ON-BOARD DATA ANALYSIS AND REALTIME INFORMATION SYSTEM - STATUS & OUTLOOK

K. Schwenk*, D. Herschmann*

* German Aerospace Center (DLR), German Space Operations Center (GSOC), Münchner Str. 20, 82234 Wessling, Germany

Abstract

In this article an overview of the current development status of the "On-board Data Analysis And Real-Time Information System" (ODARIS) developed at the German Aerospace Center (DLR) is presented. We will focus on an upcoming technology demonstration experiment of the ODARIS concept planned around 2023-2025 and include topics as the utilization of “low-cost real-time communication channels”, on-board data evaluation, the communication concept and the benefits and limitations of our approach.

Keywords

Embedded systems, Edge computing, Situational awareness, AI image analysis, Satellite platform

1. INTRODUCTION

For satellite operation, having the capability to communicate with the satellite system, anytime, anywhere, with low communication latency and high network bandwidth, would be beneficial in many satellite missions. At the moment communication with the space segment can only be established, when the satellite passes a communication window of a ground station network. For many satellite missions in Low Earth Orbits (LEO) this is equivalent of having a timeframe for communication of typical 10-20 minutes within one or two contacts a day. The connectivity for space operations can be increased via additional ground control stations to the ground segment network, or by utilizing relay satellite networks like the European Data Relay Satellite (EDRS) [1]. Due to high operational cost, these solutions are currently only utilized by large space missions, as for example the Sentinel-1 and -2 satellites in the Copernicus Programme of the European Space Agency (ESA). But for many space missions, the high transmission bandwidth offered by these solutions are not mandatory. They would already largely benefit from low-bandwidth communication channels, which are uncomplicated to deploy and offer transmit- and receive-latencies around a few minutes. An example would be the transmission of warning messages or the capability to request satellite state parameters without having a ground station in sight.

Taking into account the above-mentioned mission benefits- & cost-characteristics, DLR is currently exploring the feasibility of a low-cost "On-board Data Analysis And Real-Time Information System" (ODARIS) for small- to medium-sized satellite missions [2]. The general approach is the utilization of existing satellite communication networks, designed for satellite telecommunication [3–5], and mask the low-bandwidth limitations, by massively increasing the information efficiency already on-board. This can be achieved by providing just the most valuable product data at times, when it is most beneficial. One key enabling factor for realizing this concept, is being able to perform data analysis and management directly on-board a satellite. With *Moore’s Law* [6] like advances in on-board computer technology, on-board data processing, or from a broader perspective edge computing on satellites, becomes viable even on small satellite platforms [7, 8].

In this article we will present the current development status of ODARIS. We especially focus on providing details of a technology demonstration experiment of the ODARIS concept, planned for a future satellite mission, scheduled around 2024. The technology demonstration will be prepared in collaboration with the DLR’s Scalable On-board Computing for Space Avionics (ScOSA) team, which develops a high-performance on-board computing platform [9]. As we are currently in a mid-level phase of the software development process of the experiment, the article will focus on the scientific mission targets, the experiment design, the current implementation methods. The paper concludes with the obstacles occurred while developing the system and the way forward.

2. THE ODARIS CONCEPT

In this section a brief overview of the ODARIS concept is presented. A more detailed introduction can be found at the article about the "On-board Data Analysis And Real-Time Information System" experiment, presented at the DLRK 2020 [2].

2.1. History

The development of an ODARIS concept started around 2012 with the experiment "Verification of Image Analysis Onboard a Spacecraft" (VIMOS). With VIMOS the feasibility to perform on-board data analysis on satellite platform should be evaluated. A cloud detection application and change detection application had been developed with the goal to extract useful information directly on board of a spacecraft, to shorten the time between image acquisition and an appropriate reaction from the operators and to avoid unnecessary data downloads [10]. VIMOS was also intended to be combined with an on-board planning system [11] for scheduling additional image acquisition or additional download passes. The experiment was scheduled to be demonstrated on the 2016 launched DLR technology demonstration satellite "Bispectral InfraRed Optical System" (BIROS) [12, 13], but due to technical problems with the satellite platform itself, the demonstration could not be performed. With the VIMOS experiment only utilizing the standard satellite-based ground communication, the reaction time was intended to be decreased slightly by direct event notification via Satellite Telemetry (TM) and by data download prioritization.
For being able to perform event notification within minutes, a real-time communication channel was required. Around 2016 several satellite mission experiments analyzed the feasibility of using satellite phone networks for satellite communication [5, 14]. Based on the results of these experiments we developed an experimental predecessor system of ODARIS, called "Autonomous real-time detection of moving maritime objects" (AMARO), suited for an aircraft flight experiment, which was successfully demonstrated in 2018 [15]. The communication system was based on a text message concept, utilizing the Iridium satellite communication system [16] and allowed us to perform real-time (RT) communication with a latency around a few minutes. The focus on the on-board data processing of AMARO was a ship detection algorithm by analyzing camera images with the open source library Open source Computer Vision library (OpenCV) [17]. As an extension of these classic object detection algorithms, a modern Artificial intelligence (AI) based approach for object detection will be evaluated in ODARIS.
Starting with begin of 2020, ODARIS joined the ScOSA Flight Experiment (ScFE) project. The aim of ScOSA is to develop a reliable, performance focused, heterogeneous and distributed computing platform, by combining Commercial Off-The-Shelf (COTS) and space qualified computing nodes with a scalable software middleware [9]. Within this project an in-orbit demonstration of ODARIS is planned around 2024 on a DLR technology demonstration satellite platform. The goal is to demonstrate ODARIS in a space environment, to evaluate the feasibility of the concept's approach for providing real-time information and identifying the usability, limits and performance of the systems. The successful demonstration shall also increase the Technology readiness level (TRL) of the system.
An additional opportunity for demonstrating ODARIS also occurred in 2022, by joining a satellite mission of the Universität der Bundeswehr München called Seamless Radio Access Networks for Internet of Space (SERANIS) [18].

2.2. Design goals

The design goals of ODARIS have been derived by analyzing typical bottlenecks of satellite operation, especially of small- and medium-sized satellite missions without access to high performance data communication networks like EDRS. In the following the extracted main design criteria for ODARIS are presented.

Real-time communication capability

The key target of ODARIS is enabling near instant access to satellite data. With ODARIS the user shall be capable to query data anytime from the satellite and get a response as soon as possible. For example, an operator can request the satellites health status anytime. These requests will be handled within the QUERY-Service of ODARIS. Furthermore, ODARIS shall provide a PUSH-Service, which enables an operator to define an event or specific criteria. If such an event or criteria occurs, a notification message will be immediately dispatched. For example, it can be used as an alarming service or for periodic updates of specific information.
We decided to keep the language for defining queries and push events more generic, since it may not be clear at design time of the satellite system, which information is required by the operator. A use case may be a step by step inspection of all system components in case of an unknown malfunction. The communication latency is mainly dependent of the communication system, but our goal is to complete a full query/respond cycle within a few minutes.

Space application service platform

ODARIS shall enable other developers to easily implement their own applications, without the need of dealing with the integration details of the space system. The goal is to provide an user-friendly -not space specific- interface, where application developers can operate and communicate with their application.

Broad support for flight platforms

Another design goal of ODARIS is the integrability on a broad variety of space platforms. The flight platforms we currently targeting are satellite platforms, but also High altitude platform (HAP) [19] or other autonomous vehicles would be possible. It shall also be scalable regarding the available computing performance and available communication system. To achieve this goal the ODARIS system shall be:

- compatible to popular on-board computing ecosystems which have a high probability to be supported by current and future space missions,
- adaptable to platform specific interfaces, as for example the communication system,
- integrable for a broad variety of data processing applications as well offer an option for an RT communication system, which can be deployed on a broad variety of space systems.

Quality of service

A crucial nonfunctional requirement of ODARIS is, to be able to provide a high quality of service and system robustness, since the system maintenance measurements may become limited during operation. This also includes the reactivity of the system, allowing low latency data processing and communication, implementation of Fault detection, isolation, and recovery (FDIR) measurements, availability of interfaces to enable thorough system monitoring & inspection and a continuously performed system validation & verification process to minimize implementation errors.

3. SPACE FLIGHT EXPERIMENTS

The ODARIS system was designed for space operability from the start. To evaluate if the concept is feasible for space operations, we started to prepare an In-orbit demonstration of ODARIS targeting a satellite platform in 2018. Within different space experiments, we target to answer the following open scientific and engineering questions, but with the main focus on the applicability of the general ODARIS concept for space operations.

Open scientific & engineering questions:

a) Can real-time satellite based telecommunication networks, designed for ground operation, be meaningfully utilized for real-time communication of (LEO-)satellites ?
b) Can ODARIS be operated reliably within a space environment ?
c) Can we deploy ODARIS on a space system ?

From these major questions we derived also scientific objectives we want to answer within the space experiments.

Scientific & engineering objectives:

- Assessment of the performance of the real-time communication system regarding connectivity
- Assessment of the performance of the real-time communication system regarding information transfer latency
- Assessment of the performance of the real-time communication system regarding data transfer volume
- Assessment of the reliability and quality of service that can be provided by real-time communication system
- Elaboration of the methods for efficiently operating a real-time information system
- Evaluation of the operability of the information system within space operation
- Evaluation of the reliability and quality of service of the information system
- Identification of space-based error sources
- Evaluation of the FDIR-model of the information system
- Evaluation of the performance of the on-board data analysis and information extraction
- Assessment of the feasibility for usage of AI-based methods for on-board data analysis
- Identification of efficient methods for on-board data extraction
- Evaluation of the computing performance achievable for on-board data analysis on medium sized satellite platforms

3.1. ScOSA Flight Experiment

ScOSA is a next generation Scalable On-board Computing for Space Avionics, developed at DLR starting in 2012 [20]. As part of the DLR project ScFE, which started in 2020, we are currently preparing a flight mission scheduled to launch around 2024/2025 on one of the next DLR technology demonstration satellites. Within ScFE, we plan to deploy a RT communication system on the satellite platform, integrate ODARIS within the ScOSA system middleware and perform multiple short- & long-term experiments in space, answering the scientific questions mentioned above. To demonstrate, that ODARIS and ScOSA are capable of supporting modern remote sensing data analysis applications, an AI-based object detection algorithm shall be performed on-board, on image data provided by Visible Spectrum (VIS) camera, a COTS optical camera system. The major challenges for the ScOSA flight experiment is providing a RT communication solution, the system integration of the ODARIS components and setting up a suitable trained AI algorithm for image analysis.
A major part of the current development phase concerns the integration of the ODARIS software components into the ScOSA software environment. For detailed information of the ScOSA software system see [9]. The ScOSA system is a distributed computing system, consisting of High Performance

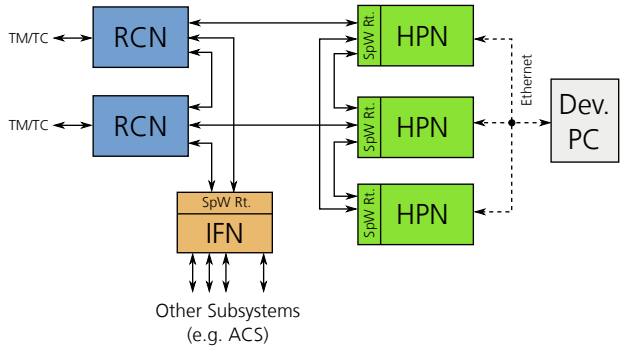

FIG 1. Block diagram of an exemplary ScOSA system configuration (source [20]).

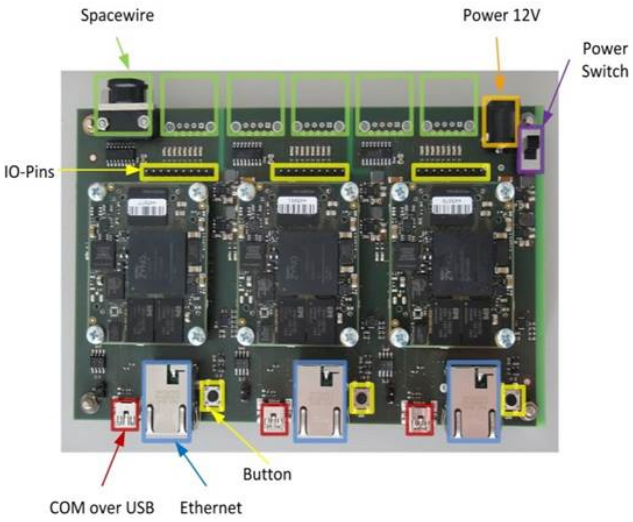

FIG 2. ScOSA development board containing three HPN (source [2])

Nodes (HPN) for high performance tasks, Interface Nodes (IFN) for attaching external components like sensors and Reliable Computing Nodes (RCN) for mission critical tasks, system monitoring and FDIR management. A block diagram of an exemplary configuration is shown in Fig. 1 and an image of an ScOSA development board containing three HPN's in Fig. 2. The HPNs are based on a COTS Xilinx Zynq7020 System on a chip (SOC), which provide a dual-core ARM Cortex-A9MPCore CPU with a frequency up to 1 GHz and an embedded Field Programmable Gate Array (FPGA). The RCNs are based on a radiation hardened LEON3 CPU [21]. Each HPN node will be equipped with 1024 MB system memory with Error correcting code (ECC) support, and 2 GB of NAND flash memory.

The ScOSA On-board computing system (OBC) will consist of 4 Multi-HPN-Modules (MHM) each housing 2 HPN's and 1 RCN. The nodes are connected using two SpaceWire networks [22]. The first one is a slower network with a star-like topology and the RCN at its center. The second one is an additional high-speed mesh-like network, connecting all HPNs. The nodes are completely independent computing systems, running their own operating system. The HPNs run a version of the embedded Poky-Linux

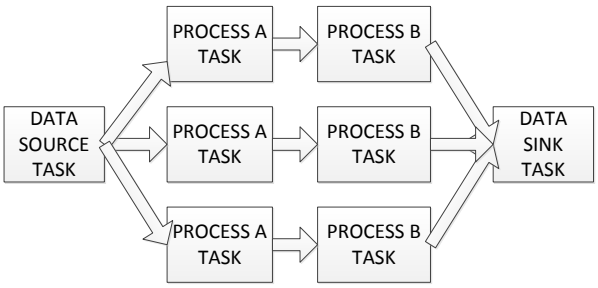

FIG 3. Exploratory ScOSA application data flow schema (source [25]).

distribution from the Yocto Project, which is an open source collaboration project, that helps developers create custom embedded Linux-based systems [23]. The Poky-Linux is a source distribution, providing an up-to-date Linux system with a package registry, offering a broad variety of common libraries and tools used in Linux development environments, such as *OpenCv* [17] and *Boost* [24]. The ScOSA distributed system consist of the ScOSA middleware and ScOSA applications, compiled as one composite executable, which runs on each HPN as a single instance. The ScOSA middleware is implemented in C++ programming language (C++) and offers an C++-API for the applications. The ScOSA middleware is responsible for many system management tasks, such as application execution, FDIR-management, system communication and configuration. The ScOSA system execution architecture is based on a dataflow driven design, which its major building blocks called tasks and channels. Tasks are callable function objects, with predefined data inputs (sources) and outputs (sinks), connected via different channels. An exemplary application data flow schema is presented on Fig. 3. If data is available within all of the data sources of a task, the task is executed, and the output is sent via the channel to the next tasks as input. The ScOSA middleware follows a static configuration paradigm, where tasks are tied to specific nodes, predefined at system design. If a node fails the ScOSA system can reconfigure the system, to migrate tasks, executed on the failing node, to an operational node. More details about the scosa middleware and application integration can be found in [9, 25].

3.2. Seamless Radio Access Networks for Internet of Space

Beginning of 2022 the DLR participates with several experiments on the upcoming SERANIS satellite mission of the Universität der Bundeswehr München [18]. The goal of the mission is to provide a satellite as a multifunctional Experiment-Laboratory for various scientific research topics, including the 6G mobile systems standard, laser communication, Internet of Things and more. Currently more than ten experiments are in development for the platform, where

ODARIS is one of them. The start for the mission is targeted in 2025. The satellite will have a LEO-Orbit and weights around 200kg. Since this project is still an early stage, preliminary designs of the experiments where defined so far. The experiment concept of ODARIS will be similar to the ScFE mission.

4. DEVELOPMENT STATUS

In the following chapter the complete status overview of all ODARIS components will be given.

4.1. System architecture

For a high level overview of the software we will present a brief insight of the ODARIS system architecture. A more detailed view will be provided here [2].
For the implementation of the ODARIS system components the Linux operating system (LINUX) environment and C++ has been chosen. LINUX and C++ are well documented and offer a widely sophisticated set of libraries and development tools. Both technologies are widely used in the embedded world and also increasingly becoming popular within the space ecosystem. One example is the "F Prime" Framework for small-scale flight software systems developed by the NASA Jet Propulsion Laboratory [26]. It is expected, that on many future space systems, LINUX applications will be supported. Additionally, the usage of C++ also delivers a high probability for application developers to reuse already implemented software components and therefore, avoid the need of re-implementing the applications for space usage.
The ODARIS software system uses a service-based approach which is illustrated in fig 4.
The system contains several service applications, each performing a specific task. For inter-service communication and data storage, the design decision was made to use a Structured Query Language (SQL) database. One key benefit of using SQL databases is the availability of SQL itself as an interface language. It is a powerful, widely used and well documented domain-specific language, designed for managing and dynamically accessing structured data.
With the decision to use the SQL language also for formulating query-request, push-events, and inter-service communication, we instantly could provide a generic well-known interface, for system operators and application developers for performing data requests. At the moment we decided to use the file based SQLite database engine (SQLite) [27]. In case it will become a bottleneck in the future development, it can also be replaced with a server based SQL database implementation with only moderate effort.
For the inter-service communication within ODARIS, a message-based approach has been established. All services can send messages to other services, by putting them into a postbox, implemented as a database table. Concurrently, they periodically poll the postbox and extract the messages assigned to them. Using a polling based approach allows a lose coupling of the services. The most critical system bottleneck is the RT communication channel between the space and ground segment. Other limitations like the service-to-service message transfer latency, restricted message throughput and limited data transfer bandwidth are therefore currently negligible to the overall system performance.
More application services can be attached to ODARIS as standalone executables, communicating with other ODARIS services via the database tables. Application developers can also provide their application core functionality in form of C++ compatible libraries. Afterwards, they can integrate their application service by using the ODARIS system libraries, which are providing user friendly interfaces for accessing the databases and other functionality like logging. The concrete implementation method for a service is often dependent on the platform ecosystem. As an example, for the ScOSA project, the services are required to be integrated in form of ScOSA tasks, which will be provided by the ScOSA middleware [9]. For the former AMARO flight experiment, the services have been implemented as standard linux applications, with the system startup, shutdown and monitoring implemented via Bourne-again shell (Bash) scripts. For the decrease of platform dependency and on the other hand the increase of code reusability, the services core functions are provided as libraries. Only a shallow platform integration layer will be used

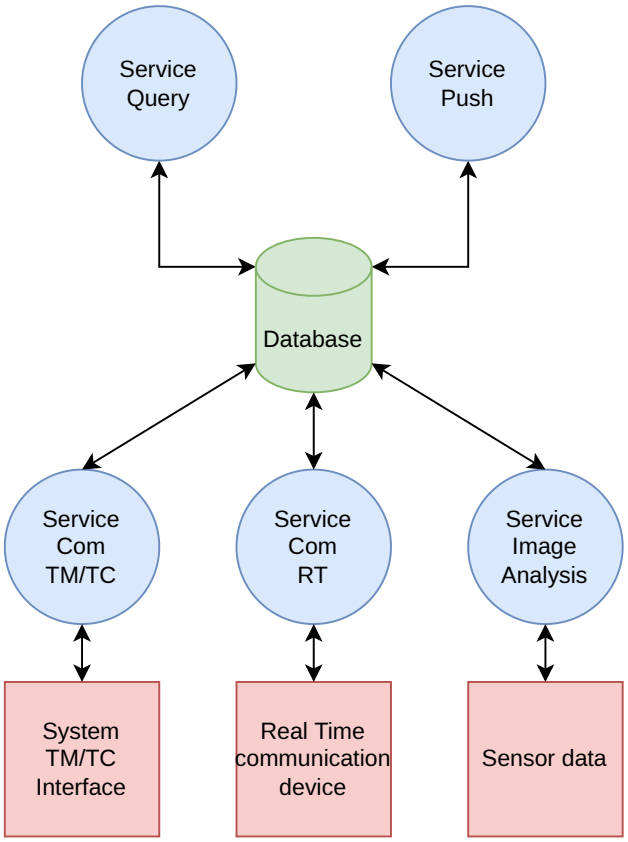

FIG 4. ODARIS context diagram

for implementing the concrete services for a specific space system [25].

4.2. ODARIS Information System

The information system is also implemented as a collection of different services. External communication interfaces like a satellite's standard Telemetry/Telecommand interface (TMTC) or additional RT channels are also meant to be implemented as services, responsible for sending and receiving messages via the satellite's communication channels. The push- and query-functionality is implemented by the accordingly named PUSH- and QUERY-service. For example, if an operator wants to access on-board data from an application service, the operator sends a query message via one of the available communication channels, with the QUERY-service as the destination address. The message will be received by the responsible ODARIS communication service and inserted into the overall message postbox. The QUERY-service pulls the message and answers it by collecting all requested data from the databases, creating a reply message and stores it in the message postbox assigned to the communication service. Finally, the communication service pulls the answer-message and send it back to the operator via the communication channel. Similarly, the PUSH-service performs event checks, by periodically executing SQL-queries stored in a database. If such an event is detected, the result data will be packed in a message and also stored in the postbox assigned to the communication service. Events can be managed and reconfigured anytime by inserting or deleting push-queries using the QUERY-service. At last, different kind of data analysis applications can provide information by inserting product data in a SQL database accessible by the information services. Databases can also be used for controlling the applications.

4.3. RT-communication system

Since our focus is on the software development of the experiment, we searched for an external partner providing a ready-to-use hardware communication solution. With Near Space Launch (NSL) we found a U.S. located company, providing a suitable solution [28] for ODARIS. We are planning to integrate one of their newest products, a half-duplex communication solution based on the Iridium network [3]. It has been designed for providing an easy to deploy TMTC interface for smaller Equatorial Low Earth Orbits (ELEO) satellite missions, without the need of a dedicated ground segment. Based on the published article, see [3], it is a package based low bandwidth solution offering a maximal transfer rate of about 1 byte per second or 80 kB per day. The tested packages of around 18 byte had a latency in the order of seconds to a few minutes. Within the ScFE project, we want to evaluate if this product is a feasible solution for our concept of providing RT capability on satellite platforms.

4.4. AI-based Application

One of the design goals for ODARIS is the usage for other application developers as a flexible space application platform, as stated in the "Design Goals" section earlier in this article. Developers shall be able to integrate their applications without dealing with the integration and communication details of the satellite system. For the In-orbit demonstration of the ScFE project an AI-based image analysis application service will be implemented to generate useful product data, which can be queried via the ODARIS information system anytime during the satellite mission. The image analysis concept will be developed in the scope of another DLR's Vorhaben called "Cognitive Autonomy for Space Systems" (CASSy). One of the goals is to evaluate different concepts to increase the autonomy of spacecrafts and perform data processing already on-board as much as possible. The process of the space application consists of the preprocessing of the available camera images, the inference of the AI-based image analysis and the storage of the analysis results in one of the ODARIS databases, accessible for the information system. Derived from the former AMARO mission, an object detection algorithm for ships or other smaller object was targeted. The camera systems for the flight experiments cannot provide a sufficient resolution to detect such small objects. But since the demonstration focuses on the applicability of such concepts, we plan to use an already well researched and established AI-based cloud detection algorithm instead. The AI-model itself shall be based on an Convolutional neural network (CNN) model, which is currently the most used deep learning algorithm for object detection. The model will be trained on-ground with datasets of satellite images with different amount of cloud coverages. To provide the necessary machine learning algorithms, we use the popular open source Framework Tensorflow (TF) developed by Google [29]. The training process will be performed in a python environment on an x86 computer system within our lab environment. One of the major advantages of TF is the availability of the lightweight version of the Framework Tensorflow Lite Library (TFLite) [30, 31], explicitly developed to run on edge devices for the inference. Furthermore, TFLite provides already a well-documented C++-Application Programming Interface (API) out of the box. The preprocessing algorithm for the satellite's images depends on the used camera system and will be implemented later in the project. In the current development phase, we face the following major challenges for the implementation of an AI-based image analysis concept:

- TFLite library integration into the ODARIS CMake-Buildsystem:
  The TF framework was not intended to be integrate as a plain standalone static third-party library. Furthermore, Google uses its own build

system Bazel [32] to compile the framework from source, which contradicts our CMake-Buildsystem approach. Our goal was to have a clean build-infrastructure for all our dependencies without mixing several build systems. In newer versions of TFLite a CMake build system file was delivered by Google, but still caused issues in CMake variables between dependency libraries of TFLite and other direct dependent libraries of ODARIS. Additionally, it is not possible to compile a self-contained static library of TFLite [33]. By resolving all CMake Variable conflicts by hand and explicitly including all necessary transitive dependencies, we could successfully integrate the TF Lite library into ODARIS.

- Suitable satellite image datasets for training of the AI-model prior to the actual satellite mission:
  A crucial aspect for the usage of current deep neural networks like CNN is the availability of large datasets from the production environment for the training process. Therefore, most known AI-system were implemented as an enhancement of already running systems or have at least enough data available from earlier equivalent systems. For satellite missions this is a major challenge, since the production environment changes on every mission, either by new orbit parameters or camera systems. Already available satellite images datasets are therefore not completely suitable to train an AI-model for upcoming missions. Therefore, we currently collecting datasets from former or current satellite missions with orbit and camera parameters as close as possible. But to reach a good performance of the AI-Model the real camera images from the mission are necessary for the training process.

- Re-training process with actual satellite image data and update of AI-model file on-board:
  Subsequent to the previous challenge, the real performance of the trained AI-Model can only be measured in the production environment itself, which in our case is In-orbit after the satellite's launch. This makes a re-training of the AI-Model file necessary after launch. A critical requirement for such a mission is the Up- & Downlink capacity of the satellite. The downlink is necessary to download enough camera images to create a suitable dataset for training and the uplink will be used to upload the updated AI-Model file or at the least a diff-file for the update's parameter values.

Since some of these challenges can only be resolved in later stages of the project and satellite's system design, we still planning the best approaches and continuously keep in touch with the responsible systems engineers and project managers.
For the SERANIS project a similar AI-based concept is in development. Due to parallel software experiments with focus on AI planned on this mission, we follow a synergetic approach, like the common usage of the TFLite Framework and share of satellite images on-ground for re-training. This would reduce the included dependencies of our software application and lower the downlink demand for the satellite images.

4.5. Application integration

To integrate the ODARIS application into the ScOSA system, three major steps are required:

- Add support for the ScOSA software system environment.
- Prepare and ensure the compatibility of the ODARIS application within the ScOSA middleware.
- Integrate the ODARIS application within the overall ScOSA experimental version.

As stated in the history section, a first version of the ODARIS system has been available at end of 2016. The system was developed and operated on an x86 OpenSuse desktop platform [34], utilizing several external libraries, such as Boost C++ library (Boost), OpenCV and SQLite. Bash scripts were used for starting and stopping the service tasks and the task's supervision. Most of the ODARIS system components have been developed as modern standard C++-14 and were updated later on to a C++-17 application, using the new available language features [35]. A port to specialized operating systems like Real-Time Executive for Multiprocessor Systems (RTEMS) or Free Real Time Operating System (FreeRTOS), which are currently widely used on space systems, is currently not planned due to the high effort it would require. Additionally, necessary third-party libraries may not be supported as well. To avoid such a practical rewrite, we decided as a critical system requirement the availability of a Linux operating system, allowing us to use modern libraries and toolsets. Within the ScFE project, a Poky-Linux distribution will be used on the HPNs, which simplifies the integration process for our software in the ScOSA system environment. An advantage of Poky-Linux is, that a lot of libraries are available as packages. For a first integration test, an equivalent Poky-linux embedded system was set up in our own lab environment, containing our library resources. In the result, most of the external packages required by ODARIS have been already available, resulting in only a few missing libraries, which needed to be compiled by ourselves. As our software infrastructure does not support package management at the moment, we added missing libraries from source via Git-Submodules and integrated the build and deployment of these libraries within our own CMake-Buildsystem.

As second step the integration of ODARIS in the ScOSA middleware was performed. Since ODARIS has a complete orthogonal design and is therefore compatible with the tasking philosophy of ScOSA, we used the ScOSA tasks as call wrappers for our

services. This gives us a clean separation between the middleware's tasking framework and our internal service structure. In the current state, the ODARIS application cannot not utilize the distributed system architecture of the ScOSA system as it has to be executed on one node. Since check pointing of large datasets is not natively supported by the ScOSA system yet, an auto-migration of ODARIS to another node is not possible, at the moment.
Currently, we are in the process of integrating the ODARIS system within the official project's overall experimental system. The major challenge consists of the export and deployment of all software components on ScOSA software system. Until completion of the integration process, we provide all our library resources and additional external dependencies as pre-build binary resources, so that first tests in the project's lab environment are already possible.

4.6. Status summary

In this section, the current development status of all ODARIS components from the previous sections will be summarized.

- A first demonstration setup of ODARIS was setup in our lab environment, where the software runs on an ARMv7-board within a lab environment. A x86-Computer will be used as terminal for the operators.
- The information system, including QUERY- and PUSH-service, is already fully functional and could be successfully verified.
- Since the RT-Communication device by NSL is still in procurement and the specific TMTC protocols of the targeted satellite are also not available yet, the simulation of the communication will be established via an Ethernet-Network between the devices. A server/client-Architecture via TCP/IPv4 between ODARIS and the operator terminal was implemented with the open source library Boost.Asio [36]. This enables us to verify the end-to-end information system concept of ODARIS, where the communication channel can be replaced later on with minimal effort.
- To verify the general functionality of an AI-based image analysis algorithm, the image analysis service currently performs an inference with a pre-trained AI-model for object detection, based on the popular CIFAR-10 database [37]. A set of test images, stored in the mass storage, will be periodically processed to simulate the input from the future camera system. The object detection results will be stored in a dedicated database accessible by the ODARIS information system.
- The operation of ODARIS can be performed via different CLI-scripts. Additionally, a GUI-Application, based on the open source library wxWidgets [38], to interact with all ODARIS features is currently in development. The integration of a minimal version of ODARIS into the ScOSA system environment was successfully performed.

5. CONCLUSIONS AND WAY FORWARD

In this paper we presented the current status of the ODARIS system developed at DLR. We are currently in the process of preparing a flight mission scheduled around 2024/2025, with the intension to evaluate, if the proposed concept for RTcommunication is feasible for space operation, if the ODARIS software system can be reliable operated in space and if we can perform state of the art AI-based image analysis using COTS software components. Within the last two years we have been able make our software compatible for Linux ARM based embedded computing platforms, which will be hosted on the projects space platform and probably utilized in many upcoming space missions. Additionally, we have set up a complete Continues Integration (CI) development platform for testing our software system on the target system platform.

With the next year we plan to perform a thoroughly quality revision of the system and implementing a basic FDIR model.
Furthermore, for the preparation of ODARIS as a software experiment on a satellite platform, several major steps are planned:
As a next step, we will implement the RT-communication service to operate an engineering model of the RT communication system, after the hardware was delivered. Also, an TMTC interface service will be implemented as soon as the necessary information are available from the system engineers.
For the AI-base image analysis service the transition to the targeted cloud detection algorithm will be completed, with the challenges of finding suitable datasets and developing appropriate processes for retraining after the real image data from the mission is available.
The integration of all upcoming ODARIS components into the ScFE project system environment will be continuously carried out.
In parallel, similar preparations for the SERANIS project, which is still in an early stage, will be performed.
At last, during the continuous progress of the system design of the satellite platforms, also typical preparations for a space mission will occur like providing a FDIR-concept, an operational mission concept, necessary technical budgets and detailed AIV processes.

Contact address:

kurt.schwenk@dlr.de

References

[1] Harald Hauschildt, Nicolas le Gallou, Silvia Mezzasoma, Hermann Ludwig Moeller, Josep Perdigues Armengol, Michael Witting, Jörg Herrmann, and Cesar Carmona. Global quasi-real-time-services back to Europe: EDRS Global. In Zoran Sodnik, Nikos Karafolas, and Bruno Cugny, editors, *International Conference on Space Optics — ICSO 2018*, volume 11180, page 111800X. International Society for Optics and Photonics, SPIE, 2019. DOI: 10.1117/12.2535952.

[2] Kurt Schwenk and Daniel Herschmann. Onboard data analysis and real-time information system. In *Deutscher Luft- und Raumfahrtkongress 2020*, Oktober 2020.

[3] Hank D Voss, Jeff F Dailey, Matthew B Orvis, and Matthew C Voss. Thin cubesats and compact sensors for constellations in vleo to deep space. In *Small Sat Conference*, 2022.

[4] Vincent J. Riot, Lance M. Simms, and Darrell Carter. Lessons learned using iridium to communicate with a cubesat in low earth orbit. *Journal of Small Satellites*, 10(1), 2 2021. ISSN: 2327-4123.

[5] Andrew D. Santangelo and Paul Skentzos. Utilizing the globalstar network for satellite communications in low earth orbit. In *54th AIAA Aerospace Sciences Meeting, San Diego, California, USA*, 2016. DOI: 10.2514/6.2016-0966.

[6] Gordon E. Moore. Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, april 19, 1965, pp.114 ff. *IEEE Solid-State Circuits Society Newsletter*, 11(3):33–35, 2006. DOI: 10.1109/N-SSC.2006.4785860.

[7] Alan D. George and Christopher M. Wilson. Onboard processing with hybrid and reconfigurable computing on small satellites. *Proceedings of the IEEE*, 106(3):458–470, 2018. DOI: 10.1109/JPROC.2018.2802438.

[8] Bradley Denby and Brandon Lucia. Orbital edge computing: Nanosatellite constellations as a new class of computer system. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '20, page 939–954, New York, NY, USA, 2020. Association for Computing Machinery. ISBN: 9781450371025. DOI: 10.1145/3373376.3378473.

[9] Andreas Lund, Zain Alabedin Haj Hammadeh, Patrick Kenny, Vishav Vishav, Andrii Kovalov, Hannes Watolla, Andreas Gerndt, and Daniel Lüdtke. Scosa system software: the reliable and scalable middleware for a heterogeneous and distributed on-board computer architecture. *CEAS Space Journal*, May 2021.

[10] Katharina Götz, Kurt Schwenk, and Felix Huber. Vimos - modular commanding and execution framework for onboard remote sensing applications. In *Conference on Big Data from Space (BiDS'14)*, 2014.

[11] Katharina Alice Maria Goetz, Felix Huber, and Maria Schönermark. Vimos-autonomous image analysis on board of biros (doi:10.3182/20130902-5-de-2040.00069). In *Automatic Control in Aerospace*, volume 19, pages 423–428. Elsevier, 2013.

[12] DLR. Dlr biros public website, 2016. https://www.dlr.de/content/en/articles/missions-projects/firebird/biros.html.

[13] Winfried Halle, Christian Fischer, Thomas Terzibaschian, Adina Zell, and Ralf Reulke. Infrared-image processing for the dlr firebird mission. In Michael Cree, Fay Huang, Junsong Yuan, and Wei Qi Yan, editors, *Pattern Recognition*, pages 235–252, Singapore, 2020. Springer Singapore. ISBN: 978-981-15-3651-9.

[14] Hank D. Voss, Jeff F.Dailey, Matthew B. Orvis, Arthur J. White, and Stefan Brandle. Globalstar link: From reentry altitude and beyond. In *2016 Small Satellite conference*, 2016.

[15] Katharina Willburger, Kurt Schwenk, and Jörg Brauchle. Amaro - an on-board ship detection and real-time information system. *Sensors*, 20(5):1324, Februar 2020.

[16] Iridium Communications Inc. Iridium short burst data (sbd), 2020. https://www.iridium.com/services/details/iridium-sbd.

[17] OpenCv. Opencv project homepage, 2022. https://opencv.org/.

[18] SeRANIS Redaktions-Team. Seranis public website, 2022. https://seranis.de/.

[19] Florian Nikodem. Entwicklung der dlr-höhenplattform und ihrer anwendungen. In *Deutscher Luft- und Raumfahrtkongress*, September 2021.

[20] Carl Johann Treudler, Heike Benninghoff, Kai Borchers, Bernhard Brunner, Jan Cremer, Michael Dumke, Thomas Gärtner, Kilian Johann Höflinger, Daniel Lüdtke, Ting Peng, Eicke-Alexander Risse, Kurt Schwenk, Martin Stelzer,

Moritz Ulmer, Simon Vellas, and Karsten Westerdorff. Scosa - scalable on-board computing for space avionics. In *International Astronautical Congress IAC 2018*, 2018.

[21] CAES. Leon3 product page, 2022. www.gaisler.com/index.php/products/processors/leon3.

[22] S.M. Parkes and P. Armbruster. Spacewire: a spacecraft onboard network for real-time communications. In *14th IEEE-NPSS Real Time Conference, 2005.*, pages 6–10, 2005. DOI: 10.1109/RTC.2005.1547397.

[23] Yocto Project. Yocto project website, 2022. https://www.yoctoproject.org/.

[24] Boost. Boost c++ libraries project homepage, 2022. www.boost.org.

[25] Kurt Schwenk, Moritz Ulmer, and Ting Peng. Scosa: application development for a high-performance space qualified onboard computing platform. In *Proc. SPIE 10792, High-Performance Computing in Geoscience and Remote Sensing VIII*, volume VIII of *High-Performance Computing in Geoscience and Remote Sensing*. SPIE Remote Sensing, 2018.

[26] Robert Bocchino, Timothy Canham, Garth Watney, Leonard Reder, and Jeffrey Levison. F prime: an open-source framework for small-scale flight software systems. In *Small Satellite Conference*, 2018.

[27] SQLite Project. Sqlite project homepage, 2022. https://sqlite.org/.

[28] NearSpace Launch. Nearspace launch company website, 2022. https://www.nearspacelaunch.com.

[29] Google Inc. Tensorflow project homepage, 2022. https://www.tensorflow.org.

[30] Google Inc. Tensorflowlite project homepage, 2022. https://www.tensorflow.org/lite.

[31] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. https://www.tensorflow.org/.

[32] Google Inc. Bazel project homepage, 2022. https://bazel.build/.

[33] Google Inc. Tensorflowlite cmake build instruction, 2022. https://www.tensorflow.org/lite/guide/build_cmake?hl=en#step_5_build_tensorflow_lite.

[34] SUSE LLC. Opensuse project homepage, 2022. https://www.opensuse.org/.

[35] Standard C++ Foundation. Standard c++ foundation, 2022. https://isocpp.org/.

[36] Christopher Kohlhoff. Boost.asio library project homepage, 2022. https://www.boost.org/doc/libs/1_76_0/doc/html/boost_asio.html.

[37] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 dataset, 2022. https://www.cs.toronto.edu/~kriz/cifar.html.

[38] wxWidgets. wxwidgets library project homepage, 2022. https://www.wxwidgets.org/.