UAS-NAS

Live Virtual Constructive Distributed Environment (LVC)

LVC Gateway, Gateway Toolbox,

Gateway Data Logger (GDL), SaaProc

Software Design Description

National Aeronautics and Space Administration

Ames Research Center
Moffett Field, CA 94035-1000

LVC SWDD-03

Rev B

Release Date: April 20, 2015

Software Design Description

Document No. LVC SWDD-03, Rev B

Release Date: April 20, 2015

Prepared By:

Srba Jovie – IT&E SW and System Integration Lead, SAIC/NASA ARC

Concur:

Jamie Willhite – IT&E – LVC Integration and Test, NASA AFRC

Neil Otto – Project Manager, SAIC/NASA ARC

Approve:

Jim Murphy – IT&E Project Engineer, NASA ARC

4.28.15

Peggy Hayes – Deputy Chief Systems Engineer, NASA AFRC

Revision Log

Version #	Date	Change #	Page #	Author	Description
Baseline	May 1, 2013	0	All	Srba Jovic	Initial Release of Document
Rev A	Feb 19, 2014		All	Srba Jovic	Revision to support the IHITL development phase.
Rev B	Apr 20, 2015		All	Srba Jovic	Revision to support the FT3 development phase.

Table of Contents

1.0 Introduction 1
1.1 Scope and Overview 4
2.0 Reference Documents 5
3.0 Software Design and Development Considerations 6
3.1 Programming languages 6
3.2 LVC Software 6
3.3 LVC Source Code Management 6
4.0 System Architecture Design 7
5.0 LVC Gateway Design 7
5.1 LVC Gateway Class Diagram 8
5.2 Data Flow 11
5.3 Sequence Diagram 15
6.0 LVC Gateway Data Logger Design 28
7.0 LVC Gateway Toolbox Design 29
7.1 LVC Gateway Toolbox Class Diagram 29
8.0 Sense and Avoid Processor (SaaProc) Design 30
8.1 SaaProc Class Diagram 31
9.0 Software Requirements Traceability and Implementation 35
Appendix A – Acronyms 36

1.0 Introduction

The desire for a capability to fly government and commercial Unmanned Aircraft Systems (UAS) in the National Airspace System (NAS) is of increasing urgency. The application of unmanned aircraft to perform national security, defense, scientific, and emergency management functions are driving the critical need for less restrictive access by UAS to the NAS. UAS represent a new capability that will provide a variety of services in the government (public) and commercial (civil) aviation sectors. The growth of this potential industry has not yet been realized due to the lack of a common understanding of what is required to safely operate UAS in the NAS.

NASA's UAS Integration into the NAS Project is conducting research in the areas of Sense and Avoid/Separation Assurance Interoperability (SAA), Human Systems Integration, Communication, and Certification to support reducing the barriers of UAS access to the NAS. To accomplish this task, the Project will conduct a series of Integrated Human-in-the-Loop (IHITL) and Flight Test activities that integrate key concepts, technologies and/or procedures in a relevant air traffic environment. Each of the integrated events will build on the technical achievements, fidelity and complexity of the previous tests and technical simulations, resulting in a body of evidence that supports the development of regulations governing the access of UAS into the NAS.

The Flight Test 3 (FT3) has three distinct test configurations, each focusing on a separate aspect of the acceptability of the SAA algorithms. These test setups can be traced back to the two test configurations shown in Figures 1 and 2. Within the test setups, there are common LVC components required for a complete integrated test environment.
Figure 1 provides a high-level diagram of FT3 System Configuration 1. System Configuration 1 is designed to support pairwise encounters of the low speed ownship - Ikhana with an intruder for the set of designed scenarios. The aircraft will be flying at Armstrong Flight Research Center (AFRC). LVC subsystems such as Conflict Prediction and Display System (CPDS), Vigilant Spirit Control Station (VSCS) with Research Ground Control Station (RGCS), SaaProc/JADEM, and Stratway+ will interface with LVC Gateway research critical messages.

The FT3 System Configuration 2, showed in Figure 2, supports the Full Mission experiment. The Vehicle Specific Module (VSM) is the subsystem that provides the data interfaces defined by the LVC ICD-03 between the live Surrogate UA and the LVC environment as presented in Figure 2.

The framework for the simulation environment is provided by the LVC via the High Level Architecture (HLA) messaging infrastructure. HLA is an industry standard that provides a well-defined set of messages, message formats defined by the Federation Object Model (FOM) specified for the given project and application programming interface (API) for conducting air traffic simulations. The virtual UAS simulator, VSCS, is be integrated into a Research Ground Control Station at used to present SAA advisories to the pilot as well as to transmit Command and Control messages with the Surrogate UA. LVC Gateway is another core component of LVC distributed test environment. The components send and receive data to other LVC components

through LVC Gateway connected to the HLA network. The constructive aircraft and ATC workstations will be co-located on the same network that communicates with other components via the HLA through their respective HLA Toolboxes. Constructive aircraft provide the required background traffic to create a realistic NAS environment. Flight simulators at Ames are connected to the LVC via HLA utilizing HLA Toolboxes. As part of FT3, the LVC's capability to provide a realistic and relevant environment, including the timeliness (e.g., synchronicity and latency) of the data and the maturity of the ATC and GCS emulation will be evaluated.

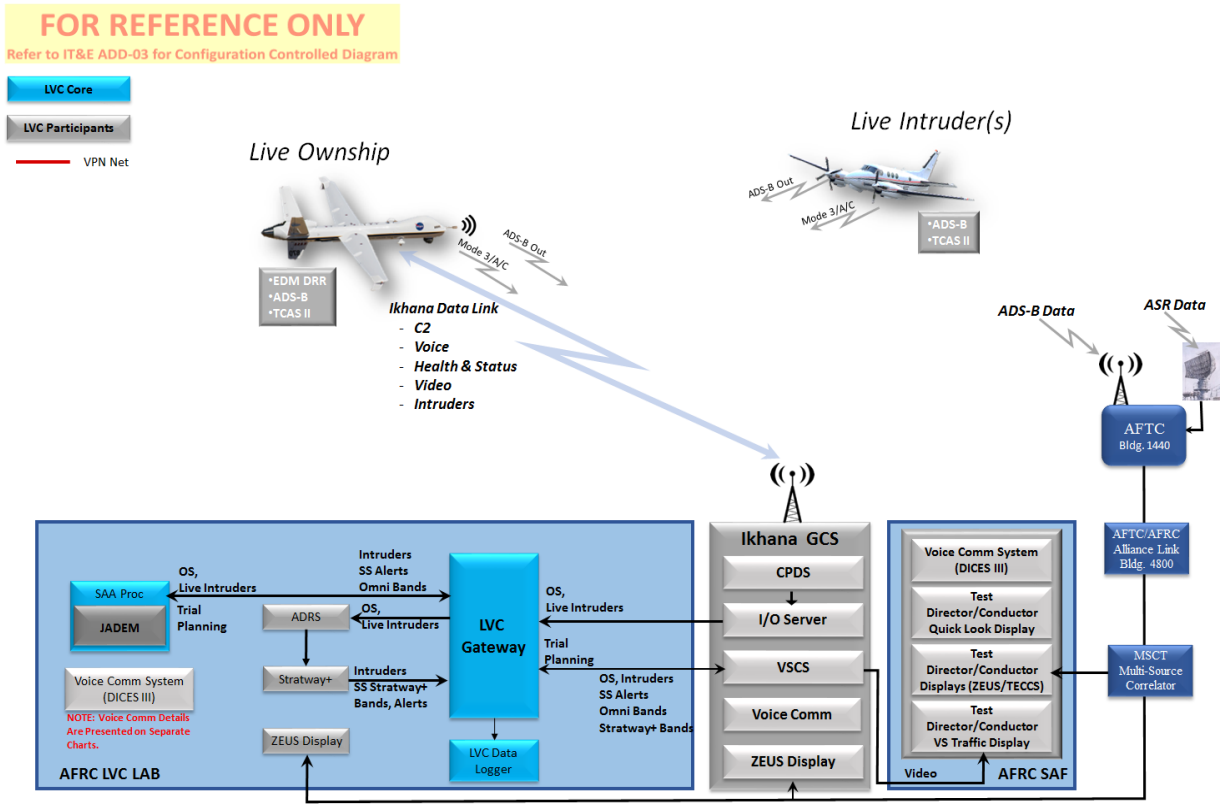

Figure 1. System Configuration 1A

LVC system components rely on the core LVC components, identified in light blue in Figures 1, 2, and 3 to exchange shared information with each other. All participating system components are integrated either through a High Level Architecture (HLA) Toolbox interface, or through the LVC Gateway. The existing HLA distributed environment has the capability to integrate constructive air traffic that is generated by the Multi-Aircraft Control System (MACS), which also contains a robust Air Traffic Control and pseudo pilot capability. In addition, MACS has the capability to publish and subscribe to all required messages (Flight Plan, Flight State, and Delete) through an HLA Toolbox. The LVC Gateway will integrate participating components that do not have the capability to connect directly to HLA into the LVC. The LVC interfaces with HLA as a server to the LVC Gateway Toolbox.

Figure 1 shows the architecture in support of pairwise encounters with the low speed Ikhana UAV as the Ownship. The CPDS with the I/O Server is used as the interface for receiving the

live aircraft data and passing on to the LVC environment.

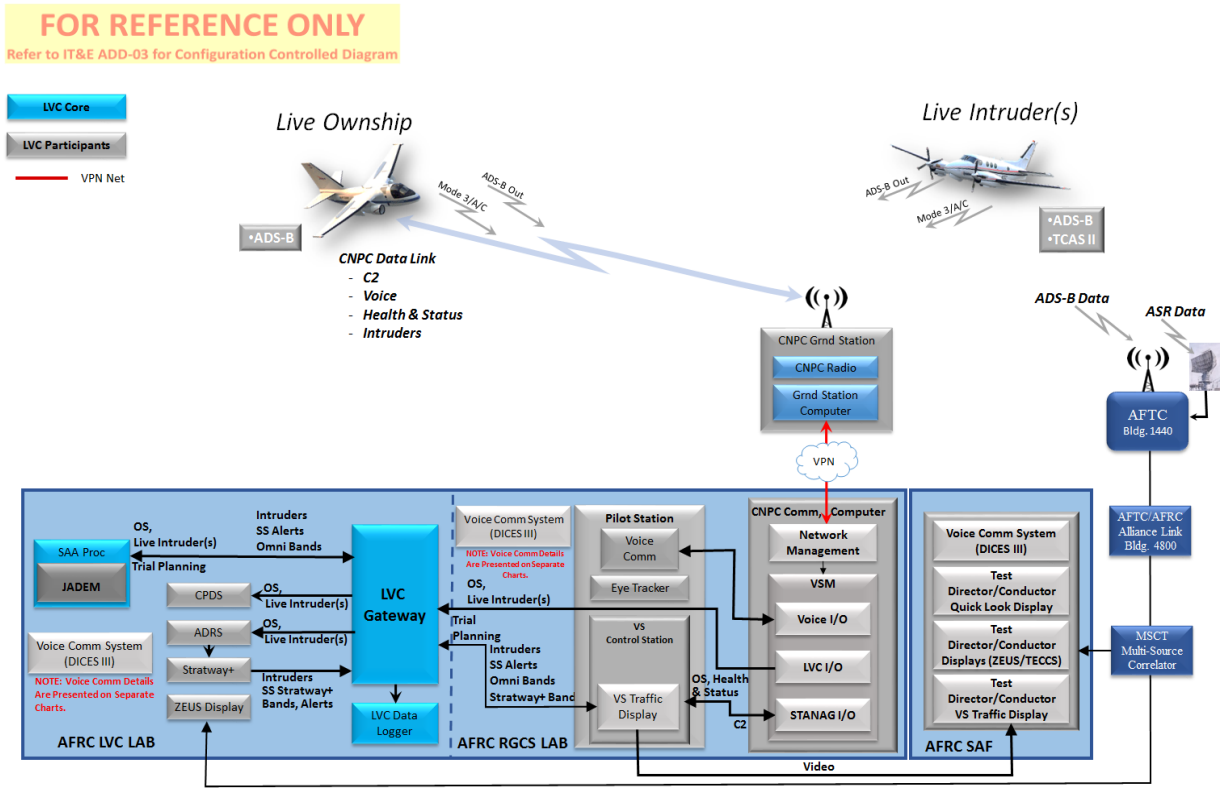

Figure 2. System Configuration 1B

Figure 2 shows the architecture in support of pairwise encounters with the high speed S3 as the Ownship. The VSM interface is used as the interface for receiving the live aircraft data and passing on to the LVC environment.

FOR REFERENCE ONLY
Refer to IT&E ADD-03 for Configuration Controlled Diagram

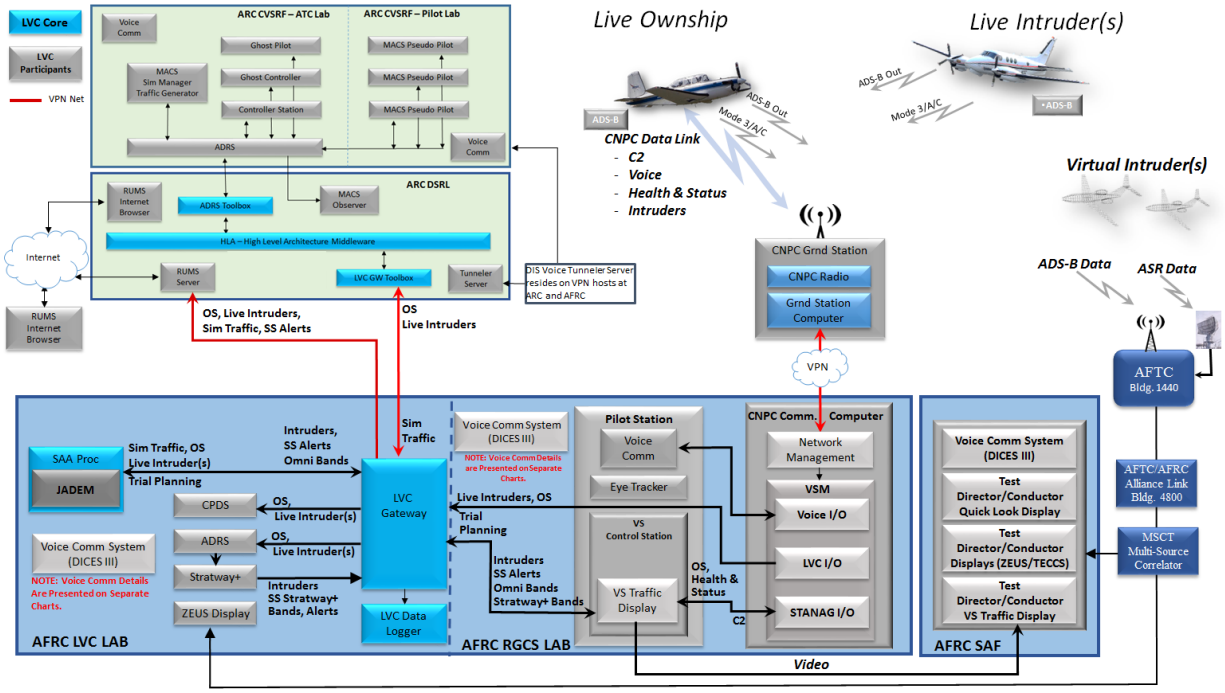

Figure 3. System Configuration 2

Figure 3 shows a high-level diagram of the FT3 System Configuration 2. As with System Configuration 1, System Configuration 2 is designed to provide connectivity among a virtual UAS GCS, constructive manned traffic generators, and virtual ATC. In this case these virtual and constructive components will be provided via facilities at NASA Langley. In addition, a flight simulator with TCAS II version 7.0 running at NASA Ames and connected via HLA will provide the primary intruder aircraft during testing. The UAS GCS at NASA Langley will contain a combined Virtual Simulator, the CDTI, and the SAA algorithm for testing.

Note that this SWDD traces implementation to the new or revised LVC SWRD-02 Rev C requirements for FT3 as follows. (See Section 9)

1.1 Scope and Overview

This document provides the software design description for the two core software components, the LVC Gateway, the LVC Gateway Toolbox, and two participants—the LVC Gateway Data Logger and the SAA Processor (SaaProc).

2.0 Reference Documents

Document Title	*Description*
APR-7150.2	*Ames Software Engineering Requirements*
IT&E-CMP-001	*IT&E Configuration Management Plan*
IT&E PP-01	*IT&E Project Plan*
IT&E RMP-01	*UAS-NAS Risk Management Plan*
IT&E SAP-01	*IT&E Software Assurance Plan*
IT&E ORD-01	*Objectives and Requirements Document (ORD)*
LVC SRD-01	*System Requirements Document*
LVC-SWRD-02	*Software Requirements Document*
IT&E SDMP-01	*Software Development & Management Plan*
LVC V&VP-01	*LVC Verification & Validation Test Plan*
LVC Gateway ICD-03	*LVC Gateway Message Interface Control Documents*

3.0 Software Design and Development Considerations

3.1 Programming languages

The LVC Gateway software development will use C++ programming language and Object Oriented Design methodology enabled by the language paradigm. The Object Oriented abstraction concept will be utilized for communication classes to capture specifics of different clients' interfaces as well as for configuring socket types as a TCP client or server. The design can accommodate other protocols such as UDP, Multicast or Loopback sockets. Some of the hardware interfaces and libraries are best integrated using the C++ language due to the timeframe in which these libraries were created.

3.2 LVC Software

The LVC is a complex system designed and developed by the combined NASA/ARC and NASA/DFRC IT&E teams to integrate with other Government off-the-shelf (GOTS) software. The processes and procedures for software development and management are described in the LVC SDMP-01 software development and management plan and the IT&E configuration management process is defined in the IT&E CM Plan. This includes the typical Configuration Change Board (CCB) and CM processes followed at SimLabs at ARC.

3.3 LVC Source Code Management

Version control for the LVC software is maintained using the Concurrent Versions System (CVS). Each revision of the source code will be checked into the source code configuration management repository using CVS, providing complete traceability of all software changes for each application build and executable.

The LVC Gateway, LVC Gateway Data Logger, LVC Gateway Toolbox, and SaaProc software have been determined to be class D software as defined in APR 7150.2.

4.0 System Architecture Design

The LVC Gateway functional diagram is shown in Figure 3. The LVC Gateway will integrate participating components that do not have the capability to connect directly to the LVC environment through the HLA. The LVC Gateway, and LVC Gateway Data Logger have been modified to handle connectivity and messaging associated with SaaProc, and Vigilant Spirit Control Station (VSCS).

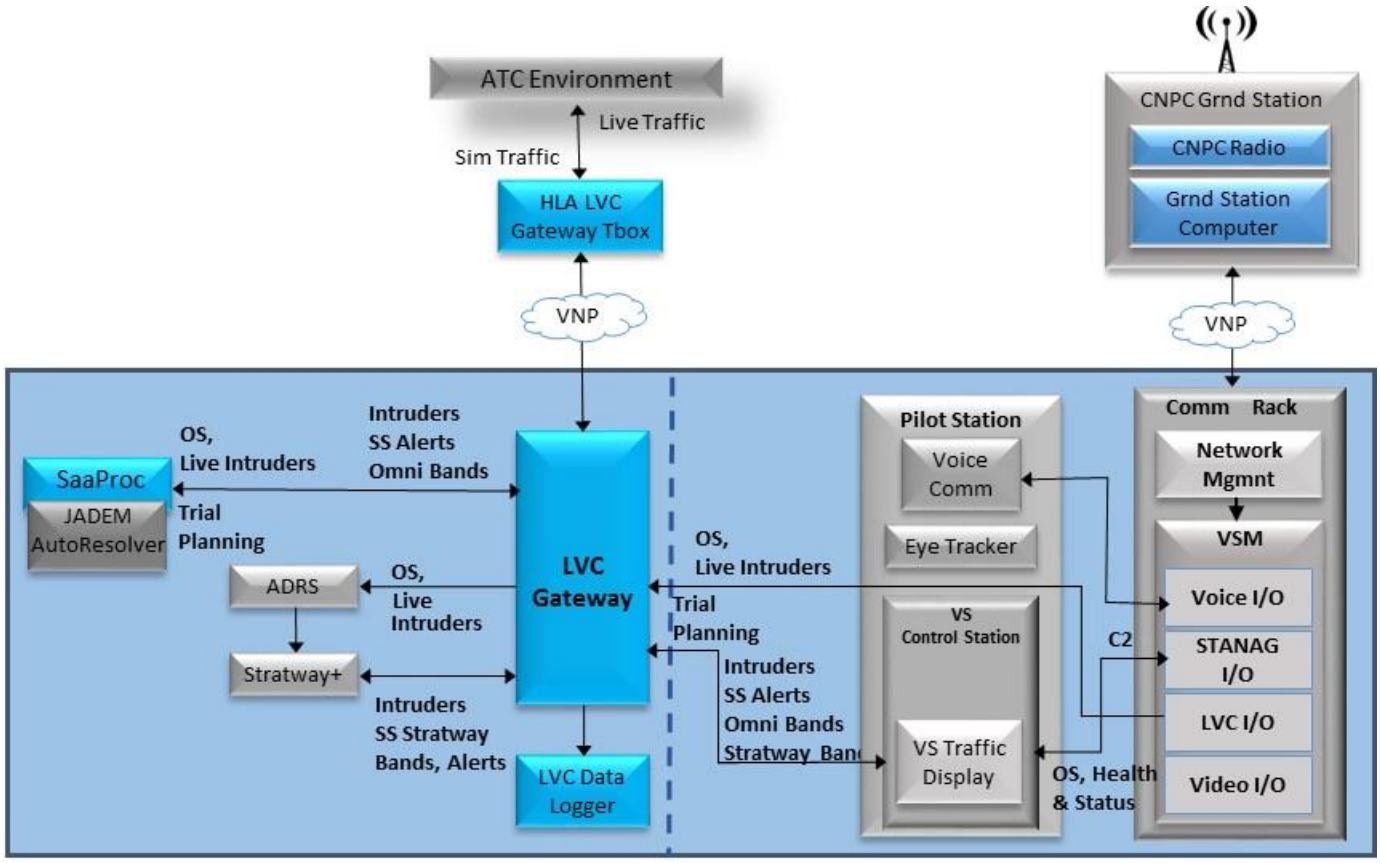

Figure 4. LVC System Architecture for One of the FT3 Configurations.

5.0 LVC Gateway Design

HLA Middleware (COTS) and the LVC Gateway constitute the core components of the LVC system. The HLA LVC Gateway Toolbox will interface with the LVC Gateway as a client. Test participants and simulation components will connect as clients to the LVC Gateway's socket server. The LVC Gateway ICD-03 Rev-B provides data structures definitions in support of the project requirements. In addition to the data structures that supported IHITL experiment, the ICD has been updated to include SAA Omni Band and Stratway Bands messages in order to meet system requirements for VSCS capabilities. Upon initial network connection, each client will send a handshake message to the Gateway defining the message types to which the client will publish and subscribe.

The ICD specifies details for data exchange between the LVC Gateway and the following clients:

- Vigilant Spirit Control Station (VSCS)
- SaaProc/JADEM
- CPDS/IOServer
- LVC I/O (VSM)
- LVC Gateway Data Logger
- LVC Data Collector
- Stratway+/DADELIS/ADRS
- LVC Gateway Toolbox

The first six clients listed above will connect to the LVC Gateway using MPI interface type on the LVC Gateway server's TCP socket. The client will be required to map its native data structure to the MPI interface.

The LVC Gateway connects to the Stratway+ interface as a client using the ADRS interface type. The ADRS client has different interface requirements. The LVC Gateway will send a periodic heartbeat message to each client to check the health of the connection. If the socket connection does not reply in a timely manner due to a client process crash, process disconnect (voluntary or involuntary) or socket failure, the LVC Gateway will send the delete message out to the LVC system for tracks associated with that client.

5.1 LVC Gateway Class Diagram

The Unified Modeling Language (UML) class diagram of the LVC Gateway design is shown in Figure 5. The figure depicts the software architecture representing the relationship between the classes of the LVC Gateway.

The Extensible Markup Language (XML) input configuration file is used to define the LVC Gateway configuration in term of interface types, interface socket types, and data structures that are transmitted by the LVC Gateway.

The GW Client Manager class is responsible for the instantiation of all LVC Gateway components in the initialization phase of the code based on the XML configuration file. The GW Client Manager contains a collection of GW Client objects. In addition, it utilizes the observer/observable pattern. Using this paradigm, the *observable* is associated with clients that publish messages defined in the ICD while the *observer* is associated with clients that subscribe to those messages.

The GW Client class is an abstract class that instantiates as the MPI or ADRS interface types as defined by the XML configuration file. Each GW Client can have many TCP and/or UDP sockets of the client and/or the server type. Each interface on the LVC Gateway side will receive the legacy data messages such as Flight State, Flight Plan, Trajectory Intent, SAA and Trial

Planning data from respective participants in MPI data format. In addition to the legacy messages, two new messages are added for FT3, Saa Omni Band and Stratway Bands messages, in support of the new FT3 requirements. All messages are time stamped, buffered and then passed to the subscribing clients.

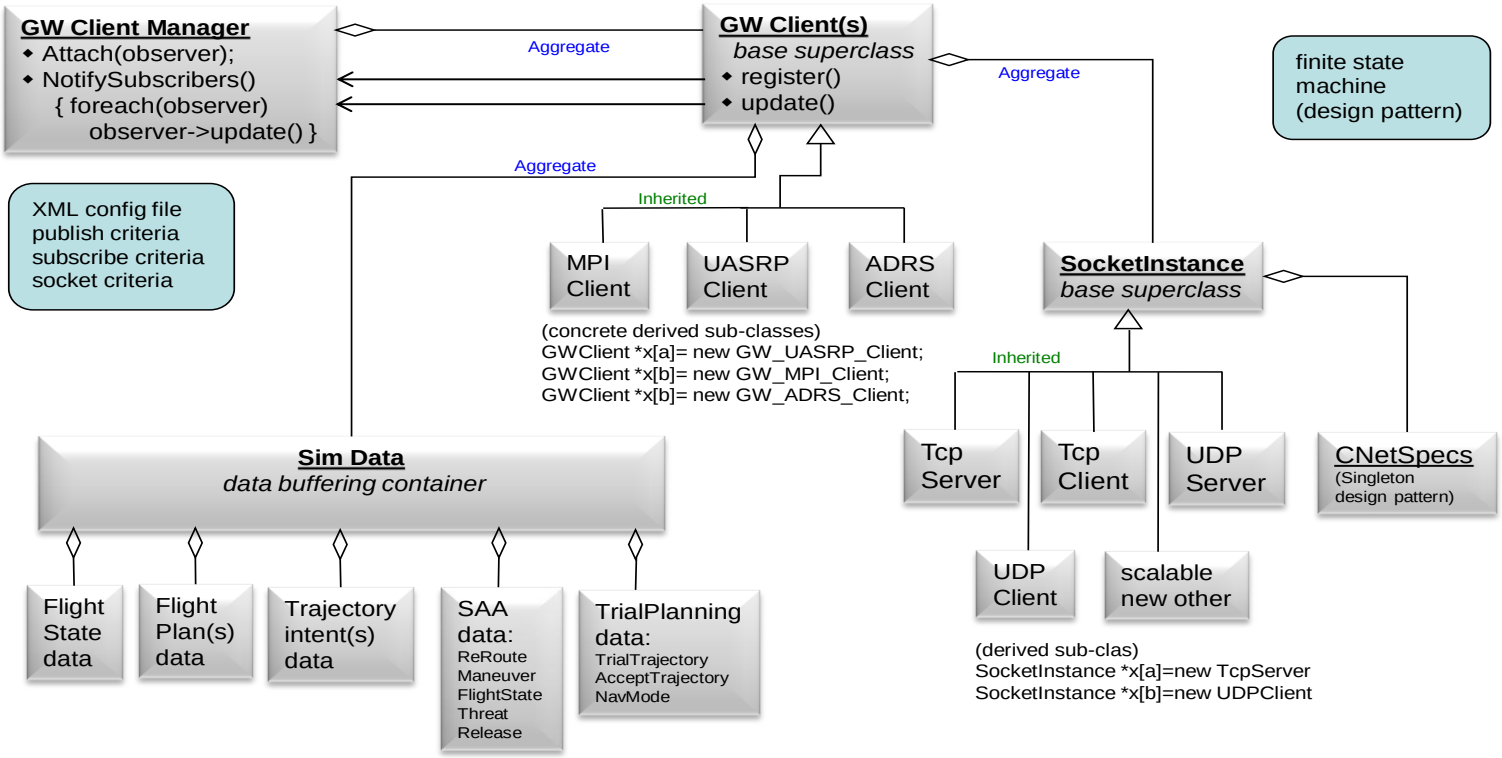

Figure 5. LVC Gateway Class Diagram

5.2 Data Flow

The LVC Gateway will exchange air traffic data and the set of Sense (Detect) And Avoid (SAA/DAA) messages with live, virtual and constructive participants, as depicted in Figure 6.

The VSCS will provide Flight Plan, Flight State, Trajectory Intent and Navigation Mode data. The initial Flight Plan is published as soon as the VSCS is launched while the Flight State message is published periodically at the rate of 10Hz once the simulation begins operating. The Navigation Mode message is event driven and is sent every time the pilot changes the ownship's navigation mode between Flight Plan, Autopilot, Override and Manual modes. The VSCS will receive a subset of the ownship's background traffic as determined by the ownship's sensor range filter via the SAA Flight State message depicted in Figure 6a, b, and c. In the current design, the sensor range filter is an integral part of JADEM and Stratway+ GCS subsystems.

Constructive Flight Plan and Flight State data generated by MACS are provided through the LVC Gateway HLA Toolbox, depicted in Figure 6c. The ownship's Flight Plan, Flight State messages will be sent to the LVC Gateway HLA Toolbox and on to MACS ATC displays.

The SaaProc/JADEM receives Flight State message updates for constructive background traffic at 1Hz data rate. In addition, it receives all flight data including navigation mode data from VSCS. The SaaProc also receives the Trial Trajectory Intent message published by the VSCS during the trial planning operation at the nominal rate of 15Hz. SaaProc receives the VSCS ownship Flight State message at the rate of 10Hz, then down-samples to a rate of 1Hz to match the data rate of the MACS constructive traffic. The new SAA Flight State message is generated by using a sensor range filter, applied to the state data of the constructive traffic, in order to emulate a sensor proximal filter. The filtered background traffic is called "intruder traffic" while the state of each intruder aircraft is defined by the SAA Flight State message. The intruder traffic is published back to LVC Gateway for display on VSCS while detected intruders are also used for conflict detection by SAA.

The SAA input is comprised of the following messages: SAA Flight State (intruders), down-sampled ownship Flight State, ownship Trial Trajectory Intent and the ownship Navigation Mode. The SAA algorithms calculate the following set of output SAA messages for intruders, as shown in the Figure 6a, b, and c: SAA Threat results (Alerts), SAA Resolution Maneuvers, SAA Resolution Reroute and SAA Trial Threat results. All messages are defined in the LVC ICD-03.

When Stratway+ GCS is configured with the system instead of SaaProc/JADEM, Stratway + GCS receives Flight State message updates for constructive background traffic at 1Hz data rate. Stratway+ GCS receives the VSCS ownship Flight State message at the rate of 10Hz (then down-samples to a rate of 1Hz to match the data rate of the MACS constructive traffic). The new SAA Flight State message is generated by using the same sensor range filter as JADEM. The intruder traffic is published back to LVC Gateway for display on VSCS while intruders are used for conflict detection using Stratway+ algorithm. The resulting Stratway Bands message are sent to VSCS for display.

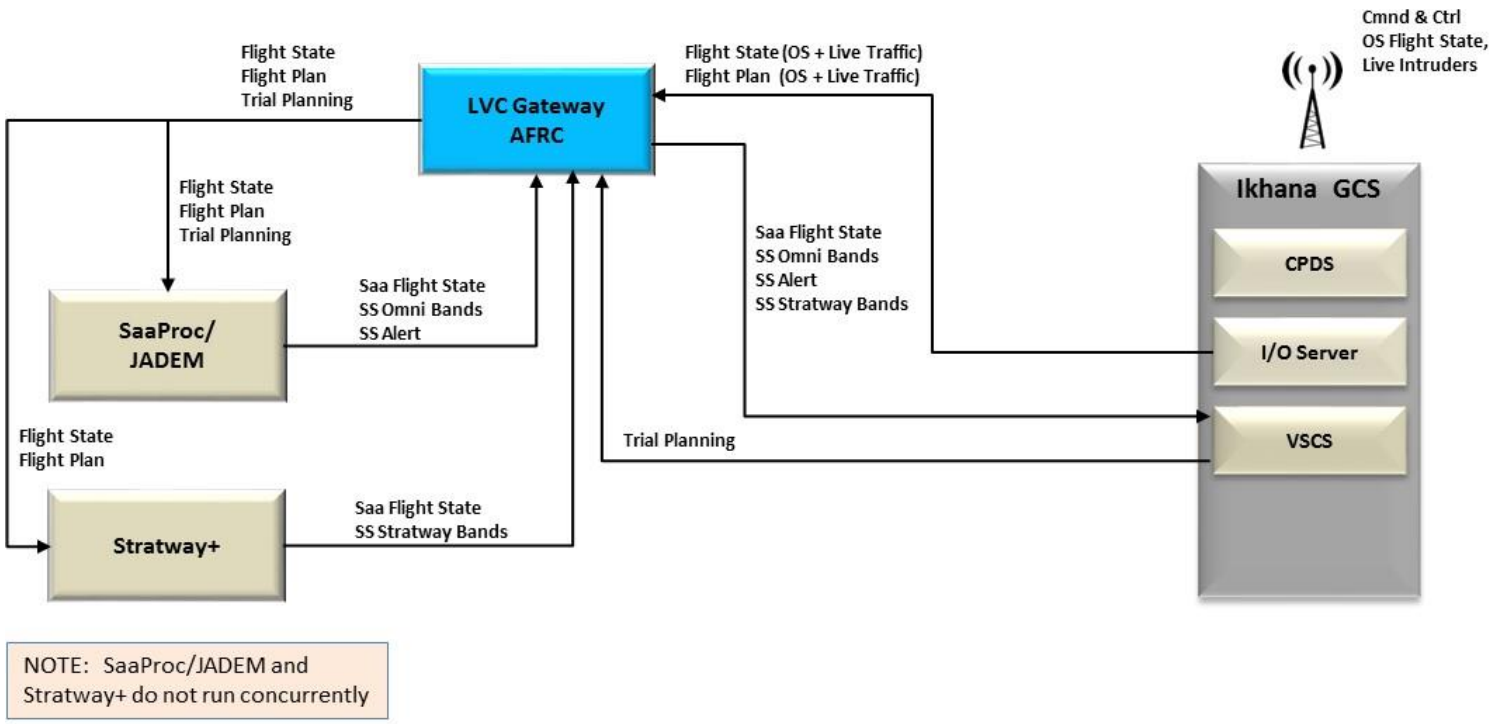

Figure 6a. Data Flow Diagram for Configuration 1A.

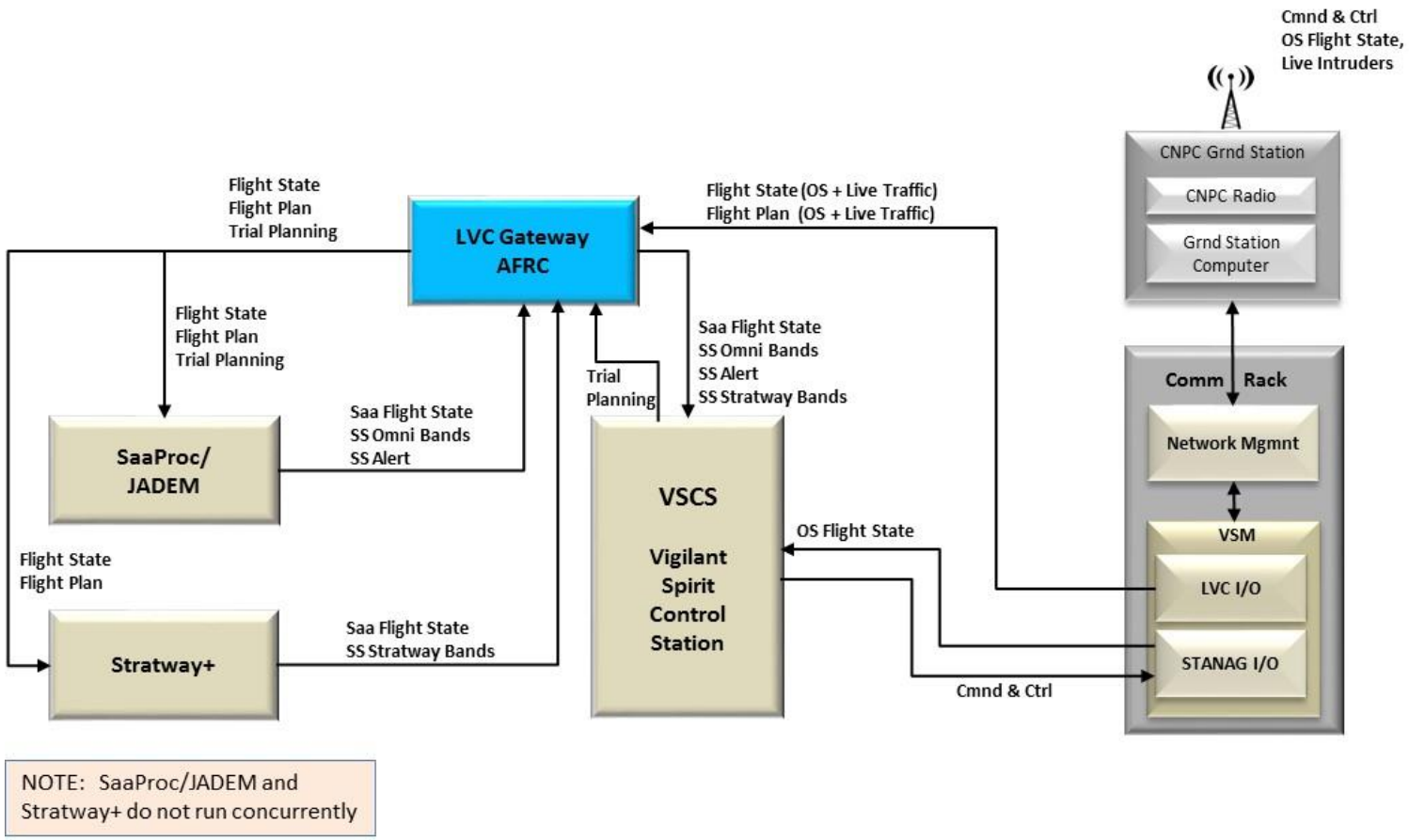

Figure 6b. Data Flow Diagram for Configuration 1B.

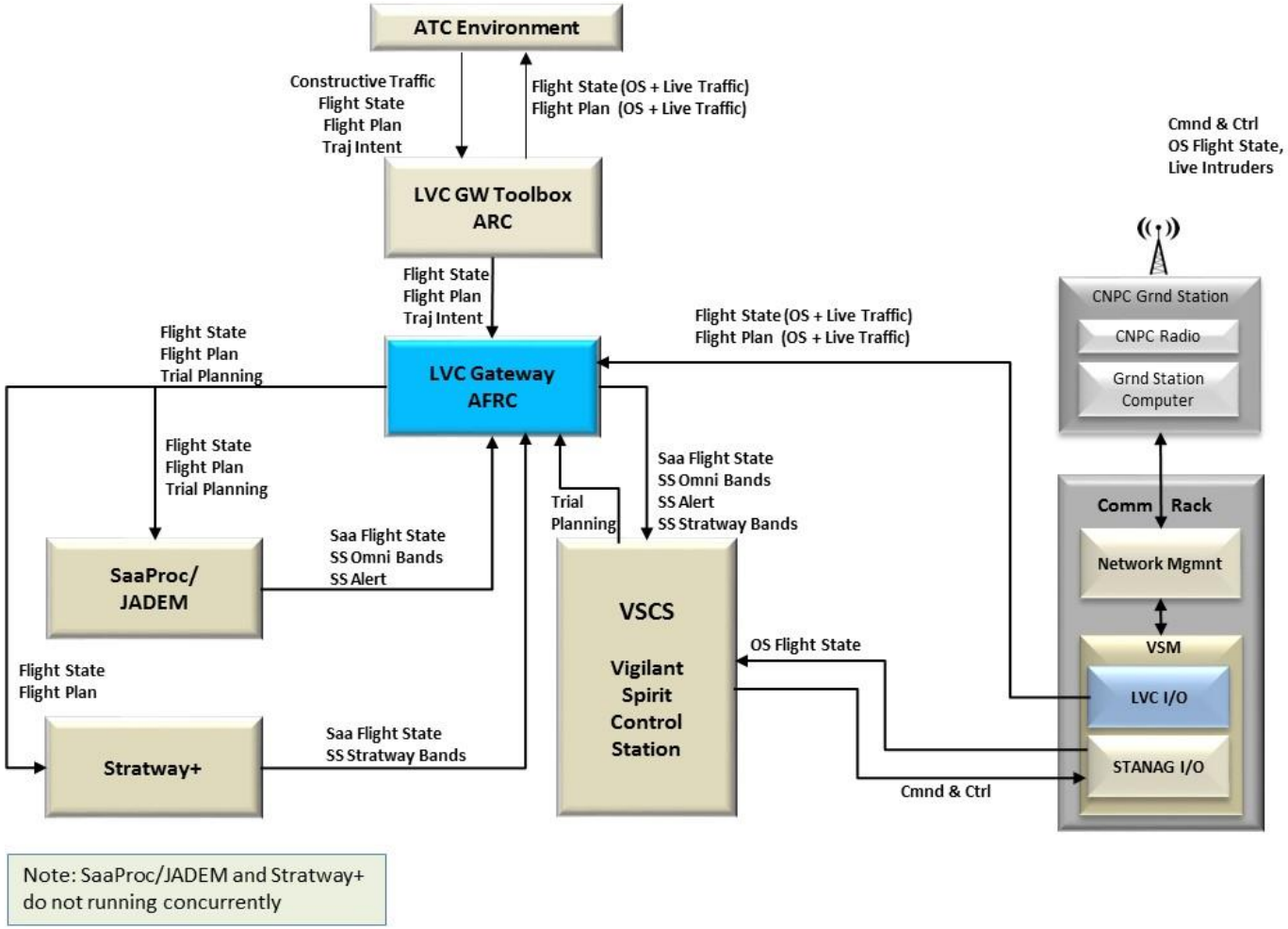

Figure 6c. Data Flow Diagram for Configuration 2.

5.3 Sequence Diagram

The total of six UML sequence diagrams associated with the message flow between the LVC system components are shown for all three system configurations (two diagrams for each configuration). The sequence diagram for the first use case for each configuration describes SaaProc/JADEM use while the second sequence diagram shown use of Stratway+ GCS. The first diagram for pairwise ownship/intruder and full mission encounter configurations are displayed in Figure 7a, 7c and 7e shows time sequence of events during the concurrent auto-resolution conflict detection, alert level, Omni Band detections and the trial planning. The second use case with Stratway+ in the loop for pairwise and full mission encounters are shown in Figure 7b, 7e and 7f showing event diagrams during the Stratway+ Bands conflict detection.

A list of the sequence of events for Configuration 1A using SaaProc/JADEM is shown below:

1. The surveillance system of the live aircraft (Ikhana) will send the state data update of the ownship (Ikhana) and intruders in the range of the on-board sensors to the Ikhana GCS.

2. The CPDS/IOServer will receive the state data messages of the live traffic. Subsequently, IOServer will map the corresponding state data fields into the MsgFlightState message data structure for the ownship and intruders.

3. The CPDS/IOServer will send the MsgFlightState messages for the live traffic to the LVC Gateway.

4. The LVC Gateway updates state data for the received traffic, including the ownship MsgFlightState, and publishes them to SaaProc/JADEM (labeled as 1Hz Thread in the diagram) at the rate of 1Hz.

5. The sensors model, which is an integral part of the SAA algorithm, determines a set of intruders based on their 3D spatial distance from the ownship.

6. SaaProc publishes only intruders back to LVC Gateway and to VSCS for display utilizing the MsgSaaFlightState message.

7. The SAA algorithm in JADEM calculates threat level for each intruder. The algorithm detects a Self Separation (SS) threat between the ownship and intruders.

8. SaaProc sends the MsgSaaThreatResult message to the LVC Gateway containing an array of data structures for each intruder. See ICD for details.

9. The LVC Gateway forwards MsgSaaThreatResults message to VSCS to display the associated threat level for each intruder.

10. The SAA algorithm in JADEM calculates and sends to The LVC Gateway SS Omni Bands for the heading and the altitude in the horizontal and vertical planes respectively. The bands are color coded advising the fly/no-fly zones for the ownship.

11. The LVC Gateway sends SS Omni Bands to the VSCS display.

12. The pilot (operator) performs trial planning to resolve the SS conflict by selecting the end of the heading vector attached to the ownship on the VSCS display. The pilot then rotates the vector around the compass rose seeking conflict free options.

13. The VSCS publishes the Trial Trajectory message (MsgTrialTrajectoryIntent) at a nominal rate of 15Hz to the LVC Gateway.

14. The LVC Gateway forwards the Trial Trajectory message to SaaProc at a rate of 15 Hz for the conflict assessment by the SAA Trial Planning code thread (labeled "15Hz Thread" at the top of the diagram).

    Note: The two threads, SAA 1Hz and SAA 15Hz, run asynchronously performing conflict detection calculations between each intruder and the ownship. See SaaProc section below for details.

15. SaaProc sends the MsgSaaTrialThreatResults message to the LVC Gateway for each trial trajectory. The message contains an array of data structures for each intruder.

16. The LVC Gateway forwards MsgSaaTrialThreatResults message to the VSCS to display the associated threat level for each intruder.

17. The pilot selects a conflict-free heading of the trial vector for the given intruder and then negotiates the heading change with ATC. Upon receiving approval from the ATC, the pilot executes the heading maneuver by pressing the "Send" button in the VSCS UI.

18. VSCS sends the MsgNavMode message to the LVC Gateway.

19. The LVC Gateway forwards the MsgNavMode to both the SAA 1Hz and the SAA 15Hz threads informing them that the ownship is now flying in the Autopilot mode vs. the Flight Plan mode prior to this maneuver.

20. The process repeats from step 1 when the new traffic update arrives into SaaProc.

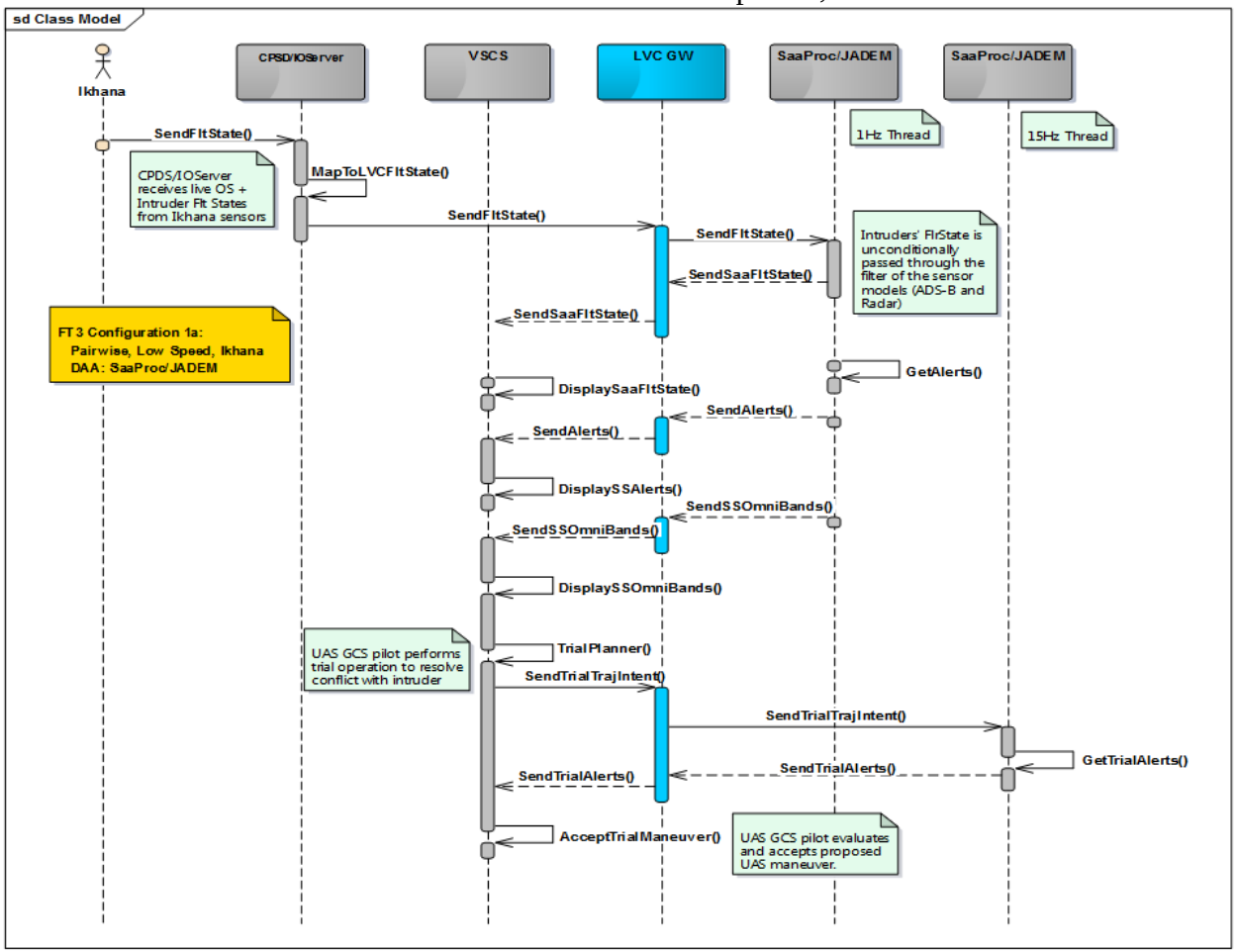

Figure 7a. UML Sequence Diagram: Conflict Detection using SaaProc/JADEM for Configuration 1A

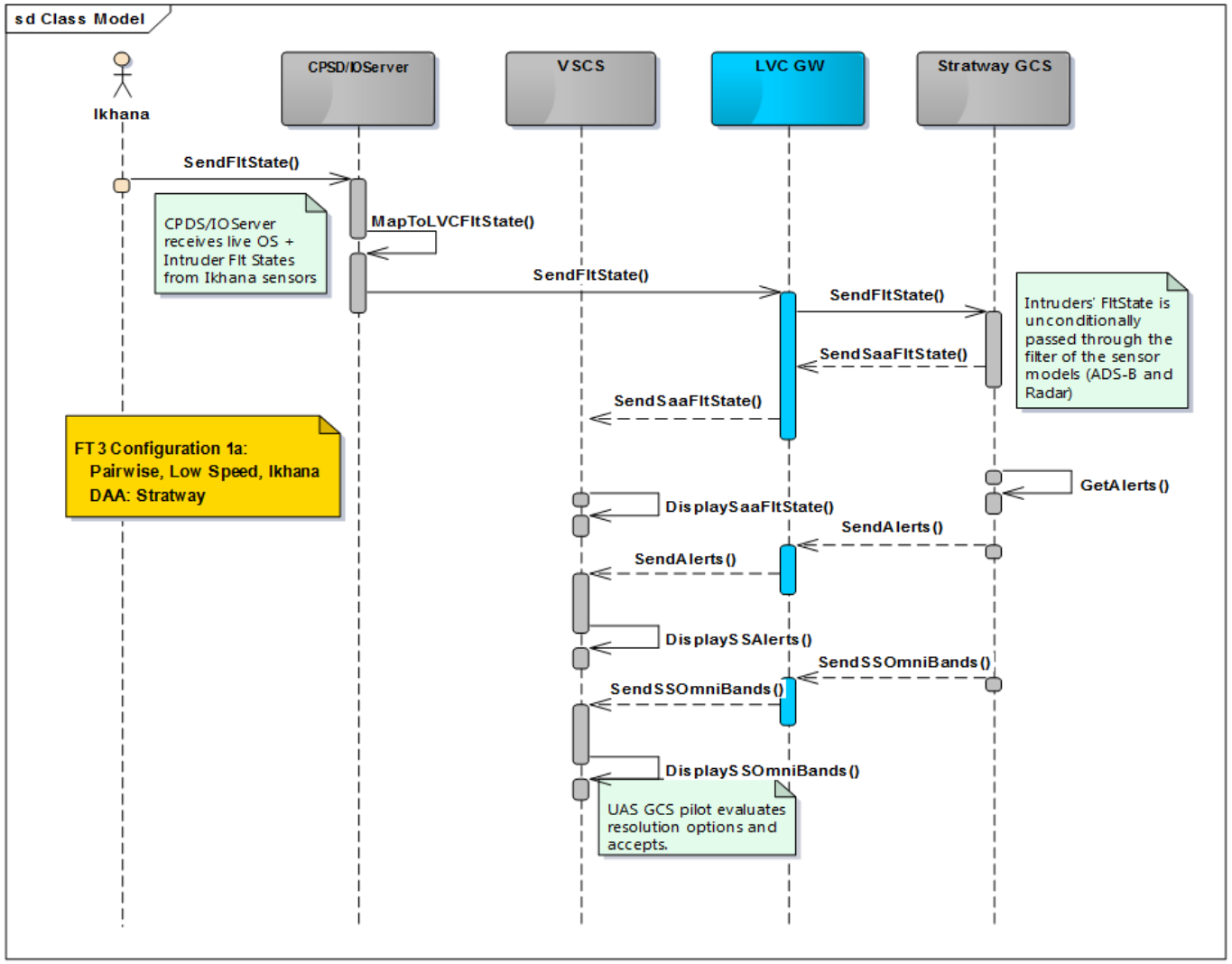

Figure 7b. UML Sequence Diagram: Conflict Detection using Stratway+ for Configuration 1A

A list of the sequence of events for Configuration 1A using Stratway+ GCS is shown below:

1. The surveillance system of the live aircraft (Ikhana) will send the state data update of the ownship (Ikhana) and intruders in the range of the on-board sensors to the Ikhana GCS.

2. The CPDS/IOServer will receive the state data messages of the live traffic. Subsequently, IOServer will map the corresponding state data fields into the MsgFlightState message data structure for the ownship and intruders.

3. The CPDS/IOServer will send the MsgFlightState messages for the live traffic to the LVC Gateway.

4. The LVC Gateway updates state data for the received traffic, including the ownship MsgFlightState, and publishes them to Stratway+ GCS at the rate of 1Hz.

5. The sensors model, which is an integral part of Stratway+, determines a set of intruders based on their 3D spatial distance from the ownship.

6. The Stratway+ GCS publishes only the intruders back to the LVC Gateway and to the VSCS for display utilizing the MsgSaaFlightState message.

7. The Stratway+ algorithm calculates no-fly bands and the threat level for each intruder.

8. Stratway+ GCS sends the MsgStratwayBands message to the LVC Gateway. See ICD for details.

9. The LVC Gateway forwards the MsgStratwayBands message to the VSCS to display the associated threat level for each intruder and Stratway+ bands. The VSCS displays four types of bands; heading, altitude, true air speed and vertical speed bands.

10. The pilot selects a conflict-free heading of the vector for the given intruder and then negotiates the heading change with ATC. Upon receiving approval from the ATC, the pilot executes the heading maneuver by pressing the “Send” button in the VSCS UI.

11. The VSCS sends the MsgNavMode message to LVC Gateway.

12. The process repeats from step 1 when the Ikhana issues a new traffic update.

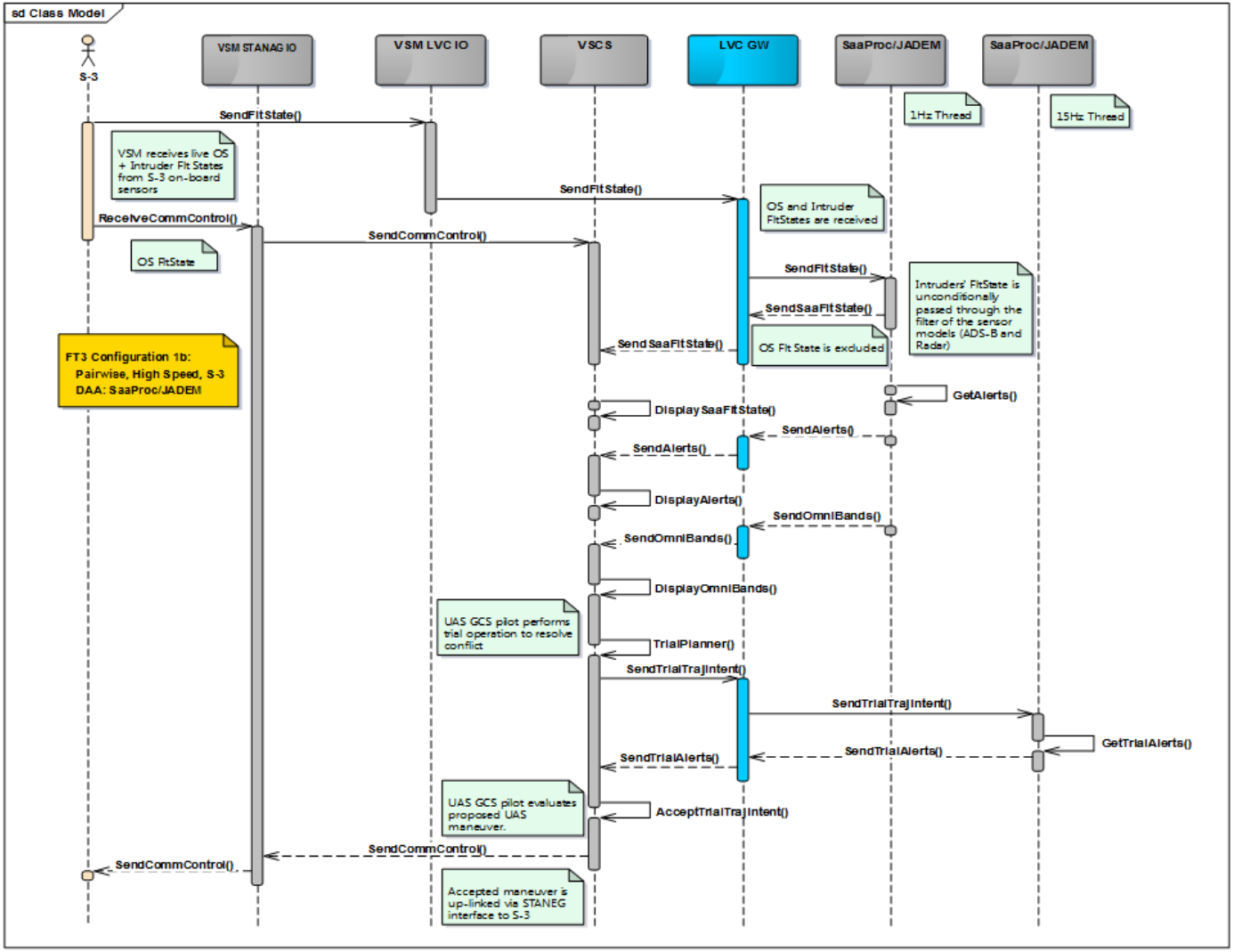

Figure 7c. UML Sequence Diagram: Conflict Detection using SaaProc/JADEM for Configuration 1B

A list of the sequence of events for Configuration 1B using SaaProc/JADEM is shown below:

1. The surveillance system of the live S-3 aircraft will send the state data update of the ownship (S-3) and intruders in the range of the on-board sensors to the CNPC Ground Station and onto the VSM LVC I/O subsystem.

2. The VSM LVC I/O will receive the state data messages of the live traffic. Subsequently, the LVC I/O will map the corresponding state data fields into the MsgFlightState message data structure for the ownship and intruders.

3. The LVC I/O will send MsgFlightState messages to the LVC Gateway.

4. Repeat Configuration 1A (when using SaaProc/JADEM) Steps 4-19, including the traffic sensor range filtering, Alerting, Omni Bands, and Trial planning.

5. Concurrently and asynchronously, STANAG 4586 standard interface containing Command and Control data is sent from the S-3 aircraft to CNPC GS and onto the STANAG I/O.

6. The STANAG 4586 message is sent directly to the VSCS.

7. The pilot might issue a heading maneuver that will be encoded into the STANAG message which will be uplinked S-3. The content of Command and Control message will be presented to the S-3 pilots for their assessment and subsequent execution of the maneuver.

8. The process repeats from step 1 when the S-3 surveillance system issues a new state update.

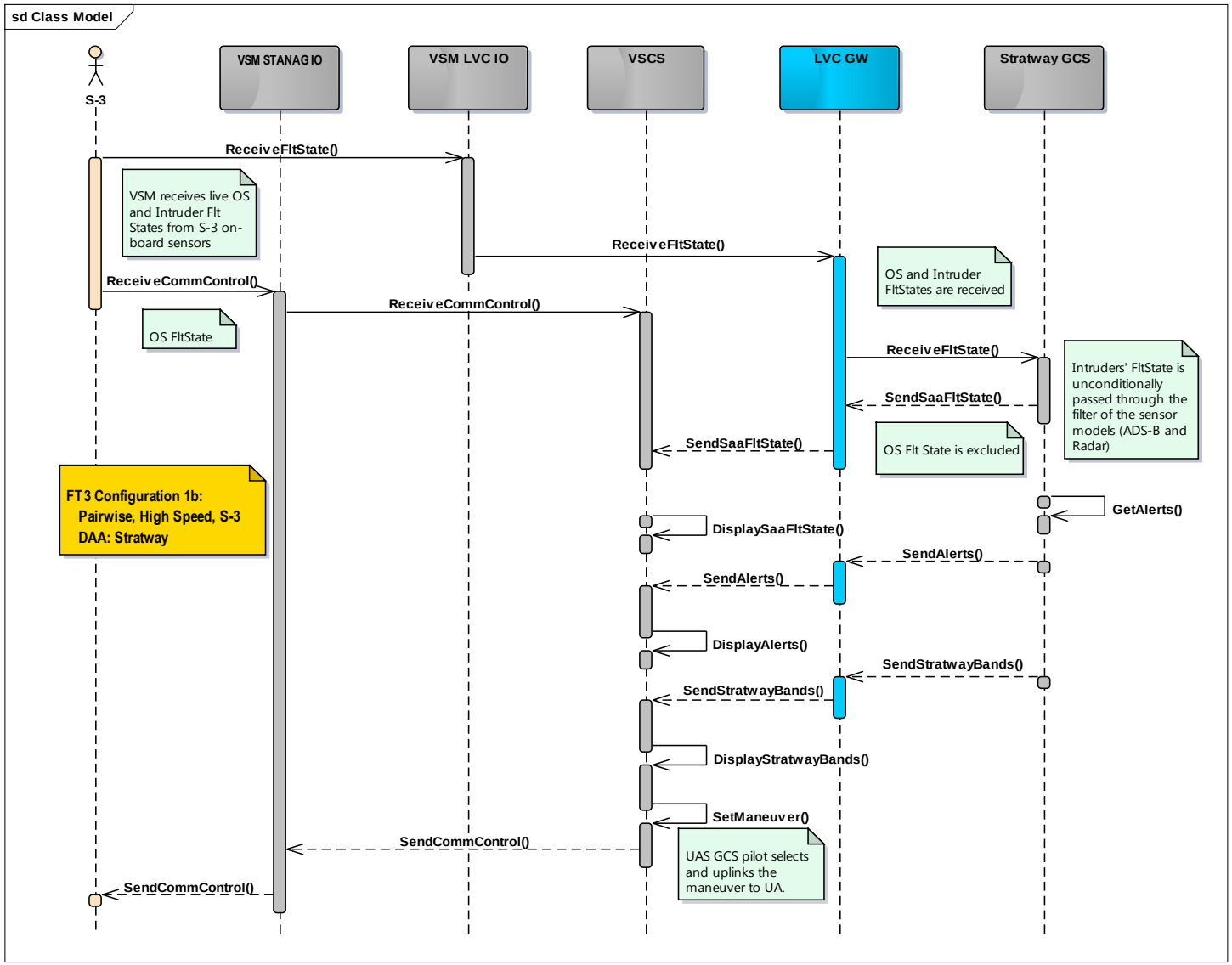

Figure 7d. UML Sequence Diagram: Conflict Detection using Stratway+ for Configuration 1B

A list of the sequence of events for Configuration 1B having Stratway+ GCS in the loop is shown below:

1. The surveillance system of the live S-3 aircraft will send the state data update of the ownship (S-3) and intruders in the range of the on-board sensors to the CNPC Ground Station and onto the VSM LVC I/O subsystem.

2. The VSM LVC I/O will receive the state data messages of the live traffic. Subsequently, the LVC I/O will map the corresponding state data fields into the MsgFlightState message data structure for the ownship and intruders.

3. The LVC I/O will send MsgFlightState messages to the LVC Gateway.

4. Repeat Configuration 1A (when using Stratway+ GCS) Steps 4-11, including the traffic sensor range filtering, Stratway+ Bands, and Alerting.

5. Concurrently and asynchronously, the STANAG 4586 standard interface containing Command and Control data is sent from the S-3 aircraft to CNPC GCS and onto the STANAG I/O.

6. STANAG 4586 message is sent directly to VSCS.

7. The pilot might issue a heading maneuver that will be encoded into the STANAG message which will be uplinked directly to the autopilot of S-3 which will be executed by the airplane.

8. The process repeats from step 1 when the S-3 surveillance system issues a new state update.

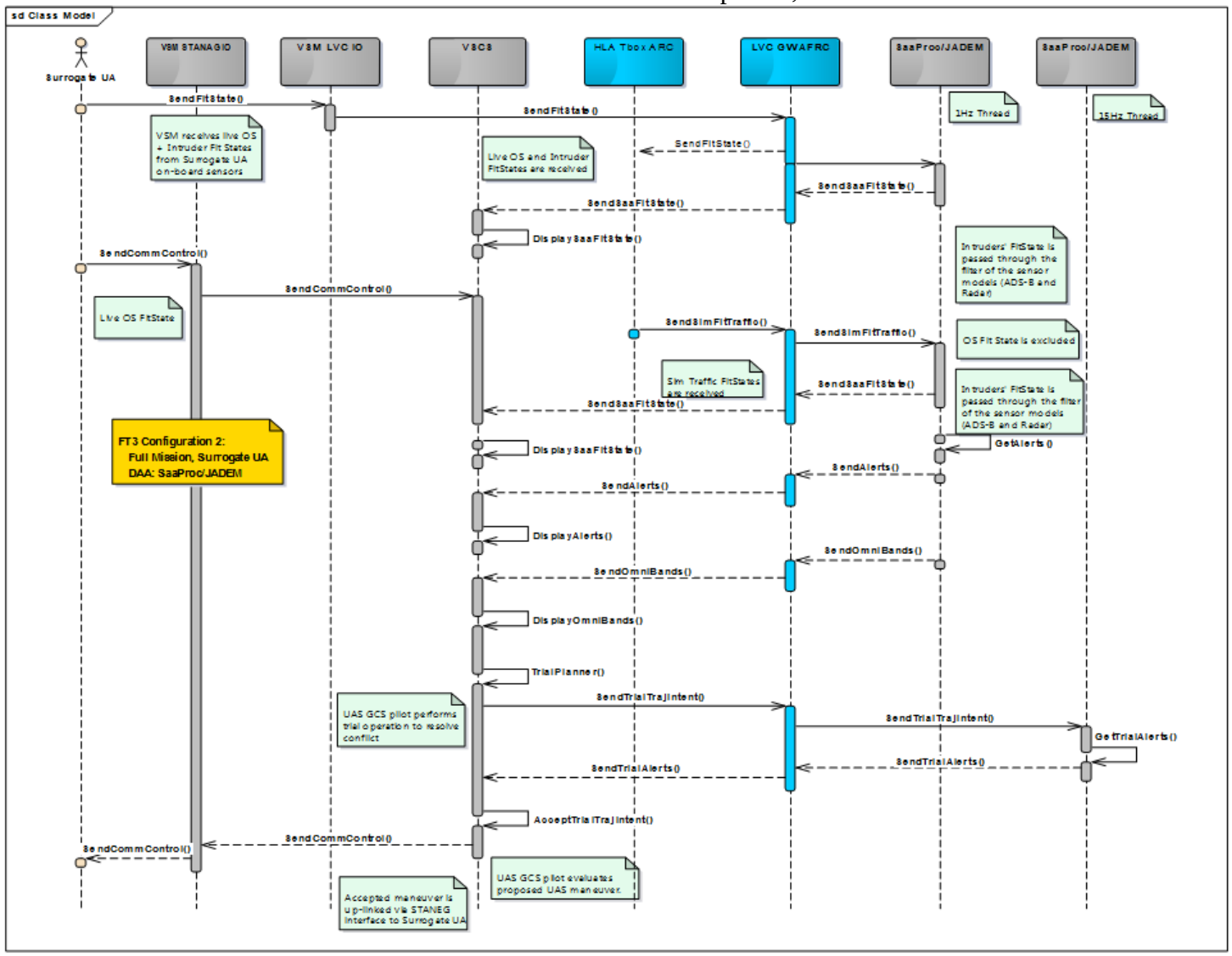

Figure 7e. UML Sequence Diagram: Conflict Detection Using SaaProc/JADEM for Configuration 2

A list of the sequence of events for Configuration 2 having SaaProc/JADEM in the loop is shown below:

1. The surveillance system of the live Surrogate UAV aircraft will send the state data update of the ownship (Surrogate UAV) and intruders in the range of the on-board sensors to the CNPC Ground Station and onto the VSM LVC I/O subsystem.

2. The VSM LVC I/O will receive the state data messages of the live traffic. Subsequently, LVC I/O will map the corresponding state data fields into the MsgFlightState message data structure for the ownship and intruders.

3. LVC I/O will send MsgFlightState messages for the live traffic to LVC Gateway.

4. Repeat Configuration 1A (when using SaaProc/JADEM) Steps 4-19, including the traffic sensor range filtering, Alerting, Omni Bands, Trial planning, and handling of the STANAG message.

5. Concurrently and asynchronously, the simulated virtual traffic (in form of MsgFlightState) by generated in the ARC SimLabs is published to LVC Gateway.

6. LVC Gateway receives state data for the live traffic, including the ownship, in the form of MsgFlightState, and publishes them to SaaProc/JADEM (labeled as 1Hz Thread in the diagram) at the rate of 1Hz.

7. The process repeats from step 1 when the S-3 surveillance system issues a new state update.

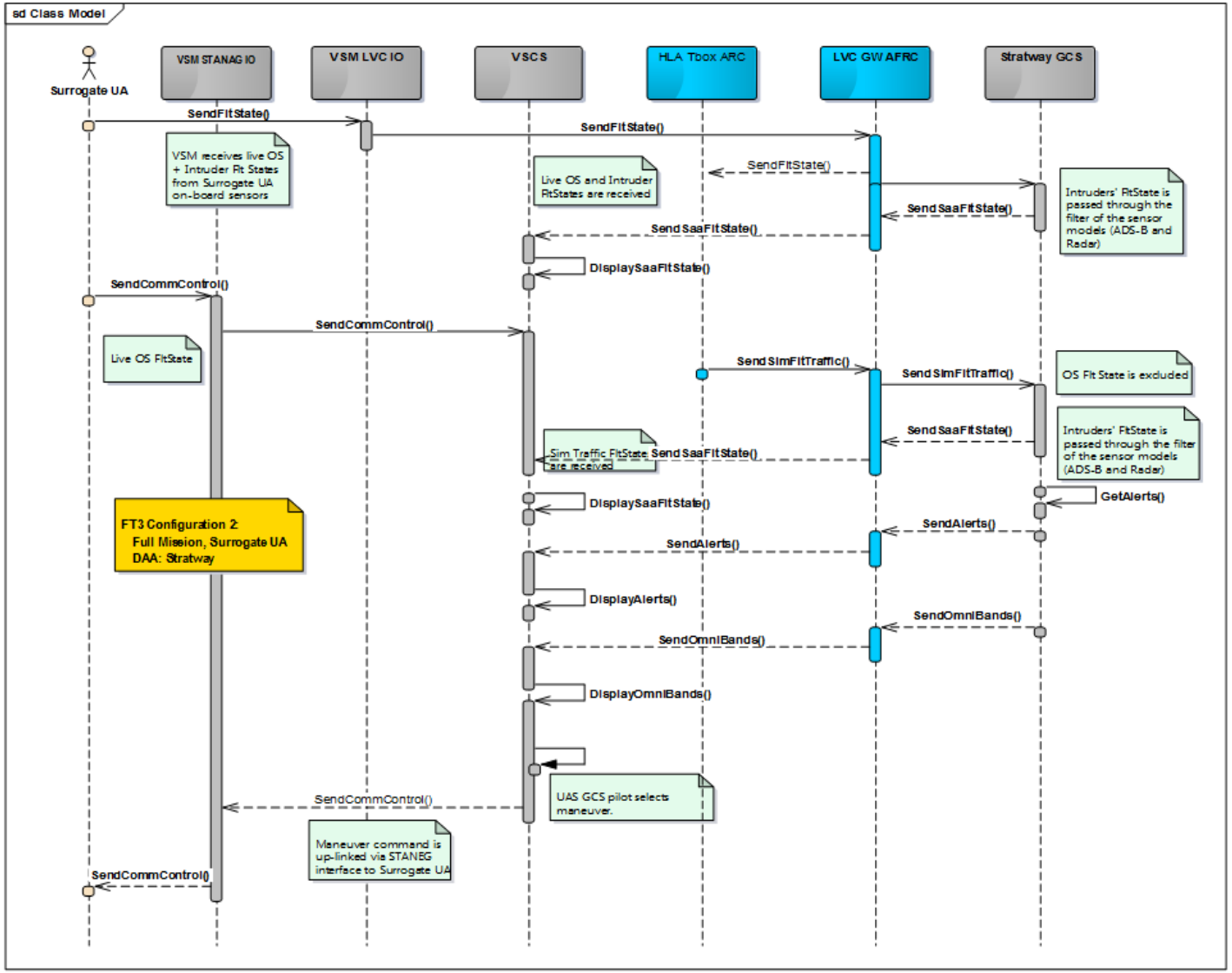

Figure 7f. UML Sequence Diagram: Conflict Detection using Stratway+ for Configuration 2

A list of the sequence of events for Configuration 2 having Stratway+ GCS in the loop is shown below:

1. The surveillance system of the live Surrogate UAV aircraft will send the state data update of the ownship (Surrogate UAV) and intruders in the range of the on-board sensors to the CNPC Ground Station and onto the VSM LVC I/O subsystem.

2. The VSM LVC I/O will receive the state data messages of the live traffic. Subsequently, LVC I/O will map the corresponding state data fields into the MsgFlightState message data structure for the ownship and intruders.

3. The LVC I/O will send MsgFlightState messages for the live traffic to LVC Gateway.

4. Repeat Configuration 1A (when using Stratway+ GCS) Steps 4-14, including the traffic sensor range filtering, Stratway+ Bands, Alerting, and handling of the STANAG message.

5. Concurrently and asynchronously, the simulated virtual traffic (in form of MsgFlightState) by generated in the ARC SimLabs is published to LVC Gateway.

6. LVC Gateway receives state data for the live traffic, including the ownship, in the form of MsgFlightState, and publishes them to SaaProc/JADEM (labeled as 1Hz Thread in the diagram) at the rate of 1Hz.

7. Next sequence event steps are identical to the steps from the step 5 through 14 including the traffic sensor range filtering, Omni Bands, Trial planning and handling of STANAG 4586.

8. The process repeats from step 1 when the surveillance system issues a new state update.

6.0 LVC Gateway Data Logger Design

The LVC Gateway Data Logger (GDL) is designed as a client to the LVC Gateway using TCP/IP client socket interface protocol. The GDL interface conforms to the LVC Gateway ICD. The handshake message will specify that the GDL will subscribe to and record all messages transmitted by the LVC Gateway (i.e., Flight Plan, Flight State, Flight Trajectory Intent, SAA, Saa Omni Band, Stratway+ Bands, Trial Trajectory Intent and Aircraft Delete) in the recording mode, while it will publish all messages when in the playback mode. Data messages of every type will be recorded in an ASCII comma-delimited CSV data file. The UML class high-level diagram of the LVC Gateway Data Logger design is shown in Figure 8.

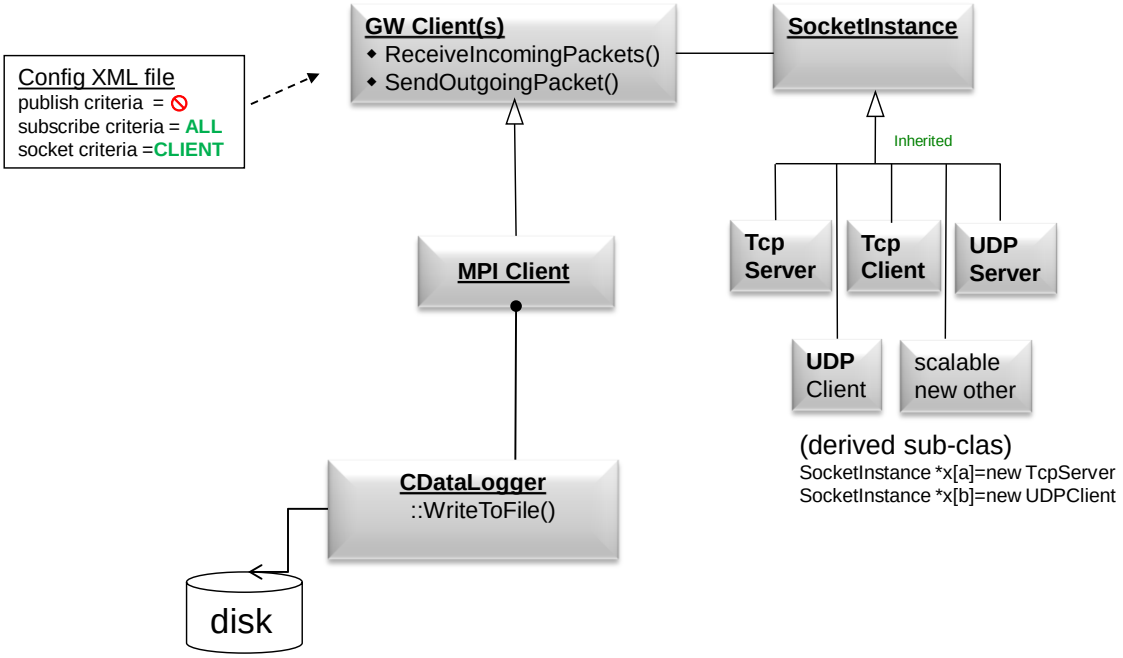

Figure 8. UML Class Diagram of LVC Gateway Data Logger

7.0 LVC Gateway Toolbox Design

An HLA Toolbox is one of the core LVC components that enables fast and cost-effective integration of participants into the distributed simulation test environment. HLA Toolboxes provide a developer with a standard interface that encapsulates Run Time Infrastructure (RTI) to the HLA Federation. Each Toolbox implements the unique requirements associated with a simulation component (or federate) that the Toolbox interfaces with. A communication interface to a federate utilizes standard Unix/Linux. The data that is transmitted to or from the HLA Toolbox is controlled by the HLA Federation Object Model (FOM) Specification.

The FOM is parsed in the HLA Toolbox development process (compile/link) to create a standard data structure that is used in the sockets interface between the HLA Toolbox and its mating simulation component. The FOM also controls the toolbox's access to simulation data and its intent to publish simulation data into the HLA Federation.

7.1 LVC Gateway Toolbox Class Diagram

The UML class diagram of the LVC Gateway Toolbox design is shown in Figure 9. The figure depicts the software architecture representing the relationship between the classes of the LVC Gateway Toolbox.

The LVC Gateway Toolbox will be implemented as a server such that the LVC Gateway will connect to it as a client. The input configuration file will be used to define the Toolbox configuration in terms of the Toolbox IP address, port number, TCP/IP type server socket, the LVC Gateway IP address, the port number, and the IP address of the HLA RTI Exec host.

The LVCMPIManager class is responsible for instantiating the LVCServer class of the MPI type based on the configuration input file. Once instantiated, the LVC Server object will parse the configuration file to obtain required parameters for the instantiation of the LVC Gateway Toolbox components during the initialization. The LVCServer object will setup the TCP/IP server socket interface with the predefined port number using InetPort and InetPortImpol classes. Subsequently, it will define a handshake message that includes messages that will be published and subscribed. This handshake message will be sent to the LVC Gateway, which will register messages that it will receive and publish from the Toolbox.

When the LVC Gateway Toolbox is launched, it will connect with the HLA RTI Exec infrastructure. The Toolbox's periodic execution loop, which executes methods that communicate with the LVC Gateway and the HLA RTI, will be launched to receive flight data updates and transmit them between the two environments.

Two methods are responsible for transmitting flight data between participants connected to HLA and the LVC Gateway. The method *DownloadFromHlaAndSendToFederate( )* receives periodic updates of flight data from HLA objects that are defined by the HLA FOM. The flight data are subsequently mapped to the corresponding MPI data structure types defined by the ICD. After the mapping is completed, the data structure is encoded to the MPI message and sent to the LVC

Gateway utilizing the TCP/IP server/client socket infrastructure.

Inversely, the method *ReceiveFromFederateAndUploadToHla( )* receives periodic updates of flight data from the federate (participant). The flight data are subsequently mapped to the corresponding HLA FOM objects. After the mapping is completed, the HLA object is passed to the RTI which delivers the flight data to the subscribing participants.

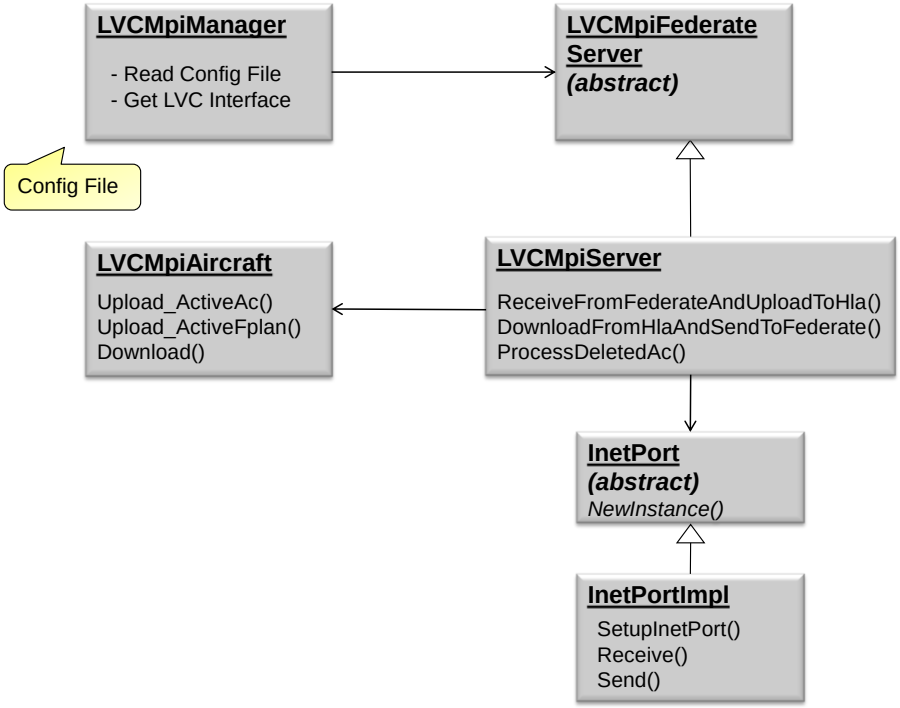

Figure 9. UML Class Diagram of LVC Gateway Toolbox

The Toolbox execution loop also hosts *ProcessDeletedAc()* method that is responsible for removing the HLA objects associated with the targets that are removed from the traffic scenario during the run time.

The low level implementation of these methods is implemented in the LVCMpiAircraft class.

With the exception of a bug fix, the LVC Gateway Toolbox has not been modified from its baseline configuration since the LVC Gateway Toolbox is not required to process SAA and Trial Planning data.

8.0 Sense and Avoid Processor (SaaProc) Design

The SaaProc subsystem is an integral component of the UAS Ground Control Station (GCS) under test, providing a framework for integrating multiple Sense and Avoid (SAA) algorithm and

traffic display combinations. The SaaProc communicates over a socket connection as a client with the LVC Gateway. The internal IO data infrastructure supports the flow of inputs to the SAA algorithm (ownship and intruder flight data), outputs from the SAA algorithm (SAA threats and conflict resolutions) to the LVC Gateway. The SaaProc class diagram is shown in Figure 10. SaaProc supports two conflict resolution functionalities: auto-resolution and trial planning. Threat levels and resolutions produced by SAA during the auto-resolution conflict detection phase as well as the results of trail planning are displayed graphically on VSCS. Sequence diagrams for the six use cases are shown in Figure 7a through 7f from the LVC Gateway perspective, while the sequence diagram in Figure 11 shows the internal sequence infrastructure for the SaaProc subsystem.

Trial Planning is a process in which a UAS operator attempts to resolve conflicts identified by the conflict detection function by manually exploring a new conflict-free trajectory using the trial planning capabilities built into VSCS. The trial planning is activated by selecting the end of the heading vector attached to the ownship on the VSCS display. The trial heading vector will become brighter and the pilot can then rotate the vector around the compass rose. During this process, the trial trajectory intent is dynamically modified and is sent to SaaProc/JADEM at a high data rate for the prediction of trial conflict alerts.

A second altitude trial planning method is also available to the pilot, who can use a graphical altitude strip to select climb/descend trial altitudes. The altitude strip is displayed in the VSCS UI. These trial plan trajectories/ altitudes are sent to the SaaProc via the LVC Gateway at the nominal 15Hz data rate to detect whether the proposed trajectories are conflict free. The resultant conflict threat information corresponding to each trial plan trajectory is sent back to the VSCS. Subsequently, appropriate visual alerts will be displayed in the VSCS display. A multithreaded infrastructure is employed to handle different data rate requirements for data management as well as handling asynchronous events associated with trial planning and JADEM's conflict detection.

An additional alerting capability is provided in the form of Omni Bands. The Omni Bands algorithm is implemented in the JADEM environment together with the alerting and Autoresolver algorithms. The Omni Bands are providing threat intervals for heading and altitude as defined in the LVC ICD.

8.1 SaaProc Class Diagram

The UML class diagram of the SaaProc design is shown in Figure 10. The figure depicts the software architecture representing the relationship between the classes of the SaaProc. Central to the SaaProc operation is the SaaIOController class, which encapsulates the SAA algorithm (SAA) implemented in the SenseAndAvoid class. The SaaIOController directs the flow of data coming in from LVC Gateway and the data going out to LVC Gateway as a result of the SAA computation. The input and output to and from the LVC Gateway are facilitated by the NetworkCom class.

Employing a bridge pattern in the input structure, SaaProc can adapt to pairing up different client input formats via the SaaInput abstract class with different data consumers via the

FlightDataInputAdaptor class. Likewise, the output structure can adapt to pairing up different algorithm output formats via the SaaResultsAdaptor class with different output data protocols via the SaaOutput abstract class. This design provides flexibility for the SaaProc to handle any algorithm, given any type of input data format.

In order to maximize its performance, SaaProc utilizes three additional worker threads running asynchronously, implemented by SaaInput, SaaOutput, SaaTrialPlanController. This allows the input/output/SAA computation to operate independently at a 1Hz nominal rate while the trial planner operates at a 15Hz rate. During the run time, these asynchronous objects communicate with each other via four synchronized objects implemented by the FlightDataLocker, ResultsLocker, TrialPlanDataLocker, and TrialPlanResultLocker.

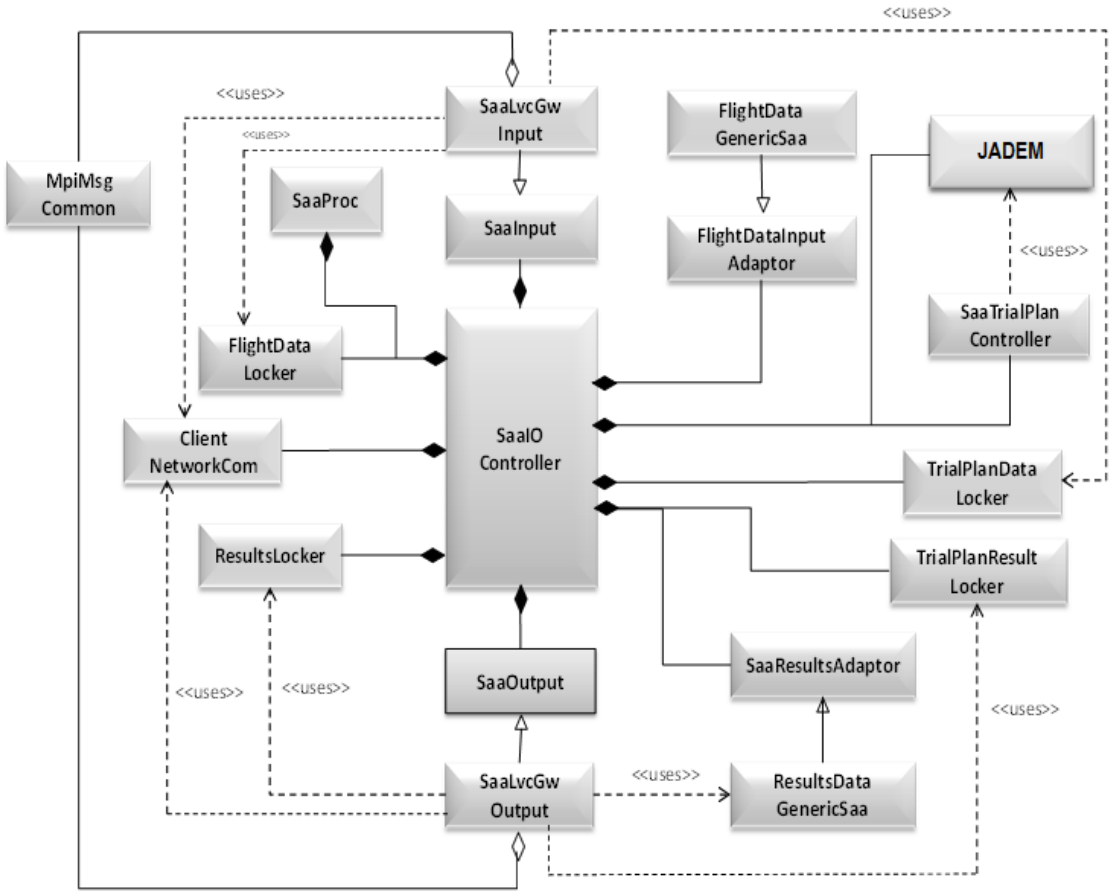

Figure 10. UML Class Diagram of SaaProc

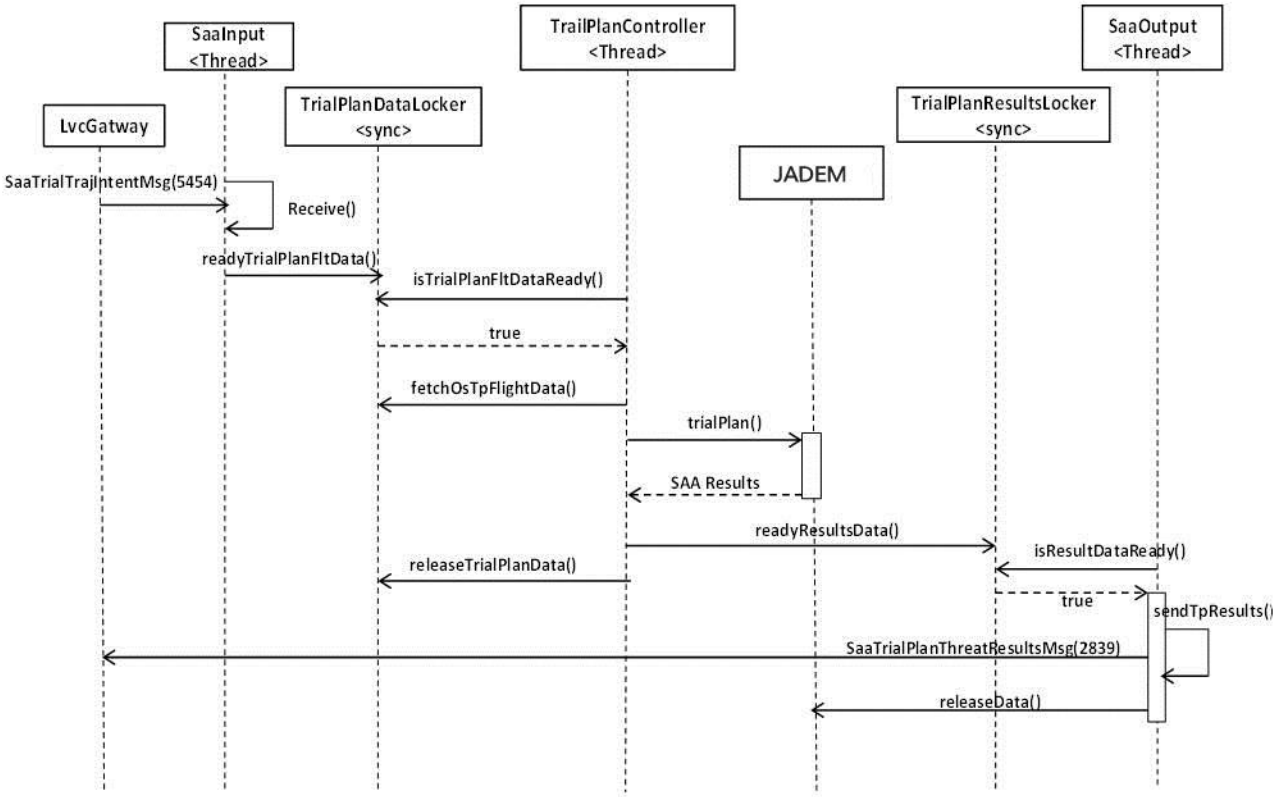

Figure 11. UML Sequence Diagram for Trial Planning in SaaProc

The sequence of steps that take place inside SaaProc when trial planning is invoked is as follows:

1. The SaaInput object detects a trial planning message (SaaTrialTrajIntentMsg) coming in from the LvcGateway. It decodes the message and deposits the trial plan trajectory data along with current flight data into the TrialPlanDataLocker object while simultaneously signaling that there is trial plan data ready and waiting.
2. The TrialPlanController continuously polls the TrialPlanDataLocker to see if any trial plan has come in. When true, it fetches the data by calling TrialPlanDataLocker.fetchOsTpFlightData().
3. The TrialPlanController then submits the trial plan request to JADEM with relevant data by calling the SaaAlgorithm.trialPlan(). Currently three SAA algorithms are implemented in JADEM, i.e. Alert levels, Autoresolver and Omni Band.
4. When trial plan results are computed, TrialPlanController deposits these results into the TrialPlanResultsLocker. Immediately afterward, it sends a release signal to the TrialPlanDataLocker so that the new input can be written into the locker.
5. The SaaOutput object continuously polls the TrialPlanResultsLocker to see if any trial plan results are available. When true, it fetches the data, encodes it into the SaaTrialPlanThreatRsultsMsg, and sends the message out to the LvcGateway via the socket. At this point, the SaaOutput sends a release signal to the TrialPlanResultsLocker so that new results can be written into the locker.
6. The process repeats indefinitely from step one.

Note that VSCS has developed integrated advanced display capabilities in support of Sense and Avoid conflict detection function, including Self Separation Alerting, Resolution maneuvers, Omni Bands and Trial Planning. VSCS will receive all SAA messages published by SaaProc as presented in sections above and will be displayed according to the requirements in the VSCS display. In addition to the regular conflict detection functionality, the infrastructure handling trial planning for heading and altitude are also implemented.

Although VSCS adds a new capability to trial plan using heading, and altitude trial planning than, the generated messages were designed to conform to the trial trajectory intent data structure as defined in the LVC Gateway ICD. Therefore, the infrastructure that is already in place in LVC Gateway and SaaProc is readily available to support heading and altitude trial planning for VSCS. The heading trial planning trajectory intent message is constructed of three waypoints, where the first waypoint is located upstream and the next two way points are located downstream of the ownship position. The VSCS trial planning operation is initiated by a mouse click at the tip of the trial vector on the display. The heading vector can then be rotated around the compass rose centered on the ownship. During this operation, trial trajectory intent messages are published at the rate of 15Hz. Corresponding threat messages will be received from SaaProc.

Altitude trial planning messages are published at a rate of 5Hz for a duration of one second followed by a one second break. A small pop-up window containing a list of buttons with preset altitudes (+/- 500 and +/-1000) centered on the ownship's current altitude is used for altitude trial planning. The trial planning operation is initiated by the push of any of the altitude buttons except the current altitude one. The VSCS trial operation is terminated by the selecting the current (middle) altitude. When a Self Separation threat message is received, the altitude button in the pop-up window changes to yellow, and is changed to red for Collision Avoidance threats.

9.0 Software Requirements Traceability and Implementation

The full set of software requirements is documented in the LVC SWRD-02 Rev C. All requirements in the Rev B (IHITL) version of the SWRD have been implemented and verified.

This SWDD traces implementation to the new or revised LVC SWRD-02 Rev C requirements as follows:

- SWRD Requirements Satisfied by Configuration 1A: LVC-20, GW-15, SWGW-36, 37, 38, 39, 40, 41, 42, 43, SWGW-44, 45

- SWRD Requirements Satisfied by Configuration 1B: LVC-20, GW-15, SWGW-36, 37, 38, 39, 40, 41, 42, 43, SWGW-44, 45

- SWRD Requirements Satisfied by Configuration 2: LVC-20, GW-15, SWGW-36, 37, 38, 39, 40, 41, 42, 43, SWGW-44, 45

Note: SWRD requirements can be traced to the system level requirements in the LVC SRD-01 Rev C which traces to the LVC Objectives and Requirements Document, IT&E ORD-01 Rev A.

Appendix A – Acronyms

ADS-B	Automatic Dependent Surveillance-Broadcast
AFSRB	Airworthiness and Flight Safety Review Board
ATC	Air Traffic Control
ARC	Ames Research Center
CCR	Configuration Change Request
CDR	Critical Design Review
CI	Configuration Item
CM	Configuration Management
CMP	Configuration Management Plan
COTS	Commercial Off The Shelf
CSD	Cockpit Situation Display
CSCI	Computer Software Configuration Item
CVS	Concurrent Versions System
DCP	Dryden Centerwide Process
AFRC	Armstrong Flight Research Center
DHB	Dryden Handbook
DOC	Document
DOP	Dryden Operational Procedure
DPR	Dryden Procedural Requirement
DR	Discrepancy Report
DSRL	Distributed Simulation Research Laboratory
FAA	Federal Aviation Administration
FMR	Flight Media Release
FRR	Flight Readiness Review
GB	Guidebook
GCS	Ground Control Station
GOTS	Government off-the-shelf
HLA	High Level Architecture
H.S.I.	Human Systems Integration
ICD	Interface Control Document
IRS	Interface Requirements Specification

ISRP	Integrated Systems Research Program
IT	Information Technology
IT&E	Integrated Test and Evaluation
IV&V	Independent Verification and Validation
JADEM	Java Architecture for DAA Extensibility and Modeling
LVC	Live Virtual Constructive Distributed Environment
MACS	Multi-Aircraft Control System
MOTS	Modifiable off-the-shelf
MPI	Multi-Purpose Interface
NASA	National Aeronautics and Space Administration
NPR	NASA Procedural Requirements
N/R	Not Required
OFP	Operational Flight Program
ORD	Objectives and Requirements Document
PDR	Preliminary Design Review
PCM	Pulse Code Modulation
REV	Revision
RFDP	Research Flight Data Processor
RGCS	Research Ground Control Station
RTCA	Radio Technical Commission for Aeronautics
SAP	Software Assurance Plan
SaaProc	Sense And Avoid Processor
SBU	Sensitive But Unclassified
SDA	Software Design Agent
SDD	Software Design Description
SDP	Software Development Plan
SRD	System Requirements Document
SWRD	Software Requirements Document
SRR	System Requirements Review
STD	Standard
STR	System (or Software) Test Report
SWDF	Software Development Folder

TBD	To Be Determined
TOC	Table of Contents
TRR	Test Readiness Review
UI	User Interface
UML	Unified Modeling Language
V&V	Verification and Validation
V&VTP	Verification and Validation Test Plan
V&VTPR	Verification and Validation Test Procedure
VDD	Version Description Document
XML	Extensible Markup Language