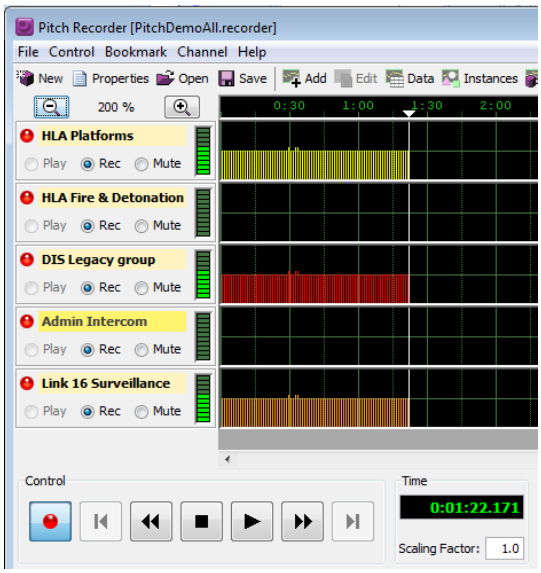

Figure 8: Channels in Pitch Recorder

All data streams can be recorded, played back, filtered, inspected and exported to other programs. Complete recordings can also be exported to a package that can be sent by e-mail or other file transfer methods. Pitch Recorder can be used stand-alone or be embedded into a solution and externally controlled by another software application.

One of the more recent features of this product is a plug-in framework that allows the addition of new kinds of data streams for recording and replay.

4.2 Scalability Experiences

Pitch Recorder can record to small local databases for modest data flows. For large federations, high end COTS databases on dedicated hosts can be used for sustained logging of tens of thousands updates per second. Typical performance for Pitch Recorder in a lab test is more than 25 000 recorded HLA updates per second on a regular desktop computer.

An interesting scalability experience from a real training application is the recent Viking 11 exercise [12]. This exercise was described in ITEC 2011 keynote as the world's premier comprehensive exercise, including civilian, military and police participants. The exercise covered the planning and execution of a UN mandated Chapter VII Peace Operation/Crisis Response Operation. On the civilian side approximately 35 Non-Governmental Organizations participated. It was based on a scenario called Bogaland that contains a large number of challenges for example piracy, irregular forces, refugees, children in armed conflicts and reconstruction. Approximately 2500 persons from 31 nations were involved, participating from 9 different sites.

Examples of participating systems were JCATS, ICC, Sitaware, Exonaut, TYR, ASCOT and VBS2. The information exchange was based on an HLA Evolved infrastructure using Pitch pRTI Evolved version 4.2.5. Data was logged using Pitch Recorder with a separate database host running MySQL, saving data to a RAID-5 disk set. More than 160 hours of exercise was recorded amounting to more than 210 GB of data. The majority of this data was position updates. Note that the data rate varies a lot over time, with a typical "idle rate" of 8000 updates per second. Voice data was also recorded using a separate Pitch Recorder since voice was handled on a separate network to reduce the risk of network overload.

One conclusion from this exercise is that it is important to fully understand how to configure the database manager (in this case MySQL) in order to guarantee that the data base sessions don't time out. Another, more obvious conclusion is the importance of powerful hardware to avoid overload during busy periods of the exercise.

4.3 Experiences from adding Link 16 support

As an engineering feasibility demonstration, a new data stream for Link 16 [5] recording and replay was added to Pitch Recorder by TNO. Tactical Data Link traffic like Link 16 is often emulated in simulation environments. Several protocols and wrappers are being used to provide the exchange of Link 16 messages between federates. The Standard Interface for Multiple Platform Link Evaluation (SIMPLE) [13] is widely supported and was selected for the engineering feasibility demonstration. The Link 16 data stream was added relatively easily to the Pitch Recorder, given that a Link 16 software library for receiving and sending Link 16 messages from/to a SIMPLE network was already available.

The plug-in framework provides a set of Java interface classes that a plug-in must implement, for example for sending and receiving data, and for providing a property window. Once the plug-in is constructed and compiled to a jar file, it is just a matter of dropping the jar file in the Pitch Recorder plug-in folder.

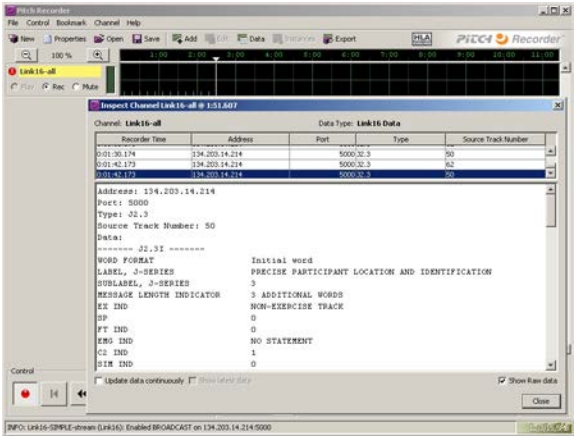

Figure 9: Screenshot of Pitch Recorder Link 16 recording.

One of the reasons to choose SIMPLE Link 16 as a first candidate plug-in is to create the ability to record, replay and analyze DIS/HLA simulation data in combination with Link 16 tactical data. This data stream combination is often found in LVC air and missile defense simulation exercises, like JPOW (Joint Project Optic Windmill) [14].

The Link 16 plug-in for Pitch Recorder was successfully tested in the JROADS (Joint Research On Air Defence Simulation) simulation environment at TNO. JROADS is an extensive simulation tool to support air defense research and CD&E for the Netherlands armed forces. At JPOW, JROADS has been used for joint experimentation, analysis, and mission training for many years.

5. Discussion

While generating requirements from the use cases, a number of challenges became obvious as to how these requirements should be implemented. This section summarizes some of them.

5.1 What data do we need to collect?

For many purposes, like after action review or analysis, there may be a requirement to use many types of data from the simulation. Some of them may be exchanged using HLA or DIS during the execution. Others may be internal variables in simulators or physical states of hardware. The challenge is how to collect the later type of data. Some approaches are to publish that data using HLA or to introduce a separate data stream for that data into the same or a different data logger. Using several data loggers creates problems when re-synchronizing the data. Sending additional data using HLA may only be practical for a limited set of data. Creating a specialized data stream for internal data from an application means a fair amount of work. The best approach has to be decided from case to case.

5.2 Data loggers and data awareness

One of the more difficult questions when designing a data logger is to what degree a data logger needs to be aware of the data it handles. Playing back data is usually more challenging than recording data and will sometimes require additional functionality in most participating simulators. Typical examples include:

- Handling of the life cycle of a simulated entity. If the playback of a DIS recording is paused and no data is sent for an aircraft for a certain time period, then listeners may delete that aircraft (unless all systems implement the freeze PDU). For HLA, a related problem is that a data logger may send out data for an aircraft that hasn't been created or that has attributes that are owned by another system.
- Handling of data where certain shared algorithms have been agreed. One example is dead reckoning where an aircraft has a certain speed that participating systems use for predicting its future position. When such data is played back at scaled time or even paused there is a risk that listeners may interpret the data in an unintended way.
- Handling of data that needs to be adapted. An example is the DIS exercise identification in a DIS data stream. The DIS exercise identification may be different on playback. Another example is time information. Time information may be adapted in order to replay

data at another simulation time than it was recorded.

As can be seen from these examples a data logger may need to have deeper insights into both the simulation standard used and particular federation agreements.

5.3 Exchanging data that has been logged

It is likely that different organizations may want to use different data logging software. The same organization may even want to use different software over time or for different projects. Therefore, it would be of great value if different data loggers could exchange data using a standardized file format. While the internal format of a data logger may be optimized for fast search and execution, a data interchange format would be optimized for generality.

One strongly related topic is a long-term data archival format that ideally would be the same as a standardized data interchange format.

6. Conclusions

This paper has presented a number of use cases, requirements and challenges for data logging in an LVC environment. Although the different use cases all have their own focus areas with respect to logging, it should be possible to provide a solution that fulfils all or most requirements. Such a solution must be open and extendable, for example by using a plug-in framework such as in Pitch Recorder. An initial demonstrator based on the Pitch Recorder plug-in framework has been described in this paper and has shown that a new data stream such as SIMPLE/Link 16 can relatively easily be added to the Pitch Recorder.

One important conclusion is the need to record several types of data in parallel to fully capture the exercise in particular in LVC and training applications. This may include both standardized data streams, like HLA, DIS and voice as well as proprietary data.

Future work on data logging and playback, in particular work related to debrief, should not only consider the requirements listed in paper, but also look at the work of the SISO Distributed Debrief Control Protocol (DDCP) Study Group [15]. The aim of the DDCP Study Group is to evaluate industry and government interest in developing a distributed debrief control protocol standard. Some of the requirements in this paper are related to this work.

References

[1] A Henninger, et. al. “Live Virtual Constructive Architecture Roadmap (LVCAR) Final Report”, US DoD, September 2008.

[2] IEEE: "IEEE 1516-2010, High Level Architecture (HLA)", www.ieee.org, August 2010.

[3] SISO, “Real-time Platform Reference Federation Object Model 2.0 ”, SISO-STD-001 SISO, draft 17.

[4] IEEE: "IEEE 1278, Distributed Interactive Simulation (DIS)", www.ieee.org.

[5] Link 16 is defined as one of the digital services of the JTIDS / MIDS in NATO's Standardization Agreement STANAG 5516. MIL-STD-6016 is the related United States Department of Defense Link 16 MIL-STD.

[6] “TENA - The Test and Training Enabling Architecture, Architecture Reference Document”, https://www.tena-sda.org/public_docmanager/userdocuments/TENA ARCHITECTURE REFERENCE/TENA Architecture Reference Document 2002.pdf.

[7] Metropolis, N. and Ulam, S. "The Monte Carlo Method." J. Amer. Stat. Assoc. 44, 335-341, 1949.

[8] 08E-SIW-003: Application of HLA in the Optimization of Rail Transport. Euro SIW 2008. T.W. van den Berg et al.

[9] 09S-SIW-008: Execution Management Solutions for Geographically Distributed Simulations. Spring SIW 2009. T.W. van den Berg et al.

[10] 11E-SIW-010: Generic Reconstruction and Analysis for simulations or live exercises. Euro SIW 2011. R. Witberg et al.

[11] Pitch Recorder web page, http://www.pitch.se/products/recorder

[12] Viking 11, http://www.forsvarsmakten.se/en/About-the-Armed-Forces/Exercises/Completed-exercises-and-events/VIKING-11/

[13] Standard Interface for Multiple Platform Link Evaluation (SIMPLE). STANAG 5602 (Edition 2). http://nsa.nato.int.

[14] Joint Project Optic Windmill, http://www.globalsecurity.org/military/ops/optic-windmill.htm.

[15] SISO Distributed Debrief Control Protocol (DDCP) Study Group, http://www.sisostds.org/StandardsActivities/StudyGroups/DDCPSGDistributedDebriefControlProtocol.aspx.

Author Biographies

BJÖRN MÖLLER is the vice president and co-founder of Pitch, the leading supplier of tools for HLA 1516 and HLA 1.3. He leads the strategic development of Pitch HLA products. He serves on several HLA standards and working groups and has a wide international contact network in simulation interoperability. He has twenty years of experience in high-tech R&D companies, with an international profile in areas such as modeling and simulation, artificial intelligence and Web-based collaboration. Björn Möller holds an M.Sc. in Computer Science and Technology after studies at Linköping University, Sweden, and Imperial College, London. He is currently serving as the vice chairman of the SISO HLA Evolved Product Support Group.

FREDRIK ANTELIUS is a Lead Developer at Pitch and is a major contributor to several commercial HLA products. He holds an M.Sc. in Computer Science and Technology from Linköping University, Sweden.

TOM VAN DEN BERG is scientist in the Modeling, Simulation and Gaming department at TNO, The Netherlands. He holds an M.Sc. degree in Mathematics and Computing Science from Delft Technical University. His research area includes simulation systems engineering, distributed simulation architectures and concept development & experimentation.

ROGER JANSEN is a member of the scientific staff in the Modeling, Simulation and Gaming department at TNO, The Netherlands. He holds an M.Sc. degree in Computing Science and a Master of Technological Design (MTD) degree in Software Technology, both from Eindhoven University of Technology, The Netherlands. He works in the field of distributed simulation and his research interests include distributed computing and simulation interoperability.