SpaceOps-2023, ID # 444

Bringing a Machine Learning Based Novelty Detection Software Tool from Research to Production

Clemens Schefels[a*], Leonard Schlag[a], Agnese Del Moro[a], Kathrin Helmsauer[a], Tobias Lesch[a], Tobias Göttfert[a]

[a] *German Space Operations Center, Deutsches Zentrum für Luft- und Raumfahrt e. V., German Aerospace Center Münchener Straße 20, 82234 Weßling, Germany,* clemens.schefels@dlr.de
* Corresponding Author

Abstract

This paper presents the process of bringing a machine learning based novelty detection software tool from research to production. Moreover, it sums up the necessary changes that needed to be done for developing a scientific software library into a software product with an application in space operations. This process considers the needs and expectations of all stakeholders.

The system for which this process is shown is the Automated Telemetry Health Monitoring System (ATHMoS) developed at the German Space Operations Center of the German Aerospace Center. In its early phase as a research software, it paved the way for the novelty detection research. After its value for the satellite engineer's daily work became visible, it evolved to a robust and resilient software tool that can be used in a productive environment to support the engineers in their routine work. Furthermore, the integration of the system into our Visualization and Data Analysis framework is explained. This framework has a web-based front-end for the interactive exploration and analysis of satellite telemetry data.

Keywords: Telemetry, Time Series, Machine Learning, Data Analysis, Space Operations, Software Development.

Acronyms/Abbreviations

American Standard Code for Information Interchange (ASCII),
Anomaly Detection algorithm based on a sparse decomposition into a DICTionary (ADDICT),
Application Programming Interface (API),
Automated Telemetry Health Monitoring System (ATHMoS),
Central Processing Unit (CPU),
Comma-Separated Values (CSV),
Computer Science and Engineering (CSE),
Density-Based Spacial Clustering of Applications with Noise (DBSCAN),
Software Development and IT Operations (DevOps),
European Organization for the Exploitation of Meteorological Satellites (EUMETSAT),
Extract, Transform, and Load (ETL),
German Aerospace Center / Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR),
German Space Operations Center (GSOC),
Gigabyte (GB),
Gravity Recovery and Climate Experiment-Follow-On (GRACE-FO),
Graphical User Interface (GUI),
k Nearest Neighbors (k-NN),
Long-Short Term Memory (LSTM),
Machine Learning (ML),
Mission Data Access (MiDA),
National Center for Space Studies / Centre National d'Études Spatiales (CNES),
Outlier Probability Via Intrinsic Dimension (OPVID),
Random-Access Memory (RAM),
Terabyte (TB),
University of California, Riverside (UCR),
Virtual Machine (VM),
Visualization and Data Analysis software (ViDA)

1. Introduction

In satellite operations, telemetry data plays a critical role to track the satellite's system status. Therefore, modern satellites collect telemetry data of thousands of parameters. For example, the GRACE Follow-On satellites, operated by the German Space Operations Center (GSOC) at the German Aerospace Center (DLR), define about *80,000* unique housekeeping parameters each. To help system engineers and operators to inspect this huge amount of data, many space operation centers put effort into the research and prototype development of machine learning based software tools for the analysis of these telemetry data. The next step, to bring such kind of software tools into productive use, is quite a challenge as it has to respect various needs of diverse user groups. Only if these needs are tackled, all users' routine work and the operation center at the whole will profit from a new software tool, which then will be used regularly.

This paper starts with a review of current novelty detection systems and related works in Section 2 about novelty detection with the focus on the space domain.

In Section 3, we first give a quick overview of the Automated Telemetry Health Monitoring System (ATHMoS), a novelty detection tool, developed at GSOC. ATHMoS uses semi-supervised machine learning methods to detect novel behavior in satellite telemetry data by learning the satellite's nominal behavior from historic data. The historic data contains nominal as well as abnormal behaviors. In order to determine if there are novelties in new data sets, we use the Outlier Probability Via Intrinsic Dimension (OPVID) algorithm. This very generic approach doesn't need tuning or adjustments for each single parameter. Moreover, ATHMoS has proven its reliability and efficiency in many research projects.

Next, in Section 4, the main part of this paper, we compare application of ATHMoS in a research environment to its application in a production environment. ATHMoS was primarily developed for research purposes. Therefore, it is a flexible framework, that can easily and quickly be adapted to data scientists' specific analysis tasks. The architecture reflects its scientific usage. To bring ATHMoS to production, it has to fulfill different requirements and support different use-cases of different stakeholders. We highlight the necessary changes in the ATHMoS framework to support all these new needs. The main part of the paper concludes with a detailed description on how we integrate ATHMoS into our Visualization and Data Analysis software (ViDA) to display the detected novelties in the satellite's telemetry data. However, we still give the users, e.g., data scientists, the possibility to access raw ATHMoS data for individual analysis tasks.

This paper finishes with a conclusion in Section 5 and gives a short outlook on follow-up projects related to ATHMoS at GSOC.

2. Related Works

Machine learning and its capabilities to detect novelties in various domains has made its way into many areas of research and applications, ranging from fraud or malware detection to medical image analysis [1]. It is no surprise that it has also made its way into the domain of satellite operations for novelty detection as well as other areas such as mission planning or image processing [2].

With regard to novelty detections, various methodologies are currently being researched and applied to satellite telemetry with the goal of benefiting operations on ground. The Centre National d'Études Spatiales (CNES) is employing a One-Class Support Vector Machine in their NOSTRADAMUS tool and investigating possible on-board applications [3]. EUMETSAT applies a distance-based k Nearest Neighbors (k-NN) approach and is gaining experience in the challenges an operational implementation brings along [4].

Investigations into multivariate anomaly detection by Tariq et al. propose a multivariate convolutional Long-Short Term Memory (LSTM) network with mixtures of probabilistic principal component analysis to be trained per subsystem [5]. Other approaches for multivariate detections by B. Pilastre and supported by CNES and Airbus Defence & Space use a sparse decomposition of a signal into a dictionary as part of a new algorithm named ADDICT [6]. This new algorithm allows including both continuous as well as discrete signals in its learned model.

As part of the ESA project Deep Learning 4 Space, AIKO also has proposed an AI-based framework for on-ground anomaly detection and root cause analysis [7].

In the next section, we will describe our machine learning system for novelty detection in more detail and present its workflows.

3. ATHMoS – Automated Telemetry Health Monitoring System

The Automated Telemetry Health Monitoring System developed at GSOC is not a single algorithm, but a sequence of steps combined in a larger workflow. We will give a short overview of the most important workflow steps comprising ATHMoS. For a detailed description of the main algorithms, see [8]. While the acronym ATHMoS contains the word *System*, it is separate from the actual software and hardware it runs on which is described in the main part of this paper in Section 4.

It is important to note that both, the core algorithm used for testing and training, and the pre- and post-processing steps, play an important role and are equally vital to the quality of the results.

3.1 Pre-processing

The raw satellite telemetry data constitutes the main input to the ATHMoS workflow. Before training a model or testing new data against the model, it needs to be pre-processed as a first step. This ensures no incomplete data is used and auxiliary data as input for the training and testing steps are derived from the raw data. Overall, there are three main pre-processing steps applied in the beginning of the workflow:

1. Detect gaps in the timeseries data hindering the derivation of senseful statistics and remove time windows containing these gaps. The removed data will be omitted in the remainder of the workflow.
2. Classify the parameter type in order to derive an optimized set of pre-defined features for the feature vector computation in the next steps. Using the different features for, e.g., telemetry parameters representing discrete flags and telemetry parameters representing physical attributes such as temperatures yields better results and allows for easier modifications to capture new behaviours.
3. In addition to the raw timeseries data, prepare a smoothed timeseries by removing possible noise. The smoothed timeseries will be used for extracting features in the next workflow steps as well.

3.2 Training

The next main workflow step takes care of training the model which can be seen as a representation of the nominal behavior of the satellite's telemetry. Especially initially, generating the model is the task requiring the most resources. Typically, a year of raw time-series data is loaded and processed for each telemetry parameter as there may be fluctuations in the parameter's behavior over this time frame.

The first step of the training is to take the pre-processed data and derive feature vectors by applying a sliding window. Using a window size of ,e.g., *1.5* hours for low Earth orbit satellites and step size of *30* minutes, this yields around *17,500* feature vectors over the training period. Details regarding the choice and advantages of the chosen features are described in [9]. As our input data is not sufficiently labeled, we also apply an enhancement of the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) clustering algorithm [8, 10]to clean up the feature vectors of obvious outliers, making the training set more robust.

To generate our model, the core algorithm [8] is applied to the feature vectors and intermediate results are stored, comprising the trained model. Doing so allows a quick model inference during the testing step which we describe next.

3.3 Testing

The current iteration of ATHMoS is not designed for real time data and uses dumped housekeeping data as its input source. Thus, it is ideally run on a daily basis for the last day or days of data.

While the pre-processing and feature vector computation are identical to the training, the testing step computes a novelty score for each feature vector and corresponding time window of the test telemetry data. The computed novelty score can be seen as a measure of how different new data behaves compared to the data used for training the model. This is done for each feature vector by temporarily extending the precomputed model (see Section 3.2) by that feature vector and deriving the score following algorithm [8] for the temporarily added feature vector only.

The novelty scores are used to label each feature vector and its time window accordingly, see Fig. 1. We use the fact that our time windows are overlapping to increase the quality of our labels. This is done by preventing single threshold breaches of the score which are not significant enough to trigger the threshold in neighboring overlapping windows to be labeled as a so-called *high priority* detection.

3.4 Model Update and Relabeling

To ensure the model contains the recent behaviors, it is recomputed periodically, typically on a weekly basis. It will always take the last year of data into account, starting from the day the recomputation is triggered on.

For generating the updated model, it only considers the feature vectors which were not flagged as high priority during testing and were not omitted during training for the model recomputation. We use this subset as our updated nominal data.

To allow the workflow to profit from our engineers' expert knowledge, engineers are given the option to change the priority labels assigned to the data by ATHMoS, thereby relabeling the data in our database. One could, e.g., set false positive detections, which ATHMoS labeled as *high priority*, back to *nominal*. The updated labels will also be

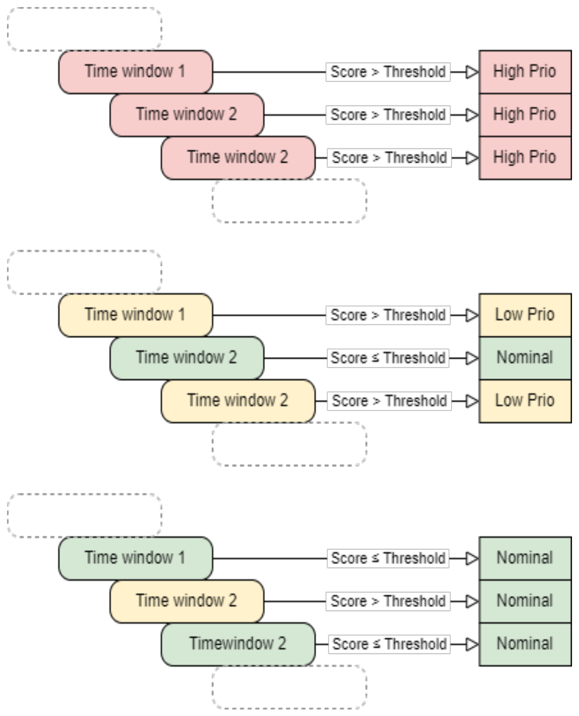


Fig. 1: Labeling of overlapping time windows

considered for the model update the next time it is executed. This further continually improves the quality of the results and ensures the model is in line with the expectation and understanding of our users.

4. From Research to Production

In this section, we describe the applications and use cases for a novelty detection system. It's split into two parts, the application in a research environment and the application in a production environment. At first ATHMoS was used as a research system but after proving its capabilities it was brought into production. For both applications, the realization of a system as described in Section 3 has to meet different requirements in respect to the software and the hardware it runs on.

4.1 Research Application

At the beginning, ATHMoS was a research project only. It paved the way for the novelty detection research at our institute. Moreover, it provided a system to gain experience with this kind of technology and enhance our knowledge. After defining its workflows and algorithms as described in Section 3, the design and architecture of its actual implementation reflect its scientific usage. In research, the single tasks of the users, i.e., the data scientists, are very specific and often unique. Data scientists use ATHMoS on a very personalized level within their own analysis environment, on their own hardware, commonly standard laptops. In general, data scientists have a lot of expert knowledge and can adapt the code of ATHMoS to their needs. Therefore, it is developed as a very flexible and adaptable Python [11] framework.

A typical workflow of a data scientist who uses ATHMoS looks as follows: at the beginning the data scientist receives a new task from e.g., a satellite system engineer. Examples of such tasks can be to explore the reason for an unexpected behavior of a satellite or just to explore the telemetry data of a satellite component in detail. Since the research activities of the DLR are not limited to the space domain only, our data scientists also use this opportunity to test and tune ATHMoS on various other kinds of data. Additionally, external partners, like international research institutes and universities, cooperate with us in our research on novelty detection. That way, we can constantly improve the accuracy of ATHMoS by developing new pre-processing methods or enhancements to the core algorithm.

After receiving a task, the data scientist has to load the provided data into their personal research environment. In our institute, scientists commonly use Python scripts or Jupyter [12] notebooks for analyzing such kind of tasks. The data is usually in a very task specific format like in CSV format, MS EXCEL file format, or just a (semi-)structured

ASCII file. For in-house satellite missions, the data are in the same format but for research activities with other institutes, the variety of data formats is broad. To name one example: the open UCR Time Series Classification Archive [13, 14] with tab-separated values, published by the Computer Science and Engineering (CSE) Department at the University of California, Riverside [15], which we use with external partners to benchmark our system. Therefore, for many of the new tasks, the data scientist has to adapt its data load pipeline to the present file format, especially when the task is not related to the satellite domain, in order to read the data.

After loading the data, the scientist has to pre-process and analyze the data. The ATHMoS framework comes with many possible adjustments to be adapted to the data it processes. For example, the data scientist can define a different set of features or a different time window length. This tuning of the hyperparameters, the finding of the right configuration for ATHMoS, is an important task during the analysis process.

After training the model with the nominal data, the scientist can test the data with the trained model as described in the section above. Next follows, the interpretation and presentation of the results, i.e., of the detected novelties. Here, each data scientist has their favorite diagram, plot, or data output to analyze the data. Last, the scientist has to present the results to the system engineer in a meaningful way and explain the findings to them.

To conclude this section, as one can see, most of these workflow steps are very task specific. In contrast to that, we will take a closer look at the workflow in a production environment in the next section.

4.2 Production Application

During the research process, the value of ATHMoS for the satellite engineer's daily work became visible. One important task of an engineer is to check the satellite parameters for unexpected, i.e., novel behavior. Modern satellites provide data of thousands of telemetry parameters which make it very time-consuming for the engineers to check each of them in detail—more system engineers would be needed. As a consequence of this, the operation of modern satellites would become more expensive. To overcome this obstacle, the usage of an automated telemetry health monitoring system that checks all parameters and informs the engineer in case of novel behaviors would be a solution. With its help, system engineers could concentrate on difficult tasks and let the software system do the routine work.

To convince satellite engineers as well as the groups in charge of operations to use a system like this, it has to meet several requirements. First, its usage has to be easy and intuitive so that the engineers do not have to spend their valuable time on learning how to use the system. More important is that the results the system generates are reproducible and understandable. The process of the calculation should be deterministic and transparent to the engineers. For deeper investigations and individual analyses, the system has to provide an Application Programming Interface (API) to access and use the calculated results.

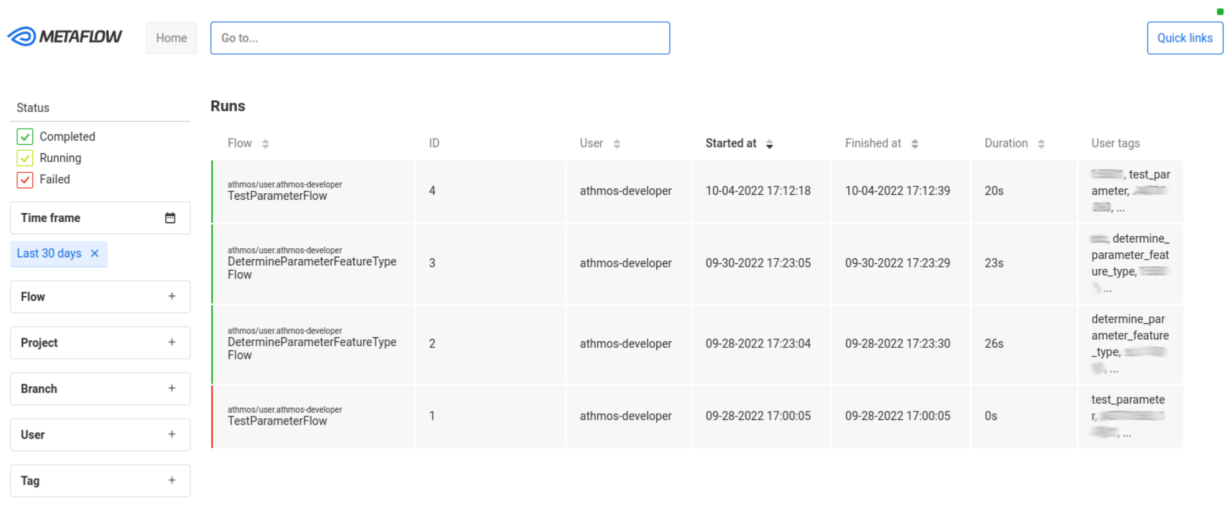
But not only engineers are customers of such a system for the routine satellite operations, but also the Software Development and IT Operations (DevOps) team that has to run and maintain the software. From their point of view, the software needs to be easy to install, operate, and maintain. For new mission, the adaptation effort has to be minimal. The requirements to the hardware need to be moderate to guarantee an economical operation. Moreover, the operational costs in total should be kept low to make it attractive for further satellite missions.

The ATHMoS Python library, described in Section 4.1, is developed for scientific purpose and does not meet all of these discussed requirements. At the beginning of the process of bringing ATHMoS into production, the ATHMoS Python library was refactored, and all parts were separated into single Python modules. That made the library easier to maintain and easier to adapt to the routine use case. Moreover, new improvements from research, e.g., a new pre-processing step, can be integrated straightforward into the production environment because dependencies are kept minimal.

After the refactoring, the extract, transform, and load (ETL) module was built. It provides the connection to the production database, containing the telemetry data, the transformation of the telemetry data into an ATHMoS suitable format, and the load methods for the workflows. In contrast to the research application, the data structure in the production database is stable and similar for all satellite missions. The persistence of the ATHMoS results is included in the production database.

One key challenge is to automate the single ATHMoS workflow steps from Section 3 in a robust and resilient way. Therefore, we use a standard ML framework, Metaflow [16], developed by Netflix for its machine learning software stack. This framework can automatically be executed by a cron job, the UNIX job scheduler, or various other task triggers based on e.g., data availability.

At last, the presentation of the ATHMoS results is implemented into the ViDA system, which is explained in detail in Section 4.2.3.



Flow	ID	User	Started at	Finished at	Duration	User tags
athmos/user.athmos-developer TestParameterFlow	4	athmos-developer	10-04-2022 17:12:18	10-04-2022 17:12:39	20s	[redacted], test_parameter, [redacted], ...
athmos/user.athmos-developer DetermineParameterFeatureTypeFlow	3	athmos-developer	09-30-2022 17:23:05	09-30-2022 17:23:29	23s	[redacted], determine_parameter_feature_type, [redacted] ...
athmos/user.athmos-developer DetermineParameterFeatureTypeFlow	2	athmos-developer	09-28-2022 17:23:04	09-28-2022 17:23:30	26s	determine_parameter_feature_type, [redacted], ...
athmos/user.athmos-developer TestParameterFlow	1	athmos-developer	09-28-2022 17:00:05	09-28-2022 17:00:05	0s	test_parameter, [redacted], ...

Fig. 2. Metaflow UI: Overview of the current status of the workflows

4.2.1 Workflow

The workflows of ATHMoS, as described in Section 3, are implemented in the Metaflow framework. Metaflow provides a unified API to the whole infrastructure stack that is required to execute data science projects from prototype to production [16]. Metaflow takes care of the low-level infrastructure: data, compute, orchestration, and versioning. Furthermore, it has a solid logging functionality which is essential for the DevOps team to quickly detect and fix system failures or errors within the workflow executions. With its web-based user interface, DevOps can get a quick overview over the current status of the calculations (see Fig. 2).

The building blocks from the scientific application could be consolidated into the following workflows.. At the beginning of each workflow, the configuration of the ATHMoS algorithms and of the database is loaded:

- *Train Parameter*: in this workflow, the historic model for the novelty detection is created. After loading the historic telemetry data and the type of the current parameter (continuous or discrete) from the database, the feature vectors and the model are calculated for each satellite parameter. The feature vectors are persisted into the database, while the models are stored as a file artifact. This step needs to be done just once at the beginning and takes the most computational time.
- *Test Parameter*: the aim of this workflow is to detect novel behavior in current telemetry data. Therefore, the current telemetry data has to be loaded from the database and tested against the model for each parameter. The historic model, calculated before, is loaded from files and the feature vectors for the current data are calculated. The feature vectors are tested against the historic and the recent models and anomaly scores are derived. The scores and feature vectors are stored into the database. This has to be done on a regular basis (e.g., each night) to provide the newest data about novel behavior to the engineers.
- *Retrain Parameter*: the aim of this workflow is to keep the models up-to-date. With the retraining, new nominal behavior is taken into account and the models are updated. This needs to be done on a regular basis but not as frequently as the "*Test Parameter*" workflow (e.g., once per week).

Next, in the Section 4.2.2, the system and hardware architecture for the production application of ATHMoS is explained.

4.2.2 System Architecture

To be able to run the described operational workflow in a continuous and reliable fashion, a suitable infrastructure is needed. Since the ATHMoS algorithm itself is relatively frugal in comparison to, e.g., deep-learning approaches, the amount of computing power to support the novelty detection is not very high. However, the desire to include more and more telemetry parameters and satellites into the monitoring, or to run more complex analyses in the future (like multivariate novelty detection or additional algorithms) makes it necessary to envision a scalable architecture from the get-go. On the other hand, since we had no readily available cluster resources and had to set up the infrastructure ourselves, the system should also start out small and only be extended once it becomes necessary.

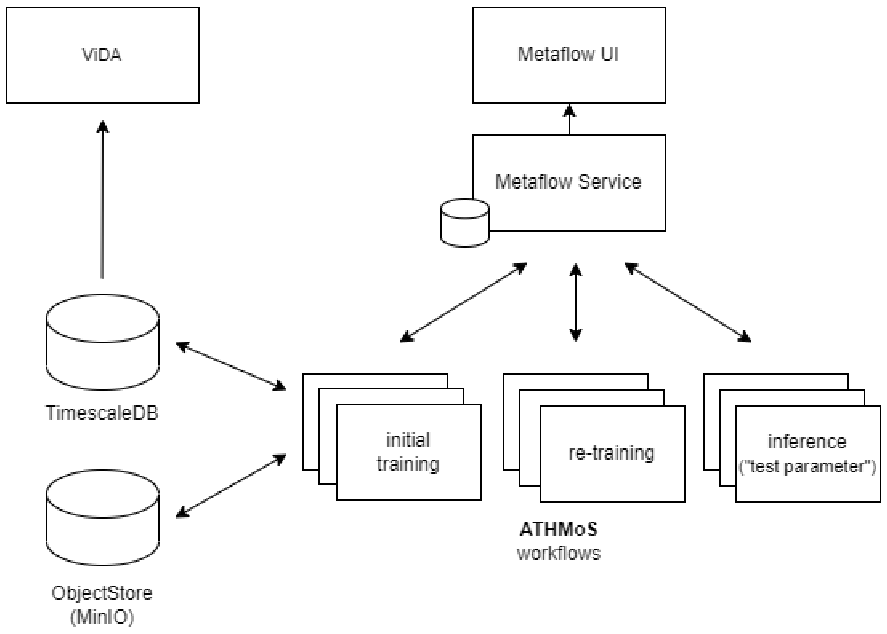


Fig. 3. Schematic overview over the services constituting the ATHMoS infrastructure. TimescaleDB holds the input data (telemetry) and stores the results of ATHMoS, which can be made available to the users (operators, spacecraft engineers) via the ViDA application. The object store is used for the persistence of data passed between the workflow steps and the trained ML models. The Metaflow service keeps track of executed workflows and their status, for the DevOps team to monitor the inner workings of the system. Scheduling and orchestration are currently handled inflexibly and are subject to improvement in the future

We evaluated two common frameworks for ML workflow deployments, Kubeflow and Metaflow, to see whether to make use of them. For a few reasons, Metaflow was chosen to be tried and adopted:

- Our initial goal was to get the ATHMoS workflows running operationally in their default, “regular” way as quick as possible. As Kubeflow is a far broader system that comes with considerable setup effort, while Metaflow is focussing on and limited to the processing pipeline, it is quicker to adopt and requires less setup.
- Kubeflow requires to be deployed on an existing Kubernetes cluster, which we might have available in the future. At the beginning, we did not want to have Kubernetes a requirement for the system.
- Metaflow is a Python library that could easily be added to the refactored ATHMoS code, which is also written in Python. It allows to test the processing workflow locally on the development computer and then push the resulting validated workflow onto a server or cluster. As we were just starting out, this allowed for a smooth transition from local algorithm development to the DevOps and infrastructure work of scaling up the algorithm.

Currently, regular ATHMoS processing is running in-house on dedicated servers hosting all VMs of the ViDA/ATHMoS system. Out of the full capacity of the infrastructure, ATHMoS currently has *16* CPUs and *96* GB RAM available (see also [17]). Current performance figures show that a common satellite mission, having *600* parameters (the most important once) monitored overnight, will keep the system busy for about one hour per night for the novelty detection (“test parameter”) and an additional *1.5* hours for the periodic model updating (“retrain parameter”), using *8* CPUs in parallel. We estimate that the current hardware setup is able to support up to *10* satellites for their routine novelty detection by dedicating more resources to the ATHMoS processes and staggering the times when they are started.

Fig. 3 shows a rough schematic view of the ATHMoS infrastructure: telemetry data is read from one or more TimescaleDB instances and the resulting feature vectors and anomaly scores are written back for ViDA to pick up the results. A small setup of a Metaflow service was deployed, including its metadata store and a MinIO S3-compatible object store, which are necessary for the internal workings of Metaflow, and the storage of the ML

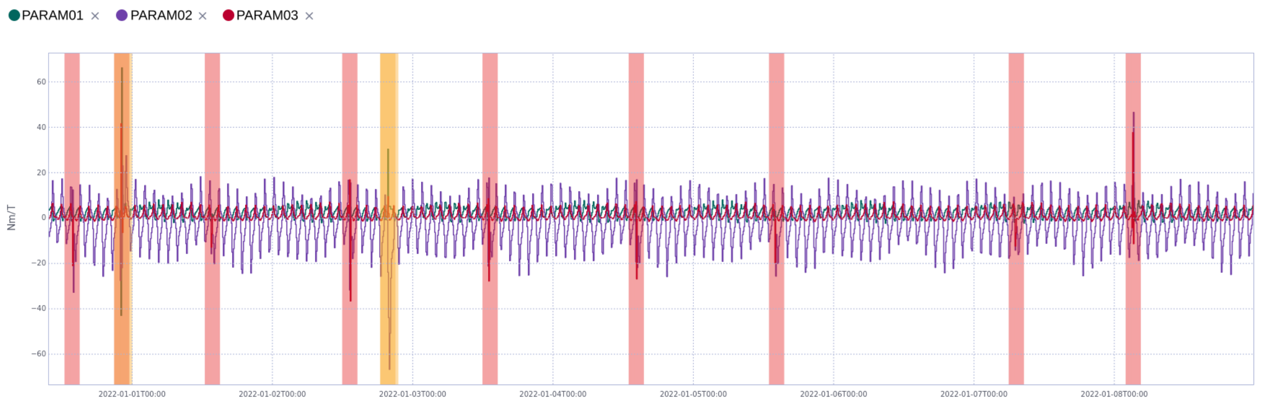


Fig. 4. Example of a multi-parameter time series plot in ViDA showing several ATHMoS detections marked as vertical bars (high priority detections in red, trending detections in yellow)

models and intermediate results of the workflow steps, respectively. For a full production-ready setup that follows Metaflow recommendations, this might have to be enlarged in the future; however, currently the system fulfills the needs of the routine processing of ATHMoS. Finally, the ATHMoS code, implemented as Metaflow workflows, is running on the processing VMs and is triggered via a simple, cron-based scheduling script. The code makes use of our Mission Data Access (MiDA) Python library for accessing the telemetry data from the TimescaleDB instances, and reports all statistics and logs to the Metaflow service to be inspectable via its GUI.

In its current form, the processing pipelines are deployed manually and started via a fixed schedule, which is fine for the momentary use case. In the future, this part of the system will be extended and more scalable and dynamic ways to orchestrate and schedule the workflows will be added.

4.2.3 Result Visualization and Data Analysis

The novelty detection results from ATHMoS are made available to the users through the Visualization and Data Analysis (ViDA) [17] tool, that serves as the ATHMoS web interface. ViDA is a comprehensive framework, consisting of a central user interface (UI) and underlying micro-services to ingest, pre-process, and access the telemetry data and other data products. Through ViDA, the engineers can monitor the status of the satellites, visualize the telemetry data up to the full mission lifetime (e.g., *>10* years of data) in seconds, and perform their daily analyses and checks.

The ATHMoS results are retrieved from the database by the ViDA backend via GraphQL queries and served to the web front-end to be displayed in dashboards as aggregated data, showing the general status of a satellite in the past day or week. The high-priority and trending novelties are also displayed in time-series plots, together with the telemetry data, to flag time ranges where potential anomalies have been found (Fig. 4). This type of visualization was realized to ease and speed up the telemetry inspection process by highlighting the parameters and time ranges where the engineers should focus on to identify potential problems.

Further types of graphical visualizations of the ATHMoS results, specifically aimed to improve the explainability of the detected novelties according to the users' needs, are under development. Besides the graphical views, ViDA will enrich the telemetry and ATHMoS data with contextual information, such as the telecommand logs, on-board event logs, anomaly reports, etc., thus providing a more complete view of the status of the satellite at any selected time range.

Besides the standard analyses and presentations of the ATHMoS data via ViDA, we provide to the users the MiDA Python library, to conduct individual analyses. MiDA supports the full range of data that is present in the database including telemetry, feature-vector data for ATHMoS analysis and several further information for housekeeping. With MiDA, a data scientist can easily obtain and query satellite data and include the results into their own analysis environment like Python scripts or Jupyter notebooks.

In future developments, ViDA will also implement an interface where the users can label the telemetry data and novelties, e.g., in case of false-positive or false-negative detection. These labels will be stored in the database and used by the ATHMoS algorithms to improve the data models for future data processing, thus making the novelty detection and classification more and more accurate and reliable over time.

5. Conclusions and Future Work

This article described the way of bringing a machine learning based novelty detection software tool from research to production. It has first briefly introduced the Automated Telemetry Health Monitoring System (ATHMoS) as a sequence of steps combined in a larger workflow. The system was originally designed as a research project to enable the novelty detection research at our institute. With a short comparison of the requirements of such a research project in contrast to the requirements of an application in an operative context, the concrete changes to the system are provided. For the presentation of the calculated novelty detections, the integration of the ATHMoS system into our ViDA framework was explained.

In the future, we want to enhance ATHMoS by bringing the experience we are now collecting from the productive application back to research. With the bigger data basis from the operational satellite service, we will improve the core-algorithm, like more parameter specific feature vectors, and pre-processing steps, like using different smoothing methods, to enhance the quality of the novelty detection. Moreover, with the collected re-labeled data by the system engineers, we can filter out maneuvers or unwanted novelty detections and build a database with standard satellite behavior.

Furthermore, we are planning to execute the novelty detection directly on the satellite in space to find novelties earlier and give the engineers more time to react. This would also include a real time analysis of the telemetry data. The frugal and effective algorithms of ATHMoS are very well suited for that approach. First attempts had already been conducted and a successful prototype for the real-time novelty detection has been tested in a lab environment.

After setting up the system for the satellite operational application, we want to expand our costumer basis further by thinking about other application domains. One obvious idea would be to check the telemetry data from the ground segment like from the satellite dishes. Domains beyond space operations like aerospace, energy, and transportation are also considered.

In the distant future, we want to take advantage of Quantum technology for our novelty detection methods. The DLR, as one of Germany's leading research centers, is investing as well in fundamental research as in software development and application analysis on quantum computing [18]. As a first step, we research on the state preparation of quantum computers for satellite telemetry data. Later on, we will research on how to extend or/and replace part of ATHMoS with quantum enhance technology.

Acknowledgements

The authors are thankful to Dr. Martin Wickler, Dr. Edith Maurer, all members of the DLR MBT-Team, and Heavens Above GmbH for their valuable support and fruitful discussions.

References

[1] R. Chalapathy and S. Chawla, “Deep Learning for Anomaly Detection: A Survey”, https://arxiv.org/abs/1901.03407/, 2019.
[2] P. Miralles et al., “Machine Learning in Earth Observation Operations: A review”, Proceedings of the 72nd International Astronautical Congress (IAC), Oct. 2021.
[3] P. Delandea, P.-B. Lambertb, M. Bouayadc, M. Zaroubiand, A. Barone, and E. Jalabert, “AI for Satellite Anomaly Detection: On-Ground Operational Feedback and Development of On-Board Experiments”, Proceedings of the 73rd International Astronautical Congress (IAC), Sept. 2022.
[4] P. L. Losco, J. Pergoli, A. De Vincenzis, and R. Dyer, “From Theory to Practice: Operational Implementation of Telemetry Outlier Detection at EUMETSAT”, Proceedings of the 16th International Conference on Space Operations, May 2021.
[5] S. Tariq, S. Lee, Y. ShinS. M. S. Lee, O. Jung, D. Chung, and S. S. Woo, “Detecting Anomalies in Space using Multivariate Convolutional LSTM with Mixtures of Probabilistic PCA”, Proceedings of the 25th ACM SIGKDD International Conference on Knowledge & Data Mining, Jul. 2019.
[6] B. Pilastre, J.-Y. Tourneret, and L. Boussouf, “Multivariate Anomaly Detection in Mixed Telemetry time-series Using A Sparse Decomposition”, Proceedings of the 2019 IEEE 8th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), Dec. 2019.
[7] L. Manca, I. Bloise, A. Spörl, and K. Helmsauer, “An Innovative AI-Based Framework for On-Ground Anomaly Detection and Root Cause Analysis”, 17th International Conference on Space Operations, Dubai, United Arab Emirates, 2023, 6 - 10 March.
[8] C. O’Meara, L. Schlag, L. Faltenbacher, and M. Wickler, “ATHMoS: Automated Telemetry Health Monitoring System at GSOC using Outlier Detection and Supervised Machine Learning”, Proceedings of the AIAA SpaceOps 2016 Conference, May 2016.

[9] L. Schlag, C. O'Meara, and M. Wickler, "Numerical Analysis of Automated Anomaly Detection Algorithms for Satellite Telemetry", Proceedings of the 14th International Conference on Space Operations, May 2018.
[10] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise", Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, Volume 96, No. 34, pp. 226-231, August 1996.
[11] Python, https://www.python.org/,1991, (accessed 21.01.2023).
[12] Project Jupyter, https://jupyter.org/, 2014, (accessed 21.01.2023).
[13] H. A. Dau, et al., "The UCR Time Series Classification Archive", October 2018, https://www.cs.ucr.edu/~eamonn/time_series_data_2018/ (accessed 21.01.2023).
[14] H. A. Dau, et al., "The UCR Time Series Archive", arXiv, 2018.
[15] Computer Science and Engineering (CSE) Department at the University of California, Riverside, https://www1.cs.ucr.edu/ (accessed 21.01.2023).
[16] Metaflow, https://metaflow.org/ (accessed: 23.01.2023).
[17] A. Del Moro, M. Dauth, T. Lesch, C. Schefels, A. Braun, V. Filip, and T. Göttfert, "A Modern Approach to Visualize Structured and Unstructured Space Missions' Data", 17th International Conference on Space Operations, Dubai, United Arab Emirates, 2023, 6 - 10 March.
[18] DLR Quantumcomputing Initiative, https://qci.dlr.de/ (accessed 25.02.2023).