The Onboard Artificial Intelligence Research (OnAIR) Platform

Evana Gizzi*[1], Timothy Chase Jr[1], Connor Firth[2], James Marshall[1], Alan Gibson[1]

[1]*NASA Goddard Space Flight Center, Greenbelt, MD, USA*
[2]*Aurora Engineering, Potomac, MD, USA*

In this paper, we present the NASA On-Board Artificial Intelligence Research (OnAIR) Platform, a dual-use tool for rapid prototyping autonomous capabilities for earth and space missions, serving as both a cognitive architecture and a software pipeline. OnAIR has been used for autonomous reasoning in applications spanning various domains and implementation environments, supporting the use of raw data files, simulators, embodied agents, and recently in an onboard experimental flight payload. We review the OnAIR architecture and recent applications of OnAIR for autonomous reasoning in various projects at NASA, concluding with a discussion on the intended use for the public domain, and directions for future work.

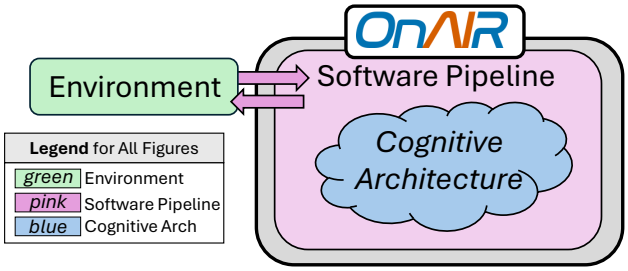

Figure 1: High-level depiction of OnAIR which shows its two main elements (cognitive architecture and software pipeline) and a third external element, the environment to be interacted with.

1 Introduction

To keep pace with the increasing development of autonomy in space, we must streamline aerospace development approaches with industry standards for artificial intelligence (AI) development. Rapid prototyping has become commonplace for algorithm development with the unprecedented and consistent growth of the field of AI [1]. However, this agile style of prototyping is challenging to employ in the aerospace domain due to its multidisciplinary complexity (leading to variation in software experience and literacy), the inherent inaccessibility of space and space data, and the conservatism of spaceflight development approaches [2–4]. These factors create a barrier to entry for developing and infusing autonomy into space systems.

To support agile algorithm development and integration of AI capabilities into space systems, we present the NASA On-Board Artificial Intelligence Research (OnAIR) platform. OnAIR is both a 1) cognitive architecture and 2) rapid prototyping framework (Figure 1) that enables domain-agnostic development of autonomous systems. We describe the OnAIR tool, its architecture, and its historical and intended use, concluding with a discussion on future work and future use by the research community. OnAIR is open-sourced at https://github.com/nasa/OnAIR.

*Corresponding author. E-Mail: Evana.Gizzi@nasa.gov

- **CSV** allows the use of a static file, which is useful for demonstration, development, or previously gathered data.
- **Redis** is a simple publish/subscribe format for ingesting data from a running system.
- **cFS** allows for the use of OnAIR with the Core Flight System flight software environment.

2 Background

OnAIR was first developed under an Internal Research And Development (IRAD) grant at NASA Goddard Space Flight Center (GSFC) which focused on generating methods for onboard fault diagnosis [5]. Researchers continued to organically reuse OnAIR as a framework for various diverse AI research applications. Project-specific needs led to unexpected and significant development time devoted to architectural/dataflow tasks. Consequently, OnAIR has also found suitable use internally as a generalized cognitive architecture to support reuse and agile development. To the best of our knowledge, OnAIR is the first cognitive architecture and rapid prototyping autonomy pipeline for aerospace applications.

OnAIR was designed for flexible use while providing the structure needed for modeling human cogni-

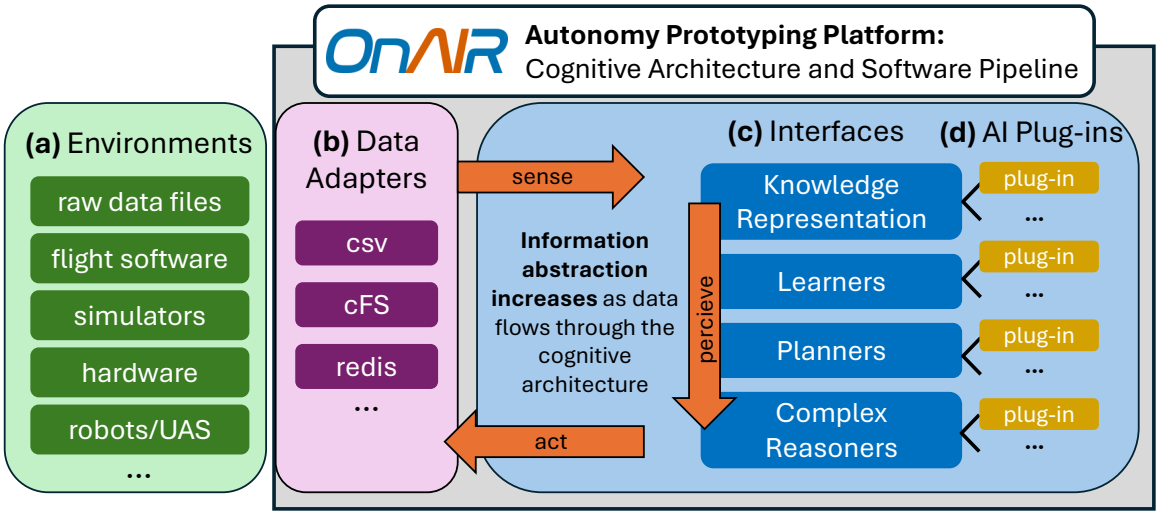

Figure 2: Low-level depiction of OnAIR, which shows (a) a non-exhaustive list of environments which can be used with OnAIR, (b) a set of data adapters which are provided with the open source version of OnAIR, (c) the four main interfaces of the OnAIR cognitive architecture, and (d) user-specified plug-ins. Note: users can provide custom data adapters in (b).

tion, which includes information discrimination and abstraction to emulate the function of the human brain. OnAIR is not intended to replicate the publish-subscribe architectures that are common in AI and aerospace applications, such as the Robot Operating System (ROS) [6, 7] and the Core Flight System (cFS) [8, 9]. OnAIR can interface with such systems to streamline AI algorithm development.

2.1 OnAIR as Cognitive Architecture

Within the field of artificial intelligence, cognitive architectures are used as models for human reasoning, consistent with a neurological and cognitive function [10]. While *traditional* cognitive architectures focus on underlying "rule-based" cognitive function (ACT-R or SOAR) [11, 12], more recently preferred *connectionist* cognitive architectures draw structural inspiration from neuroscience, considering both functional cognition and its underlying neurobiological basis [13]. OnAIR takes a hybrid approach, accounting for both rule-based and emergent cognitive outputs. The main abstraction layers of OnAIR are enforced by four interfaces which host user defined plugins as they increase in abstraction from "neurological firings" to cognitive/behavioral output. Thus, the OnAIR interfaces emulate discrete layers of cognition. Table 1 describes each interface as a brain analog, drawing from major researchers in the fields of psychology, neuroscience, philosophy, and computer science, which collectively comprise the conglomerate of AI.

2.2 OnAIR as a Prototyping Pipeline

OnAIR provides a plug-and-play architecture that allows user code (implemented as one of the four plugin types listed above) to receive low-level data (from an external source), or high-level data (from previous plugins within the pipeline); see Figure 2. OnAIR's lightweight implementation makes it suitable for rapid deployment and co-development across a team of systems. Its ability to host several self-contained systems makes it a natural prototyping platform. OnAIR has seen significant use in NASA internal demonstrations, running onboard mobile platforms like robots and drones. The modular architecture is well-suited to rapid prototyping tasks, especially to the integration of separately-developed tools. Data adapters are provided to ingest data from static sources like comma-separated (CSV) files or dynamic data sources such as Redis servers or NASA's cFS.

3 How to Use OnAIR

This section provides a high-level overview of how OnAIR is configured, receives data, and moves data through the pipeline. Built with extensibility in mind, OnAIR algorithms are implemented through user-constructed plugins that are written in Python.

3.1 Configuration File

To define how the OnAIR pipeline will run, a user must first specify required information in a configuration file. The configuration file has four main sections:

	Knowledge Representation	Planners and Learners	Complex Reasoners
Interface Description	Takes in low-level data (vehicle telemetry) and creates needed information representations for the brain	Takes in low-level data needed for AI learning and planning algorithms and outputs labels (learners). Takes in high-level data to determine possible actions (planners)	Traverses over all synthesized high-level outputs to make a combined, logical, informed decision
Neurosymbolic Representation (input → output)	Sub-symbolic → Symbolic	Sub-symbolic (learners) or Symbolic (planners) → Symbolic	Symbolic → Symbolic
Neurobiological Premise	Neurons firings to output neural circuitry patterns	Input circuitry firings in a specific brain region for specialized cognitive function. Example: Amygdala (emotional response), Hippocampus (memory recall), etc	Outputted behavior. Example: Prefrontal Cortex (process a stimulus and remember that it is not a threat – proceed with action)
Freud Levels of the Mind	Unconscious	Pre-Conscious	Conscious
Russel Norvig Rational Agent	Sense	Perceive	Act

Table 1: Each OnAIR plug-in interface can be mapped to philosophical, psychological, and neurological brain functions.

- The **files** section comprises telemetry and metadata paths and files. Telemetry file is used by static data sources (like CSV) while all sources use metadata to comprehend the structure of the incoming data.
- The **data handling** section contains the location of the data source that will be used to process incoming data.
- The **plugins** section defines the user-created plugins that will comprise the data pipeline. It has entries for all 4 cognitive types and the pipeline is defined by order of entry, left to right within each type and top to bottom: Knowledge Representation, Learner, Planner, and Complex Reasoner.
- The **options** section contains programmatic settings for OnAIR such as enabling textual outputs and debugging.

3.2 Data Adapters

Users must consider which data adapters to use with their chosen environment. The user has the choice of creating a custom adapter or utilizing one of the provided types. The purpose of data adapters is to process packets of data from the environment into a `DataSource` object and insert new data into OnAIR's low-level data frame (defined in the metadata file). OnAIR provides the following adapters to the user:

- **CSV** allows the use of a static file, which is useful for demonstration, development, or previously gathered data.
- **Redis** is a simple publish/subscribe format for ingesting data from a running system.
- **cFS** allows for the use of OnAIR with the Core Flight System flight software environment.

The cFS adapter offers an excellent real-world example of how a pre-existing system may be used with OnAIR. Interfacing with cFS is complicated by the fact that cFS is not designed to allow external processes access to its internal message bus (unlike Redis). Additional open source tools are required to gain access: the Software Bus Network (SBN) [14] is configured to publish cFS messages to a local UDP socket. The OnAIR data adapter then uses the Software Bus Network Client [15] to connect to SBN.

3.3 Plugin Interfaces

OnAIR has four interfaces – Knowledge Representation, Learners, Planners, and Complex Reasoners – which host all user-defined plugins. The data frame is passed from the `DataSource` as the low-level data to each plugin sequentially to drive internal state updates.

- The first interface to receive data is the **Knowledge Representation** interface, which contains all plugins responsible for synthesizing information from low-level perceptive data into higher-level representations (such as logical predicate descriptors, spacecraft status, or instrument state).

- The next interface to receive data is the **Learners** interface, which contains plugins that leverage data driven AI methods (like machine learning algorithms). The Learners interface receives both the low-level data frame *and* the symbolic information synthesized by Knowledge Representations, acquiring new semantic information about the environment over time. For example, a neural network Learners plugin would need both low-level feature data and high-level ground truth labels for training.
- The next interface to receive data is the **Planners** interface, which houses all plugins that perform planning and scheduling algorithms. Planners receive all incoming data rendered from Knowledge Representation and Learners plugins, which is used to update the internal state of OnAIR in preparation for the execution of user-defined planning algorithm plugins.
- Semantically significant information generated from all interfaces drives high-level reasoning in **Complex Reasoner** plugins. Complex Reasoners receive the output of each previous plugin, utilizing all available environmental and internal information to produce a high-level reasoning output. For example, complex reasoners may go against planner recommendations to preserve vehicle safety limits. Complex Reasoners are meant to emulate conscious human decision-making.

The reasoning cycle then restarts with the data source receiving new data and inserting it into the frame. This data reflects the changes observed in the environment since the last frame was given to the pipeline (i.e., the last timestep). The system will run until it runs out of data (static file) or indefinitely (awaiting data from a dynamic source).

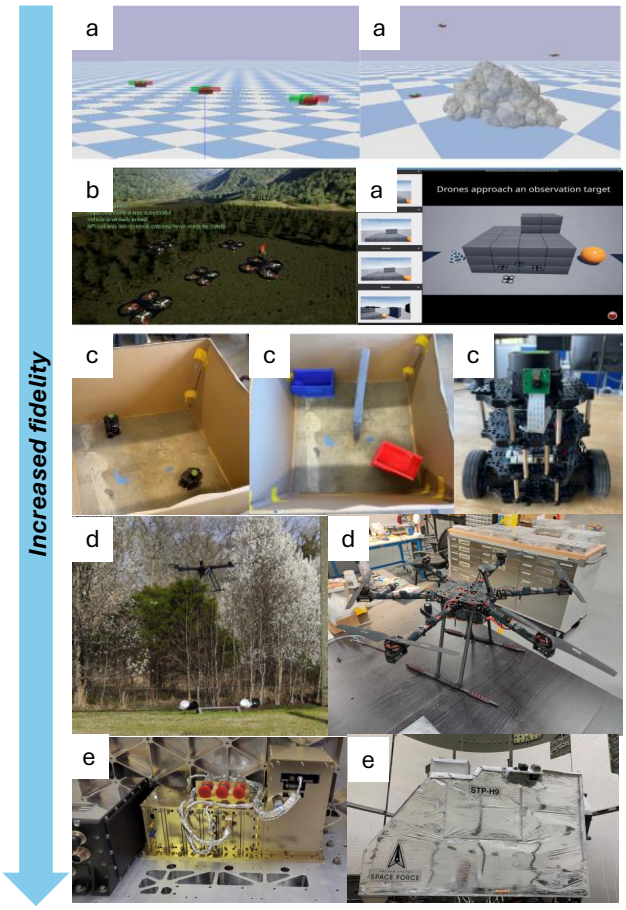

Figure 3: Applications of OnAIR for autonomy projects at NASA GSFC encompass specialized simulations developed in (a) PyBullet and (b) Unreal Engine, as well as embodiments in (c) Turtlebot, (d) GSFC sUAS, and (e) STP-H9 SCENIC in-flight platforms.

4 Results

To date, OnAIR has been used to support reasoning in many diverse autonomy applications at NASA GSFC, and in concert with our collaborators (see Figure 3). As mentioned, OnAIR was first used as an architecture for resilience research. During this time, OnAIR leveraged CSV data corresponding to sounding rocket telemetry, data from Kerbal Space Program for education and research use, Dellingr [16] satellite telemetry, and basic emulated toy data.

OnAIR was directed for use as the cognitive architecture to support NASA GSFC's Distributed Systems Mission (DSM) internal research and development project, led by Engineering and Technology Director Chief Technologist Michael Johnson. During this time, OnAIR was used for all reasoning, which included resilience, opportunistic science discovery, basic sensing, planning, and scheduling, among others. OnAIR was used with collected **real-world** science data [17–19], with **simulated data** (small satellite flights using the NASA Operational Small Satellite Simulator [20, 21]), with publicly available robot simulators (PyBullet, Gazebo, JMAVSim, Unreal Engine, AirSim [22]), and on **embodied agents** (Turtlebots and GSFC custom built small unmanned aerial vehicles (sUAS), ModalAI Starlings).

OnAIR was used to support reasoning on-board sUAS in the Network for Assessment of Methane Activity in Space and Terrestrial Environments (NAMASTE) project by principal investigator: Dr. Mahmooda Sultana. Lastly, OnAIR was **run onboard** to support a basic Kalman-filter experiment on the STP-H9-SCENIC payload [23] under principal investigator Dr. James Marshall to test onboard tractability, bringing OnAIR to Technology Readiness Level 7.

5 Discussion

In this paper, we reviewed the OnAIR tool, its architecture, and described recent applications of OnAIR for onboard reasoning for autonomy projects at NASA. OnAIR has been used in a highly interdisciplinary and application-agnostic manner. We aim for researchers to utilize OnAIR for their autonomy experiments, both within the aerospace domain and in other fields. OnAIR was intentionally open-sourced, and created with ease of use in mind for this highly sought widespread application. It is our goal to reduce the barrier to entry for increased partnership across academic, private, and public sectors. We believe the creation and sustained operation of OnAIR is a promising first step. Plans are in place to continue its use at NASA for future distributed systems missions research, with applications in Earth Science, Planetary Science, Astrophysics, and Heliophysics. This is being done in support of the most recently available decadal survey goals which seek distributed passive and active observatories [24–27].

6 Acknowledgements

We would like to acknowledge the following interns who have contributed to OnAIR (and its earlier iterations): Ibrahim Haroon, Jeffrey St. Jean, Hayley Owens, Nicholas Pellegrino, Gabriel Rasskin, James Staley, William Zhang, and Charles Zhao. Thank you to the contributors to the OnAIR open source repository github.

Thank you to the Goddard Space Flight Center Internal Research and Development program and the Distributed Systems Missions project for supporting this work.

References

1. Perrault, R. & Clark, J. Artificial Intelligence Index Report 2024 (2024).
2. Ribeiro, J. E. F., Silva, J. G. & Aguiar, A. Weaving Agility in Safety-Critical Software Development for Aerospace: from Concerns to Opportunities. *IEEE Access* (2024).
3. VanderLeest, S. H. & Buter, A. *Escape the waterfall: Agile for aerospace* in *2009 IEEE/AIAA 28th Digital Avionics Systems Conference* (2009), 6–D.
4. Monteiro, J. P., Rocha, R. M., Silva, A., Afonso, R. & Ramos, N. Integration and verification approach of ISTSat-1 CubeSat. *Aerospace* **6,** 131 (2019).
5. Gizzi, E. *et al. Autonomous System-Level Fault Diagnosis in Satellites using Housekeeping Telemetry* 2022.
6. Koubâa, A. *et al. Robot Operating System (ROS).* (Springer, 2017).
7. Macenski, S., Foote, T., Gerkey, B., Lalancette, C. & Woodall, W. Robot Operating System 2: Design, architecture, and uses in the wild. *Science robotics* **7,** eabm6074 (2022).
8. McComas, D. *NASA/GSFC's Flight Software Core Flight System* in *Flight Software WorkshopFlight Workshop* (2012).
9. McComas, D., Wilmot, J. & Cudmore, A. *The core flight system (cFS) community: Providing low cost solutions for small spacecraft* in *Annual AIAA/USU Conference on Small Satellites* (2016).
10. Russell S., N. P. Artificial Intelligence: A Modern Approach, 935 (2011).
11. Ritter, F. E., Tehranchi, F. & Oury, J. D. ACT-R: A cognitive architecture for modeling cognition. *Wiley Interdisciplinary Reviews: Cognitive Science* **10,** e1488 (2019).
12. Jones, R. M., Lebiere, C. & Crossman, J. A. *Comparing modeling idioms in ACT-R and Soar* in *Proceedings of the 8th international conference on cognitive modeling* (2007), 49–54.
13. Petersen, S. E. & Sporns, O. Brain networks and cognitive architectures. *Neuron* **88,** 207–219 (2015).
14. *cFS Software Bus Network (SBN)* `https://github.com/nasa/sbn`.
15. *cFS Software Bus Network (SBN) Client* `https://github.com/nasa/sbn-client`.
16. Clagett, C. *et al.* Dellingr: NASA Goddard Space Flight Center's First 6U Spacecraft (2017).
17. Theiling, B. P. *et al. A science-focused artificial intelligence (AI) responding in real-time to new information: Capability demonstration for ocean world missions* in *2024 Astrobiology Science Conference* (2024).
18. Firth, C., Greenlee, L., Barrie, A., Theiling, B. & Clough, L. *The Module for Event Driven Operations on Spacecraft: Applying MEDOS to Science Data* in *23rd Meeting of the American Geophysical Union (AGU)* (2023).
19. Williams, C., McKinney, L., Clough, L. A., Theiling, B. & McKinney, B. *Autonomous Science: Simulated Solar System Mission to Enceladus, Icy Ocean Moon of Saturn* in *The University of Tulsa Student Research Colloquium* (2024).
20. Grubb, M., Morris, J., Zemerick, S. & Lucas, J. Nasa operational simulator for small satellites (nos3): Tools for software-based validation and verification of small satellites (2016).
21. Geletko, D. M. *et al.* NASA operational simulator for small satellites (NOS3): the STF-1 cubesat case study. *arXiv preprint arXiv:1901.07583* (2019).
22. Shah, S., Dey, D., Lovett, C. & Kapoor, A. AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. *CoRR* **abs/1705.05065.** arXiv: 1705.05065. `http://arxiv.org/abs/1705.05065` (2017).
23. Geist, A. *et al.* NASA SpaceCube Next-Generation Artificial-Intelligence Computing for STP-H9-SCENIC on ISS (2023).
24. Of Sciences, N. A. *et al. Thriving on our changing planet: A decadal strategy for Earth observation from space* (National Academies Press, 2019).
25. Of Sciences, N. A. *et al. Visions into Voyages for Planetary Science in the Decade 2013-2022: A Midterm Review* (National Academies Press, 2018).
26. National Academies of Sciences, E., on Engineering, M. D., on Physics, P. S. S. S. B. B., for a Decadal Survey on Astronomy, A. C. & (Astro2020), A. 2. *Pathways to Discovery in Astronomy and Astrophysics for the 2020s* (National Academies Press, 2023).
27. Council, N. R. *et al. Solar and space physics: A science for a technological society* (National Academies Press, 2013).