An Extensible Architecture for Avionics Sensor Health Assessment Using Data Distribution Service

Sumant Tambe[1], Fernando Garcia Aranda[2], Joe Schlesselman[3]
Real-Time Innovations, Inc. Sunnyvale, CA, 94089

Avionics Sensor Health Assessment is a sub-discipline of Integrated Vehicle Health Management (IVHM), which relates to the collection of sensor data, distributing it to diagnostics/prognostics algorithms, detecting run-time anomalies, and scheduling maintenance procedures. Real-time availability of the sensor health diagnostics for aircraft (manned or unmanned) subsystems allows pilots and operators to improve operational decisions. Therefore, avionics sensor health assessments are used extensively in the mil-aero domain. As avionics platforms consist of a variety of hardware and software components, standards such as Open System Architecture for Condition-Based Maintenance (OSA-CBM) have emerged to facilitate integration and interoperability. However, OSA-CBM is a platform-independent standard that provides little guidance for avionics sensor health monitoring, which requires onboard health assessment of airborne sensors in real-time. In this paper, we present a distributed architecture for avionics sensor health assessment using the Data Distribution Service (DDS), an Object Management Group (OMG) standard for developing loosely coupled high-performance real-time distributed systems. We use the data-centric publish/subscribe model supported by DDS for data acquisition, distribution, health monitoring, and presentation of diagnostics. We developed a normalized data model for exchanging the sensor and diagnostics information in a global data space in the system. Moreover, Extensible and Dynamic Topic Types (XTypes) specification allows incremental evolution of any subset of system components without disrupting the overall health monitoring system. We believe, the DDS standard and in particular RTI Connext DDS, is a viable technology for implementing OSA-CBM for avionics systems due to its real-time characteristics and extremely low resource requirements. RTI Connext DDS is being used in other major avionics programs, such as FACE™ and UCS. We evaluated our approach to sensor health assessment in a *hardware-in-the-loop* simulation of an Inertial Measurement Unit (IMU) onboard a simulated General Atomics MQ-9 Reaper UAV. Our proof-of-concept effectively demonstrates real-time health monitoring of avionics sensors using a Bayesian Network –based analysis running on an extremely low-power and lightweight processing unit.

Nomenclature

DDS = Data Distribution Service
XTypes = Extensible and Dynamic Topic Types
OMG = Object Management Group
IMU = Inertial Measurement Unit
IVHM = Integrated Vehicle Health Management
CBM = Condition-Based Maintenance

---

[1] Senior Software Research Engineer, Real-Time Innovations, Inc. Sunnyvale, CA.

[2] Software Engineer, Real-Time Innovations, Inc. Sunnyvale, CA.

[3] Director of Market Development (A&D), Real-Time Innovations, Inc. Sunnyvale, CA.

Acknowledgement: This research is funded by the Air Force Research Laboratory, WPAFB, Dayton, OH under Small Business Innovation Research (SBIR) contract # FA8650-11-C-1054. Manuscript approved for publication by WPAFB: PA Approval Number: 88ABW-2013-3209.

I. Introduction

Dependability of mission critical systems requires novel techniques for fault detection, isolation, and system recovery. The state-of-the-art systems need to provide a global picture of the systems' health for complete situational awareness to improve operators reaction time. Furthermore, the mission success depends on ensuring healthy operational conditions of integrated sensor and software systems. In spite of rigorous verification and validation (V&V) of the physical sensors and embedded control software, latent bugs and unexpected environmental conditions may jeopardize the reliability of the overall system and consequently the mission itself. Therefore, to ensure survival against unpredictable faults, next-generation airborne systems must monitor the health of each component, notify operational anomalies, isolate the affected components to prevent systemic risk and ensure continued operation of the system.

Real-Time Innovations (RTI) is developing solutions for improved situational awareness using sensor health monitors into a Common Cyber Operating Picture (CCOP) for real-time airborne networks, combat management networks, and near-real-time intelligence Surveillance Reconnaissance (ISR) networks. RTI is building an open, extensible infrastructure for collecting, visualizing, analyzing, and reacting to health information in real-time. It leverages open-source and COTS technologies to collect the most important data and distribute it to diagnostics and prognostics agents for analysis.

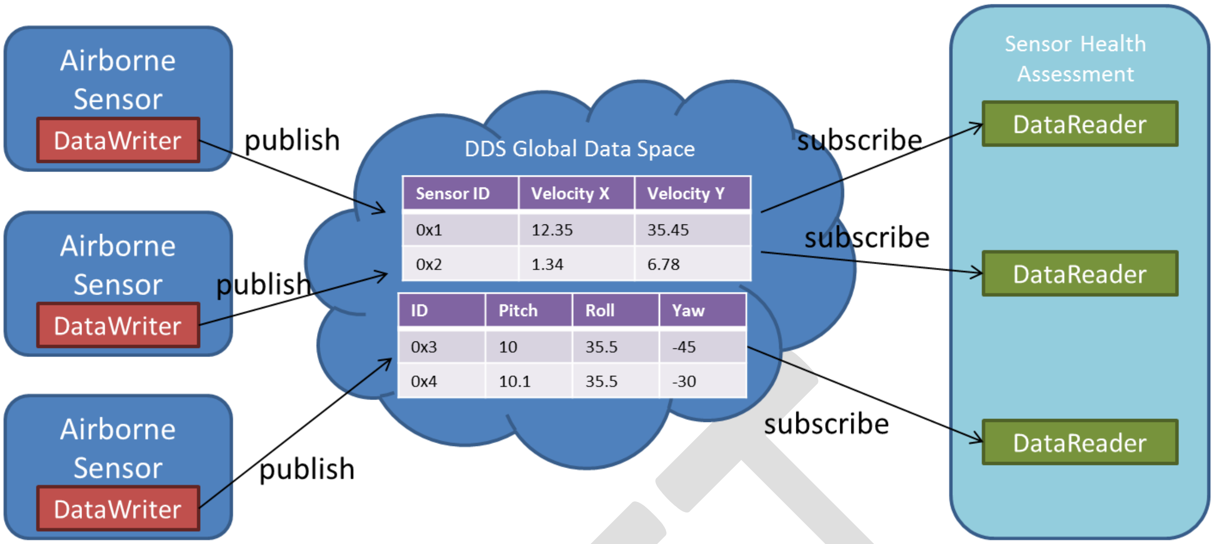
The goal of diagnostics and prognostics is to continuously monitor the health of a system by collecting data from sensors, analyzing it using artificial intelligence (AI) techniques, and notifying any anomalies and/or impending failures to the stakeholders. Integrated Vehicle Health Management (IVHM) is a generalization of diagnostics and prognostics of all the subsystems in a vehicle. In a UAV, for example, major subsystems include navigation, electrical, mechanical, communications, engine, propulsion, and the payload. Using myriad of onboard sensors, it is now possible to estimate the overall health of the UAV or a specific subsystem.

With increasing sophistication of diagnostics and prognostics, however, developing and maintaining IVHM systems have become more expensive. With the shrinking defense budgets and the drive to improve acquisition flexibility, different vendors build different subsystems in the aircraft and they are assembled on the airframe by the integrators. The COTS components must interoperate with each other as well as the overall IVHM system for effective prognostics and health management (PHM). The architecture must not couple the sensors with the PHM system because more than one type of PHM system could be operational simultaneously and the sensors and/or the PHM system may evolve independently as the providers of those systems may be different. When the sensors and/or the PHM system evolve, it is imperative to ensure that existing PHM dataflows are not disturbed and the new sensor capabilities are seamlessly integrated into the overall PHM system.

RTI's framework is based on the Object Management Group (OMG) Data Distribution Service (DDS) standard. DDS enables portable, interoperable, real-time data delivery. Both defense and commercial systems have adopted DDS; its network-centric architecture is proven in many mission-critical applications, ranging from combat-management systems to unmanned vehicles to high-speed trading floors to air-traffic control. DDS provides the ability to connect disparate, heterogeneous systems, control quality of service, and support highly dynamic networks.

II. Data-Centric Solution Architecture

To address these challenges we have developed a distributed architecture for integrating airborne sensors and the PHM modules using Data Distribution Service (DDS)[1]. DDS provides *data-centric* public/subscribe paradigm for system integration. Data-centric integration is different from conventional *message-centric* integration in various ways. First, in data-centric model, the integration data model is fully described in the *global data space* (see Figure 1) using type description facilities and the model is discoverable and evolvable. The type description facility, standardized as the XTypes[2] specification, allows composition, nesting, and inheritance of structures to capture the relationship of types. It specifies rules (a structural type system) that govern compatibility evolved (version N) types with their predecessors (i.e., version 1 to N-1). Second, data-centric model distinguishes between messages and data-objects such that data-objects are uniquely identified across the data-space (much like a row in a database table) and any state updates to the data-object (a.k.a. changes in attribute values) are distributed as messages. Third, data-centric model provides built-in capabilities to notify creation and destruction of data-objects, which would require manually crafted messages in the message-centric world.



Sensor ID	Velocity X	Velocity Y
0x1	12.35	35.45
0x2	1.34	6.78

ID	Pitch	Roll	Yaw
0x3	10	35.5	-45
0x4	10.1	35.5	-30

Figure 1. DDS Global Data Space. *Sensors publish data and the sensor health assessment system performs the diagnostics and prognostics.*

The global data space provides the logical connectivity between DataWriter and DataReaders. The extensibility of our architecture is primarily due to the use of Extensible and Dynamic Topic Types (X-Types[2]) standard, which provides expressive ways to describe the data model in the global data space and the relationship between different types. Data types may be related by aggregation, inheritance, and structural assignability. If the sensor evolves and provides additional information than the previous version, the analysis component need not be rewritten or restarted. DDS will ensure that the right subset of data is available at the consumer cite without disrupting the existing dataflows coming from the old sensors. Likewise, if the analysis component has new capabilities and can process new sensor attributes, the old sensors need not be aware of the new capability.

III. Overview of Data Distribution Service (DDS)

Data Distribution Service (DDS) is the first open international middleware standard addressing publish-subscribe communications for distributed, real-time and embedded systems. The DDS API and several other satellite technical specifications, such as DDS Wire Interoperability specification[7], Extensible and Dynamic Topic Types (X-Types) specification[2], modern language bindings for C++[8] and Java[9] are standardized by the Object Management Group (OMG)—an internationally recognized standards body most widely known for UML and CORBA.

The core DDS specification defines a data-centric publish-subscribe architecture for connecting anonymous information providers with information consumers. DDS promotes loose coupling between system components. The information consumers and providers are decoupled with respect to time (*i.e.*, they may not be present at the same time), space (*i.e.*, they may be anywhere), flow (*i.e.*, information providers must offer equivalent or better quality-of-service (QoS) than required by the consumers), behavior (*i.e.*, business logic independent), platforms, and programming languages. A data provider publishes typed data-flows, identified by names called “topics”. The coupling is expressed only in terms of topic name, datatype schema, and the required and offered QoS attributes of consumers and producers respectively. DDS allows fine control over data delivery by means of standard QoS policies, such as durability, reliability, history, deadline, time-based filtering, liveliness, transport priority, resource limits, and more.

In DDS, the producing and consuming entities communicate through pre-established *topics*. The topics are *typed* conduits of the information and can have associated quality-of-service (QoS) attributes. DDS supports a large variety of QoS attributes such as reliability, history, durability, resource-limits, deadline, ownership, lifespan to fine tune the delivery model of the data from the sensors to the analysis components. In DDS, *DataWriter* is the producing entity whereas *DataReader* is the consuming entity. DataReaders and DataWriters are coupled only terms of the topic, topic type, and the QoS.

Many DoD programs have embraced DDS, including SOSCOE, and Prime contractors working on Open Architecture initiatives. Users include virtually all major US Prime contractors, US defense research laboratories, and many commercial telecommunications, transportation, and financial companies.

A. Overview/Definitions of Select DDS Concepts Used in Our Solution

1. Data-Centric Architecture

DDS facilitates data-centric architecture where applications share a *global data space* governed by schemas specified using the XTypes[2] standard. Each point-to-point dataflow is described using a structured datatype (*e.g.,* an Interface Definition Language *struct*). The datatype could be keyed on one or more fields. Each key identifies an *instance* (similar to a primary key in a database table) and DDS provides mechanisms to control the lifecycle of instances. Instance lifecycle supports CRUD (create, read, update, delete) operations. Complex delivery models can be associated with dataflows by simply configuring the topic QoS.

2. DataWriter and DataReader

DataWriter and DataReader are end-points applications use to write and read typed data messages (samples) from the global data space. DDS ensures that the end-points are compatible with respect to the topic name, datatype, and the QoS. Creating a DataReader with a known topic and datatype implicitly crates a *subscription,* which may or may not match with a DataWriter depending upon the QoS.

3. Data Caching

DDS is not just a messaging middleware, although it can be configured to behave like that. DDS, DataReader in particular, provides caching of samples and different APIs to traverse the cached data samples so that applications need not make copies. DDS distinguishes between read and take, where read keeps the data in middleware cache until it is removed by either calling take or overwritten by subsequent samples. Resource limits QoS prevents middleware caches from growing out of bounds. The DataReader cache can be queried using specific instances as well as iterate over all the instances observed by the system. Finally, query conditions provide a powerful mechanism to write SQL-like expression on the datatype members and retrieve samples that satisfy the predicate.

4. Content-Filtered Topics

It specifies a refined subscription that filters samples that do not match an application-specified predicate. The predicate is a string encoded SQL-like expression based on the fields of the datatype. The query expression and the parameters may change dynamically. Filtering of samples could take place before publication or upon reception.

5. DDS Quality-of-Service

The DDS standard supports 22 different QoS policies to control various aspects of data delivery including reliability, persistence, deadline, resource usage, fault-tolerance, and more. We describe the significance of QoS policies used in our solution below:

Reliability

It controls the reliability of the dataflow between each pair of DataWriters and DataReaders. BEST_EFFORT and RELIABLE are two possible alternatives. BEST_EFFORT reliability does not use any cpu/memory resource to ensure delivery of samples. RELIABLE, on the other hand, uses an ack/nack based protocol to provide a spectrum of reliability guarantees from *strict* (fully reliable) to BEST_EFFORT. The reliability can be tunes using the History QoS policy

History

This QoS policy specifies how much data must be stored by DDS a DataWriter or DataReader. It controls whether DDS should deliver only the most recent value (i.e., history depth=1), attempt to deliver all intermediate values (i.e., history depth=unlimited), or do something in between.

Time-based Filter

High rate dataflows can be throttled using the time-based filter QoS policy. A *minimum separation* configuration parameter specifies the minimum period between two successive receptions of data samples. When configured, the DataReader receives only a subset of data.

Deadline

This QoS policy specifies the maximum separation (deadline) between two successive updates for each instance. When specified on a DataWriter, the application updates each instance at least once every *period*. When specified on a DataReader, the middleware will notify the application (via callback) if the DataWriter breaks the deadline QoS contract. Note that DDS makes no effort to meet the deadline it only notifies if a deadline is missed.

Resource Limits

Controls the memory used to cache samples. Maximum samples-per-instance as well as maximum number of instances can also be specified.

IV. Background: RTI Connext DDS in Defense Systems

Real-Time Innovations (RTI), Inc. is the market-leading DDS vendor. RTI Connext™ DDS[10] is available in both commercial and free Open Community Source (OCS) editions. RTI Connext DDS is deployed in myriad of operational defense systems in Navy, Air Force, and Army. RTI DDS is a Technology Readiness Level (TRL) 9 technology and has been operational for several years in military systems including Aegis, DDG 1000, SSDS, LPD, LCS, CEC, E-2, AWACS, B-1, B-52, Army FCS, and JLENS. Below, we present select projects using RTI Connext DDS.

A. Future Airborne Capability Environment (FACE™)

FACE™ [6] responds to the Department of Defense (DoD) aviation community's need to reduce acquisition costs. FACE is a standard for a software computing environment that promotes interoperability of software product lines across different air platforms. FACE is managed by The Open Group, which includes leading U.S. industry suppliers, customers and users. The goal of FACE is to reduce software development and integration costs and reduce time to field new avionics capabilities. FACE establishes a common computing software infrastructure supporting portable, capability-specific software components across DoD avionics systems. RTI is an active contributor to the FACE Consortium, working to reduce the time and cost to develop, integrate and launch avionics capabilities for military applications.

As part of our involvement in FACE RTI has developed a reference implementation of the Transport Services Segment (TSS). The TSS is an abstraction layer in the FACE Reference Architecture. It provides a standard interface between portable applications, common services and platform-specific services. This approach ultimately allows portable avionics applications to be affordably integrated with disparate architectures and aviation platforms.

RTI applies its core technology to the FACE TSS. This core technology is built on the Data Distribution Service (DDS) standard developed by the Object Management Group (OMG). A TSS built on RTI Connext DDS eases systems integration and maximizes deployment flexibility.

B. UAS Control Segment (UCS)

The UAS Control Segment (UCS)[5] architecture is a framework representing the software-intensive capabilities of current and emerging UAS programs in the U.S. Army, Navy, and Air Force inventories. The goal is to develop an architecture, based upon Service Oriented Architecture (SOA) principles, that will be adopted by each of the Services as a common basis for acquiring, integrating, and extending the capabilities of the control systems for UAS. RTI Connext middleware provides and/or facilitates the fundamental UCS capabilities such as, interchangeability, integrability, replaceability, extensibility, and interoperability.

V. An Extensible Architecture for Condition Based Maintenance using DDS and OSA-CBM

Figure 2 shows our proposed architecture for combining OSA-CBM and DDS to develop an extensible architecture for Condition-Based Maintenance applications. In our architecture, OSA-CBM compliant applications communicate using DDS instead of other commonly used integration technologies, such as Web Services or raw sockets.

The key benefit of using DDS is that the application integration problem is substantially simplified due to the bus architecture. A DDS application requires only one additional *connection* to the global data space to enable communication with all the existing applications already connected to the bus. The middleware ensures that the applications that use compatible topic name, topic type, and QoS discover each other automatically and begin exchanging data. Point-to-point integration technologies such as Simple Object Access Protocol (SOAP) and sockets on the other hand would require one connection for *each* existing application that it exchanges data with. Therefore, to communicate with N applications, N point-to-point links are necessary as opposed to just one in the case of DDS.

Furthermore, point-to-point communication makes independent evolution of applications extremely difficult. When the data provider application evolves, it cannot communicate with older versions of consumer applications unless the data model differences are explicitly addressed at the application-level. With each new version of the application the problem of data model differences exacerbates, which increases development, testing costs and time to market.

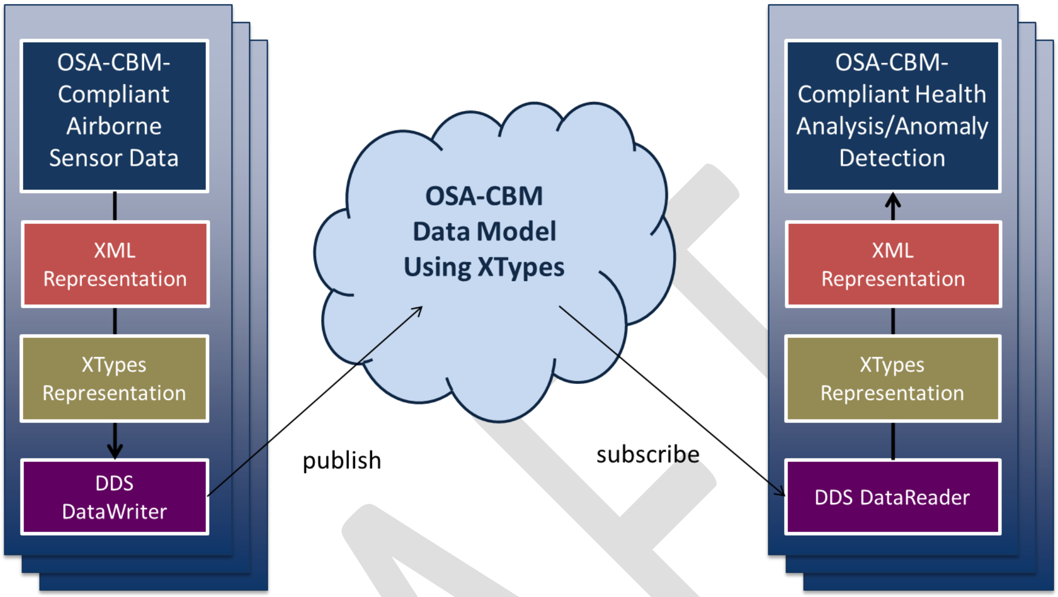


Figure 2. An Extensible Data Bus Architecture for Condition-Based Maintenance using DDS

In our architecture, the producer and consumer applications continue to use the OSA-CBM compliant XML data format for sharing information. The global data space (the cloud in Figure 2) is a full replica of the OSA-CBM data model captured using XTypes. Having the model captured in XTypes allows the model and/or the applications to evolve without disrupting the applications that do not evolve. XTypes provides expressive type model capabilities in IDL, XML, and XSD. XTypes can use the entire OSA-CBM data model defined in the XSD.

XML-encoded CBM data must be converted into the data representation used by the DDS global data space. The conversion is achieved using an adapter that converts the OSA-CBM compliant XML data into its equivalent XTypes data representation (and vice versa). The adaptation is semantically equivalent because both data representations are obtained from the common OSA-CBM semantic data model standardized by MIMOSA. The adapter process is realized using RTI Routing Service, which provides a pluggable framework to develop protocol bridges.

VI. Proof-of-Concept for DDS-Driven Avionics Sensor Health Assessment

We built a built a proof-of-concept (POC) to validate our hypothesis. The primary objective behind the POC is to demonstrate that data-centric publish/subscribe is a powerful paradigm for developing a flexible, extensible architecture for *onboard* sensor health assessment. We use RTI Connext DDS—the leading implementation of the OMG DDS—to demonstrate the POC. The POC is a *hardware-in-the-loop* simulation where we analyze the health of a physical Inertial Measurement Unit (IMU) with respect to the simulated flight data of a quasi-flying General Atomics MQ-9 Reaper UAV[13]. The IMU is mounted on a BeagleBone, which is an ultra-low power processor with ARM Cortex-A8 CPU running at 720 MHz and 256 MB of RAM.

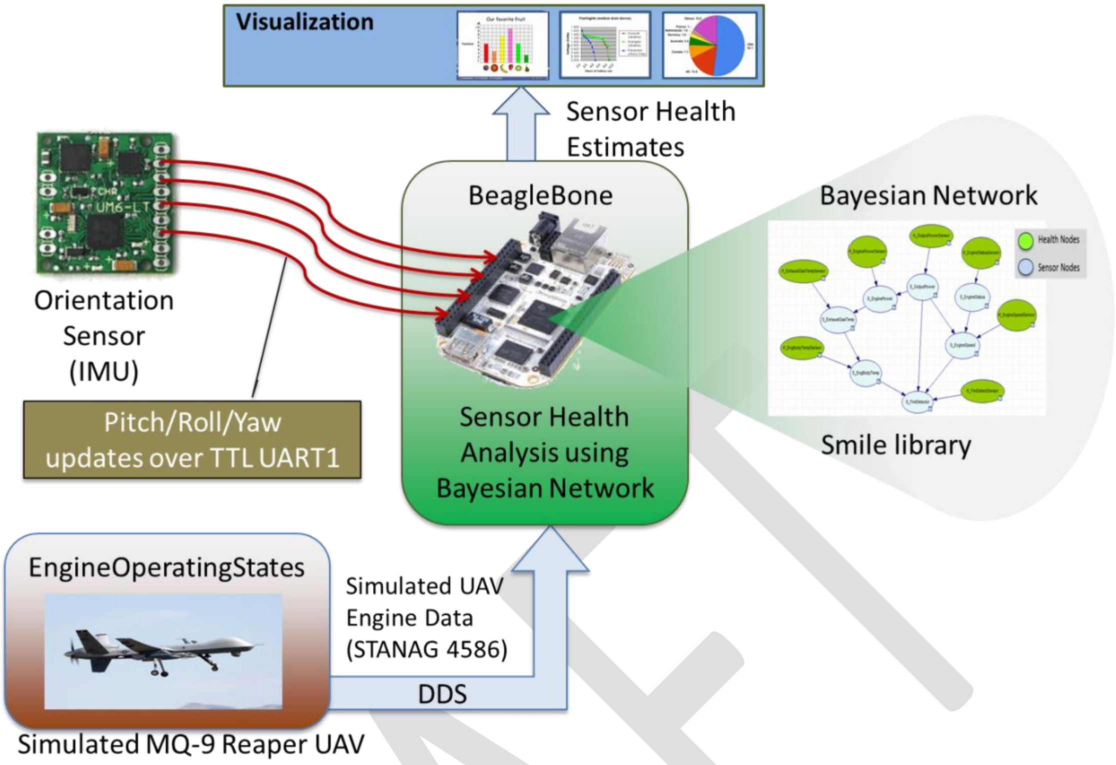


Figure 3. Topology of the proof-of-concept demonstration for sensor health assessment using DDS. *The POC demonstrates real-time onboard sensor health assessment capability using a simulated MQ-9 Reaper UAV and orientation data from a physical IMU deployed on a Beaglebone (700 MHz, 256 MB) running RTI Connext DDS 5.0.*

As show in Figure 3, the data streams from the real IMU and the engine operating states generated by the UAV simulator are distributed using DDS and analyzed on the Beaglebone to detect anomalies. The diagnostics are performed on the Beaglebone, in real-time, using an ARM port of the Smile[3] Bayesian inference library. The Bayesian Belief Network takes into account the engine operating states and the real-time IMU data and produces health estimates for the IMU. The health estimates published back into the global data space for visualization or potentially further downlink dissemination.

A. Simulating a UAV using the X-Plane Simulator

We used X-Plane[12] flight simulator for realistic simulation of the engine operating states. X-Plane provides valuable information for a variety of elements (e.g., engine, mechanical, or inertial subsystems). X-Plane has also built-in support for modeling failures on airborne sensors. It can simulate failures on instruments, engines, flight controls, control cables, and many other systems, which is especially important to determine the accuracy of our health diagnostics system.

We use our prototype implementation of STANAG 4586[11] data model to obtain sensor data from the simulated UAV. STANAG 4586 is a NATO standard that defines architectures, interfaces, communication protocols, and message formats for the interoperability of UAV systems. Message #105 of this standard—Engine Operating States—provides sensor information from different elements of an UAV's engine (e.g., engine speed, engine power, exhaust gas temperature, etc.).

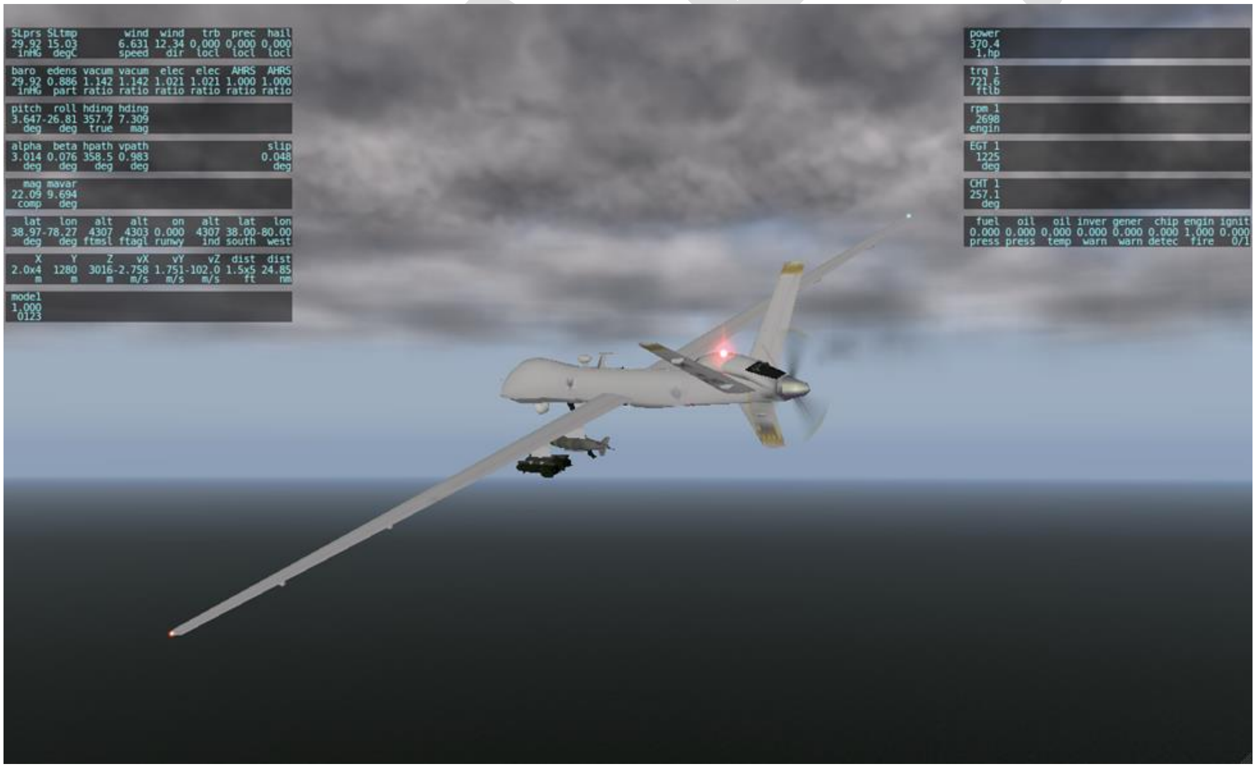
X-Plane distributes sensor information using the User Datagram Protocol (UDP). To integrate X-Plane with the different elements of our diagnostics system, we have developed an extensible application that decodes X-Plane's UDP protocol and transforms the information extracted into DDS topics. XPlane2DDS provides an extensible framework to normalize the information generated by X-Plane, defining a modular architecture in which the publication of information over DDS relies on a set of plugins. In particular, we have created three plugins to interoperate with each of the STANAG 4586 messages (i.e., Inertial States, Air and Ground Relative States, and Engine Operating States). Table 1 shows the normalized data models captured using XTypes.

Table 1: Normalized XTypes data model for STANAG message #105 (Engine Operating States) and message #102 (Air and Ground Relative States)

```
struct EngineOperatingStates {
  double  timestamp;
  string  vehicleId;    //@key
  long    engineStatus;
  double  engineSpeed;
  double  enginePower;
  double  exhaustGasTemperature;
  double  engineBodyTemperature;
  double  outputPower;
  long    fireDetectionSensor;
};
```

```
struct AirGroundRelativeStates {
   double timestamp;
   string vehicleId;     //@key
   double angleOfAttack;
   double angleOfSideship;
   double trueAirspeed;
   double indicatedAirspeed;
   double windSpeed;
   double windDirection;
   double altimeterSetting;
   double barometicAltitude;
   double barometicAltitudeRate;
   double aglAltitude;
};
```

Figure 4 shows a snapshot of a demo flight of an MQ-9 Reaper using X-Plane.



```
SLprs SLtmp         wind  wind   trb  prec  hail
29.92 15.03        6.631 12.34 0.000 0.000 0.000
 inHG  degC        speed   dir  locl  locl  locl

baro  edens vacum vacum  elec  elec  AHRS  AHRS
29.92 0.886 1.142 1.142 1.021 1.021 1.000 1.000
 inHG  part ratio ratio ratio ratio ratio ratio

pitch  roll hding hding
3.647-26.81 357.7 7.309
  deg   deg  true   mag

alpha  beta hpath vpath                     slip
3.014 0.076 358.5 0.983                    0.048
  deg   deg   deg   deg                      deg

  mag mavar
22.09 9.694
 comp   deg

  lat   lon   alt   alt    on   alt   lat   lon
38.97-78.27  4307  4303 0.000  4307 38.00-80.00
  deg   deg ftmsl ftagl runwy   ind south  west

    X     Y     Z    vX    vY    vZ  dist  dist
2.0x4  1280  3016-2.758 1.751-102.0 1.5x5 24.85
    m     m     m   m/s   m/s   m/s    ft    nm

model
1.000
 0123

power
370.4
 1,hp

trq 1
721.6
 ftlb

rpm 1
 2698
engin

EGT 1
 1225
  deg

CHT 1
257.1
  deg

 fuel   oil   oil inver gener  chip engin ignit
0.000 0.000 0.000 0.000 0.000 0.000 1.000 0.000
press press  temp  warn  warn detec  fire   0/1
```

Figure 4: Demo flight of the GA MQ-9 Reaper UAV model using X-Plane

B. Distributing Real-Time Inertial States using DDS

To obtain the Inertial States, we integrated a COTS Inertial Measurement Unit (IMU) on the Beaglebone and used DDS to publish the high-rate (200 Hz) inertial data. The orientation sensor was mounted on the Beaglebone and responded in real-time to human gestures. The *phi* and *theta* attributes described in the Inertial States structure were also used in the Bayesian Network.. The *phi* attribute represents the aircraft pitch whereas the *theta* attribute represents the roll angle. Table 2 shows the XTypes datatype we used to distribute Inertial States.

```
struct InertialStates {
   double timestamp;
   string vehicleId; //@key
   double latitude;
   double longitude;
   double altitude;
   double uSpeed;
   double vSpeed;
   double wSpeed;
   double phi;    // pitch
   double theta;  // roll
   double magneticVariation;
};
```

C. Diagnostics using Bayesian Networks

The purpose of our POC is to analyze the STANAG 4586 Engine Operating States and the Inertial States produced by the IMU sensor and identify anomalies in the IMU sensor. We designed a Bayesian Network to detect incoherent sensor information, providing the probability of being good or bad for each sensor.

Figure 5 shows the Bayesian Network that uses the Engine Operating States and Inertial States (phi and theta only) as evidence. Nodes in light blue—prefixed by "S_"—are sensor nodes, whereas green nodes—prefixed by "H_"—indicate the health of the sensors. The Bayesian Network in Figure 5 has been defined using the Smile and Genie[3] tool-chain developed by the Decision Systems Laboratory (DSL) at the University of Pittsburgh. The DSL has also developed SMILE, a C++ library for the implementation of graphical decision-theoretic methods. Using SMILE to interact with the Bayesian Network, we can extract conclusions out of it.

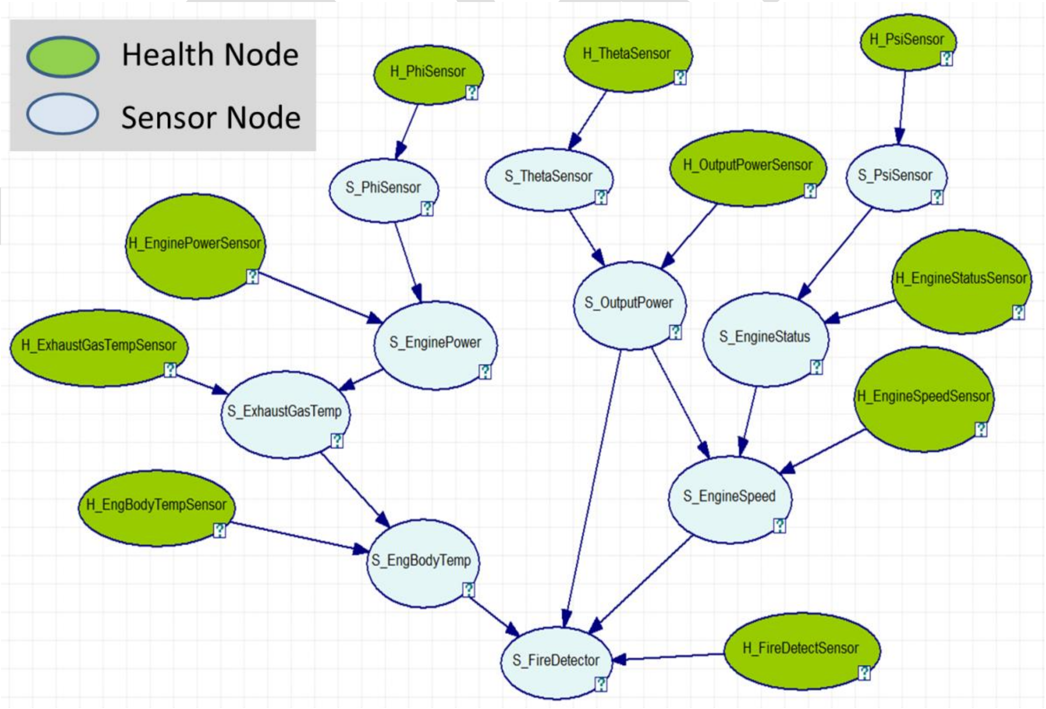


Figure 5. A Bayesian Network based on the Engine Operating States

To run the Bayesian Network analysis, we implemented a modular application called BN2DDS. BN2DDS is capable of subscribing to different normalized STANAG 4586 messages using DDS, through the definition of input plugins. In particular, we implemented input plugins for STANAG Engine Operating States and Inertial States. Given a Genie compatible Bayesian Network, BN2DDS will periodically set new evidences on the network using the information gathered by different input plugins. The extraction and publication of information from the network (e.g., health status of the different sensors) is performed by a set of output plugins. For this POC, we have implemented an output plugin that publishes the probability of being good or bad for each of the sensors.

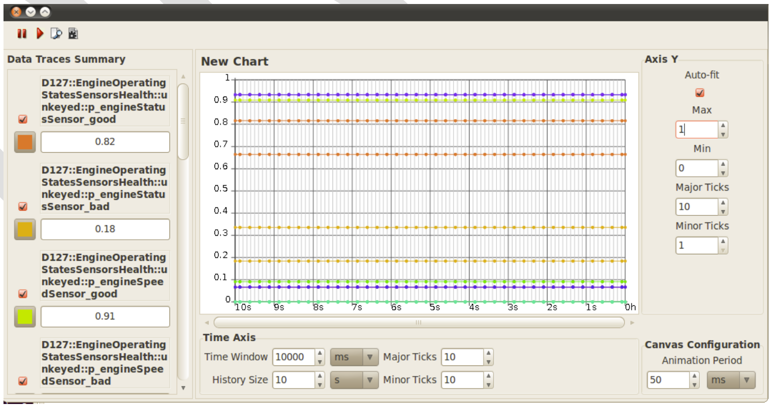
Table 3 shows the XTypes data model we used to distribute prognostics and health assessment of the engine sensors. Two probability values are published for each sensor. The values indicate the probability of sensor being good/bad. Probability value below a system-dependent threshold would indicate an anomaly in the system.

Table 3: Normalized data model for cyber health status of sensors in an engine

```
struct EngineOperatingStatesSensorsHealth {
  double p_engineStatusSensor_good;
  double p_engineStatusSensor_bad;
  double p_engineSpeedSensor_good;
  double p_engineSpeedSensor_bad;
  double p_enginePowerSensor_good;
  double p_enginePowerSensor_bad;
  double p_exhaustGasTempSensor_good;
  double p_exhaustGasTempSensor_bad;
  double p_engineBodyTemperature_good;
  double p_engineBodyTemperature_bad;
  double p_outputPowerSensor_good;
  double p_outputPowerSensor_bad;
  double p_fireDetectionSensor_good;
  double p_fireDetectionSensor_bad;
};
```

D. Data Visualization

The use of RTI Connext DDS for information sharing eases the addition of new elements to the global data space. The middleware builds a common operational picture, where new elements can be added in a seamless and scalable manner. For instance, we easily added visualization elements to show the cyber health status of the sensors of our UAV's engine and the IMU sensor by creating adequate subscriptions on the DDS global data space. Figure 6 shows the use RTI Visualizer for this purpose.



☑ D127::EngineOperatingStatesSensorsHealth::unkeyed::p_engineStatusSensor_good
0.82

☑ D127::EngineOperatingStatesSensorsHealth::unkeyed::p_engineStatusSensor_bad
0.18

☑ D127::EngineOperatingStatesSensorsHealth::unkeyed::p_engineSpeedSensor_good
0.91

☑ D127::EngineOperatingStatesSensorsHealth::unkeyed::p_engineSpeedSensor_bad

Auto-fit ☑
Max 1
Min 0
Major Ticks 10
Minor Ticks 1

Time Window 10000 ms
Major Ticks 10
History Size 10 s
Minor Ticks 10

Animation Period 50 ms

Figure 6: RTI Visualizer

E. Extensibility of the Data Model

XTypes supports evolution of any subset of data types at a time. Should any of the system components (sensors, X-Plane simulator, Bayesian analysis, visualization) are extended to include more data attributes, the dataflow in the system still works without any interruptions. Any component in the system can be updated with an updated data definition. The system components need not be aware of the other updated system components. When the datatypes of the publishing and the subscribing components are *assignable*, Connext DDS automatically does the necessary data translation as per the rules described in the XTypes specification.

VII. Related Work

Sreenuch et al.[4], successfully used RTI Connext DDS to implement the Open System Architecture Condition-Based Maintenance (OSA-CBM) standard, which is aimed at reducing the development and integration cost of Integrated Vehicle Health Management (IVHM). The authors argue that the primary benefit of using the publish/subscribe architecture is to abstract the message transport from the software components and allows them to be decoupled. Furthermore, deployments can be done rapidly. Our research not only support this prior research but also goes a step further by demonstrating the use of XTypes to create an extensible data model and distributed architecture for sensor health assessment. The novelty of our approach lies in extensibility, the normalized data model for sharing sensor data, the use of the Bayesian inference library and RTI Connext DDS in an extremely low-power processing unit, and the *hardware-in the-loop* simulation of the orientation sensor combined with the software simulation of the UAV engine operating states.

VIII. Conclusion

Data Distribution Service (DDS) provides a powerful architecture for implementing real-time onboard sensor health assessment systems. Using the standard DDS API and the standards-based extensible data model for exchanging sensor data, sensors and the analysis components can be strongly decoupled allowing flexibility in the evolution and ultimately acquisition of the onboard sensors and the analysis components. RTI Connext DDS is the leading implementation of the DDS and the XTypes standard, which can be deployed in extremely low power and resource constrained environments to drive real-time sensor health analytics.

IX. Acknowledgment

This research is funded by the Air Force Research Laboratory, WPAFB, Dayton, OH under Small Business Innovation Research (SBIR) contract # FA8650-11-C-1054. Manuscript approved for publication by WPAFB: PA Approval Number: 88ABW-2013-3209.

References

[1] Data Distribution Service, OMG document formal/2007-01-01, http://www.omg.org/spec/DDS/1.2/

[2] Extensible and Dynamic Topic Types, OMG document formal/2012-11-10, http://www.omg.org/spec/DDS-XTypes/1.0/

[3] SMILE and GENIE: Structural Modeling, Inference, and Learning Engine http://genie.sis.pitt.edu/.

[4] Application of the Data Distribution Service for Implementing OSA-CBM Standard; Tarapong Sreenuch, Antonios Tsourdos, Ian Jennions, and Peter Silson, Infotech@Aerospace 2011.

[5] UAS Control Segment, https://ucsarchitecture.org/

[6] The Future Airborne Capability Environment (FACE™) Consortium, an Open Group Managed Consortium, http://www.opengroup.org/getinvolved/consortia/face

[7] The Real-Time Publish Subscribe DDS Wire Protocol, v2.1, http://www.omg.org/spec/DDSI/2.1/

[8] ISO/IEC C++ 2003 Language DDS PSM (DDS-PSM-Cxx) http://www.omg.org/spec/DDS-PSM-Cxx/

[9] Java 5 Language PSM For DDS (DDS-Java) http://www.omg.org/spec/DDS-Java/

[10] RTI Connext™ DDS, http://www.rti.com/products/dds/

[11] STANAG 4586 Navy (Edition 2)—Standard Interfaces for UAV Control System (UCS) for NATO UAV Interoperability

[12] X-Plane Flight Simulator, http://www.x-plane.com/desktop/home/

[13] General Atomics MQ-9 Reaper Factsheet, http://www.af.mil/information/factsheets/factsheet.asp?id=6405