Execution Management Solutions for Geographically Distributed Simulations

T.W. van den Berg
H.G.M. Janssen
R.E.J. Jansen
L.M. Prins
TNO Defence, Security and Safety
PO Box 96864,
2509JG The Hague, The Netherlands
tom.vandenberg@tno.nl, henk.janssen@tno.nl, roger.jansen@tno.nl, louwrens.prins@tno.nl

Keywords:
Execution Management, Federation Management, Scenario Management, HLA.

ABSTRACT: *Managing the initialization, execution control and monitoring of HLA federates is not always straightforward, especially for a geographically distributed time managed federation. Issues include pre and post run-time data distribution and run-time data collection; starting, stopping and monitoring each federate application from a single point of control; the sheer amount of data collected during a session; software and network stability when performing long lasting, uninterrupted sessions, such as for Monte Carlo simulation; network latency and time synchronization; execution control functions - in particular legacy applications may not support all required functions.*
One recent project is used as example to study in more detail how these Execution Management issues are solved. The project SIGDM&S establishes a geographically distributed HLA federation for the analysis of maritime missile defense scenarios using Monte Carlo simulation. This paper provides an overview of the Execution Management issues, the solutions provided, lessons learned and recommendations for improvements in the area of execution management.

1. Introduction

Managing the initialization, execution control and monitoring of HLA federates is not always straightforward, especially for a geographically distributed time managed federation. Issues include pre and post run-time data distribution and run-time data collection; starting, stopping and monitoring each federate application from a single point of control; the sheer amount of data collected during a session; software and network stability when performing long lasting, uninterrupted sessions, such as for Monte Carlo simulation; network latency and time synchronization; execution control functions - in particular legacy applications may not support all required functions and require adaptations.

The availability of robust and reliable tools that support the user in managing a geographically distributed HLA federation is critical to the success of experimentation and analysis. These tools must support a high degree of automation in order to prevent the execution of otherwise manual tasks as repeatedly starting and stopping federate applications during a Monte Carlo simulation, collecting log files and monitoring progress.

In 2007 an international project was initiated between the United States, Canada, Australia, Germany and The Netherlands with the objective to establish a Simulation Architecture for the evaluation of future maritime Battle Management Command, Control, Communication & Computing Information (BMC4I) architectures. This project was named “Secure International Geographically Distributed Modeling and Simulation” (SIGDM&S). Due to the fact that most of the national simulation models within the participating countries are restricted for release, a geographically distributed HLA based simulation has been developed for the performance analysis of maritime missile defense Command & Control Architectures and systems. Through stochastic (Monte Carlo) simulation the behavior of a complex system can be explored by using random samples of parameters or inputs.

This paper provides an overview of the project and execution management solution used. The paper is structured as follows:

- Chapter 2 provides an overview of the project, including the background and HLA federation architecture.

- Next, chapter 3 discusses the concept of execution management; what do we mean with federation execution management?
- Chapter 4 discusses the federation execution management solution that was applied and the lessons learned.
- Finally, chapter 5 summarizes the conclusions and provides some recommendations for improvements in the area of execution management tools.

2. SIGDM&S

2.1 Overview

In 1999 a Maritime Theater Missile Defense Forum was established as an informal gathering of United States, German and Netherlands Naval Flag Officers to identify areas of common interest in Ballistic Missile Defense and associated programs. This forum has now evolved to eight nation participation (the United States, Canada, Australia, Germany, The Netherlands, United Kingdom, Spain and Italy), with the key focus on coalition interoperability between the maritime platforms of the participating nations. The Forum's imperative provides protection against the proliferation of short, medium and long-range Ballistic Missiles, Advanced Anti-Ship Cruise Missiles threats through the creation of an interoperable sea-based defense capability among coalition nations. The Forum's Coalition Maritime forces will provide protection across the full spectrum of these threats, utilizing existing interoperable sea-based systems to protect against current threats while progressively improving and developing systems and system-of-systems to remain effective against evolving threats. Especially the use of distributed network centric architectures can multiply the capabilities of the coalition forces. Within these architectures, the Maritime forces exchange information to achieve situational awareness, threat evaluation, weapon assignment and target engagement at the force level.

In developing these distributed network centric architectures, Modeling and Simulation is of vital importance for testing, evaluating and performing assessment of proposed future architectures in an early stage of development. Therefore, the Forum established the Modeling and Simulation Working Group. A key objective of this Working Group is to cooperatively develop, demonstrate and maintain a comprehensive distributed modeling and simulation framework for the benefit of supporting concept development, design, analysis, evaluation and testing of the envisioned network centric architectures.

As a first initiative a proof of concept was defined that establishes an internationally distributed simulation to be extended by additional capabilities in later increments. The simulation had to be geographically distributed in order to permit the execution of national (proprietary) models in a multi-national geographically distributed simulation. Figure 1 shows the geographical location of the selected HLA federates in the simulation.

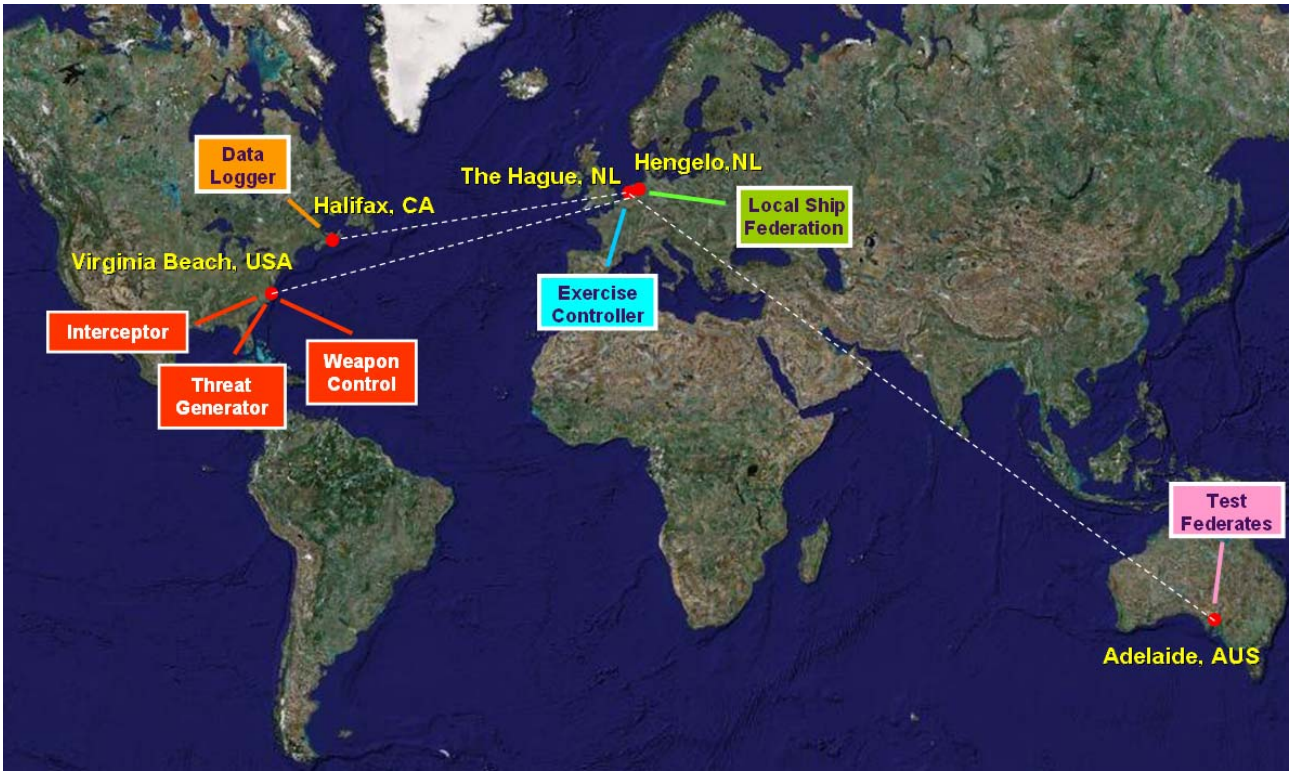

Figure 1: Locations of the selected HLA federates in the simulation.

2.2 Defended Footprint Analysis

The initial capability and proof of concept established a geographically distributed HLA federation with the purpose to perform a defended footprint analysis (DFA) for a single ship-interceptor-TBM scenario. DFA determines the regions that a ship can defend against TBM attacks launched from a specified launch area. The analysis involves a stochastic (Monte Carlo) simulation where the ship position is varied over a pre determined set of grid points in the defended area (see Figure 2). For each grid point multiple Monte Carlo runs are performed, using random perturbations in some of the parameters of the simulated systems. The outcome of the simulation shows amongst others the performance of the sensor system and provides the ship positions that provide best defense in the given scenario.

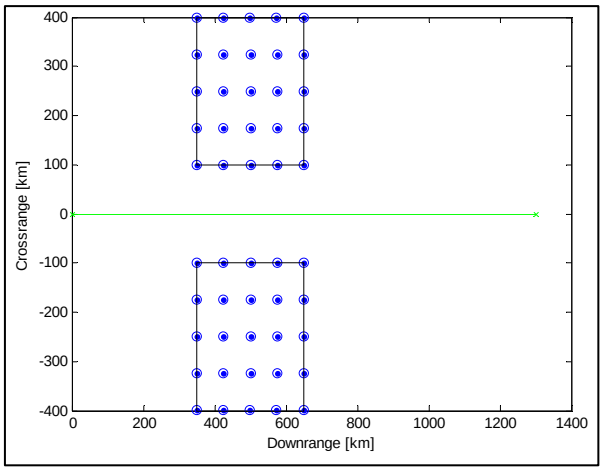

Figure 2: Defended footprint. In green the TBM trajectory, in blue the grid points (mirrored across the trajectory).

For DFA, repeatability is a highly desirable characteristic. This allows unexpected or unusual analysis results to be investigated in greater detail to explain the underlying phenomenon. Repeatability requires a conservative time management strategy in which federates are both regulating (can control federation time advance) and constrained (wait for federation time advance) and interactions are time stamped. Conservative time management generally results in slower execution times.

2.3 Federation Architecture

The federation architecture is shown in Figure 3 and the geographical distribution of the selected federates is shown in Figure 1. The architecture consists of two connected HLA federations: a Maritime Theater Missile Defence (MTMD) Federation and a Local Ship Federation. Below follows a brief description of the HLA federates in each federation and the HLA Run Time Infrastructure (RTI).

Local Ship Federation (Hengelo, NLD)

The local ship federation consists of federates, representing the behavior of several ship subsystems (a long range radar surveillance system, radar tracking system, combat management system and ship platform representation) which communicate with each other through the HLA Run Time Infrastructure. The weapon control system is logically part of the ship, but this system is modeled in a separate federate in the MTMD federation (see weapon federate).

Ship Federation Manager Federate (Hengelo, NLD)

This federate is used to bridge the geographically distributed MTMD federation and the local ship federation. This bridge allows the use of distinct sub federations (with different FOMs), which minimizes changes to existing federates and hence facilitates model reuse and which enables data shielding.

Weapon Federate (Virginia Beach, USA)

The weapon federate consists of a number of components. For various historical reasons these components have been assembled in a single federate application, but can be viewed as individual federates. The components are:

- The threat generator is a component that simulates a Tactical Ballistic Missile threat. The position of the threat is sensed by the ship's search radar, which will publish the position of the threat as a track for use by the weapon control system.
- The weapon control system is logically part of the ship, simulated by the local ship federation. It receives track information from the ship's onboard sensors and controls the interceptor with amongst others launch commands and guidance information.
- The interceptor simulates a 'generic' interceptor possessing the following characteristics: a booster, mid-flight guidance, and terminal IR homing phase.

Exercise Controller Federate (The Hague, NLD)

The Exercise Controller is a federate that controls the state transitions in the federation and coordinates the various Monte Carlo runs for each grid point. It has a graphical user interface to load a scenario file, adapt some scenario parameters, start and stop the simulation, and to monitor the current federation state.

Logger Federate (Halifax, CAN)

A data logger federate was developed that saves data in both an XML format and a format usable by the 3D viewer, SIMDIS. Both files are packaged by the logger

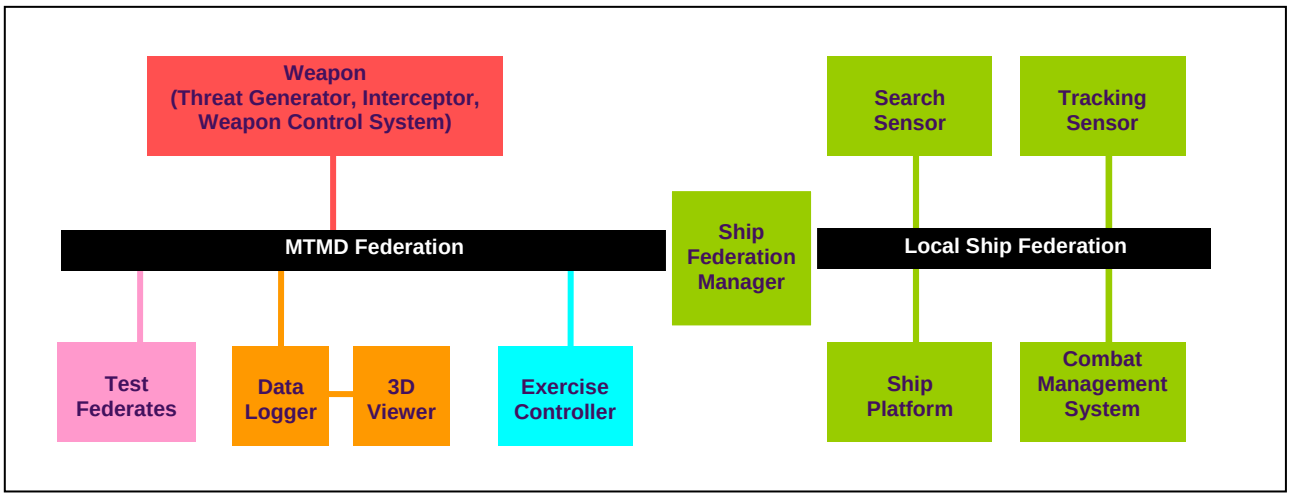

Figure 3: Federation Architecture.

federate into a single compressed binary file. The 3D Viewer (not an HLA federate by itself) can connect on line to the data logger to visualize the data that is currently processed by the data logger.

3D Viewer (Halifax, CAN)
The 3D viewer is based on SIMDIS from the US Naval Research Labs. The 3D viewer can act as both an off line and an on line viewer. As off line viewer it visualizes the data as previously recorded by the data logger. As on line viewer it connects (from any geographical location) to the data logger and visualizes the data as it is currently processed by the data logger. Platforms are represented by a 3D visual representation of the real-life hardware. Missiles and threats are likewise represented in 3D. SIMDIS provides situational awareness of the unfolding simulation within a visual environment.

Test Federates
A suite of "Mock" test federates was developed to provide low fidelity implementations of the behaviors defined for the real federates. The test federates act and react the same as the real federates with respect to the FOM objects and interactions and HLA services. These test federates were used as substitutes for the real federates while each site was independently developing their own deliverable federates. This was advantageous since the development of each federate was therefore much less dependent on the progress of the development of the other federates. Also, while some real models were not allowed to be released to other participating countries, the test federates made local testing possible at each site.

Run Time Infrastructure
The Run Time Infrastructure (RTI) is provided by the TNO-RTI. This RTI is a mixed mode RTI and supports both IEEE 1516 and HLA 1.3 compliant federates in the same federation without additional tools. Note that only the Weapon federate is still HLA 1.3. The TNO-RTI is a partial RTI implementation but provides all of the required RTI services for both the MTMD federation as well as the local ship federation.

3. What is Federation Execution Management?

There is no single agreed definition of "Federation Execution Management", but generally it covers both the topics "federation management" and "scenario management".

Federation management is about the creation, control, modification and deletion of the federation execution as defined in the HLA Federate Interface Specification [1], as well as the tasks to configure, start, monitor and stop federate applications, to distribute pre run time data and to collect post run time data. Therefore, federation management includes more than just HLA federation management.

For scenario management we use the statement from the paper on Scenario Management - Common Design Principles and Data Interchange Formats [2]. Scenario management covers "All tasks associated with development and execution of a scenario including scenario development, initialization, modification and execution control." The execution of a scenario concerns for example the tasks to start, pause, resume and stop a scenario, to request the creation and deletion of object instances and to provide initial values for object instances.

A summary of the federation execution management tasks is as follows:

- Start and stop of applications

- Monitoring of applications
- Configuration of applications
- Data distribution and collection
- Network monitoring
- Performance monitoring
- Creation and deletion of federation execution
- State control using synchronization points and save and restore points
- Scenario initialization (i.e. create, modify and delete object instances, provide initial values)
- Scenario control (i.e. start, pause, resume, stop, distribute control information such as scenario name, run number and seed values)

When looking at federation execution management then the tasks related to the development of a scenario are not relevant. These tasks are usually performed before the federation is executed. Therefore we only consider the tasks scenario initialization and scenario control. Also tasks related to federation deployment and (re)booting of systems are not considered to be part of federation execution management in this paper.

4. Federation Execution Management Solution

4.1 Introduction

From federation execution management point of view the objective is to support a federation that amongst others:

- Executes over a WAN, with geographically distributed federates.
- Executes at least 48 hours (uninterrupted) with hundreds of Monte Carlo runs.
- Executes unattended.
- Produces a log file for each Monte Carlo run.
- Requires scenario initialization data for each Monte Carlo run.

The execution management tasks have been allocated to the following components:

Task:	**Allocated to:**
1. Start and stop of applications	Pitch Commander
2. Monitoring of applications	Pitch Commander
3. Configuration of applications	Pitch Commander
4. Data distribution and collection	Pitch Commander, FTP client and server
5. Network monitoring	Various network tools
6. Performance monitoring	TNO-RTI, MS Excel
7. Creation and deletion of federation execution	Exercise Controller
8. State control	Exercise Controller
9. Scenario initialization	Exercise Controller
10. Scenario control	Exercise Controller

Where the components are:

- Exercise Controller: an HLA federate to control the federation state as it progresses through the various Monte Carlo runs and to provide scenario data to other federates.
- Pitch Commander: a commercial off the shelf application to distribute and collect data and to start, monitor and stop federate applications. See [3].
- Various network tools: both commercial and open source tools to monitor network usage.
- FTP client and server: off the shelf FTP server and client for file transfer.
- TNO-RTI: an RTI implementation from TNO (see chapter 2.3, Federation Architecture) including functionality to collect performance data for off-line monitoring.
- MS Excel: Microsoft Excel for producing trend charts from data collected by the TNO-RTI.

These components are described in more detail in the next chapter as part of the solution description.

4.2 Solution

This chapter discusses the solution applied for each federation execution management task.

1. Start and stop of applications

The start and stop of federate applications is managed with Pitch Commander [3]. Pitch Commander consists of a Commander application and a number of so called Agent applications. An Agent is installed on each host that runs a federate and controls on behalf of the Commander the startup, monitoring and shutdown of the local federate. The Agent also provides the status of the federate and its host computer to the central Commander.

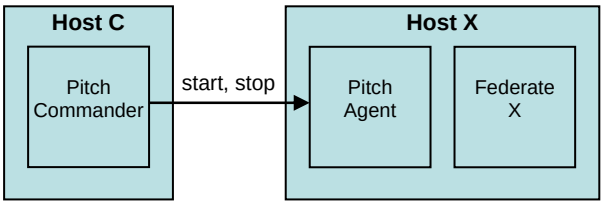

Figure 4: Start and stop with Pitch Commander.

Each federate supplier provided start and stop (Linux) shell scripts or (Windows) bat files for their federate. These scripts or bat files are invoked by the Agent on behalf of the central Commander. The start and stop of all federate applications in the federation was also

automated using a single (Python) script that was executed from the Commander scripting environment. Figure 4 illustrates the functioning.

2. Monitoring of applications

Pitch Commander provides a number of basic monitors to monitor for example CPU load, memory usage, disk usage and network statistics per host computer. It is also possible to monitor whether an application is running or has been stopped.

3. Configuration of applications

The start and stop of a federate application on a host is managed by the local Pitch Agent. Each federate supplier has defined in the Agent the available actions to start or stop their application. Each action is associated with a script or bat file to start or stop the application, including any configuration settings for the application.

The Agent actions are invoked by the central Commander. Thus by invoking different Agent actions from the Commander the appropriate federate configuration can be started.

4. Data distribution and collection

Pitch Commander provides File Transfer functionality to transfer files between a host that runs the Pitch Agent and the host that runs the Pitch Commander. A transfer from one or more hosts to the central Commander is called “collect”. A transfer from the central Commander to one or more hosts is called “deploy”.

This File Transfer functionality was initially used to automatically collect the log files from the Data Logger federate after the completion of all Monte Carlo runs and transfer these files to the central Commander. With this solution the complete process of starting the federation execution, performing the federation execution, stopping the federation execution and collecting the log files was automated.

However, the amount of data generated by the simulation was large and data collection over a wide area network took much time while waiting for the next federation execution. Therefore, the solution of automatic data collection was abandoned. Log files were manually placed on an FTP site at a convenient point in time. An FTP client was used to collect these files.

5. Network monitoring

Network monitoring was provided by What’s Up Gold [4]. This (commercial) application provided continuous monitoring of network connectivity and visual maps for quick recognition of potential network problems. Figure 5 provides an example of an imaginary network.

The open source software iperf [5] was utilized to determine and measure actual bandwidth under load conditions. These measurements were conducted prior to and between federation executions to minimize interference and provide representative values of available bandwidth.

MTUroute [6] was a freeware application utilized to verify that the correct Maximum Transmission Unit (MTU) values were properly configured in encryption devices and protocol stacks on host computers. MTU is the size (in bytes) of the largest packet or frame that a layer of a communications protocol can pass onwards.

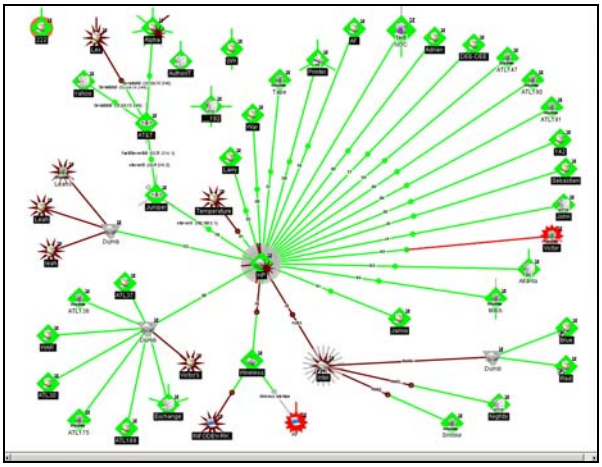

Figure 5: A Whatsupgold network map (see [4]).

6. Performance monitoring

For this project a rather detailed analysis of the federation performance was desired. We wanted to know the simulation speed over time and what amount of time each federate was spending on its main task or on waiting due to HLA time management constraints. Also it was interesting which part of this waiting time was caused by the communication network. Using the Management Object Model (MOM) [7] this kind of detailed information cannot be gathered. Moreover, if it could be gathered using the MOM it would result in additional network traffic as more messages need to be passed around. Additional on-line (during execution) and off-line (after execution) monitors are needed for the data that is maintained inside the RTI itself.

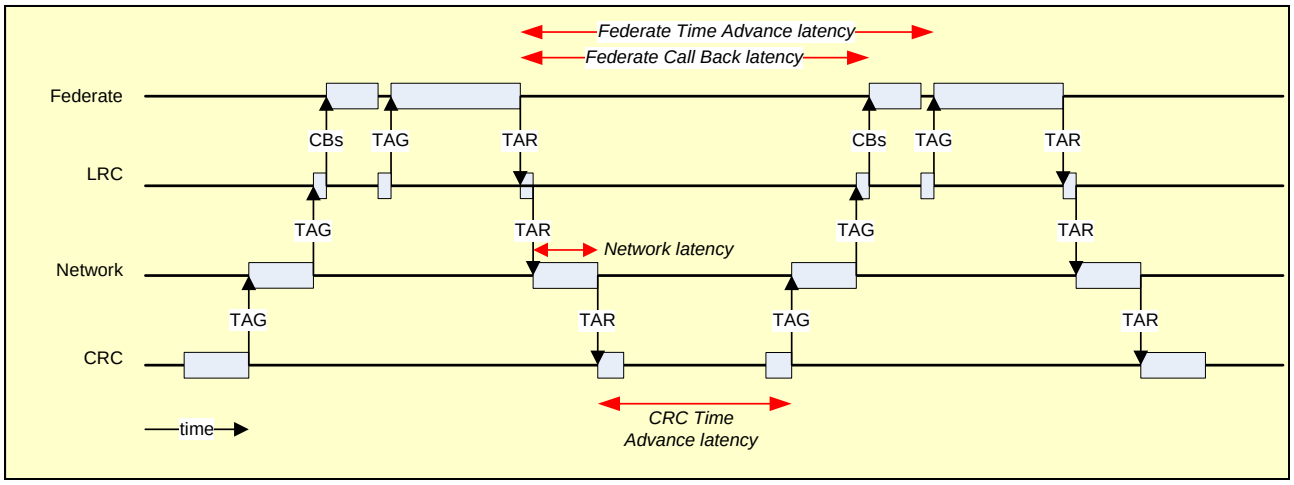

Figure 6: General time advancement pattern for the federate.

To enable this kind of "inside" RTI monitoring a federation performance model was constructed with the key parameters to be measured by the RTI. The TNO-RTI was adapted to collect this data for off-line monitoring.

The performance model is explained in Figure 6. This figure shows the general time advancement pattern for the simulation execution. The figure shows a timeline for the major components in the federation, the processing by each component (shown as a grey block) and the messages that are exchanged between the components (shown by a vertical arrow).

The components are:

- **Federate**: this is either the Weapon federate, the Ship Federation Manager federate, or the Logger federate.
- **CRC**: this is the Central RTI Component that manages, amongst other centralized HLA services, the simulation time in a federation. This component is deployed at the host that also runs the Exercise Controller.
- **LRC**: this is the Local RTI Component that is directly linked with the federate software on each host that runs a federate. Federates use the LRC to communicate amongst each other and to communicate with the CRC.
- **Network**: this represents the various hardware and software parts that make up the physical network (this includes the network protocol stack on each host, routers, cryptos, and RTI parts for handling messages to and from the LRC).

The CRC calculates if and when a Time Advance Grant (TAG) is allowed for each federate. If the LRC receives a TAG from the CRC, then the LRC first delivers all messages as call backs (CBs) to the federate according to the HLA Time Management rules, prior to delivering the TAG to the federate. On its turn the federate issues a Time Advance Request (TAR) to request the advancement of simulation time.

Figure 6 does not show the case where the LRC can locally grant a TAR from the federate. For example, when a federate requests time advances that are smaller than all the other federates, then some of these requests can be granted locally without permission from the CRC. This optimization decreases latency times for all federates. For a local TAG the time advance latency is assumed zero. Local grants are taken into account in the RTI measurements.

Using this performance model we were able to monitor per individual federate the fraction of time spent on simulation related processing (busy ratio) and network related processing (network ratio).

To determine the busy ratio of a federate for the whole federation execution the following formula is applied:

```
Busy ratio =
    1 – (Total Federate Call Back latency
    / Effective Federate Execution Time)
```

Where the **Effective federate execution time** is the total wall clock time that the federate is in the simulate state.

To determine the network ratio of a federate for the whole federation execution the following formula is applied:

```
Network ratio =
    Total Network latency /
    Effective Federate Execution Time
```

Where the **Total Network latency** is defined as:

```
Total Network latency =
    Total Federate Call Back latency –
    Total CRC Time Advance latency
```

LRC processing time is neglected in the calculation of the network ratio as it is relatively small with respect to federate processing time.

Using the same data we were also able to monitor both the simulation speed and the simulation time over time for each federate. Figure 7 shows for example the simulation speed over time for a federate with in red the moving average with a period of 10. Figure 8 shows the progress of simulation time over time.

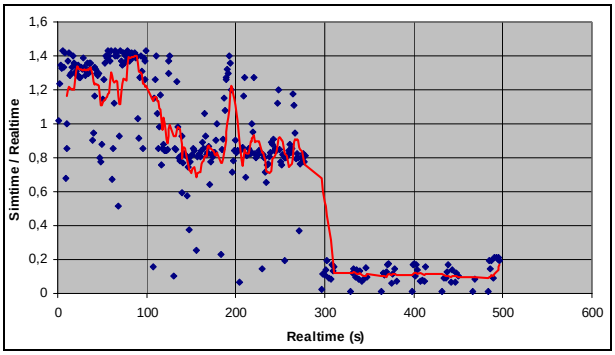

Figure 7: The simulation speed over time for a federate.

Simulation speed is defined as the ratio of simulation time and real time. In stochastic simulation the simulation speed is not constant but varies over the course of the execution (as opposed to real time simulation, where the speed is around '1').

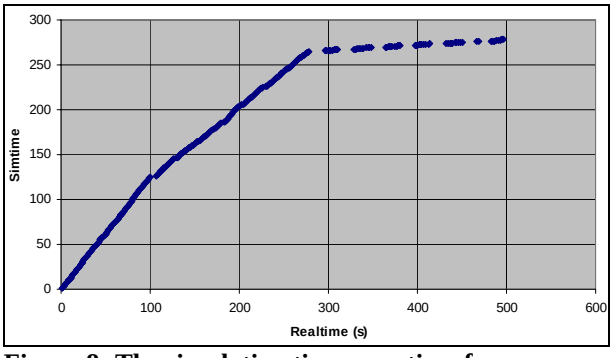

Figure 8: The simulation time over time for a federate.

7. Creation and deletion of federation execution

The federation execution is created by the first federate to start, in this case the Exercise Controller. With the federation configuration used in this project the Central RTI Component (CRC) was run on the same host that runs the Exercise Controller (location The Hague, The Netherlands).

8. State control

A dedicated Exercise Controller was developed to control the state transitions in the federation. One of the reasons for a dedicated application was that the scripting environment of Pitch Commander did not support the handling of RTI callbacks, such as the receipt of HLA interactions.

The Exercise Controller initiated the transitions between states using HLA synchronization points and HLA save and restore. The state transition diagram is shown in Figure 9. The transition triggers and guards have been left out of the diagram for simplification. The state diagram shows two (nested) loops: one loop to go through all grid points and one loop to go through all runs for a given grid point. The state diagram is for illustration purpose only and is not further described.

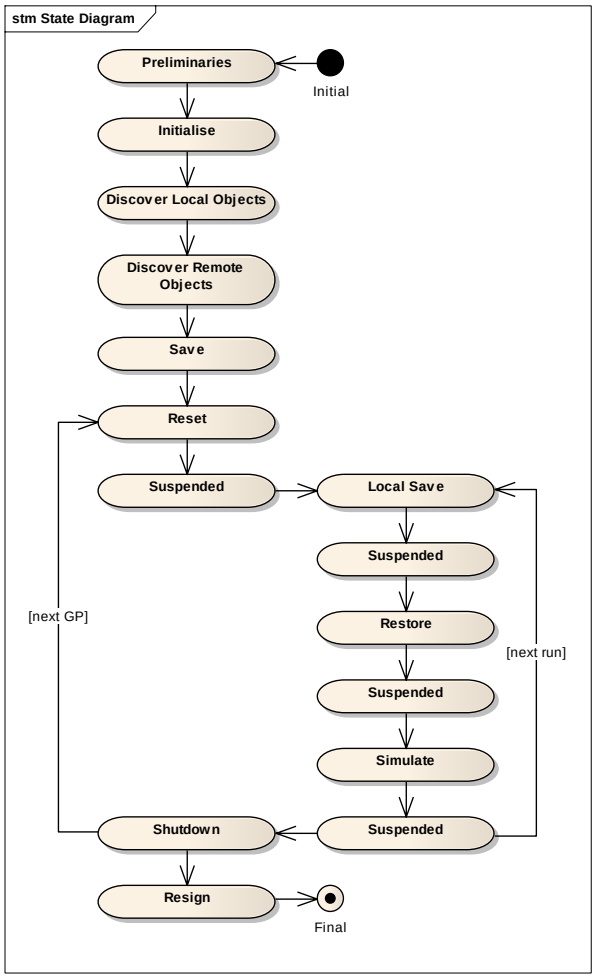

Figure 9: Exercise Controller state diagram.

The Exercise Controller has a GUI to start and stop the simulation and to display the amount of wall clock time spent on each Monte Carlo run and each grid point.

The remaining federates (Weapon, Ship Federation Manager and Data Logger) are execution managed federates and receive the state transition requests from the Exercise Controller via the HLA Federation Management services.

9. Scenario initialization

Scenario initialization is completely managed from the Exercise Controller by sending HLA interactions. No scenario initialization data is distributed pre-run time to the federate applications.

The scenario initialization data is specified in an XML scenario file that is only read by the Exercise Controller. Amongst others this file specifies the entities (i.e. ship, interceptor and TBM), their initial position and the TBM launch times. The Exercise Controller uses HLA interactions to:

- request the creation of an entity,
- set the initial position of an entity and
- set the launch time of a TBM.

10. Scenario control

Like scenario initialization discussed above also scenario control is fully managed by the Exercise Controller. Amongst others the following data is specified in the XML scenario file: the size of the defended footprint (in terms of grid points), the number of runs per grid point, the initial state of the random number generator and the execution mode (run or replay).

Depending on the execution state (see earlier discussion on state diagram) the Exercise Controller sends an HLA interaction to indicate to other federates:

- the mode of execution,
- the initial state of the random number generator,
- the current grid point and run number and
- the end of a range of grid points.

4.3 Lessons Learned

This chapter lists some of the lessons learned with respect to the solution applied.

1. Start and stop of applications

The “black box” approach of using a Pitch Agent provided a flexible solution to start and stop individual federates at any time, without the involvement of remote operators in a different time zone and physical location.

2. Network monitoring

Network latency is the dominating parameter in overall federation execution time. For the network, hardware and software configuration chosen for this project, the execution time of the distributed federation is estimated to be twice the execution time of the federation running co-located on a LAN.

3. Performance monitoring

The additional monitors for federation performance provided essential insight in federate busy and idle times and network related processing. This kind of information is generally needed for tuning federation performance during federation development.

4. Creation and deletion of federation execution

The performance of the HLA time management services depend very much on where the CRC is running. Since the time advancements characteristics of all federates, the network latency and the way how time management is implemented by the RTI all have influence on the overall performance, it is recommended to investigate the best location of the CRC for a particular federation. Performance monitoring tools are an important asset for this.

As we have two federations (MTMD and Local Ship), it would be better that the Ship Federation Manager federate could connect to two different CRCs. In this case, a CRC running on a host at Hengelo for the Local Ship federation and a CRC running on a host in the USA for the MTMD federation would be optimal in terms of reducing latency times according to our performance analysis.

The CRC of the TNO-RTI creates for each federation execution an application called ‘fedex’ that coordinates amongst others the time advancements in a federation execution. So another solution could be that the CRC can be configured manually or even determines automatically where to run the fedex application for optimal performance results.

5. State control

During the development of the Exercise Controller it was noted that the management of the execution state diagram can be generalized using a recent development briefly described below. Due to time constraints however a more project specific approach was followed.

A recent development that will certainly benefit the development of a “generic” Exercise Controller is that of State Chart XML (SCXML) of the W3C [8]. SCXML is an XML based markup language for the definition of state machines. Such a generic Exercise

Controller can have as input an SCXML file that specifies the state diagram along with the triggers, conditions and actions it has to perform on each state transition. Actions are for example the initiation of an HLA Save or Restore, or the registration of a synchronization point.

6. Scenario initialization and control

For this project the scenario was relatively simple and the (proprietary) XML scenario file was created manually. For a more complex scenario a graphical (2D map) scenario editor to generate the required XML scenario file would be desirable and recommended. Also the use of MSDL as scenario file format should be studied.

7. General

Central federation execution management capable of controlling the state of each federate proved to be efficient for distributed federations at locations having different time zones. However, voice communication remains necessary, despite the high degree of automation of federation execution management.

5. Conclusions

This chapter summarizes the conclusions for federation execution management.

In the SIGDM&S project a plurality of "point tools" was used for federation execution management, for various technical reasons. A conclusion from this project is that the need remains for a more integrated solution for all federation execution management tasks. For example network, host, federation and federate monitoring should be more integrated. Pitch Commander provided a number of capabilities as first steps to this integrated solution.

During federation development and execution often detailed monitoring information is required to understand the behavior of the federation with respect to time. Current federation execution management tools do not provide these monitoring capabilities. Future federation execution management tools should provide more simulation related performance monitoring capabilities. For example: trend lines, snapshots and averages for e.g. federate busy and idle times, federate network related processing, federate simulation speed and federate object attribute values. It has to be investigated whether it is desirable to extend the MOM to provide more inside RTI information.

With current federation execution management tools state control logic is usually coded in a programming language like Java (Exercise Controller) or Python (Pitch Commander). Future federation execution management tools should support a common document format for the specification and exchange of state diagrams for federation state control. It is recommended to research the application of State Chart XML (SCXML) from the W3C [8] as a possible candidate. Reference [9] discusses the application of SCXML for state control design patterns and demonstrates an HLA Execution Manager federate that is capable of executing SCXML.

Future federation execution management tools should provide support for resuming a long duration (Monte Carlo) simulation from an earlier "save point" in case of interruption.

6. References

[1] IEEE Std 1516.1-2000, IEEE Standard for Modeling and Simulation (M&S), High Level Architecture (HLA), Federate Interface Specification.

[2] Björn Löfstrand, et al: Scenario Management – Common Design Principles and Data Interchange Formats; Simulation Interoperability Workshop (04E-SIW-070); 2004.

[3] Pitch Commander: http://www.pitch.se.

[4] Whats Up Gold: http://www.whatsupgold.com.

[5] Iperf: http://iperf.sourceforge.net.

[6] MTUroute: http://www.elifulkerson.com/projects.

[7] IEEE Std 1516.2-2000, IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA). Object Model Template (OMT) Specification, March 2001.

[8] W3C SCXML: http://www.w3.org/TR/scxml.

[9] T.W. van den Berg, et al: Design Patterns for and Automation of Federation State Control; Simulation Interoperability Workshop (09S-SIW-009); 2009.

Author Biographies

TOM VAN DEN BERG is scientist in the M&S department at TNO Defence, Security and Safety, The Netherlands. He holds an M.Sc. degree in Mathematics and Computing Science from Delft Technical University. His research area includes distributed processing and simulation systems, software architectures and software process improvement.

HENK JANSSEN is Senior Project Manager in the M&S Department at TNO Defence, Security and Safety in the Netherlands. He holds a M.Sc. degree in Aerospace Engineering, and has to date more than 20 years experience on the Modelling and Simulation of complex weapon systems, distributed simulation systems and simulation development process improvement.

ROGER JANSEN is a member of the scientific staff in the M&S department at TNO Defence, Security and Safety in 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.

LOUWRENS PRINS is a member of the scientific staff in the M&S department at TNO Defence, Security and Safety in the Netherlands; has a Bachelor of Science degree and works in the field of software engineering for various projects since 1998.