Prognostics for Autonomous Electric-Propulsion Aircraft

Johann Schumann[1], Chetan Kulkarni[2], Michael Lowry[3], Anupa Bajwa[4], Christopher Teubert[5], Jason Watkins[6]

[3, 4, 5] *NASA Ames Research Center, Moffett Field, CA, USA*
michael.r.lowry@nasa.gov
anupa.r.bajwa@nasa.gov christopher.a.teubert@nasa.gov

[1, 2, 6] *KBR, NASA ARC, Moffett Field, CA, USA*
johann.schumann@nasa.gov
chetan.s.kulkarni@nasa.gov
jason.watkins@nasa.gov

Abstract

An autonomous unmanned aerial system (UAS) needs, during the flight, accurate information about the current failure state of the aircraft and its capabilities in order to safely perform its mission and properly react to contingencies. The flight battery of an electric-propulsion aircraft is its most relevant resource. Model-based prognostics algorithms are used to obtain good estimates of its current state of charge and remaining capacity. However, these algorithms can have a large computational footprint. We present Prognostics-as-a-Service, a hybrid approach combining on-board computation with server-based prognostics on the ground.

In this paper, we focus on the role, battery prognostics plays for the safe operation of a highly autonomous aircraft: prognostics for (1) continuous on-board safety monitoring, (2) for UAS operations, and (3) for contingency planning. We present the NASA Autonomous Operating System (AOS) and discuss how the autonomous components closely work together with on-board and server-based ground prognostics systems. We will illustrate the system with case studies on small NASA unmanned aircraft.

1. Introduction

Unmanned Aerial Systems (UASs) are being increasingly used in different application areas. These range from package delivery (Beckman, Haskin, Rolnik, & Vule, 2017; Lisso, 2017), automatic surveying (Gašparović & Gajski, 2016), precision agriculture (Gómez-Candón, De Castro, & López-Granados, 2014), search-and-rescue (Polka, Ptak, & Kuziora, 2017), medical package delivery (Thiels, Aho, Zietlow, & Jenkins, 2015; Jones & Despotou, 2019), etc., to applications in Urban Area Mobility (UAM) to transport humans in small un-piloted aircraft over short distances.

The goal of all these applications is to break away from piloted remote control toward full autonomous operations, where the flight computer of the UAS has full authority over the aircraft and cargo. Obviously, most of these UAS applications are, at least to some extend, safety-critical. Failures during operation could potentially lead to loss of human life in UAM applications, could cause damage on the ground, or, at least would result in the loss of the mission and vehicle. Therefore, the autonomous components (AUC) controlling the UAS must be able to detect system degradation, failures, environmental anomalies, as well as be aware of the system's current health state and capabilities. Only with that information the autonomous component can react accordingly and come up with necessary contingency actions that avoid safety violations and minimize risks while trying to perform the current mission/task as good as possible.

Typical examples include fault detection, diagnosis, and recovery (FDDR) on critical subsystems like the engines, electrical subsystem, or actuators. While the FDDR system reacts on failures that have already occurred, reliable knowledge about the vehicle's resources and likelihood about future failures is extremely important so that the AUC can make informed decisions.

For most UAS applications, electric-propulsion aircraft are used. This means, one or more high-power batteries are driving all motors and propellers of the vehicle. For such an aircraft, the flight battery is obviously the most relevant resource. Its status and remaining charge needs to be carefully measured and monitored. Also, the batteries for avionics and on-board computation should be monitored as well, as they

https://doi.org/10.36001/IJPHM.2021.v12i3.2940

might use up to 25% of the overall UAS power consumption (Caccamo, 2017).

At each point in time during the flight, the AUC needs reliable information about the current state-of-charge (SOC) of the battery and if the battery has enough remaining capacity to safely fly the entire mission. Due to the complex and non-linear battery characteristics, these values cannot be directly measured as it is possible with a gas-powered aircraft. *Prognostics* systems have been developed to be able to exactly address these questions. Mostly model-based, the prognostics system monitors the battery, provides up-to-date statistical information about the current state-of-charge (SOC) and the rest of useful life (RUL) for the flight battery.

In this paper, we focus on the role, prognostics plays for the safe operation of highly autonomous aircraft. We identify three major application areas: (1) prognostics for continuous safety and performance monitoring of the UAS components and battery, (2) prognostics for improved operations and mission planning, and (3) prognostics for contingency planning.

However, accurate prognostics algorithms have a large computational footprint, which may prohibit it from running on the on-board flight computer. In this paper, we therefore investigate interfacing the on-board autonomous components to a hybrid prognostics architecture: PaaS. PaaS (Prognostic-as-a-Service) (Teubert, Daigle, Sankararaman, Goebel, & Watkins, 2017) is a ground-based server system that communicates with the on-board systems and provides results for complex prognostics tasks, which cannot be executed by the on-board computer hardware. That way, prognostics tasks, even going beyond just one vehicle, can be executed as effectively and accurately as possible, given current restrictions of on-board computational resources, availability of ground communication links, and required timeliness of prognostics solutions.

As our environment in this paper, we use the *Autonomy Operating System* (AOS), a software system and framework (Lowry et al., 2018) that has been developed at NASA Ames. AOS provides important capabilities that are necessary for the autonomous operation of a UAS and features automatic execution of flight plans, natural-language communication with Air Traffic Control, as well as contingency planning, diagnostics, and prognostics.

In this paper, we discuss different prognostics applications within AOS and illustrate them with case studies carried out on a simulated fixed-wing UAS and on actual test flights with small multi copters.

The rest of this paper is structured as follows: Sections 2 and 3 provide the background on the Autonomous Operating System (AOS) and battery modeling and prognostics. In Section 4, we present the hybrid PaaS system. Section 5 focuses on the role of prognostics for on-board monitoring. In Section 6 we present how prognostics can help with mission planning, and in Section 7 we demonstrate the role of prognostics for contingency planning. Section 8 presents related work and Section 9 summarizes the paper, discusses future work and concludes.

2. BACKGROUND: THE AUTONOMY OPERATING SYSTEM (AOS)

The Autonomy Operating system (AOS) is a software system that enables core capabilities for the autonomous operations of an unmanned aircraft (Lowry et al., 2018). It is based on the NASA Core Flight System (cFS) system (McComas, 2012) and provides a higher-level layer of infrastructure, applications, and communication mechanisms.

The AOS architecture, shown in Figure 1, provides capabilities for the execution of flight plans, natural-language communication with Air Traffic Control (Lowry, Pressburger, Dahl, & Dalal, 2019), Diagnostics, Prognostics, and contingency planning (Schumann, Mahadevan, Sweet, et al., 2019). The underlying cFS system provides a "software bus," which is a publish-subscribe architecture that is used for communication between the different components or applications of the system. These can be activated on regular schedule, in our case with a rate of up to 10Hz. AOS is communicating with a low level flight control software (Figure 1 bottom) to obtain sensor and aircraft status information and to issue low level commands. In our case, we use a slightly modified version of the Open-source ArduCopter software,[1] running on a PixHawk hardware,[2] which directly interfaces with sensors and controls the motors of the UAS and which is in charge to keep the aircraft in the air and perform waypoint-to-waypoint flights.

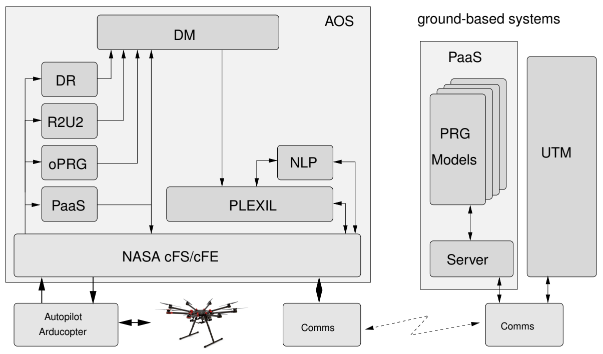


Figure 1. High-level architecture of the AOS system

Our model-based diagnostics, prognostics, and contingency planning framework is implemented as several applications on top of the cFS system and centered around the Decision Maker (DM). Figure 1 shows the flow of information: Sen-

[1]`http://ardupilot.org/copter/`
[2]`pixhawk.org`

sor and status data from the UAS are transmitted by the ArduCopter flight software into the AOS system. These data are used to perform diagnosis, prognostics, and monitoring (Figure 1, left). The Diagnostic Reasoner (DR) performs model-based detection and isolation of failures, using an efficient algorithm based upon diagnosability matrices (Schumann, Mahadevan, Sweet, et al., 2019). The R2U2 (Realizable, Responsive, Unobtrusive Unit), which will be described in more detail below is continuously monitoring the system behavior using temporal logic observers and Bayesian reasoners. The on-board prognostics (oPRG) engine performs model-based determination of SOC and RUL for the flight battery, using the Kalman filter based algorithm as described earlier in this paper. More elaborate prognostics tasks that would consume to too much on-board resources are handled by the PaaS (Prognostics as a Service) client, which would use a ground-based server for the computation.

The current health status of the UAS as obtained by these modules are updated with a 0.5Hz rate and then handed over to the DM. The DM performs logic-based search to find (a) active diagnostic procedures to improve the diagnostic resolution (if necessary) and (b) contingency flight plans, which can be safely executed under the current circumstances. If necessary, such contingency plans might contain emergency actions like, for example, cutting short the flight, diversion to a nearby airport for emergency landing, or an immediate ditch by activating an on-board parachute. The generated active diagnosis or contingency plan is sent to PLEXIL (Verma, Jonsson, Pasareanu, & Iatauro, 2006). PLEXIL is an event-driven planner that has been customized to execute flight plans for nominal/off-nominal operations, which might require Air Traffic Control (ATC) interaction. Spoken ATC commands are processed by the Natural Language Processing (NLP) unit in AOS (Lowry et al., 2019). During plan execution, PLEXIL emits a sequence of commands and waypoints to the low-level autopilot that the UAS will follow.

AOS can optionally communicate with a ground station (Figure 1, right), which contains the PaaS server with it multiple prognostics models and algorithms as well as UTM (Unmanned Traffic Management) interfaces, which provide services for aircraft separation, management of reserved volumes, and coordination between multiple UAS sharing the airspace (Federal Aviation Administration, 2020).

3. BACKGROUND: BATTERY MODELING AND PROGNOSTICS

In order to predict end-of-discharge (EOD) as defined by a voltage cutoff, the battery model must compute the voltage as a function of time given the current drawn from the battery. There are several electro-chemical processes that contribute to the cell's potential that make this a difficult problem. For the purposes of on-line prognostics, we focus here on a lumped-parameter ordinary differential equations form that still considers the main electro-chemical processes.

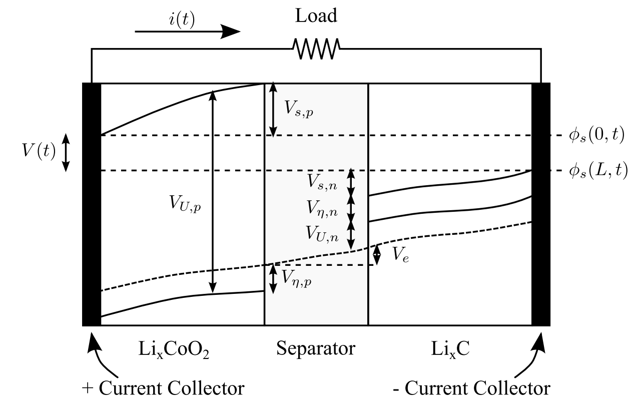


Figure 2. Battery voltages (from (Daigle & Kulkarni, 2013))

The voltages of a single cell in a battery pack are summarized in Figure 2. The overall battery voltage $V(t)$ is the difference between the potential at the positive current collector, $\phi_s(0,t)$, and the negative current collector, $\phi_s(L,t)$, minus resistance losses at the current collectors (not shown in the diagram). As shown in the figure, the potentials vary with the distance $d \in [0, L]$, because the loss varies with distance from the current collectors.

Battery potentials as seen in Figure 2 at the current collectors are described by several voltage terms. At the positive current collector is the equilibrium potential $V_{U,p}$. This voltage is then reduced by $V_{s,p}$, due to the solid-phase ohmic resistance, and $V_{\eta,p}$, the surface overpotential. The electrolyte ohmic resistance then causes another drop V_e. At the negative electrode, there is a drop $V_{\eta,n}$ due to the surface overpotential, and a drop $V_{s,n}$ due to the solid-phase resistance. The voltage drops again due to the equilibrium potential at the negative current collector $V_{U,n}$. Details of the developed battery model are discussed in (Daigle & Kulkarni, 2013).

3.1. State of Charge

State of Charge (SOC) of a battery is conventionally defined to be 1 when the battery is fully charged and 0 when the battery is discharged to a set voltage threshold. In this model, it is analogous to the mole fraction x_n, but scaled from 0 to 1. There is a difference here between nominal SOC and *apparent* SOC. Nominal SOC would be computed based on the combination of the bulk and surface layer control volumes in the negative electrode, whereas apparent SOC would be computed based only on the surface layer. That is, a battery can be discharged at a given rate, and reach the voltage cutoff, i.e., apparent SOC is then 0 as discussed earlier. But, once the concentration gradient settles out, the surface layer will be partially replenished and the battery can be discharged further, i.e, apparent SOC increases whereas nominal SOC remains the same.

Nominal (n) and apparent (a) SOC are defined by equations below:

$$SOC_n = \frac{q_n}{0.6q^{\max}}, \quad \text{and} \quad SOC_a = \frac{q_{s,n}}{0.6q^{\max_{s,n}}},$$

where $q^{\max_{s,n}} = q^{\max}\frac{v_{s,n}}{v_n}$. The factor $1/0.6$ comes from the fact that the mole fraction at the positive electrode cannot go below 0.4 (Daigle & Kulkarni, 2013), therefore SOC of 1 corresponds to the point where $q_n = 0.6q^{\max_{s,n}}$.

The model contains as states $\mathbf{x}$, $q_{s,p}$, $q_{b,p}$, $q_{b,n}$, $q_{s,n}$, V'_o, $V'_{\eta,p}$, and $V'_{\eta,n}$. The single model output is V. Model validation for a variable loading scenario is shown in Fig. 3. The load changes every 2 minutes (Fig. 3A), resulting in corresponding changes in voltage. Fig. 3B shows that the voltage predictions are fairly accurate in response to changes in load. Some errors are still present that may possibly be accounted for by including temperature effects.

3.2. Prognostics

In this section we discuss our developed battery prognosis framework following the general estimation-prediction methodology of model-based prognostics (Luo, Pattipati, Qiao, & Chigusa, 2008; Orchard, Tobar, & Vachtsevanos, 2009; Daigle & Goebel, 2013; Daigle & Kulkarni, 2013). Similar approaches have been used for prognosis of pneumatic valves (Daigle, Kulkarni, & Gorospe, 2014; Kulkarni, Daigle, Gorospe, & Goebel, 2014) as well as for Current/Pressure (I/P) Transducers (IPT) (Teubert & Daigle, 2013, 2014). The formulation of the prognostics problem is summarized below followed by a brief description of the estimation and prediction approach.

3.2.1. Problem Formulation

The system model may be generally defined as

$$\mathbf{x}(k+1) = \mathbf{f}(k, \mathbf{x}(k), \boldsymbol{\theta}(k), \mathbf{u}(k), \mathbf{v}(k)), \text{and}$$
$$\mathbf{y}(k) = \mathbf{h}(k, \mathbf{x}(k), \boldsymbol{\theta}(k), \mathbf{u}(k), \mathbf{n}(k)),$$

where k is the discrete time variable, $\mathbf{x}(k) \in \mathbb{R}^{n_x}$ is the state vector, $\boldsymbol{\theta}(k) \in \mathbb{R}^{n_\theta}$ is the unknown parameter vector, $\mathbf{u}(k) \in \mathbb{R}^{n_u}$ is the input vector, $\mathbf{v}(k) \in \mathbb{R}^{n_v}$ is the process noise vector, $\mathbf{f}$ is the state equation, $\mathbf{y}(k) \in \mathbb{R}^{n_y}$ is the output vector, $\mathbf{n}(k) \in \mathbb{R}^{n_n}$ is the measurement noise vector, and $\mathbf{h}$ is the output equation.[3]

In prognostics, the occurrence of an event E is to be predicted, that is defined with respect to the states, parameters, and inputs of the system. The event is defined as the earliest instant that some event threshold $T_E : \mathbb{R}^{n_x} \times \mathbb{R}^{n_\theta} \times \mathbb{R}^{n_u} \to \mathbb{B}$, where $\mathbb{B} \triangleq \{0, 1\}$ changes from the value 0 to 1. That is, the time of the event k_E at some time of prediction k_P is defined as

$$k_E(k_P) \triangleq \inf\{k \in \mathbb{N} \colon k \geq k_P \wedge T_E(\mathbf{x}(k), \boldsymbol{\theta}(k), \mathbf{u}(k)) = 1\}.$$

The time remaining until that event, Δk_E, is defined as

$$\Delta k_E(k_P) \triangleq k_E(k_P) - k_P.$$

For system health management, T_E is defined via a set of performance constraints that define what the acceptable states of the system are, based on $\mathbf{x}(k)$, $\boldsymbol{\theta}(k)$, and $\mathbf{u}(k)$ (Daigle & Goebel, 2013). For batteries, we are interested in predicting the end of discharge (EOD) time, i.e., the time at which the battery voltage will deplete below the voltage threshold V_{EOD}.

Models of the system components are constructed in this paradigm that capture both nominal behavior, as well as faulty behavior and damage progression. Using these models, observations can be mapped back to the health state of the system as represented in $\mathbf{x}$ and $\boldsymbol{\theta}$. An estimation algorithm, such as the Kalman filter (KF), unscented Kalman filter (UKF), or particle filter (PF), is used to solve these types of problems (Daigle, Saha, & Goebel, 2012). In this work an UKF based approach in implemented. This state-parameter estimate, along with a prediction of the future usage of the component, is used as input to a prediction algorithm that computes the time to EOD. The difference between EOD and current time is called the remaining useful life (RUL) (Daigle & Goebel, 2013; Daigle, Saxena, & Goebel, 2012).

3.2.2. Prognostics Architecture

In our model-based prognostics architecture (Daigle & Goebel, 2013), there are two sequential problems, (*i*) the *estimation* problem, which requires determining a joint state-parameter estimate $p(\mathbf{x}(k), \boldsymbol{\theta}(k)|\mathbf{y}(k_0{:}k))$ based on the history of observations up to time k, $\mathbf{y}(k_0{:}k)$, and (*ii*) the *prediction* problem, which determines at k_P, using $p(\mathbf{x}(k), \boldsymbol{\theta}(k)|\mathbf{y}(k_0{:}k))$, a probability distribution $p(k_E(k_P)|\mathbf{y}(k_0{:}k_P))$. The distribution for Δk_E can be computed from $p(k_E(k_P)|\mathbf{y}(k_0{:}k_P))$ by subtracting k_P.

The prognostics architecture is shown in Figure 4. In discrete time k, the system i.e. any model in this case li-ion battery model, is provided with inputs $\mathbf{u}_k$ and provides measured outputs $\mathbf{y}_k$. The estimation module uses this information, along with the system model, to compute an estimate $p(\mathbf{x}(k), \boldsymbol{\theta}(k)|\mathbf{y}(k_0{:}k))$. The prediction module uses the joint state-parameter distribution and the system model, along with hypothesized future inputs, to compute the probability distribution $p(k_E(k_P)|\mathbf{y}(k_0{:}k_P))$ at given prediction times k_P.

Estimation. A detailed electro-chemistry (EC) based physics model of component behavior is developed using nominal data from the testbed (Daigle & Kulkarni, 2013). This work

[3]Bold typeface denotes vectors, and n_a denotes the length of a vector $\mathbf{a}$.

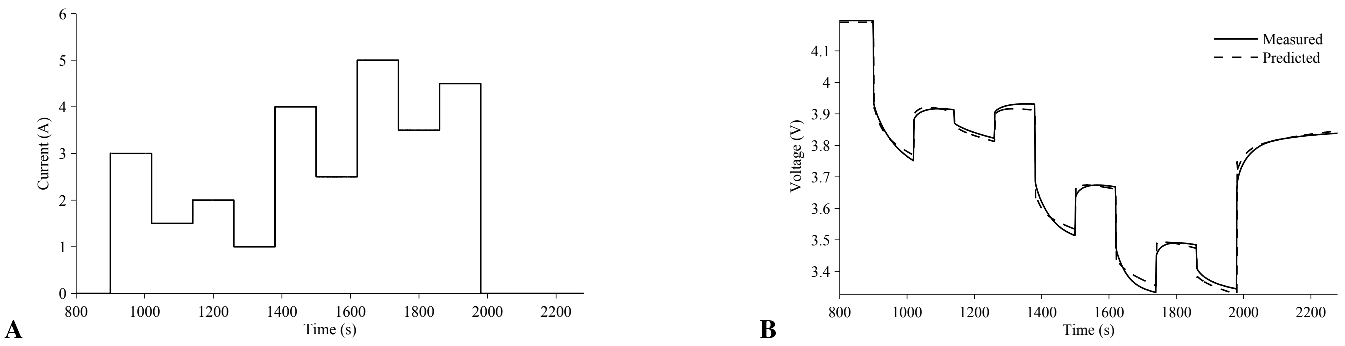


Figure 3. Model validation for variable loading. Current over time (A); measured and predicted discharge curves (B).

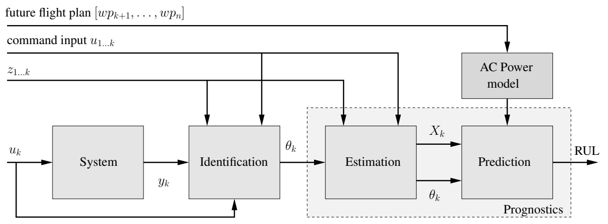


Figure 4. Prognostics architecture with identification, state estimation and prediction component that is driven by power requirements based on the future flight plan.

implements the developed EC model to estimate and predict health state of the battery. An unscented Kalman filter (UKF) is implemented to obtain the state estimate from the sensor measurements. Details of the implemented framework for battery modeling are discussed in (Daigle & Kulkarni, 2013).

Table 1. Prognostics algorithms for monitoring M and prediction P of different tasks: from battery prognostics, Aircraft (AC) with multiple Failure Modes (FM), to system wide prognostics. Algorithms include Extended (EKF) and Unscented (UKF) Kalman Filters, Particle Filters (PF), and Bayesian techniques (Bayes).

Task	M	P	Algorithms
Threshold U, I	Y	N	
SoC	Y	Y	EKF
RUL (const load)	Y	Y	EKF, UKF
RUL (flight plan)	Y	Y	EKF, UKF, PF
RUL (FP, error bars)	Y	Y	PF, Bayes
RUL (multiple FPs)	-	Y	PF
RUL + FMs	Y	Y	PF, Bayes
Multiple AC	Y	Y	PF, Bayes

Prediction. Whereas the SOC value can be calculated directly from the current state, the calculation of remaining-useful life (RUL) requires an estimate of the future load profile. With that information, a Monte Carlo forward integration is used to obtain the mean of RUL and its distribution. For an electric UAS, the power requirements are directly related to the planned future flight path. Here, we assume that the flight computer, the sensors, and the avionics is operated using a different battery, which is not considered here[4]. Our model-based prediction architecture shown in Figure 4 features an extended model-based prediction component which, given the planned future flight plan first estimates a load profile and then uses that to perform the prediction. For this work, we used a simplified model for small fixed-wing aircraft described in (Schumann, Roychoudhury, & Kulkarni, 2015).

There are different approaches being implemented in the prognostics framework. Model based approaches include first principles based, lumped parameter physics based, empirical models etc. Data driven approaches include DNNs, machine learning approaches and several others. Depending upon the computational complexities the processing can be done onboard (Gorospe, Daigle, Sankararaman, Kulkarni, & Ng, 2017) or off-board as in the case of PaaS (Teubert et al., 2017) which will be discussed in the next section.

Though in this work decision making is based on battery state of charge estimation, the frame work is able to take different state of health inputs for vehicle like motor/ESC, power conditioning circuit, sense-and-avoid info as well as external factors like weather etc. Resulting outputs from respective algorithms can then combined be combined to indicate overall health index based on severity and priority to the decision making framework.

The input parameters for the algorithm as well as the prognostics estimates vary depending on the application framework. In this case the inputs to the algorithm are voltage, current while the output is SOC estimate as well as RUL for the SOC to reach set threshold. The lower threshold bound is defined by the operator and can change based on the safety require-

[4]In general, the batteries for avionics and on-board computation should be monitored as well, as they might use up to 25% of the overall UAS power consumption (Caccamo, 2017).

ments. Based on the threshold bounds the prediction horizon changes which is updated in the initialization parameters.

4. PROGNOSTICS AS A SERVICE (PAAS)

Most approaches for prognostics require significant computational resources. This is especially true for prognostics of critical aviation systems, where risk-tolerance is very low. For sample-based prediction approaches commonly used in prognostics, the number of samples required to predict with the confidence needed for these risk-intolerant applications is prohibitively great for many aircraft. This observation led to the team's creation of the Prognostics As-A-Service (PaaS) Domain Specific Software Architecture (DSSA) (Watkins, Teubert, & Ossenfort, 2019).

4.1. Architecture Decision: On-board, As-A-Service, and Hybrid

The design decision of where to perform prognostics (on-board, as-a-service, or some combination) is an important one. Below are a few considerations that should be considered:

Advantages of a PaaS architecture:

- *Greater Computational Resources*: Computational demand is a function of risk tolerance, prognostic horizon, and the number of prognosed systems. Many platforms cannot host the computer needed to perform prognostics to their performance requirements. Cloud resources increases resources enabling prognostics with greater precision, prognostic horizon, and number of systems.
- *Resource Sharing*: On-board prognostics applications are dormant when the platform is not in use. Cloud-based PaaS architectures take advantage of on-demand resources, reducing costs for some applications.
- *Intelligent Data Leveraging*: In the PaaS architecture, data from many systems are fed to a centralized location. This opens the door for algorithms that leverage all this data to improve models and provide better service.
- *Update and Installation Ease*: For many applications physical access to the machine is limited, and updating software is difficult. The PaaS architecture enables fleet-wide deployment of new features and bug fixes.

Advantages of on-board architecture:

- *No Link Dependence*: PaaS architectures require stable communication to transmit sensor data and predictions. Applications where communications are unstable and the failures are time-critical have no tolerance for dropouts.
- *Reduced Latency*: Communication with PaaS typically adds a round-trip latency of at least 150ms. This is acceptable for most applications, but for a few rare extremely time-critical targets, this latency is too high.
- *Security*: Though there are technologies to mitigate these, the communication link creates additional security risks.
- *No Link Needs*: On-board architectures reduce the platform communication requirements (bandwidth, etc.).

For some applications the advantages of an on-board architecture may not apply. This is true for long-term degradation where the aircraft can interact with PaaS off-line upon landing using a reliable hard line connection, or when communication is guaranteed reliable and secure.

Conversely, when communication is spotty and systems degrade quickly the advantages of the on-board architecture may far outweigh those of the PaaS architecture.

For many cases, a hybrid architecture may be ideal. Here are a few different approaches for hybrid prognostics architectures:

1. *Perform critical functions on-board*: critical functions, such as those that degrade quickly (time-critical) and are critical to safety are done completely on-board.
2. *Perform link critical functions on-board*: For bandwidth constrained applications, those functions that require the largest amount of bandwidth are done completely on-board
3. *Perform a minimum amount of prognostics for all systems on-board*: Some basic prognostics can be done on-board, with lower precision and a shorter time horizon. Longer, more precise prognostics for strategic decision making can still be done on a PaaS architecture. This maintains just enough capability to preserve safety in the event of loss of communications.

Some additional notes on the decision to perform prognostics on-board or as-a-service can be found in (Sankararaman & Teubert, 2017).

The architecture described in Section 4.3 is designed to be used to supplement on-board PHM capabilities, or provide prognostics capabilities to platforms without any on-board PHM.

4.2. General Use-Case

PaaS architectures like the one described here have common elements in the way they are used. These are illustrated in Figure 5. Understanding the basics of how a PaaS architecture is used is important for understanding the architecture itself.

First, pilots, operators, or the vehicles themselves request prognostics services from a menu of available services (Step 1), beginning their prognostics session. They must have set up accounts and configured them to describe the systems on-board their fleet beforehand (this needs only to be done once).

Once the session has begun, sensor data required for prognostics (which is specific to the system(s) being prognosed) is

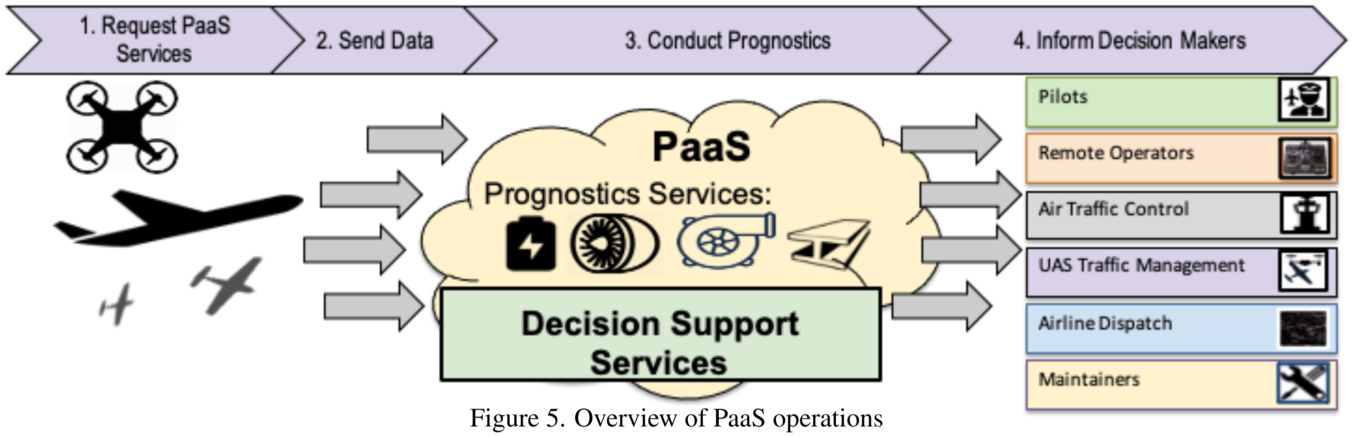


Figure 5. Overview of PaaS operations

streamed to the PaaS service (Step 2). The PaaS service uses this and the pre-supplied configuration to perform prognostics (Step 3). The results from prognostics are used by decision-makers to inform decisions to preserve safety and maintain efficiency of the vehicle (Step 4). Decision-makers may include on-board like pilots (both machine or human), ground operators or supervisors, or fleet managers. They can also be other stakeholders like air traffic controllers, automated air traffic management services, maintainers (predictive maintenance), or even emergency responders.

Steps 2–4 repeat for the duration of the mission. Once the mission is completed, the session is closed.

4.3. PaaS Architecture

PaaS uses a layered architecture style. There are three distinct layers: the REST (Representational State Transfer) API, the Service Layer, and the Execution Layer, further described in Sections 4.3.1-4.3.3. Each of these only communicate with their adjacent layers.

Understanding the function of these layers requires an understanding of the hierarchy of entities used to organize information in the PaaS Architecture. The relationships between the different entities is illustrated in Figure 6 and each of the entities are described below.

User. A single user of the PaaS service. Can own multiple platforms.

Platform. A collection of prognostic targets (systems) to be analyzed together, such as an aircraft or vehicle. Each platform has one or more systems associated with it and are owned by Users.

Session. A session is created when a user requests prognostics for systems in one of their platforms. The user terminates the session when they no longer need prognostics. Sessions are tied to a single platform.

System. A discrete object on a platform that can be analyzed

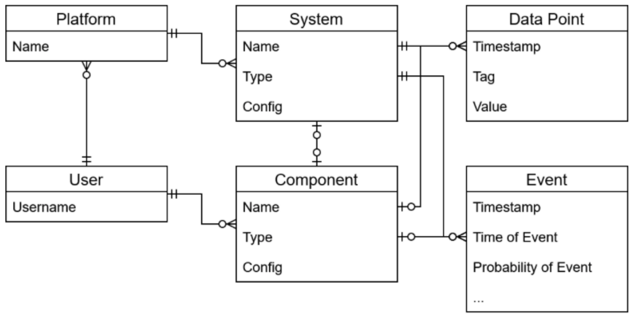


Figure 6. PaaS Entity Relationships

(e.g., ’battery1’ on ’UAS3’). Systems are owned by the specific platform (e.g., aircraft) that they are on.

Component. A physical object that is used in a system (e.g., battery with serial abc123). The component stores configuration information that is specific to a particular physical device. They are used to track health between multiple sessions. Components are assigned to a system (e.g., component ’battery abc123’ is assigned as system ’battery1’ on platform ’UAS3’).

Data Point. A single sensor value, such as a battery voltage or vehicle latitude.

Event. A single prediction of an event. Includes current state (e.g., State of Health), prediction of time of event, and event state at various ”save points” between now and time of event.

4.3.1. Layer 1: REST API

Representational State Transfer (REST) is a web architecture that allows for both querying and updating of resources using HTTP requests. The REST style provides a uniform stateless architecture that fits well into today’s web-centric world. The API is not without limitation however. In particular, the REST format (and HTTP in general) do not provide any mechanism for push-based notifications. Due to this limitation, the current API requires that clients poll periodically to receive new events.

Table 2. PaaS URIs

URI	GET	POST	PUT	DELETE
/component	X	X	X	X
/data		X		
/event	X			
/platform	X	X	X	X
/session	X	X		X
/system	X	X	X	X
/user	X	X		X

The PaaS API was designed according to the OpenAPI specifications (Swagger, 2020). The primary Uniform Resource Indicators (URIs) are summarized in Table 2. GET requests are used to request information, POST requests create something new, PUT requests update information, and DELETE requests delete the specified resource.

Note that REST was used for the preliminary implementation of the PaaS Architecture, but future implementations could use alternate protocols.

4.3.2. Layer 2: Service Layer

The service layer executes the primary "business logic" of the system. This includes input validation, session management, storage and retrieval of data from the database. Session management activities include the spinning off of a new "Prognostics Application" in the execution layer for each new session, then configuring the application to perform prognostics on the systems for that platform by passing configuration parameters for the systems of interest to the prognostics application.

4.3.3. Layer 3: Execution Layer

The execution layer implements the logic for prognostics described in Section 3.2. This layer consists of multiple prognostics applications, one for each active session.

In PaaS, Prognostics Applications are built on top of the Generic Software Architecture for Prognostics (GSAP), and publicly released software framework for building prognostics applications. The GSAP Architecture is described in greater detail in the 2017 GSAP paper (Teubert et al., 2017), and on its website[5]. Since the publishing of that paper, the GSAP architecture has been updated to adopt an event-driven architecture style, where observers and predictors communicate asynchronously. GSAP was adapted to support the needs of the PaaS framework.

4.4. Notes on Architecture in Practice

In practice, a PaaS architecture could provide services to a large number of aircraft. PaaS could be potentially be implemented as a single service supplier for a broad range of customers—a Supplemental Data Service Provider (SDSP) in the UAS Traffic Management (UTM) architecture. In this case the SDSP would provide services to customers for a price (either subscription or demand-based).

Alternately, PaaS architectures could be implemented by organizations that operate a large number of vehicles to provide private prognostics services to their fleet. The prognostics services could be extended beyond aircraft to also provide prognostics for ground support equipment.

This architecture was adapted by the NASA System Wide Safety (SWS) Project to create the SWS Service (Corbetta, Banerjee, Okolo, Gorospe, & Luchinsky, 2019), a REST service for providing estimates of safety metrics and predictions of how safety will change with time. It was also used in the NASA Convergent Aeronautics Solutions (CAS) PaaS Incubation Project, and the NASA CASAS autonomous decision making effort.

[5]https://software.nasa.gov/software/ARC-17748-1A

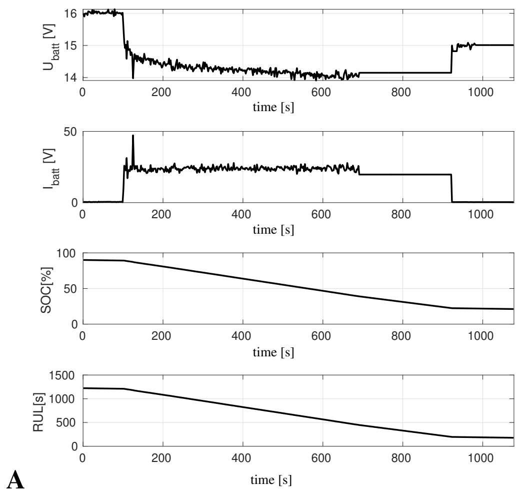
5. PROGNOSTICS FOR ON-BOARD MONITORING

For safe and reliable autonomous operations, the UAS must continuously monitor its state and health. In case of failures, deviations, or unfavorable prognostics results, the flight software for the UAS must be able to, without any ground support, plan and execute appropriate contingency measures that help to keep the UAS safe in the air and allows it to complete the mission as best as possible. In order to achieve this goal, the AOS system has several advanced diagnostic engines, which can in combination with the on-board prognostics engine and PaaS provide and process current health information and future estimates.

5.1. The R2U2 Monitoring System

Whereas DR as a diagnostic engine is mainly used for fault detection and diagnosis (Schumann, Mahadevan, Sweet, et al., 2019), the R2U2 (Realizable, Responsive, and Unobtrusive Unit) (Rozier & Schumann, 2017; Reinbacher, Rozier, & Schumann, 2014; Geist, Rozier, & Schumann, 2014) is an on-board monitoring system to continuously monitor system and safety properties of the aircraft during flight. Health models within this framework (Schumann, Rozier, et al., 2015) are defined using Metric Temporal Logic (MTL) and Mission-time Linear Temporal Logic (LTL) (Reinbacher et al., 2014) for expressing temporal properties as well as Bayesian Networks (BN) for probabilistic and diagnostic reasoning. A signal processing unit reads in continuous sensor signals or information from the prognostics unit and performs filtering and discretization operations.

A large number of safety and performance properties can be formulated using temporal logic. Properties range from simple instantaneous ones, like $U_{batt} > 13.5\,\mathrm{V}$ indicating that



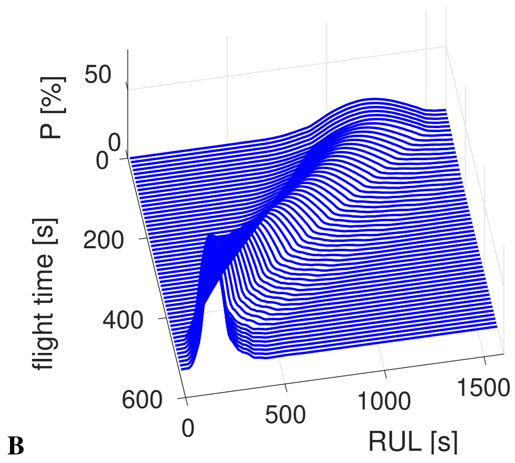


Figure 7. A: Measured battery voltage and current, estimated SOC and RUL for a constant load profile. The UAS takes off at $t = 100s$ and touches down at $t = 700s$. The system is turned off at $t = 930s$. B: Probability distribution for RUL over flight time.

the battery voltage must never drop below 13.5 V to complex temporal specifications, for example

$$\Box((I_{batt} > 30\,\mathrm{A})\mathcal{U}_{[0\,\mathrm{s},29\,\mathrm{s}]}(I_{batt} \leq 30\,\mathrm{A}))$$

This property is violated, if more than 30 A are drawn for more than 30 consecutive seconds. The temporal "until" operator $\mathcal{U}$ monitors the temporal requirement. For the definition of all temporal operators and more examples see (Rozier & Schumann, 2017; Schumann, Roychoudhury, & Kulkarni, 2015).

There is a number of safety properties that define flight safety with respect to the future flight path of the aircraft. For example, the current flight plan requires the aircraft to fly over a high mountain. Obviously the climb to the top uses up considerable battery power, so the pilot (or autopilot) must ensure that during the 10-minute climb, the battery level, i.e., the average State of Charge (μ_{SOC}) always stays above 30%:

$$\Box(\texttt{climb-mountain} \rightarrow \Box_{[10\,\mathrm{min}]}(\mu_{SOC} > 30\%))$$

There is only one catch: R2U2 is no magic crystal ball, which can look into the future. Thus the above formula can only be evaluated after 10 minutes; prior to that time, R2U2 returns `maybe`. This is, of course, not helpful for the auto-pilot since it has to make the decision to fly over the mountain or not *prior* to even attempting the climb.

We therefore integrate the prognostics engine into the R2U2 framework. We now can formulate safety properties that directly access prognostics information. The above safety property now would be simply formulated as:

$$\Box(\texttt{climb-mountain} \rightarrow \mathit{RUL}(\texttt{FP}) > 10\,\mathrm{min}),$$

where $\mathit{RUL}(\texttt{FP})$ is the rest of useful life estimate after given the flight plan `FP` has been executed. This formula can be evaluated immediately and be the basis of the autonomous decision to keep or change the current flight plan (see Section 6 for a detailed example).

5.2. Example: Monitoring the SOC Estimate

Figure 7 shows the results of a test flight with AOS running on an X8+ octo-copter at the NASA Ames Research Center. The X8+ was powered by two parallel banks of Tattu 22.2V 25C 6S 22000mAh LiPo Battery packs. Figure 7A shows measured battery voltage and current as well as the SOC in % during the flight as estimated by the on-board prognostics. The bottom panel shows the estimated RUL for a predicted constant current draw of $\bar{I} = 20A$. Since this scenario does not involve any strong climbs, the estimation of RUL with a constant load seems to be appropriate. Uncertainties in the current draw can be modeled in our architecture by carrying out the RUL estimation for samples from a Gaussian distributed load current $I = \mathcal{N}(\bar{I}, \sigma^2)$. Figure 7B shows how the probability density function (PDF) for load current of $I[A] = \mathcal{N}(20, 4)$. develops during the flight. As expected, the mean of RUL is a straight line as in Figure 7A and the variance of RUL decreases toward the end of the flight experiment.

6. PROGNOSTICS FOR ON-BOARD OPERATIONS

For successful and safe autonomous operations, the UAS must be capable of adjusting the current mission and flight plan according to the current system status and environmental conditions. The prognostics component play a central role in finding out, which flight plans and trajectories are possible and compatible with the current and predicted battery status. Note that this dynamic analysis and re-planning of trajectories is a part of nominal autonomous aircraft operations and in contrast to contingency planning, which will be discussed in Section 7.

In this paper, we present a simulation case study with a simulation model of the Langley Edge R5 aircraft (Hogge et al., 2015). This aircraft is a large RC-style aircraft with 6 ft wingspan and a take-off weight of approximately 18 kg. This aircraft is in active use at NASA LaRC for UAS-related research. An existing model for the dynamics and the battery of the aircraft made it a suitable candidate for this case study. The Edge was powered by 2S3P configuration of Thunderpower 7800mAH battery packs. In an UAM (Urban Area Mobility) scenario, the aircraft must find as suitable trajectory between points A and B. Such a trajectory must obey numerous constraints (restricted zones, flight time, other aircraft, weather, etc), one of the most important ones concern the availability of enough battery power. In our simple demonstration scenario, the shortest direct path is flying over a mountain, a distance of approximately 14.3nm , requiring a strong climb of 4,600 vertical feet. An alternative trajectory around the mountain does not require any climbs/descends, but is substantially longer (17.3nm). Here, we analyze (a) flying over the mountain, (b) going around the mountain at level altitude with high speed (60 kts), and (c) going around the mountain with an energy-conserving 40 kts. While the aircraft is in the air, R2U2 and the prognostics engines continuously monitor the battery of the aircraft and calculate SOC and RUL for existing flight plan. If there is an opportunity or need to evaluate a different future flight plan (e.g., to slowly go around the mountain), PDFs for the RUL of the various trajectories are calculated on-board.

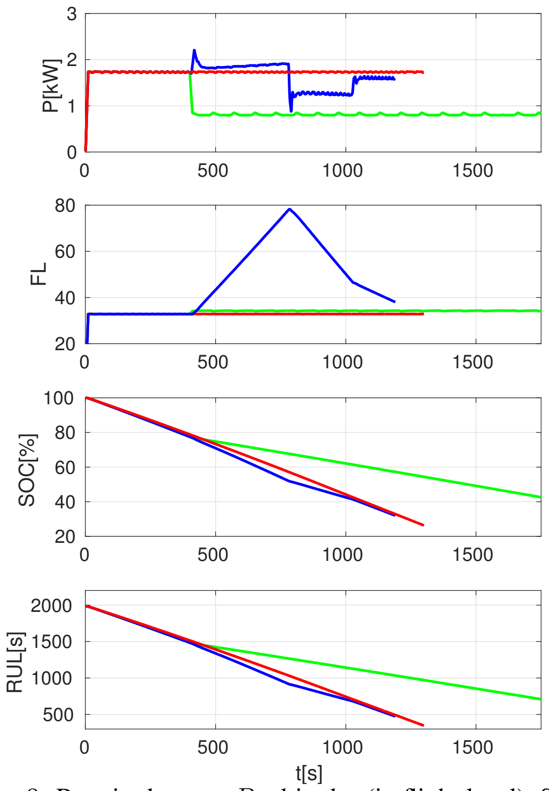


Figure 8. Required power P, altitudes (in flight level), SOC, and RUL with constant power use for three trajectory variants (a)—blue, (b)—red, and (c)—green.

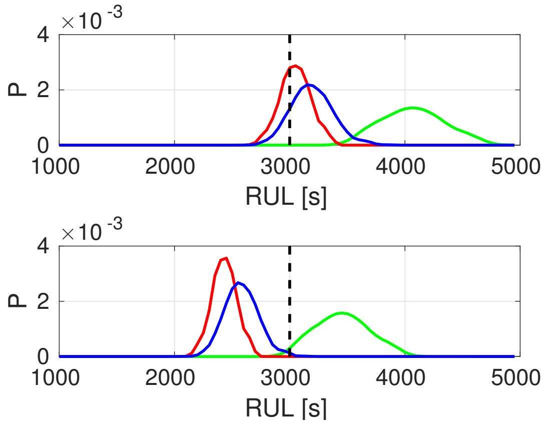


Figure 9. Probability densities for RUL at the beginning of the flight $t = 0s$ (top) and at the decision point $t = 400s$ (bottom) with $\sigma_{pwr} = 0.1$ for variants (a)—blue, (b)—red, and (c)—green.

For these flight-plans, the power consumption is being calculated using a simplified model (Schumann, Roychoudhury, & Kulkarni, 2015). The RUL calculation during the flight uses the actual battery state into account, which is updated using measured power consumption of the aircraft. Figure 8 shows the actual power requirements for trajectory variants (a), (b), and (c), and the altitude profile in flight levels (FL). For the first 400 seconds all trajectories are the same. Obviously, the higher power consumption of (a) and (b) cause a lower SOC, which goes below 50%. Note that the mountain route and fast flight around the mountain reaches the destination earlier; hence the lines are shorter. If we calculate the RUL according the to high power usage similar to that in the initial flight segment, the faster, but power-hungrier variants end up with a shorter RUL.

If we require a minimum RUL of at least 3,000 seconds at the destination, both SOC and RUL estimation do not allow help us in the early stages of the flight to make a decision on which route can actually be flown. Only some time after our decision point at $t = 400$, we could see that only the slow route around the mountain allows for enough safety margin. We therefore perform RUL predictions using the variants of the intended flight path. In order to accommodate uncertainties, which might, for example, arise due to weather, we draw the power requirements from a Gaussian distribution. If current wind-speed and direction were available, an even more accurate prediction of RUL would be possible.

Figure 9 shows the resulting RUL distribution at the beginning of the flight ($t = 0$). At that point in time, all three variants seem to be possible, albeit variant (b) is risky, when we require a minimum RUL of 3000s (top panel). At $t = 400s$ (just before the decision has to be made), the situation looks differently: variants (a) and (b) are not safe, only the slow flight around the mountain obeys our safety requirements,

7. PROGNOSTICS FOR CONTINGENCY PLANNING

In the case of a failure or an off-nominal situation, an autonomous aircraft must be able to detect and correctly diagnose the fault conditions and prepare a suitable contingency plan to mitigate the failure or safely modify or end the mission, all without human intervention or mandatory ground support. Here again, prognostics play an important role during the construction of the contingency plan. If, for example, the UAS has to divert to an emergency airport, then prognostics is needed to determine if there is enough battery capacity left to even reach that airport and land there. Otherwise, other contingency measures (e.g., land at nearest landing spot) need to be considered right away.

Within the AOS system, the Decision Maker (DM) is in charge of contingency planning. As shown in Figure 1 above, the DM receives continuous inputs from the diagnostics, monitoring, and prognostics components carrying information about the current system health. Based upon the current information, the DM first tries to disambiguate the failure modes using active diagnosis procedures (see (Schumann, Mahadevan, Lowry, & Karsai, 2019) for details), before using a logic-based search algorithm to construct a contingency plan using model-based contingency schemas. Contingency actions include: active diagnosis, aircraft reconfiguration (e.g., using the second battery pack), taking a shortcut (e.g., skipping a delivery spot), deviate to an emergency airport, land a the nearest safe landing spot, or pull the parachute for a soft crash landing. These schemas, which are listed in Table 3, are then instantiated and "plugged" together to form a contingency plan that is ultimately sent to the PLEXIL planner for execution. During that schema-based search, the prognostics engine must be consulted multiple times as most schemas in Table 3 need prognostics evaluation. Any potential modification of the flight plan, e.g., by taking a short cut or deviate to an emergency airport, but also an aircraft reconfiguration triggers a prognostics evaluation. Only if the resources are estimated to be sufficient, a contingency plan can be considered. In our AOS architecture, DM uses a simplistic but efficient on-board prognostics module during the search; potential contingency plans are then checked by the hybrid prognostics engine.

Let us consider the following small example: the UAS, currently at position X (Figure 10) has to fly along waypoints $[A, B, C, D, E, L]$, where L is the final destination airport. Between waypoints C and D there is a substantial mountain range. The regular flight plan calls for a traverse over the mountains; an alternative, but much longer route around the mountains would be $[A, B, C, F, G, E_2, L]$. E_1 and E_2 denote alternate emergency airports. At the current point X, the aircraft is subjected to a failure, which is diagnosed as a stuck elevator (see (Schumann, Mahadevan, Sweet, et al., 2019) for details on the failure mode). The stuck elevator has substantial impact on the performance of the aircraft; the DM consults the on-board failure models to determine that, in this case, strong climbs are not safe.

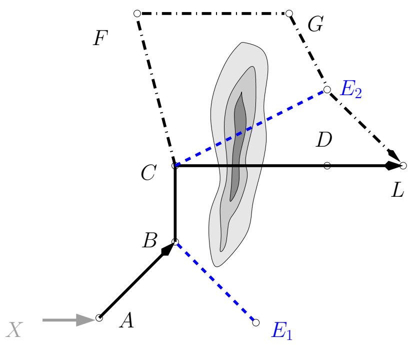
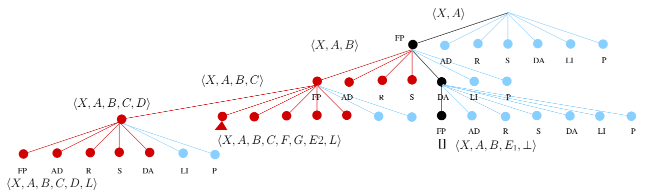


Figure 10. Diagram of waypoints with alternatives (dot-dashed), possible divergence routes (dashed) and emergency airports E_1, E_2.

The DM now performs a search, trying to apply the schemas from Table 3 starting with the one with least severity and impact. During the planning search, the DM finds out that it could follow the original flight plan along the waypoints $[A, B, C]$, but cannot traverse to D due to the mountain range. A short-cut (fly directly to L) or a deviation to E_2 are not possible because of the mountains. The alternative route, continuing via $[F, G, E_2, L]$ is ruled out, because the battery resources would not be sufficient, as determined by the on-board prognostics. The search now back-tracks to waypoint B: short-cut to D is not possible, again, because it would lead over the mountains. However, a deviation to E_1 is possible, because no strong climb is needed. During the search the potential use of a schema also requires calls to the prognostics



$\langle X, A\rangle$
FP
AD
R
S
DA
LI
P
$\langle X, A, B\rangle$
FP
AD
R
S
DA
LI
P
$\langle X, A, B, C\rangle$
FP
AD
R
S
DA
LI
P
$\langle X, A, B, C, D\rangle$
$\langle X, A, B, C, F, G, E2, L\rangle$
[] $\langle X, A, B, E_1, \perp\rangle$
FP
AD
R
S
DA
LI
P
$\langle X, A, B, C, D, L\rangle$

Figure 11. Search tree of DM for situation in Fig. 11. Current path (black), unsuccessfully explored (red), potentially explorable (blue). The small red triangle corresponds to the search along the alternative route $[A, B, C, F, G, E_2, L]$, which is rejected by prognostics due to insufficient battery resources.

Table 3. Contingency schemas with severity S. w_i are waypoints and PRG denotes if prognostics is needed for schema evaluation.

	Action $\mathcal{A}$	S	Contingency plan	PRG	Description
[]	empty	0	[]	N	mission G concluded, no action to be taken
FP	flight-plan	0	$\langle w_i, ...w_\perp\rangle$	Y	follow flight plan to next waypoint w_i
AD	active_diagnosis	0–2	$\langle w_k, D_a, ...\rangle$	N	perform active diagnosis D_a at waypoint w_k
R	reconfig	1	$\langle w_i, w_{i+1}, ...\rangle$	Y	reconfigure aircraft (e.g., alternate battery or sensors)
S	shortcut	1	$\langle w_{i-1}, w_{i+1}, ...\rangle$	Y	skip waypoint w_i. Fly directly to w_{i+1}
DA	deviate-airport	2	$\langle w'_1, w'_2, \ldots, w'_\perp\rangle$	Y	deviate to emergency airport $w'_\perp$. Mission G terminated
LI	land-immediate	3	$\langle w'\rangle$	Y	land at nearest safe waypoint w'. Mission is terminated
P	parachute	4	[]	N	pull parachute at current location. G is terminated

system. In our case, the DM uses prognostics information to find out if there is enough battery remaining to fly the route $[X, A, B, E_1]$, which it is. Finally, Figure 11 shows a graphical representation of the search tree for that scenario.

This example is part of a simulation case study for a fixed-wing UAS described in (Schumann, Mahadevan, Sweet, et al., 2019). A related scenario concerning an altimeter sensor failure has been successfully test-flown at NASA ARC on a DJI S1000+ (Schumann, Mahadevan, Lowry, & Karsai, 2019).

8. RELATED WORK

Systems Health Management framework applied to aerospace domain has been discussed in (Johnson et al., 2011; Orsagh, Brown, Romer, Dabnev, & Hess, 2005; Ashby & Byer, 2002; Hess, 2002; Hess, Calvello, & Frith, 2005; Millar, 2007) ranging from structural, machine, avionics systems, etc. Prognostics-based decision making in the Aerospace domain has been addressed, for example in (Balaban & Alonso, 2012). Here, techniques from optimization and game theory have been used for Dynamic Constraint Redesign (DCR), which enables decision making in a continuous space and deals with mission reconfiguration. The underlying numerical algorithms, based, for example on Particle Filters (Sweet et al., 2014) have been employed for different autonomous vehicles. This formulation can deal with complex and continuous mission re-configurations but is mathematically more challenging and has a substantial computational footprint.

Due to restricted on-board computing capabilities, contingency management for autonomous and electric aircraft have been restricted to fail-safe operations, like loitering, immediate landing or return to home plate. Approaches to more complex contingency planning for UASs have been developed for the planning and pre-flight assessment (DiFelici & Wargo, 2016) or concern a "holistic" multi-level contingency management system that spans UAS, communications, weather, and battle teams (Franke, Hughes, & Jameson, 2006). The actual planning uses the Lockheed Martin tool TeamWorks (Franke et al., 2006) and only has limited control or monitoring capabilities on-board. On-board path planning by dynamic probabilistic reconfiguration is described in (Wzorek & Doherty, 2006), but does not incorporate diagnosis or failure-based contingency management.

A system for flight mission planning, which addresses the specific problem of autonomous battery recharging is presented in (Tseng, Chau, Elbassioni, & Khonji, 2017). It uses elaborate graph optimization algorithms, but is not being run on-board.

9. CONCLUSIONS

In this paper, we discussed how prognostics plays an important role for autonomous electric-propulsion aircraft. Obviously, the current state of the battery as well as good estimates on the rest of useful life (RUL) for the flight battery is most essential for all aircraft with electric propulsion, whether piloted or not. We presented a chemo-electrical battery model that allow the accurate determination of the state of charge and discussed methods for prognostics and general prognostics architecture.

Our main focus of this paper is on how prognostics can facilitate or even enable safe and effective autonomous operations. Based upon the case study of the NASA Autonomous Operating System (AOS), we discuss: prognostics for safety and health monitoring during the flight, prognostics for flight operations and trajectory planning, and prognostics for on-board contingency planning. For each of these application areas, we motivated the necessity and presented results of a simulation case study or an actual test flight with an electric UAS.

Although all autonomy components are operating on-board the vehicle, the high computational footprint of model-based prognostics algorithms makes it viable to run certain prediction tasks on a ground-based server architecture. In this paper, we presented Prognostics as a Service (PaaS) and discussed it architecture and integration into the on-board AOS autonomy software.

The results of our case studies show that accurate, real-time prognostics play an important role for autonomous aircraft during nominal operations and off-nominal situations. However, there is a number of open areas, which will part of future work. Prognostics is always associated with uncertainties, which advanced prognostics algorithms can estimate. Carrying over this probabilistic information toward the on-board reasoning will make it possible to yield more justifiable decisions with attached confidence metrics. Research will also be necessary on how to extend the application of prognostics beyond a single mission of a single UAS: how can long-term effects like battery ageing and prognostics for multiple vehicles or fleets be modeled and efficiently integrated into the on-board systems. Finally, model-based prognostics of other safety-critical hardware components, like engines, motor bearings, or hydraulic subsystems must be integrated into the autonomous software system in order to improve safety, reliability, and effectiveness of UAS operations and maintenance.

REFERENCES

Ashby, M. J., & Byer, R. J. (2002). An approach for conducting a cost benefit analysis of aircraft engine prognostics and health management functions. In *Proceedings of the 2002 IEEE Aerospace Conference* (Vol. 6).

Balaban, E., & Alonso, J. (2012). An Approach to Prognostic Decision Making in the Aerospace Domain. In *Proceedings of the Annual Conference of the Prognostics and Health Management Society 2012.*

Beckman, B. C., Haskin, M., Rolnik, M., & Vule, Y. (2017). *Maneuvering a package following in-flight release from an unmanned aerial vehicle (UAV).* Google Patents. (US Patent 9,567,081)

Caccamo, M. (2017). *Power-aware emulation environment for long-endurance solar UAVs.* Retrieved from `rtsl-edge.cs.illinois.edu/UAV/inc/CNS-1646383_Poster_Nov-2017.pdf`

Corbetta, M., Banerjee, P., Okolo, W. A., Gorospe, G. E., & Luchinsky, D. G. (2019). Real-time UAV Trajectory Prediction for Safety Monitoring in Low-Altitude Airspace. In *AIAA 2019-3514. Session: UAS Traffic Management IV.* doi: 10.2514/6.2019-3514

Daigle, M., & Goebel, K. (2013). Model-based prognostics with concurrent damage progression processes. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *43*(4), 535-546.

Daigle, M., & Kulkarni, C. (2013). Electrochemistry-based Battery Modeling for Prognostics. In *Proceedings of the Annual Conference of the Prognostics and Health Management Society 2013* (p. 249-261).

Daigle, M., Kulkarni, C., & Gorospe, G. (2014). Application of Model-based Prognostics to a Pneumatic Valves Testbed. In *Proceedings of the 2014 IEEE Aerospace Conference.*

Daigle, M., Saha, B., & Goebel, K. (2012). A comparison of filter-based approaches for model-based prognostics. In *Proceedings of the 2012 IEEE Aerospace Conference.*

Daigle, M., Saxena, A., & Goebel, K. (2012). An efficient deterministic approach to model-based prediction uncertainty estimation. In *Proceedings of the Annual Conference of the Prognostics and Health Management Society 2012* (p. 326-335).

DiFelici, J., & Wargo, C. (2016). UAS safety planning and contingency assessment and advisory reseach. In *2016 Integrated Communications Navigation and Surveillance (ICNS)* (pp. 8E3-1 – 8E3-16).

Federal Aviation Administration. (2020). *Unmanned Aircraft System (UAS) Traffic Management (UTM): Concept of Operations V2.0.* Retrieved from `https://www.faa.gov/uas/research_development/traffic_management/media/UTM_ConOps_v2.pdf`

Franke, J. L., Hughes, A., & Jameson, S. C. (2006). Holistic contingency management for autonomous unmanned systems. In *Proceedings of the AUVSIs Unmanned Systems North America.*

Gašparović, M., & Gajski, D. (2016). Unmanned Aerial Photogrammetric Systems in the Service of Engineering Geodesy. In *International Symposium on Engineering Geodesy-SIG* .

Geist, J., Rozier, K. Y., & Schumann, J. (2014). Runtime Observer Pairs and Bayesian Network Reasoners Onboard FPGAs: Flight-Certifiable System Health Management for Embedded Systems. In *Proceedings Runtime Verification (RV14)* (pp. 215–230). Springer.

Gómez-Candón, D., De Castro, A., & López-Granados, F. (2014). Assessing the accuracy of mosaics from unmanned aerial vehicle (UAV) imagery for precision agriculture purposes in wheat. *Precision Agriculture*, *15*(1), 44–56.

Gorospe, G. E., Daigle, M. J., Sankararaman, S., Kulkarni, C. S., & Ng, E. (2017). GPU Accelerated Prognostics. In *Proceedings of the Annual Conference of the Prognostics and Health Management Society 2017.*

Hess, A. (2002). Prognostics, from the need to reality-from the fleet users and PHM system designer/developers perspectives. In *Proceedings of the 2002 IEEE Aerospace Conference* (Vol. 6, p. 6-6). doi: 10.1109/AERO.2002.1036118

Hess, A., Calvello, G., & Frith, P. (2005). Challenges, issues, and lessons learned chasing the "Big P". Real predictive prognostics. Part 1. In *Proceedings of the 2005 IEEE Aerospace Conference* (p. 3610-3619). doi: 10.1109/AERO.2005.1559666

Hogge, E., Bole, B., Vazquez, S., Strom, T., Hill, B., Smalling, K., & Quach, C. (2015). Verification of a Remaining Flying Time Prediction System for Small Electric Aircraft. In *Proceedings of the Annual Conference of the Prognostics and Health Management Society 2015.*

Johnson, S., Gormley, T., Kessler, S., Mott, C., Patterson-Hine, A., Reichard, K., & Philip Scandura, J. (2011). *System Health Management with Aerospace Applications*. Wiley & Sons.

Jones, R. W., & Despotou, G. (2019). Unmanned aerial systems and healthcare: Possibilities and challenges. In *14th IEEE Conference on Industrial Electronics and Applications (ICIEA)* (pp. 189–194).

Kulkarni, C., Daigle, M., Gorospe, G., & Goebel, G. (2014). Validation of Model-Based Prognostics for Pneumatic Valves in a Demonstration Testbed. In *Proceedings of the Annual Conference of the Prognostics and Health Management Society 2014.*

Lisso, G. K. (2017). *Delivery of packages by unmanned aerial vehicles.* Google Patents. (US Patent 9,536,216)

Lowry, M., Bajwa, A. R., Pressburger, T., Sweet, A., Dalal, M., Fry, C., … Mahadevan, N. (2018). Design Considerations for a Variable Autonomy Exeuctive for UAS in the NAS. In *AIAA Information Systems-AIAA Infotech @ Aerospace.*

Lowry, M., Pressburger, T., Dahl, D., & Dalal, M. (2019). Towards Autonomous Piloting: Communicating with Air Traffic Control. In *AIAA Scitech Forum.*

Luo, J., Pattipati, K. R., Qiao, L., & Chigusa, S. (2008, September). Model-based prognostic techniques applied to a suspension system. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, *38*(5), 1156 -1168.

McComas, D. (2012). NASA/GSFC's Flight Software Core Flight System. In *Flight Software Workshop.*

Millar, R. C. (2007). A Systems Engineering Approach to PHM for Military Aircraft Propulsion Systems. In *Proceedings of the 2007 IEEE Aerospace Conference* (p. 1-9). doi: 10.1109/AERO.2007.352840

Orchard, M., Tobar, F., & Vachtsevanos, G. (2009, December). Outer feedback correction loops in particle filtering-based prognostic algorithms: Statistical performance comparison. *Studies in Informatics and Control*, *18*(4), 295-304.

Orsagh, R., Brown, D., Romer, M., Dabnev, T., & Hess, A. (2005). Prognostic health management for avionics system power supplies. In *Proceedings of the 2005 IEEE Aerospace Conference* (p. 3585 - 3591).

Polka, M., Ptak, S., & Kuziora, L. (2017, 12). The Use of UAV's for Search and Rescue Operations. *Procedia Engineering*, *192*, 748-752.

Reinbacher, T., Rozier, K. Y., & Schumann, J. (2014). Temporal-Logic Based Runtime Observer Pairs for System Health Management of Real-Time Systems. In *Tools and Algorithms for the Construction and Analysis of Systems - 20th International Conference, TACAS* (Vol. 8413, pp. 357–372). Springer.

Rozier, K. Y., & Schumann, J. (2017). R2U2: Tool Overview. In *Proceedings of RV-CuBES 2017* (pp. 138–156).

Sankararaman, S., & Teubert, C. (2017). Prospective architectures for onboard vs cloud-based decision making for unmanned aerial systems. *Proceedings of the Annual Conference of the Prognostics and Health Management Society 2017*.

Schumann, J., Mahadevan, N., Lowry, M., & Karsai, G. (2019). Model-Based On-Board Decision Making for Autonomous Aircraft. In *Proceedings of the Annual Conference of the Prognostics and Health Management Society 2019.*

Schumann, J., Mahadevan, N., Sweet, A., Bajwa, A. R., Lowry, M., & Karsai, G. (2019). Model-based System Health Management and Contingency Planning for Autonomous UAS. In *AIAA Scitech Forum.*

Schumann, J., Roychoudhury, I., & Kulkarni, C. (2015). Diagnostic Reasoning using Prognostic Information for Unmanned Aerial Systems. In *Proceedings of the Annual Conference of the Prognostics and Health Management Society 2015.*

Schumann, J., Rozier, K. Y., Reinbacher, T., Mengshoel, O. J., Mbaya, T., & Ippolito, C. (2015). Towards Real-time, On-board, Hardware-supported Sensor and Software Health Management for Unmanned Aerial Systems. *International Journal of Prognostics and Health Management*.

Swagger. (2020). *Open API Specification.* Retrieved from `https://swagger.io/docs/specification/about/`

Sweet, A., Gorospe, G., Daigle, M., Celaya, J. R., Balaban, E., Roychoudhury, I., & Narasimhan, S. (2014). Demonstration of Prognostics-Enabled Decision Making Algorithms on a Hardware Mobile Robot Test Platform. In *Proceedings of the Annual Conference of the Prognostics and Health Management Society 2014.*

Teubert, C., & Daigle, M. (2013). I/P Transducer Application of Model-Based Wear Detection and Estimation using Steady State Conditions. In *Proceedings of the Annual Conference of the Prognostics and Health Management Society 2013* (p. 134-140).

Teubert, C., & Daigle, M. (2014). Current/Pressure Transducer Application of Model-Based Prognostics using Steady State Conditions. In *Proceedings of the 2014 IEEE Aerospace Conference.*

Teubert, C., Daigle, M. J., Sankararaman, S., Goebel, K., & Watkins, J. (2017). A Generic Software Architecture for Prognostics (GSAP). *International Journal of Prognostics and Health Management*, *8*(2).

Thiels, C. A., Aho, J. M., Zietlow, S. P., & Jenkins, D. H. (2015). Use of Unmanned Aerial Vehicles for Medical Product Transport. *Air Medical Journal*, *34*(2), 104 – 108.

Tseng, C., Chau, C., Elbassioni, K. M., & Khonji, M. (2017). Flight Tour Planning with Recharging Optimization for Battery-operated Autonomous Drones. *CoRR*, *abs/1703.10049*.

Verma, V., Jonsson, A., Pasareanu, C., & Iatauro, M. (2006). Universal Executive and PLEXIL: Engine and Language for Robust Spacecraft Control and Operations. In *Spacecraft Control and Operations, American Institute of Aeronautics and Astronautics Space 2006 Conference.*

Watkins, J., Teubert, C., & Ossenfort, J. (2019). Prognostics As-A-Service: A Scalable Cloud Architecture for Prognostics. *Proceedings of the Annual Conference of the Prognostics and Health Management Society 2019.*

Wzorek, M., & Doherty, P. (2006). Reconfigurable Path Planning for an Autonomous Unmanned Aerial Vehicle. In *International Conference on Hybrid Information Technology* (Vol. 2, pp. 242–249).

BIOGRAPHIES

Dr. Johann Schumann received his PhD (1991) and German habilitation degree (2000) in Computer Science from the Technische Universität München (TUM) in Germany. He is engaged in research on system and software health management, autonomy for UAV, V&V of advanced air traffic control systems, and the automatic generation of reliable code. Dr. Schumann is author of a book on theorem proving in software engineering and has published numerous articles on automated deduction, automatic program generation, V&V of safety-critical systems, and neural network oriented topics.

Dr. Chetan Kulkarni received the B.E. degree from University of Pune, India in 2002 and the M.S. and Ph.D. degrees from Vanderbilt University, Nashville, TN, in 2009 and 2013, respectively. His current research interests include physics-based modeling, model-based diagnosis and prognosis for complex systems, hybrid prognostics frameworks, decision making. Dr. Kulkarni is Associate Fellow AIAA and SM IEEE. He is Associate Editor of IEEE Advances in Prognostics and System Health Management and SAE Journal of Aerospace.

Dr. Michael Lowry is the NASA chief scientist for Reliable Software Engineering. After receiving his BS/MS from MIT and PhD from Stanford, all in computer science, he joined the Kestrel Institute as PI working on program synthesis. In 1993 he joined NASA Ames as group lead then area lead, and was promoted to chief scientist in 2008. Dr. Lowry is the editor of MIT Press "Automating Software Design" and serves on the editorial board of the journal Automated Software Engineering. He has published numerous papers principally on the topics of program synthesis and software V&V.

Dr. Anupa Bajwa is the Tech Area Lead for Discovery and Systems Health in the Intelligent Systems Division. As a researcher at NASA Ames for over two decades she has enabled autonomous capabilities for spacecraft and for unmanned aerial vehicles. She co-led ARMD's Autonomy Operating System project at Ames and was project manager for STMD's Autonomous Systems. She led the Fault Management team on the 2013 LADEE lunar orbiter mission. She has worked on implementing Integrated Systems Health Management for launch vehicles on projects such as PITEX and Constellation using model-based systems engineering. She has a Ph.D. in Aerospace Engineering from the Pennsylvania State University, an M.S. in Aeronautical and Astronautical Engineering from Ohio State University, and a B.Tech. in Aeronautical Engineering from the Indian Institute of Technology, Bombay. She holds a certificate in Corporate Innovation, a LEAD Program in Stanford's Graduate School of Business.

Christopher Teubert is the principal investigator for the Generic Software Architecture for Prognostics (GSAP). Christopher received his B.S. in Aerospace Engineering from Iowa State University in 2012 and his M.S. in Computer Science and Engineering from Santa Clara University in 2019. Chris worked as a research engineer with Stinger Ghafarrian Technologies (SGT) at NASA Ames Research Center from 2012-2016. Since 2016, Chris has been a computer engineer and lead with the NASA Diagnostics and Prognostics group. Since 2018, Chris has been serving as technical lead for autonomous systems.

Jason Watkins is a software engineer with KBR working for the Diagnostics and Prognostics group at NASA Ames Research center. He currently works on the System-Wide Safety and Volatiles Investigating Polar Exploration Rover (VIPER) projects at NASA Ames. Jason received his B.S. in Computer Science from University of California, Irvine in 2018. Prior to graduating and joining KBR full time, Jason completed several undergraduate internships with NASA. As part of those internships, Jason has worked on several NASA software projects, including the open source projects X-Plane Connect and Generic Software Architecture for Prognostics (GSAP).