Mastering Training Data Generation for AI - Integrating High-Fidelity Component Models with Standard Flight Simulator Software

Andreas Löhr[1], Conor Haines[2]

[1,2]*Linova Software GmbH, München, Bavaria, 80805, Germany*
andreas.loehr@linova.de
conor.haines@linova.de

ABSTRACT

The German state-funded aviation research project "Real-time Analytics and Prognostic Health Management" (RTAPHM) envisioned fully automated urban air services executed by autonomous drones and infrastructure controlled by a digital system. Research was focused on utilizing onboard real-time diagnostics to enable AI-driven UAV capability predictions. These predictions increased the reliability of upfront service commitments. The use case selected to demonstrate these elements was organ transport. The project delivered an end-to-end demonstrator incorporating a virtual fleet of drones with onboard diagnostics to provide data for the platform decision logic.

The project followed a „digital-twin-first" approach to overcome a common bootstrapping problem faced by data-driven applications. That is, the lack of in-service data for exploration, prototyping and training of diagnostic and prognostic approaches during the concept and early development phases. Due to the upfront development of physical high-fidelity simulation models for the monitored components, a digital twin – of the portion of the twin that resembles the physical behavior – was used to generate data and facilitate preliminary exploration, prototyping and training. Digital twins were further employed to allow evaluation of what-if scenarios and identify the optimal future operation parameters of a drone.

Development of the RTAPHM digital twin involved a multi-disciplinary team of members distributed across different organizations and locations. Successful realization of the digital twin depended on early integration testing, performed in high frequencies, which generated continuous feedback regarding technical and conceptual issues. Within the research project we developed MOLE, an engineering tool for automating the integration of distinct simulation components, into a single system simulation driven by commercially available flight simulator software. Here, we showcase the internal mechanisms of the tool and demonstrate its abilities to generate a Docker-based executable for efficient data generation in the cloud. We also show our approach to online visualization, fault insertion, batch integration testing and debugging the digital twin executable. We also report on the utilization of MOLE in assembling the final RTAPHM demonstrator (Löhr, 2023).

1. OUTLINE

The document first introduces the RTAPHM project with a focus on the use and purpose of digital twins. This leads to our primary project contribution: MOLE, a software tool assisting in the fast integration of digital twin components. After describing the core principles of MOLE, we report from our experience in using MOLE to build the digital twin for the RTAPHM demonstrator. We conclude with a suggestion for areas of future work.

2. INTRODUCTION

The project Real-time Analytics and Prognostic Health Management (RTAPHM) was a joint government-funded research endeavor in Germany It was in the aviation domain and included SMEs, academia, and industry as partners. The project concluded in June 2023. We participated as an SME partner focused on simulation and software engineering IVHM systems.

The project contributed to the field of urban air services, such as person and cargo transport, imaging services, etc. The vision of a fully automated urban air services platform drove the work. Here, customers could book a variety of different services via low-threshold access channels, e.g., via their smart phone. Once booked, the service would be carried out by autonomous unmanned air vehicles (UAVs) supported by autonomous infrastructure, e.g., for mounting payload or loading cargo. The platform would have to work digitally to

benefit from automation. AI technologies would support the orchestration multiple booked services in parallel and the handling of problems which could not be sufficiently formalized upfront.

Obviously, such a bold vision cannot be accomplished by a single research project. Therefore, project research focus was narrowed to a single use case highlighting a specific problem within that use case. The project selected organ transport as that use case. By the means of a digital service platform, the personnel of a donor hospital would book (air) transport to a specific receiver hospital, as depicted in Figure 1. The last mile between the hospitals and the next logistics hub should be covered with transport UAVs.

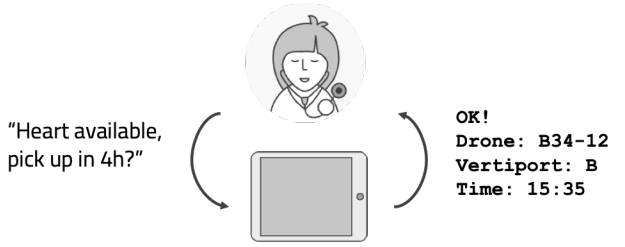

Figure 1. Synopsis of selected use case.

Within the use case, the project focused on the issue of how to make reliable service commitments given a fleet of UAVs (or other transportation vehicles for long haul routes) depending on their current and future health state. That is, build a platform capability to assess whether a specific service request can be completed with the available resources. This question should be answered by a comprehensive situational awareness picture of the fleet's conditions, mainly driven by real-time diagnostics and prognostic capabilities, integrated into the onboard data processing of the UAVs. Figure 2 depicts the research focus on a high level: for selected components of an artificial UAV (fuel cell, servo and pusher motor) a monitoring concept should be established and implemented. The obtained (gradient of the) health data should be used to make reliable maintenance predictions for the components. Finally, by having predicted the future maintenance burden – thus, the availability of individual UAVs – incoming service bookings for organ transports would be committed reliably within a timeframe where the selected UAV is available and maintenance free.

3. RTAPHM DIGITAL TWIN

The project selected three components to be monitored: servo motors, a fuel cell and pusher propeller system. For all components, a monitoring concept, including diagnostics and prognostics, was developed, and included in the laboratory demonstrator.

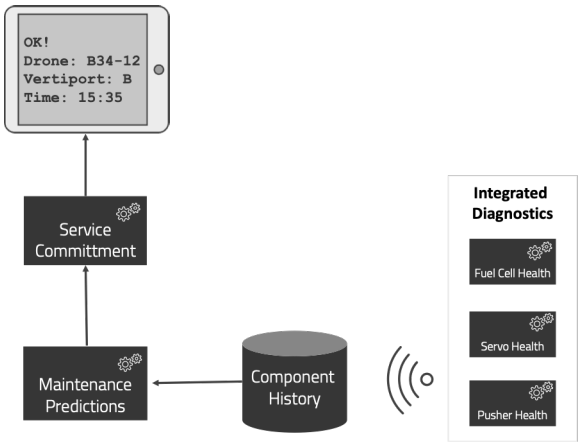

Figure 2. Research focus.

3.1. Motivation

The project faced a common bootstrapping problem, as depicted in Figure 3: incomplete, insufficient, or total absence of operational data from the components in question (the specific configuration of the targeted UAV, as well as the reason for the data absence, is out of the scope of this writing). There was no foundation for conducting exploration or conceptual studies.

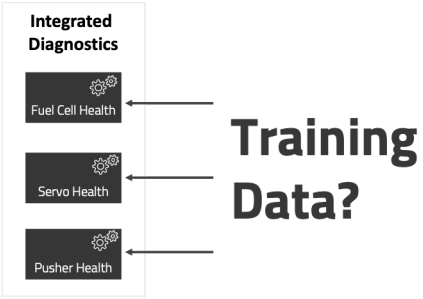

Figure 3. Problem statement.

3.2. Digital-Twin-First Approach

The concept of a digital twin is known in both academia and industry. A digital twin is typically derived by observing (data from) an existing system using specific frameworks and methodologies, as for instance Vuckovic, Prakash, and Burke (2023) have shown. The project decided to overcome the bootstrapping problem by employing simulation. The project already incorporated the usage of digital twins within the digital platform. This existing capability and the absence of existing physical components informed the decision to proceed with a "digital-twin-first" approach instead of creating the digital twin afterwards.

To accomplish this, the partners agreed to provide validated high-fidelity component simulation models adapted from previous undertakings. These models formed the bases of the

digital twin. Within the project, these models were adapted according to the performance requirements of the targeted UAV platform and equipped with additional technical features to facilitate integration. Additionally, to stimulate the models with realistic input data, the project decided to use a commercially available flight simulator software.

The solution approach is depicted in Figure 4: a commercial flight simulator should be equipped with a plugin, to extract and send specific data streams (e.g., electrical power demand, thrust demand, environmental data, etc.) to the simulation models as input. Optionally, the models should be operated in a closed loop with the flight simulator, depending on its capabilities. Being stimulated in a realistic way, the simulation models should provide simulated sensor data for the health assessment algorithms. Also, as Darrah, Frank, Quinones-Grueiro, and Biswas (2021) have pointed out, simulation allows generation of run-to-failure data – which would be an unsafe undertaking for a real system.

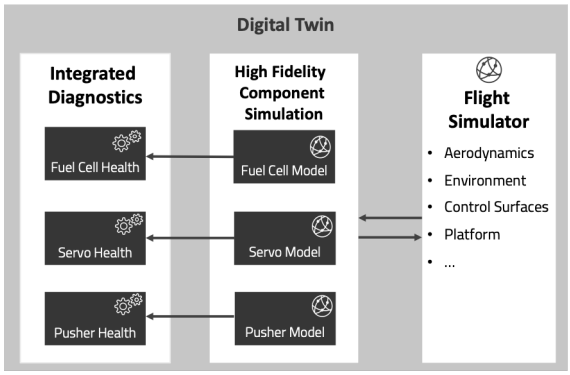

Figure 4. Solution approach.

3.3. Conceptual Challenges

Performing digital-twin-first imposes certain challenges to the modeling process and the overall concept.

This approach is used to obtain data from something that does not yet exist. But how can something that does not yet exist, be modeled? The project addressed this challenge by adapting existing work and narrowed the challenge down to finding the right scaling for the target environment.

There is a further challenge of modeling the right inputs and output of each simulation component. This was addressed by employing agile development techniques, such as a high frequency of iterations and many integration attempts with close feedback amongst all partners.

Finally, the challenge of representativeness and validity remains. Additionally, one must answer the question of how credible the resulting diagnostic and prognostic approaches can be, if they are developed on pure simulated data. There is a risk that simulation development is driven to produce what the exploiting modules expect and vice versa rather than reflecting a realistic and useful abstraction of the potential platform. This challenge was addressed by using (adapting) simulation models which had been created independently and validated in isolation. However, using simulations in this context can only be a first measure to parallelize development. As soon as the first set of "real" operational data becomes available, it must be used to tune the simulation.

3.4. Integration Challenges

Our task in the RTAPHM project, amongst others, was the provision of an integrated executable digital twin to be used by the partners to generate data according to their scenarios. Along with the task itself came the necessity to not only perform the physical integration just once towards the end of the project, as it would have been in a waterfall organized project. Instead, to adhere to the agile mindset of the digital twin development process, we had to be able to perform the physical integration as often as possible. We identified a set of challenges in the context of a multi-organization project that we had to address.

- in what form should a model be delivered?
- how can the intellectual property of a component providing partner be protected?
- what needs to be provided so that a specific model can be integrated with others?
- how can the interaction of two models be tested?
- how to be test the nominal and erroneous behavior?
- how can a quick turnaround be performed

4. MOLE

While the conceptual challenges were addressed by our partners, we worked on providing a solution to the integration challenges that the project was facing. We provided that solution on the form of MOLE – a desktop software tool for automating large portions of the digital twin integration process. Besides the capability to automatically create an executable digital twin from the provided simulation models, the tool provided support for the actual development cycle, and could be used directly by the partners.

4.1. Model Format

Our partners agreed to deliver the models as C code. Some chose to generate the C code using graphical modeling tools, such as Matlab/Simulink, while others provided custom code. Whatever the source, all models to adhered to a common interface concept, which was derived from the way that Matlab exports models. It consisted of:

- structures representing the internal state of a model
- an initialization function for the structures
- a stepping function, accepting (pointers to) the structures

- a custom naming scheme for parameters to establish semantic consistence

Intellectual property was protected by giving the partners the option to provide their models in pre-compiled object code, accompanied with their respective header files. Partners whose models depended on the models of other partners agreed on the specific information to be exchanged, and then specified the required technical parameters. We acted as a central parameter registry to enforce consistency.

As part of the conventions, simulated fault behavior was triggered via a set of up to four parameters. This set consisted of one input parameter acting as a plain on/off switch, a second defining the degree (severity) of the simulated fault, a third setting the point in time at which the fault should become effective, and a fourth setting whether the fault should become effective immediately, or if a degradation towards the specified severity should be simulated.

4.2. Automated Wiring

A significant portion of the integration work consists in the correct wiring of a model's outputs to one or mode inputs of dependent models, according to a specification given by the model creators. By "wiring" in the technical sense, we mean the temporal storage of a model's output in memory, so that it can be read by all dependent models once they start calculating their next cycle. Figure 5 shows an example of MOLEs mapping browser, a visualization of all detected inputs of a specific model, and the assigned outputs for each inputs based on naming conventions.

Model Variables	Type	Source	Initial Value	Internal Resolution
EMA_CBS	Simulink			1.0E-5
Inputs				
EMA_DBBs_PsBll[120]	double			
EMA_DNt_PsNtl[12]	double	EMA_DNt.EMA_DNt_PsNtl[12]		
EMA_DSc_PsScl[12]	double	EMA_DSc.EMA_DSc_PsScl[12]		
EMA_TRM_TNt[9]	double	EMA_TRM.EMA_TRM_TNt[9]		
EMA_TRM_TSc[9]	double	EMA_TRM.EMA_TRM_TSc[9]		
SIMCONTROL_DMG	double		1	
Outputs				
EMA_CGB	Simulink			1.0E-5
EMA_CPB	Simulink			1.0E-5

Figure 5. Component I/O and mapping browser.

To relieve the burden of creating specification documents and harmonizing those documents among the partners, we exploited the project wide naming conventions for model parameters (inputs/outputs). Having established the conventions, we programmed MOLE to use them in inspecting (parsing) each provided simulation module and generating "glue" code for correct parameter exchange. To cater for exceptional cases, we incorporated the ability to manually override the automated wiring.

Figure 6 aims at illustrating what we mean with "wiring". The figure depicts a simulation of two aircraft systems, the hydraulic system and the fuel system. Each system simulation is decomposed into smaller simulation blocks, such as pumps, circuitry, and actuators. Each block exposes specific inputs (upper ports of each block) and outputs (lower blocks), whereas each output models either a sensor (e.g., a pressure sensor) or a specific physical property (e.g., a specific pressure) that provides input for another simulation block (e.g., the pressure output of the pump acts as the input of various actuators). A specific interface block models the interdependency between the two systems. Finally, some of the inputs will be fed from an external aircraft simulation. MOLE is able to generate code (glue code) for the data exchange, even for high exchange frequencies (see section 4.3).

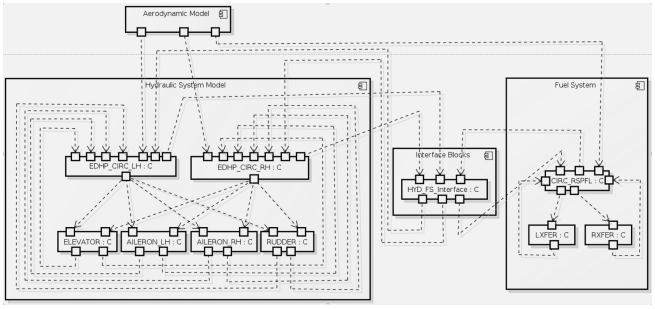

Figure 6. Glue code for automated wiring.

4.3. Parallel Computing

Due to their intended use as source for health assessment algorithm training data, the models performed complex calculations demanding high CPU power. The necessity for small temporal solutions, in the magnitude of 10^{-4}s – 10^{-6}s, drove the need for computational resources. Additionally, different models exhibited different temporal solutions, which were in general not multiples of each other. To benefit from multi-core processors, we introduced a concept for executing each model block in isolation with constant inputs for a maximum number of steps before the model block becomes unstable, and then halting the execution while each model block was updated with external inputs according to the specified wiring (while respecting overrides by expression). The concept was based on two frequencies. The "internal I/O frequency" determined the rate by which the model block execution was halted for the sake of parameter updates. As this lower frequency is reduced, overall execution time is also reduced, as a smaller relative fraction of time was used for parameter exchange and threading overhead. The "external I/O frequency" determined the rate in which supporting functions like graph plotting, data recording or other custom plugins were triggered. Based on that concept, MOLE supported setting a constant number of utilized CPU cores or dynamically adjust this number based on runtime performance analysis.

Figure 7 depicts the main parallel execution loop. First, each simulation block is iterated in its own thread, with constant input values for a block-individual maximum number of iterations that leaves the block numerically stable. Once each block is finished, the respective block outputs are copied to

the inputs according to the wiring. Then, further supporting functions are executed (typically on the current outputs), see sections 4.4 and 4.5. Then the loop repeats.

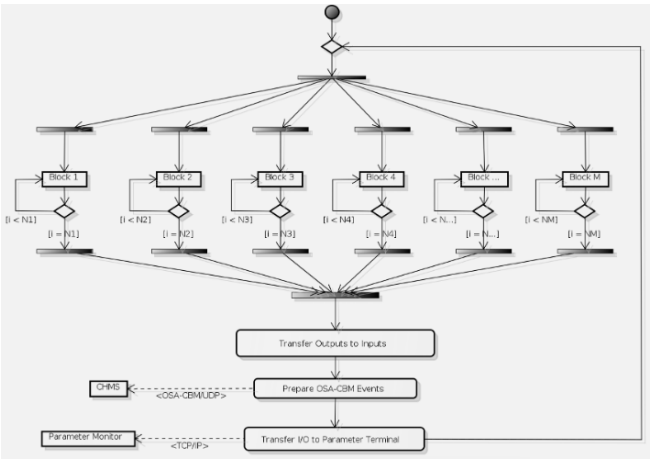

Figure 7. Main execution loop.

4.4. OSA-CBM Sampling

Based on our experience with implementing MIMOSA standards (Löhr & Buderath, 2014) we encouraged the project to follow the OSA-CBM design principles for the data processing chain. Consequently, the integrated diagnostics component of the digital twin was equipped with an OSA-CBM compliant sensor data interface, which was implemented using our proposed implementation of the binary OSA-CBM messaging protocol described by Drever, Naughton, Nagel, Löhr and Buderath (2016), which also discuss challenges and opportunities of MIMOSA standards in general. To drive the integrated diagnostics with data from the simulation, we enhanced MOLE with an OSA-CBM-compliant sampling function that was tied to the external I/O frequency. Selected model outputs, specifically those that simulated sensors, were sampled with the frequency required by the defined monitoring concept in the diagnostics layer. Then, the samples were automatically wrapped into OSA-CBM DM DataSeq events and transmitted to the integrated diagnostics via UDP.

4.5. Supporting Features

The core features of MOLE kept the inputs and outputs consistent, and ensured all models could be compiled. This allowed for quick iterations. Further, we added the following functions to enhance the testing process:

- Stepping: running a single model or a set of models, and halting/resuming the execution on demand, to inspect the current parameter sets
- Recording: marking a set of model inputs and outputs to be written to a CSV file for offline inspection, or for exchange within the project
- Fault Insertion: picking up on the naming conventions for faults, the fault input quadruples were recognized, and presented to the developer in a graphical user interface for interactive control
- Visualization: plotting the development of selected parameters over time to visually inspect the behavior of the model. An example is given in Figure 8.
- Parameter Override: we added an online compiler for simple expressions bound to model parameters. If set, the result of the expression was set (input parameter) or distributed (output parameter), resulting in a dynamic override of the original model wiring.
- Scripting: we added a simple scripting feature which allowed the association of model executions with time-indexed parameter overrides for that execution. By supporting an operation mode in which MOLE executes all scripts in a certain directory tree, users could generate data for different constellations, thus, having a set of repeatable test cases at their disposal.

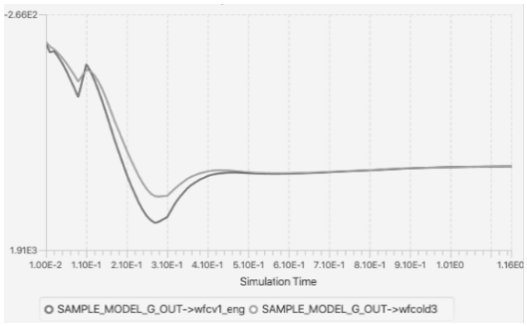

Figure 8. Integrated visualization.

5. VALIDATION

Validation of MOLE functionality was accomplished by its use in supporting the projects continuous integration process and building of the final deliverables. The constituents of the RTAPHM digital twin were:

- model of a pusher motor in several blocks
- model of a fuel cell in several blocks
- model of a servo motor in several blocks
- diagnostic module for pusher motor
- diagnostic module for fuel cell
- diagnostic module for pusher motor
- prognostic module for fuel cell
- interface module for pushing telemetry data to digital platform

- interface module for exchanging data with COTS flight simulator software
- bundled flight simulator module consisting of Gazebo, PX4, and a communication adapter towards the RTAPHM interface module

As validation evidence and as deliverables to the RTAPHM project, we created two artifacts.

First, a MOLE-based integration process for the digital twin. Here, Gazebo/PX4 was used as the "aerodynamics driver" to stimulate three high-fidelity physical simulation models. These models, in turn were used to generate sample and training data for three individual implementations of monitoring concepts.

Second, an interactive digital twin was created as a result of the integration process. This was used during project demonstration runs to create a live feed of (simulated) sensor data into the running diagnostic and prognostic modules, to show the interactive reaction to different fault scenarios.

Figure 9 depicts the result: simulation models for fuel cell, servo and pusher motor formed the RTAPHM system simulation. Some of the system simulation's inputs were fed with data from a customized fixed wing VTOL-capable UAV of which the physical and aerodynamic properties were simulated using Gazebo. Both system simulation and Gazebo were step-synced. PX4 was used to auto-pilot the UAV along a set of waypoints.

Figure 9. RTAPHM virtual aircraft.

6. Future Work

The validation work points out paths for future work. The most significant findings, which we are currently pursuing, are presented below.

- ***support further modelling tools***: to enlarge the possible target audience for MOLE users, the support of further modelling tools, other than just Matlab/Simulink, seems promising.
- ***support open standards***: another aspect of widening the target audience of MOLE emerges from the support of open interface standards, such as the Functional Mockup Interface (FMI), instead of relying on parsing proprietary coding patterns.
- ***integrate with Asset Administration Shell***: the current efforts around (AAS) focus on standardizing the digital representation of assets. MOLE could be integrated with AAS compliant repositories to allow users to choose from pre-build simulation components.
- ***cloud-native technology***: currently, MOLE is designed as a desktop-based tool, and thus is limited by the resources of the computing platform it runs on. By extracting the headless MOLE computing core and basing in on state-of-the-art cloud-native technology, these limitations can be overcome, and data stakeholders can scale the required resources on demand.
- ***service platform***: finally, combining all the previously mentioned streams of future work, we envision a cloud-based service platform, where stakeholders can upload own contributions, and build complex simulations from own and 3rd party simulation components, depending on their specific needs for training) data, and – by employing AAS and FMI concepts – without having to expose their intellectual simulation property to the public.

Using the MOLE digital twin, the project was able to showcase a core use case for automated functional dependency analysis: the injection of a fault in the cooling system's filter (clogging) caused the relevant areas to heat up, causing a decrease in the efficiency of the fuel cell, in turn, decreasing the remaining useful life of the fuel cell.

7. Conclusion

We presented MOLE, a tool for supporting the automation of the integration of software-based system simulations. The tool facilitates short integration cycles for agile project setups and provides specific debug and testing features. We showed the benefits of a structured and automated integration process for distributed research projects with different interests of individual partners.

We reported on the benefits of MOLE for rapid assembling of demonstrators for PHM research projects (though MOLE is not limited to that application domain). MOLE was used to support the integration process of the RTAPHM digital twin, consisting of commercial flight simulator software driving high-fidelity models of aircraft components.

We also showed that the bootstrapping and early development phases of data-driven application projects can benefit from artificially generated data to overcome the lack of initial data for first exploration and training. In particular, the data and communication architecture of the envisioned system can be explored regardless of the origin of the utilized data.

We finally recommend that OSA-CBM design principles and communication standards should not only be adopted for productive modules of a data processing chain. Designing data generators as OSA-CBM compliant data sources (see 4.4) facilitates integration testing, as the productive interface of the modules can be transparently stimulated.

ACKNOWLEDGEMENT

Gefördert durch:
Bundesministerium für Wirtschaft und Klimaschutz
aufgrund eines Beschlusses des Deutschen Bundestages

This work was supported by the ministry of commerce and climate protection (Bundesministerium für Wirtschaft und Klimaschutz) based on a resolution of the German Government (Deutscher Bundestag).

REFERENCES

Asset Administration Shell (AAS), https://www.iec.ch/dyn/www/f?p=103:38:607572709001913::::FSP_ORG_ID,FSP_APEX_PAGE,FSP_PROJECT_ID:1250,23,103536, IEC, International Electrotechnical Commission,

Darrah, T., Frank, J., Quinones-Grueiro, M., & Biswas, G. (2021). *A Data Management Framework & UAV Simulation Testbed for the Study of System-level Prognostics Technologies*. Annual Conference of the PHM Society, 13(1). https://doi.org/10.36001/phmconf.2021.v13i1.3030

Drever, J., Naughton, H., Nagel, M., Löhr, A., & Buderath, M. (2016). *Implementing MIMOSA Standards*. PHM Society European Conference, 3(1). https://doi.org/10.36001/phme.2016.v3i1.1647,

Löhr, A. (2023). *Realisierung einer auf operationellen Daten basierenden Plattform zur Qualitätskontrolle und zur Fortschrittsüberwachung bei der Erbringung von digitalen oder digital gestützten Flottendiensten*, *Technische Informationsbibliothek (TIB)*, June 2023

Löhr, A., & Buderath, M. (2014). *Evolving the Data Management Backbone: Binary OSA-CBM and Code Generation for OSA-EAI.* PHM Society European Conference, 2(1). https://doi.org/10.36001/phme.2014.v2i1.1487,

Functional Mockup Interface (FMI), https://fmi-standard.org, Modelica Association c/o PELAB, IDA, Linköpings Universitet S-58183 Linköping Sweden

Vuckovic, K., Prakash, S., & Burke, B. (2023). *A Framework for Rapid Prototyping of PHM Analytics for Complex Systems using a Supervised Data-Driven Approach*. Annual Conference of the PHM Society, 15(1). https://doi.org/10.36001/phmconf.2023.v15i1.3480

BIOGRAPHIES

Andreas Löhr received his M.Sc. degree in Computer Science from the Technical University of Munich in 2001 (Informatics, Diplom) and earned his doctoral degree in Computer Science from Technical University of Munich in 2006. For 6 years he worked as a software engineer at Inmedius Europa GmbH in interactive technical publications and researched in the field of wearable computing. He co-founded Linova Software GmbH in 2008 as managing director and worked as coding architect and consultant in both industry and academia projects in the field of aircraft maintenance and fleet optimization. Next to growing his company and concentrating on its strategic development, he specializes in aircraft MRO topics, with a research focus on condition-based maintenance, digital twins and data architectures.

Conor Haines received his B.Sc. degree in Aerospace Engineering from Virginia Polytechnic Institute and State University in 2003 and his M.Sc. degree in Computational Science from the Technical University of Munich in 2011. For 3 years Conor was a test engineer supporting the NASA Near Earth Network, providing simulation support used to guide system development. At his current post with Linova Software GmbH, he is focused on designing and developing embedded IVHM/ISHM architectures, along with a special interest in the application of digital twin simulation frameworks for the generation of early training data for data-driven application, such as enhanced embedded diagnostic and prognostic capabilities.