A Virtual Laboratory for Aviation and Airspace Prognostics Research

Chetan Kulkarni* and George Gorospe†
SGT, Inc. NASA Ames Research Center, Moffett Field, CA 94035, USA

Chris Teubert‡
NASA Ames Research Center, Moffett Field, CA 94035, USA

Cuong C. Quach§
NASA Langley Research Center, Hampton, Virginia 23681, USA

Edward Hogge¶
Northrop Grumman Technology Services, NASA Langley Research Center, Hampton, Virginia 23681, USA

Kaveh Darafsheh‖
NASA Langley Research Center, Hampton, Virginia 23681, USA

Abstract

INTEGRATION of Unmanned Aerial Vehicles (UAVs), autonomy, spacecraft, and other aviation technologies, in the airspace is becoming more and more complicated, and will continue to do so in the future. Inclusion of new technology and complexity into the airspace increases the importance and difficulty of safety assurance. Additionally, testing new technologies on complex aviation systems and systems of systems can be challenging, expensive, and at at times unsafe when implementing real life scenarios. The application of prognostics to aviation and airspace management may produce new tools and insight into these problems. Prognostic methodology provides an estimate of the health and risks of a component, vehicle, or airspace and knowledge of how that will change over time. That measure is especially useful in safety determination, mission planning, and maintenance scheduling. In our research, we develop a live, distributed, hardware-in-the-loop Prognostics Virtual Laboratory testbed for aviation and airspace prognostics. The developed testbed will be used to validate prediction algorithms for the real-time safety monitoring of the National Airspace System (NAS) and the prediction of unsafe events.

In our earlier work[1] we discussed the initial Prognostics Virtual Laboratory testbed development work and related results for milestones 1 & 2 . This paper describes the design, development, and testing of the integrated tested which are part of milestone 3, along with our next steps for validation of this work. Through a framework consisting of software/hardware modules and associated interface clients, the distributed testbed enables safe, accurate, and inexpensive experimentation and research into airspace and vehicle prognosis that would not have been possible otherwise. The testbed modules can be used cohesively to construct complex and relevant airspace scenarios for research. Four modules are key to this research: the virtual aircraft module which uses the X-Plane simulator and X-PlaneConnect toolbox, the live aircraft module which connects fielded aircraft using onboard cellular communications devices, the hardware in the loop (HITL) module

*Intelligent Systems Division, Discovery and Systems Health Area, MS 269-3, AIAA Senior Member.
†Intelligent Systems Division, Discovery and Systems Health Area, MS 269-3, AIAA Member
‡Intelligent Systems Division, Discovery and Systems Health Area, MS 269-3, AIAA Member
§Safety Critical Avionics Systems Branch
¶TEAMS2 Contractor, Safety Critical Avionics Systems Branch, AIAA Member
‖Safety Critical Avionics Systems Branch

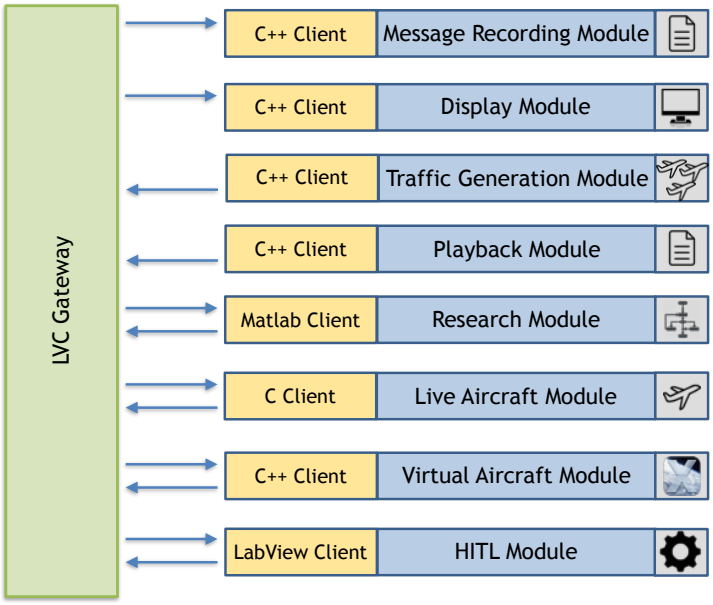

Figure 1. Schematic of the Virtual Lab Testbed

which connects laboratory based bench-top hardware testbeds and the research module which contains diagnostics and prognostics tools for analysis of live air traffic situations and vehicle health conditions. The testbed also features other modules for data recording and playback, information visualization, and air traffic generation. Software reliability, safety, and latency are some of the critical design considerations in development of the testbed.

I. Introduction

Our goal is to integrate airborne assets, virtual aircraft, and hardware-in-the-loop elements within a single virtual development environment to safely study various real-time air traffic scenarios. These real-time scenarios will be used for testing developed NAS safety algorithms safely, quickly and with low personnel or equipment overhead. This unique fusion of virtual aircraft, UAVs, hardware-in-the-loop elements, and prognostic research tools opens up many opportunities for future research into providing a safer and more efficient intelligent airspace for the future.

To enable this we've created a testbed framework designed to be scalable and allowing for plug-and-play addition of field and lab assets from researchers/operators. The framework is primarily designed as a cloud-resource with a web interface, taking advantage of the increased access to eliminate the need for each center to have its own server and avoiding any duplication of resources.

Using this flexible testbed framework we can design experiments piecewise, controlling the flow of data across the network from data sources to sinks. We can control and study interaction between aircraft or operate the the testbed as an observer performing analysis on the airspace as complex, risky, dangerous situations evolve. Furthermore, the testbed framework includes tools for integration, development, research, benchmarking, and testing for rapid feasibility studies of innovative airspace concepts.

The paper is organized as follows. Section II mentions the previous research work and current state of the art. Sections III and IV present the testbed Architecture framework and testbed modules, respectively. The final section V concludes with discussions and future work.

II. Background and Current Research

In our earlier work [2,3] we used a scaled Edge 540 aircraft as seen in Fig.2, to verify and validate testing procedures and prognostic algorithms capable of predicting the expected flying time from a battery set prior to actual flight testing. From these tests, we developed a new level of charge alarm and warning system to

notify the pilot when the estimated remaining flying time falls below a certain threshold.
During this time prognostic algorithms operated on a base station while the vehicle was in flight, these algorithms were further developed and refined offline utilizing recorded flight data. In the interest of generating more flight-like data and testing algorithms pre-flight within the laboratory, a new isolated propulsion system testbed was proposed. With this new capability in mind, planning for the Prognostics Virtual Laboratory involved a desire to perform more complicated flight experiments with a focus on the airspace system and the vehicle state. The virtual lab architecture was designed to allow for three primary types of experiments:

1. Local airspace experimentation and demos including the real-time interaction of live, virtual piloted, and scripted traffic. This would allow for experiments involving complex interactions of multiple assets in a local airspace. By using a combination of piloted (autonomous or not) live aircraft, virtual aircraft, and scripted traffic these interactions could occur in real-time in a single virtual airspace without risking a real collision. Each aircraft would be aware of other aircraft through the sharing of Flight State Messages.

2. Remote prognostics and prognostics decision making based on onboard systems. For this class of experiment, sensor information could be sent in real-time from the aircraft to the research module, which can be hosted anywhere. The research module then would, in real time, conduct prognostics and send the results to a decision maker. The decision maker then updates the waypoints or flight plan and sends the results to the pilot (automated or not).

3. Live-loaded hardware-in-the-loop twin fault injection. In this experiment the HITL testbed would receive telemetry information from a fielded aircraft in flight, then match the throttle levels, and battery current draw, while injecting a non-destructive fault into the system that simulating increased mechanical resistance. A higher current draw results in decreased capability to complete the flight plan. Prognostic decision making methods would then be executed to produce a new flight plan within the capabilities of the faulted system.

To enable these experiments the Prognostics Virtual Laboratory was designed around a modular framework utlizing gateway clients to connect different hardware/software to a network facilitating message publication and subscription. This network, the LVC Gateway, was developed at NASA by the Live Virtual Constructive Distributed Environment (LVC-DE) project. The project created the LVC Gateway to integrate key concepts, technologies and/or procedures in a relevant air traffic environment. [4] The LVC Gateway is a capable of connecting virtual manned, virtual unmanned, and real UAVs in the field from various centers into a single real-time virtual airspace and has been proven to be a powerful tool for training and research.[5,6]

Figure 2. Edge 540 Vehicle

In addition to the LVC Gateway we integrated the NASA Airborne Science Mission Tools Suite (MTS) [a] which is a collection of web-based tools to assist with the planning, operations, and overall management of airborne missions. The main objectives of the MTS are (a) to support tactical decision-making and distributed team situational awareness during a flight; (b) to facilitate team communication and collaboration throughout the mission life-cycle; and (c) to both consume and produce visualization products that can be viewed in conjunction with the real-time position of aircraft and airborne instrument status data. The intent of the system is to encourage more responsive and collaborative measurements between instruments on multiple aircraft, satellites, and on the surface in order to increase the scientific value of the measurements, and improve the efficiency and effectiveness of flight missions. MTS is a product of NASA's Airborne Science Program. The MTS contains a core set of tools with capabilities such as: remote monitoring real-time aircraft location, viewing of current and archived flight tracks, ability to add information overlays from a curated product registry, team communication and collaboration tools etc.

[a] http://mts.nasa.gov

We are working to extend both the LVC-Gateway and MTS to better support the local airspace prognostics, safety, and efficiency research undertaken by groups at the aero centers. Our research work adds the following functionalities to LVC-DE:

- The integration of additional labs and lab resources into the LVC-DE architecture
- The addition of virtual aircraft simulated in X-Plane Flight Simulator
- Extending the LVC Gateway to handle hardware-in-the-loop airspace components
- Setup of a network resource for cloud computation supporting research and communication management
- Built-in support for future rapid feasibility studies and research in local airspace prognostics, control, and safety.

In addition to the experiments the test bed can be used as a research platform where researchers can develop algorithms which then can be tested in the NAS scenario under different conditions. Some of the research tests cases we propose are listed below and discussed later in the paper.

- Airspace experiments with interaction between live and virtual (simulated aircraft)- Conflict avoidance, without risking loss of aircraft.
- Degraded capability/ fault injection with HITL elements without risking aircraft. Simulate conditions on HITL elements.

III. Testbed Architecture

In this section we discuss the design of the Prognositc Virtual Lab testbed and the functionality of each module in the testbed and methods and steps used to integrate, test and validate the modules. The Virtual Lab software was written with the following factors in mind:

Performance Messages had to be passed between components correctly, and with very little latency to be used in HITL component driving or decision making.

Flexibility The infrastructure had to capable of supporting a wide range of planned distributed experiments. Any of the modules must be capable of running from any machine that has network connection.

Ease of Use This infrastructure needs to make experimentation easier and simpler.

Extendability The infrastructure was designed with the future in mind. As time goes on the needs of the group will change, so we have to be able to expand the capabilities of the testbed easily.

Considering these factors and the project requirements, a modular architecture was chosen. In this architecture communication is conducted by a message exchange server that handles the receipt and distributions of messages from a variety of sources. For our project we chose the LVC Gateway, a NASA product that handles a variety of aerospace messages though a publisher/subscriber pattern.[5,7] The modules each add one feature and use a version of the client to communicate with the LVC Gateway. Depending on the experiment, you can activate different modules of the infrastructure. The modules are described further in Table 1. In this table, the source is indicated to show which software products were reused. Four categories of sources are indicated: commercial off the shelf (COTS), government off the shelf (GOTS), mixed, and new. The modules marked as new are the ones that were made from scratch without any significant software reuse.

Many of the individual software components required for the virtual lab already exist and are currently being used by other projects.[2] When this is the case, we have tried to incorporate those software components.

A. Virtual Lab Clients

Each module uses a virtual lab client to communicate with the gateway. Each client handles the formation, receipt, and communication of messages with the LVC Gateway. For this project we developed C, C++, LabVIEW, and Matlab clients. Each client communicates with the LVC Gateway through a TCP connection. The clients are designed so that any number of clients of different languages can be connected to the same gateway sending and receiving messages. This is necessary to support the modular architecture critical to this application.

Table 1. Infrastructure Modules

Module Name	Source	Description
Message Recording Module	GOTS	A message recording system designed for the LVC Gateway. This module records any messages handled by the gateway.
Display Module	Modified GOTS	A module for displaying the current aircraft states, and sensor information in a web-accessible graphical user interface (GUI). This is useful for providing state awareness to researchers. For this, the NASA Airborne Science developed Mission Tools Suite (MTS) was used. Mission Tool Suite is a collection of web-based tools to assist with the planning, operations, and overall management of airborne missions. [b]
Traffic Generation Module	New	Creates multiple passive virtual aircraft of a set pattern.
Playback Module	New	A configurable playback agent capable of reading a number of different playback file formats.
Research Module	New	A module for connecting research algorithms to the LVC gateway using the Generic Software Architecture for Prognostics (GSAP).
Live Aircraft Module	New	A module for connecting UAV 540 aircraft to the virtual lab.
Active Virtual Aircraft Module	COTS/GOTS Mixed	A module for connecting virtual controllable aircraft into the virtual lab.
HITL Module	New	A module for connecting hardware in the loop elements to the gateway. These elements are dynamically loaded based on the live flight information passing through the gateway.

B. LVC Gateway Extension-Additional Messages

The LVC Gateway is capable of managing a number of messages to convey information about an aircraft's state, and intended trajectory. These are very important for enabling airspace experiments. Absent from these messages however is a message to carry data collected from sensors internal to an aircraft, such as that from a thermocouple on a motor, or a voltage sensor on a battery. This information is critical for performing diagnostics and prognostics of aircraft components. Diagnostic and prognostic algorithms use this information to estimate the health state of the system and predict how that will degrade with time.

To carry data from these sensors the LVC Gateway was extended to support the SensorDataPoint message, defined below. This message was designed to be flexible, capable of carrying messages from a wide range of sensor types, and compact to limit bandwidth use. The entire message is 52 Bytes. The unit types enumeration will be expanded to support other units as needed. The added sensor message is defined below as a C++ struct.

```
#define SENSORNAME_LENGTH          24
#define ACID_LENGTH                12

typedef struct MsgSensorDataPoint_t {
    char                m_name[SENSORNAME_LENGTH];
    enum_UnitTypes      m_unit;
    char                m_acid[ACID_LENGTH];
    double              m_value;
    double              m_timeCollected;
} structMsgSensorDataPointType;
```

The sensor data message is consumed by displays and the research module, which uses that to produce prognostic results. The prognostic results are then sent out using the following messages:

```
#define EVENTNAMELENGTH          24

typedef struct MsgProgEvent_t {
    int                 id;
    char                name[EVENTNAMELENGTH];
    double              poe; // Probability of Event
    DistType            toeType;
    double              toe[8]; // Time of Event
    double              predHorizon;  // Horizon for prediction
} structMsgProgEventType

typedef struct MsgProgState_t {
     int                 eventID;
     int                 index;
     DistType            dataDist;
     double              data[8];
} structMsgProgState_t

typedef enum DistType_t : int {
    Gaussian,
    Percentiles,
    LowPercentiles,
    MidPercentiles,
    HighPercentiles
} DistType
```

Here the prognostic results are split into multiple messages, using the nomenclature introduced by Indranil Roychoudhury et. al.[8] For some systems it might just be one event: End of Life (EOL). Other's might have multiple events, such as a battery which could have End of Life (EOL) and End of Discharge (EOD). The prediction of each event is sent in a separate message. The Probability of Event (POE) field represents the probability that the event will occur in the prediction horizon. The Time of Event (TOE) field represents the time that the event will occur, with uncertainty.

The time of event with uncertainty is represented by a vector of doubles, which is interpreted differently depending on the distribution type used. How to interpret each distribution type is described in Table 2. Multiple messages can be sent to represent more than 8 points of uncertainty. For example, sending a low, mid, and high percentiles for toe will define 24 points along the distribution. Additional distribution types will be added in the future.

Table 2. Interpretation by Distribution Type

Distribution Type	Interpretation
Gaussian	Mean: toe[0] Standard Deviation: toe[1]. For example {3.2, 0.1}
Percentiles	Value, Percentile, Value2, Percentile2. For example {3.2, 0.5, 3.1, 0.9,...}
Low/Mid/High Percentiles	Set values for specific percentiles High {0.75, 0.8, 0.85, 0.9, 0.95, 0.99, 0.995, 0.999} Mid {0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7} Low {0.05, 0.1, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3}

The prognostic state message is used to convey related information tied to a specific event, such as state of health (SOH), wear rates, etc. Each of these states is identified by a number (the index) and is correlated to an event using its eventID. The value of the state is represented with uncertainty in the same method as the time of event.

These messages can then be used to inform a decision of some sort, taken by either a human or autonomous agent. The agent consuming this message could conceivably be a pilot, maintenance scheduler, air traffic controller, or any other decision maker.

IV. Testbed Modules

A heterogeneous network of systems with common interfaces sharing a virtual environment creates new possibilities for research. Central to this research is the integration of the live aircraft, virtual aircraft, and HITL modules. These modules make use of the features of other modules to create scientifically interesting scenarios. Experiments can make use of one or many of these.

A. Live Aircraft

Live aircraft are field-assets which can be deployed for experimentation and are connected to the virtual lab using the live aircraft module. This module is run on the embedded computer on the aircraft and communicates to the gateway through a cellular connection using the C Client. There are currently two versions of the Live Aircraft Module in use: a stand alone version and a Core Flight System (cFS) service version. cFS is a platform and project independent reusable software framework and set of reusable software applications [c]. The cFS version of the module integrates into cFS running onboard the aircraft.

For our lab we have a number of Edge 540 UAVs that can run this module. The Edge 540 is a small electric UAV with a wing span of approximately 100" and a weight of approximately 50 lbs. This UAV is a 33% sub-scale version of the Zivko Aeronautics Inc. Edge 540 T tandem seat aerobatic aircraft . This aircraft has been actively used by researchers at NASA LaRC to facilitate the rapid deployment and evaluation of remaining flying time prediction algorithms for electric aircraft since 2010.[9] Remaining flying time prediction algorithms focus on the prediction of battery charge depletion over an e-UAV flight. A lower-bound on the battery state of charge (SOC) that is considered safe for flight is set at 30% in this work. Flying the vehicle with batteries below 30% SOC is considered to be a high-risk mode of operation that violates the vehicles safe operating guidelines. Such violations of operating guidelines are referred to here as a functional failure of the vehicle's mission. [2]

B. Virtual Aircraft

Virtual aircraft are a simulation based source from which aircraft data are generated. Data from virtual aircraft are passed to the gateway to be consumed by other modules. Virtual aircraft enable a number of experiments that are not technically feasible with live aircraft, such as scenarios that might lead to failure. Additionally, experiments using live aircraft require significant effort and time to prepare and run, and are subject to weather. This is not true for virtual aircraft. Virtual aircraft are often used with live aircraft for complex scenarios or for testing experiments before using live aircraft.

There are two types of virtual aircraft: active aircraft which are controlled actively by a user or passive aircraft that follow some set path without knowledge of the other aircraft in the airspace. Active aircraft have the advantage of being able to react to events in the airspace.

1. Active Virtual Aircraft

Active virtual aircraft are simulated using the X-Plane commercial flight simulator. [d] X-Plane provides powerful flight simulation and visuals for simulating aircraft. Pilots can drive the aircraft using a joystick or their mouse. Additional autopilot systems can pilot the aircraft using plugins. The active virtual aircraft module connects with X-Plane using the NASA developed XPlaneConnect (XPC) toolbox [e]. It passes the state of the piloted virtual aircraft to the gateway in the form of FlightState messages and displays the other

[c] https://cfs.gsfc.nasa.gov
[d] http://www.x-plane.com/desktop/home/
[e] https://github.com/nasa/XPlaneConnect

aircraft in the experiment in X-Plane for the pilot to see. X-Plane allows for the simulation of a variety of different aircraft in varying geographies and weather conditions. A virtual lab pilot station (Figure 3) can be used to pilot virtual aircraft

2. Passive Virtual Aircraft

There are two separate modules available for use in the virtual lab which can create passive virtual aircraft: A playback module, and traffic generation module.The playback module creates virtual aircraft based on information contained in a playback file. The playback module is configurable to accept a wide variety of playback file formats. These playback files are useful for running exact scenarios or rerunning flights conducted by live aircraft. This module can also be useful for demonstrations.The traffic generation module is used to create multiple passive virtual aircraft of a set pattern. The generated traffic is often used to add complexity to a scenario by adding traffic for a live or active virtual aircraft to avoid.

In addition to these virtual aircraft modules custom virtual aircraft can be created using the clients.

Figure 3. The Virtual Lab Pilot Station

C. HITL Module

The hardware-in-the-loop module utilizes a LabVIEW-based client to connect laboratory benchtop hardware testbeds to the LVC Gateway. Through this connection, the current testbed, an electric propulsion system testbed, can be easily connected with the LVC Gateway. This electric propulsion system testbed is designed to operate an isolated electric UAV propulsion system and apply realistic flight-like loads on the system as desired. The Edge 540 UAV was used as a reference aircraft from which the electric propulsion system testbed was built. The propulsion elements from the Edge 540 UAV consist of batteries, electronic speed controllers, and a brushless DC motor.

The addition of hardware-in-the-loop elements such as a bench top propulsion system testbed to the virtual laboratory testbed gives researchers easy access to a scientifically relevant portion of the aircraft without the overhead and dangers encountered during actual flight. Additionally, hardware to produce mechanically or programmatically induced faults can be easily integrated without the space or weight limitations of a fuselage. This enables researchers to rapidly prototype and implement new flight hardware or aircraft control software. Finally, the bench top propulsion system testbed has comparatively fewer consequences to propulsion or energy storage system failure than the actual aircraft.

D. Research Module

The research module provides an environment for the development, testing, and profiling of Prognostic Algorithms. It integrates into the Prognostic Virtual Lab, receiving any required data and sending the prognostics results messages. The research module subscribes to sensor data, flight state, and trajectory intent messages, integrating the latest messages into a databus that the algorithms draw on to produce their results. The research module also utilizes configuration files describing the configuration and the systems for each aircraft. These configuration files are used to configure the models to the aircraft flying. The context of the research module can be seen in Figure 4.

For the research module we utilized the previously developed Generic Software Architecture for Prognostics (GSAP). This is an open, extendable domain architecture for applying prognostics. It makes applying prognostics easier by implementing many of the common elements across prognostic applications. The standard interface enables reuse of prognostic algorithms and models across systems using the GSAP framework. The architecture of the framework can be seen in Figure 5.

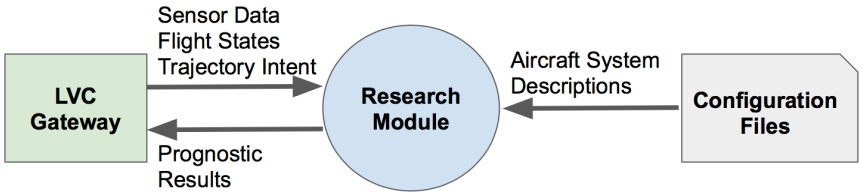

Figure 4. Research Module Context

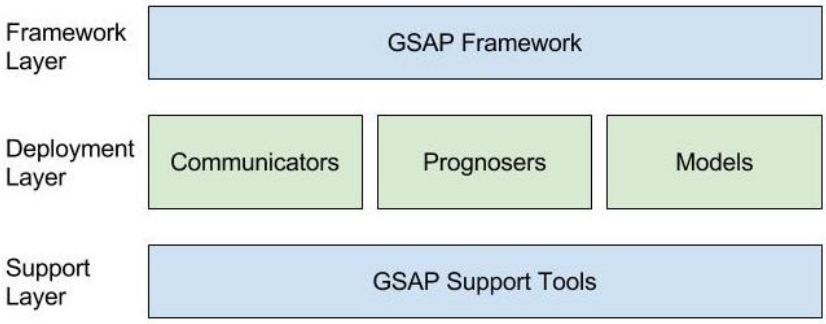

Figure 5. GSAP Architecture

The GSAP framework is used through the creation of communicators, prognosers, or models (the deployment layer). The elements of the deployment layer plugs into the framework and use the tools of the support layer. These elements are described further below:

Communicators Communicators are used to communicate data with the outside world. These function as interfaces with various data sources and sinks. Some examples could be a playback agent that reads from a file, a GUI for displaying prognostic results, an automated report generator, or a client that connects into a network messaging system (for example: Supervisory Control and Data Acquisition (SCADA)). These systems can receive data which will be used by prognosers or communicate the results with operators.

Prognosers This is the core of the GSAP system. Inside the prognosers is the core logic for performing prognostics. A new prognoser is created to support a new method for performing prognostics. Many prognostics systems follow a common model-based structure. Those systems do not require the creation of a new prognoser, only the creation of a new model that will be used by the model-based Prognoser.

Models Models are a method of representing the behavior of a component. A common way of performing prognostics is using a model that describes both the healthy and damaged behavior of the components. The model-based Prognoser uses the models to perform prognostics.

Each of these components is configured through the use of configuration files. This allows for the GSAP deployment to be configured to a new configuration or system without any software changes. The Research Module was tested with OS X 10.11, Red Hat Linux, Debian Linux, and Windows 7.

The developed algorithms for prognostics and decision making are ported to the GSAP framework through the Deployment layer. For this work, we are testing our battery prognostic algorithm along with our developed electro-chemistry battery model. Decision Making for prognostics can be defined as a methodology for selecting and performing certain actions taken based on prediction outputs informing future health state of the system.

In this case the prognostics output is end of discharge (EOD) threshold of the battery voltage level. Based on EOD output the decision-making algorithm performs actions/feedback related to safe operation and return of the Edge 540 UAV to base. Under the current development work the feedback for decision making is taken either by a ground pilot or an onboard autopilot board.

V. Discussion and Comments

In this milestone we successfully integrated our developed research module with GSAP. A Virtual Lab Communicator was developed and tested with GSAP which allows the communicator to subscribe to messages required for prognostics. Further the module is capable of publishing the prognostics output results on the bus which could be used upstream for display and decision making efforts.

Our next research goal is working towards Milestone 4, where the HITL testbed will be operating in the lab which is then synchronously loaded with similar conditions onboard the Edge 540 UAV in flight. This test will utilize the new sensor data message set and the HITL motor loading assembly within the electric propulsion system testbed. Beyond milestone 3, the team is working towards completing the research module, and creating a module to generate out-the-window visuals for live aircraft using X-Plane.

The end goal of this research work is to create an experimental control GUI for enabling/disabling modules, and exploring experiments involving autonomous, health-aware, aircraft decision making.

Acknowledgments

The team would like to thank James Murphy, Neil Otto, Srboljub Jovic, and the rest of the LVC Gateway team for their help getting us started with their LVC Gateway software, and instruction on how to add additional message sets. Additionally, the team would like to thank Aaron Duley and the MTS Team for the generous work they put into deploying their MTS display for our project.

References

[1]Kulkarni, C., Teubert, C., Gorospe, G., Bruggett, D., Quach, C. C., and Hogge, E., "Design, Development, and Testing of a UAV Hardware-in-the-Loop testbed for Aviation and Airspace Prognostics Research," AIAA Aviation, June 2016.

[2]Bole, B., Daigle, M., and Gorospe, G., "Online Prediction of Battery Discharge and Estimation of Parasitic Loads for an Electric Aircraft," *Second European Conference of the Prognostics and Health Management Society 2014*, July 2014, pp. 23–32.

[3]Bole, B., Teubert, C., Quach, C., Hogge, E., Vazquez, S., Goebel, K., and Vachtsevanos, G., "SIL/HIL replication of electric aircraft powertrain dynamics and inner-loop control for V&V of system health management routines," *Annual Conference of the Prognostics and Health Management Society*, 2013.

[4]NASA, Ames Research Center, Moffett Field, CA 94035, *Live Virtual Constructive Distributed Environment (LVC) LVC Gateway, Gateway Toolbox*.

[5]Murphy, J. R., Jovic, S., and Otto, N. M., "Message Latency Characterization of a Distributed Live, Virtual, Constructive Simulation Environment," *AIAA Infortech @ Aerospace Conference*, 2015.

[6]NASA, Armstrong Flight Reesarch Center, Edwards, CA 93523, *Interface Control Document (ICD) for the LVC Gateway (Rev B)*.

[7]Murphy, J. R. and Hoang, T., "UAS Integration in the NAS Project: Integrated Test and LVC Infrastructure," *NASA Technical Reports*, 2015.

[8]Roychoudhury, I., Spirkovska, L., Daigle, M., Balaban, E., Sankararaman, S., Kulkarni, C., Poll, S., and Goebel, K., "Predicting Real-Time Safety of the National Airspace System," *AIAA Infortech @ Aerospace Conference*, January 2016.

[9]Hogge, E., Bole, B., Vazquez, S., Celaya, J., Strom, T., Hill, B., Smalling, K., and Quach, C. C., "Verification of a Remaining Flying Time Prediction System for Small Electric Aircraft," *Annual Conference of the Prognostics and Health Management Society 2015*, 2015, pp. 1–10.