SSC21-P1-18

Operations System vs. Operating System: Towards a Ground System Supporting Satellite Application Programming

Kai Leidig, Steffen Gaißer, Susann Pätschke, Robin Schweigert, Sebastian Wenzel, Sabine Klinkner
University of Stuttgart, Institute of Space Systems
Pfaffenwaldring 29, 70569 Stuttgart, Germany
leidig@irs.uni-stuttgart.de

ABSTRACT

The term operating system refers to a software component, which traditionally controls the resources and the processes of a computer, and by providing the appropriate interfaces allows for the implementation of custom user applications. This is a common definition, working very well for ordinary computer systems. Yet, what if the operating system and a corresponding application are physically separated, because the computer is within a satellite in space, while the user program is executed on ground? Then, capabilities must be created to connect both, which is of course complicated by the natural boundaries in satellite communication, for example the limited satellite contact times.

Over the past decades, several systems have been developed, which are capable of managing satellite resources and the mission schedule from ground. Although this covers quite well the purpose of an operating system, other terms have evolved in this domain: operations system, ground system, mission control system, ground data handling, etc. The problem though is, those systems primarily focus on the exchange of data and satellite TM/TC, rather than the actual control process. This creates an artificial barrier between ground and space, which harms the development capabilities for ground based satellite applications.

This paper introduces a novel approach for an operations system architecture, which can be considered as a ground extension of the satellite's operating system. This approach shall not break with the existing conventions and definitions, especially in terms of operating systems, but shall introduce a new view on satellite operations. In a layered, functional software architecture, the operating system is the lowest layer between the hardware and the application. Through the definition of the appropriate interfaces in the ground system, a software architecture can be created that actively supports outsourcing parts of the satellite control process to ground.

The proposed approach has great potential for various applications in satellite operations. It supports the implementation of automatic system control processes, the implementation of custom payload applications, and the integration of respective activities into the satellite schedule. As applications and operators interact with a verified schedule, and operations is thus no longer limited to low-level commanding, the approach further reduces the risk of the mission being jeopardized by human mistake.

INTRODUCTION

Imagine the following case! A smartphone app shall be written, enabling the user to select a spot on earth and to trigger an observation of that spot. Further imagine, that the respective satellite system for the job is in place, including the appropriate ground segment. How can such an app be connected to the system?

To answer that question, one needs to understand how satellite operations systems are designed, and what interfaces they provide.

Usually, the development of an operations concept is data driven, and starts bottom-up. Assuming a ground station is in place, this means a TM/TC routing needs to be set-up first. After that, means of decoding Telemetry (TM) and encoding Telecommand (TC) packets have to be implemented, as well as the appropriate user interfaces. The latter involve displays for telemetry evaluation and interfaces for commanding. Both, telemetry and telecommands are persistently stored in archives.

A system as sketched here already provides enough functionality to allow satellite operations. Respective solutions are commonly referred to as Mission Control System (MCS).

As satellite operations evolved, more and more features were added to these systems. Features like a user management, interfaces for automatic command schedule execution,[1] or scripting interfaces. The result are massive and very complex software systems, difficult to engage and to manage.

What if such a system shall be integrated into an automatic operations concept? Unfortunately, an MCS like the described one is not designed for such a use case. Indeed, they provide means for automatic procedure execution, but they don't allow to control the state of the satellite properly. The latter is a key requirement for an operating system though.

An appropriate operations concept must therefore reduce the scope of the MCS, and implement a neat control process instead. Purpose of that control process would be

the managing of the satellite state, like operating systems manage the state of the underlying computing hardware. That would allow somebody to write an application for that system, with all problems like scheduling and system state management being intercepted.
How this can be achieved, and how to cope with the natural limitations in satellite operations, shall be discussed in the following.

CONCEPT

Control Process

Following Save and Feuerberg, automation is a four stage process, as follows: (1) data acquisition, (2) data analysis, (3) decision making, and (4) action implementation.[2] Depending on the degree of automation, a human can be involved into each of these four stages differently, beginning with the stage being executed by the human, and ending with a fully automatic solution.[3]
Purpose of this effort is to automate operations to a degree that a human is only involved into the decision making, and neither into extensive TM analysis, nor the implementation of an action. So, the user can become the operator of the system without even knowing what is happening inside.

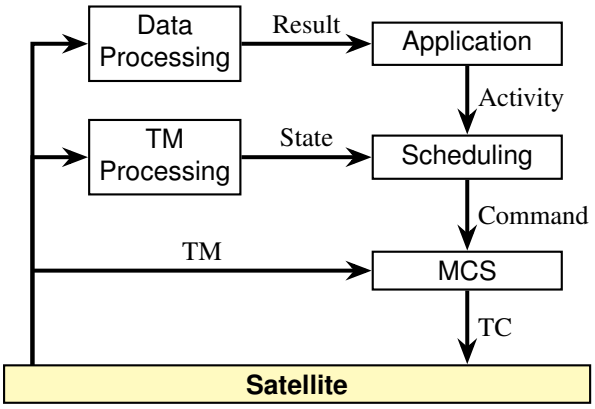

Figure 1: Hierarchy in Satellite Communication

At first, satellite operations is a data handling process. In a very simple form it is shown in figure 1. That process happens on different abstract layers.
On the lowest layer, satellite communication relies on the exchange of TM and TC. That layer provides very limited capabilities in terms of data analysis (e.g. parameter displays), and almost no decision making support. Any decision must be made by the operator on the basis of received TM parameters. Implementing an action means sending an appropriate TC to the satellite and verifying its execution.
On the next layer, TM is further analyzed and transferred into a system state vector. Purpose of this layer is to manage that state vector. Unlike before, an operator does not actively send TCs anymore, but schedules processes, called *activities*. By means of the generated schedule, the satellite state can be forecasted. This is also necessary because activities must be scheduled into the future, and this can only happen on the basis of a given system state. Any command to the satellite is automatically extracted from that schedule.
Purpose of the top layer is the presentation of fully processed results (e.g. imagery data) to the user, and to allow the integration of mission specific applications into the operations system. Since satellite schedule and state are completely maintained by the two bottom layers, such applications can focus on the pure user requirements and don't have to take care about operational issues anymore. Based on the user decisions, an application creates an activity to be added into the mission schedule. The subsequent action implementation can be hidden from the user, but does not necessarily have to.

Unlike decision making, which always happens within one layer, data analysis, and action implementation happen across these layers. Which means that for the implementation of such a high level operations concept, all the underlying layers must work fully automatic.

Interface Type Definition

In computer science, an interface is a connection between two components, over which information can be exchanged. This can be both a hardware or a software connection. Those interfaces are necessary, since they define a common communication format between components. Most components in a software system are written using different programming languages and different data formats. If they need to exchange information, they need to agree on a common format and path for the communication. Within the mission operations system described here, interfaces define the connection between different software components. For example, this can be the connection between a scheduling component and the MCS, over which commands can be displayed, issued or removed from the current schedule.
In this context, the interface does not describe the low level connection over which the actual data is being transferred. This is done through a Message Oriented Middleware (MOM) over which components can exchange information via so called messages. These messages can contain any type of data understood by both sides. This exchange is handled asynchronously through the MOM, meaning the receiving component does not have to be available at the exact moment, the sending components posts its request. The requests are persistently buffered within the MOM and will be delivered to the receiving side, as soon, as it is ready. Below this, the raw data is transferred through a default IP-based network. However, all this is fully hidden from the com-

ponents.

On a higher level though, each component defines an interface, through which data and information can be requested from them or send to them. This definition contains a set of functions, that can be executed by the component with all necessary parameters and return values, including their types.
As described earlier, the underlying messaging structure works asynchronously. Therefore, the defined interfaces also have to be able to handle the asynchronous communication. This means, that the requesting side has to be able to send a request and forget about it until an answer is received. Therefore, a request ID has to be attached to every request, that requires an answer, so the requesting side can correlate the incoming answers to their respective requests.
In the following sections, a selection of important interfaces are described. This includes for example the command interface at the MCS, the telemetry interface and the activity interface. These interfaces are important within the system, since they are the points different components can attach to. They define the possible functions of the overall system and allow for an easy transfer of information across abstraction layers.

Telemetry / Telecommand

A system capable of operating an arbitrary satellite mission must cope with different communication protocols on the space link. The minimum requirement was to be able to fulfill the ECSS PUS Standard[4]
Assuming that a typical link uses any sort of frame or packet based protocol, an abstraction layer must be added to hide the specific protocols from the rest of the operations system. Translating between any low-level mission specific protocol and the abstract implementation is part of the MCS. This allows for the implementation of generic, mission independent scheduling methods by higher-level system components. To elaborate the concept, this paper focuses on the uplink path, although the problem applies similarly to the downlink.
The exchange of commands inside the system is based on the abstract command described in the next section. At first, the MCS requests the command definition from the respective mission database. After that, the translation unit parses the command and parameter information and builds an abstract command. Eventually, the abstract command is translated into the mission specific protocol and released as specified.

Abstract Command

Based on the assumption of a packet based protocol, an abstract command layer can be used to hide the real commanding protocol from the rest of the system. In the current implementation the abstract command has two representations: *CommandDefinition* and *Command*.
The *CommandDefinition* contains the complete command information including limits, calibrations and complex parameter dependencies such as the grouped parameters from ECSS PUS. All parameters provide a description and the type as well as all possible limits and calibrations. In addition it contains the execution time and the release information like release time or interlocks.
The *Command* contains only the command identification and the parameters with their value and type as well as the release and execution information.

Activity

The *activity* is a means of interfacing with the satellite on system level. Unlike the abstract command or a TC, the activity does not address a particular element in the satellite on-board software though. Instead, it indirectly alters and verifies the system state in a predefined manner.

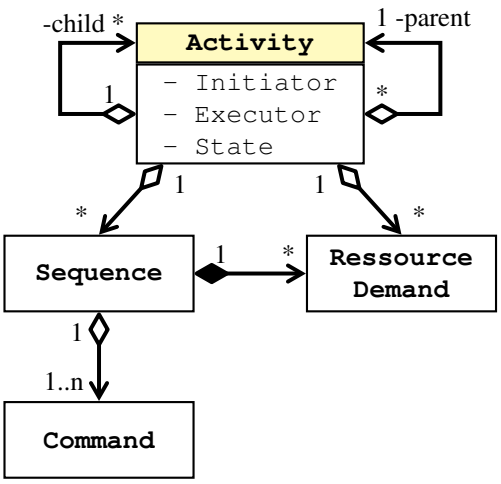

Figure 2: Structure of the Activity Class

The basic structure of an activity object is shown in figure 2. Each activity has an *initiator* and *executor*. The initiator is the entity on ground, e.g., an operator or a software application, which has requested the activity. Identifying the initiator from the activity allows providing activity information such as status updates to that entity. The executor is a representation of the system, which processes the activity. Normally, activities are processed by satellites, but it can also be any other type of system, e.g. a ground station. Within this operations concept, each executor is represented by an individual component, called the Mission Planning Tool (MPT).
The *state* object is a tuple indicating on the one hand the stage of the execution progress, and on the other hand whether the activity is currently processed or not.
Commands, which are executed during the activity are organized within a *sequence*. Processing that sequence means that the satellite must provide an appropriate

amount of resources. The required resources, for a successful execution are specified in a *resource demand* object.
Another important quality of an activity is its nesting capability. This is realized by the fact that each activity can reference a variable number of other activities as child objects. Vice versa, an activity can reference one other activity as a parent object.
Building an Activity in this way has the following advantages:

- Decomposing a (parent) activity into a number of child objects allows splitting of tasks into subtasks.
- The fact that each child activity can have an independent executer allows dividing tasks amongst different collaborating systems.
- The resource demand object allows to estimate the future resource consumption and thus the prediction of the future system state.
- The implemented resource modeling approach is further a means of resolving conflicts between activities.[5]

IMPLEMENTATION

Main purpose of the proposed operating systems is to simplify the life of the satellite operator as much as possible. It therefore provides abstract interfaces that can be used to perform operational tasks such as mission planning and TM monitoring. To enable the user to do so, the previously proposed interfaces can be used to communicate with the four core parts of the operating system. The part closest to the satellite is the virtual ground station, followed by the Mission control system (MCS), the Flight Dynamics (FD) and the Mission Planning Tool (MPT). The interaction between these parts is illustrated in figure 3.

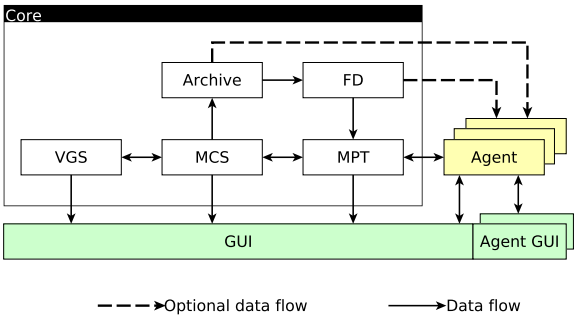

Figure 3: Overview of the operating system components

The virtual ground station (VGS) is responsible for the network connection to the respective antenna system, e.g. via a virtual private network (VPN).

Mission control is translating abstract commands into specific TCs that can be sent to the satellite. It therefore represents a border between generic instructions and mission specific commands. The MCS is also receiving the acknowledgment for the sent commands, ensuring a successful reception and execution on board the operated system.
Through the evaluation of housekeeping TM Flight Dynamics is keeping track of the satellite orbit. The Flight Dynamics tool can provide information about contact times with observation targets or ground stations.
Mission planning can be considered as the central scheduling element of the operating system and is therefore responsible for activity and resource management. It contains the mission schedule and therefore holds all activities from the past and the future.
Custom, external applications can interact with these four components using the interfaces described above.

The operating system can be seen as abstraction layer for the applications in a similar way as an operating system for a computer behaving.

A special type of external application is a software component called agent. Agents use the activity interface to insert activities into the mission planning. They collect information from the TM/Payload archives, or flight dynamics. Agents have a specific task and use the operating system to fulfill it. These arbitrary tasks will vary from mission to mission. Nevertheless, many missions might need similar ones like:

- Ground station pass scheduling for the satellite and the antenna
- TM data dump planning
- Payload mission planning
- Payload data dump planning
- Satellite monitoring and user notification

The proposed design enables access to the operated system without deeper knowledge of the underlying system and the satellite.
To manage, maintain and monitor the operating system a browser based Graphical User Interface (GUI) is used. This GUI is also controlling the user access and authentication to the operating system. Depending on the used scenario and the role of the user, the previously mentioned agents are also controlled from the same GUI, creating a seamless experience for the user. Nevertheless, by using external agent GUIs, applications can be completely separated from the underlying operating system. This is a valuable feature for high level users, not interested in satellite operation itself, but the access of data.

OUTLOOK

After introducing the novel approach for an operations system architecture which supports satellite application programming, different possible outlooks can be contemplated. Since the user is only involved into the decision making, the degree of automation in the satellite operation process is maximized while the probability of human error occurring is minimized.
Currently, the entry barrier for potential satellite operator personnel is high due to the complexity of the satellite operations, lacking automation. This also leads to not fully using the satellite resources due to the complexity of mission planning but also on the lack of adapting to the current operations states. However, the number of satellites in space is continuously growing which potentially increases the number of unused resources on the satellite platforms as well as the complexity on ground.
The higher degree of automation of the here-presented ground system architecture, allows an easier adaption and scaling of the ground system to new satellite missions or new ground stations. This easier adaption and scaling also improves the operation of a single satellite mission since the ground system can better react to anomalies, human errors and emergency mission planning.
The easier adaption and scaling could also open the access for satellite missions to a bigger community which could be directly involved in the planning of the satellite mission. The better adaption of the satellite operations could allow potential customers to actively take part in the satellite mission and increase its mission return. This strongly differs from the state-of-the-art satellite operations concept where the satellite operation is performed by highly-qualified personnel and the scientific data is just forwarded to the customers. Thus, the customers cannot directly plan a detailed observation for a defined spot.
Here, we can come back to the use case described in the introduction: The described ground system in this paper could provide an intuitive and user-friendly frontend through web access, or even through a smartphone app, to plan custom Earth observations. The customer selects a spot on Earth and triggers the observation with a satellite missions, that matches the set requirements. The customer does not have to interact directly with the ground system or to have deep understanding of the satellite system. After the satellite has performed the observation and downlinked the corresponding data, the customer can access and download the custom observation data.

CONCLUSION

This paper presented a novel satellite operations concept, aiming for the interception of all data handling processes for the user.
At first, an automation concept was introduced, which implements a satellite control process on different abstract data handling layers. After that, the required interfaces for the realization of that process were introduced in a bottom-up approach. By means of the activity as the top-level interface, users and application are able to interact with a satellite schedule, which is autonomously managed by the system.
Subsequently, the system design and a the most critical elements in terms of data handling were introduced. With all the process scheduling and the system state being managed internally, the proposed approach therefore fulfills two major qualities of an operating system.
The design further provides the necessary means, which allow the integration of customized applications into the system. Some use cases, and the worth for the scientific community were discussed in the outlook.

References

[1] Pearson, S., Reid, S., and zur Borg, W., "A Full End-to-end Automation Chain with MOIS, PLUTO, MATIS, SMF and SCOS-2000," *SpaceOps 2014 Conference*.

[2] Save, L. and Feuerberg, B., "Designing Human-Automation Interaction," *Human Factors: a view from an integrative perspective*, edited by D. de Waard.

[3] Sheridan, T. B. and Verplank, W. L., "Human and Computer Control of Undersea Teleoperators," Tech. rep., Man-Machine Systems Laboratoatory, MIT.

[4] ECSS, "ECSS-E-ST-70-41C Telemetry and telecommand packet utilization," Tech. rep., ECSS.

[5] Leidig, K., Gaißer, S., Mohr, U., Schweigert, R., Wenzel, S., Klinkner, S., and Eickhoff, J., "Multi-Mission Operations System supporting Satellite Constellations," *16th International Conference on Space Operation*, No. SpaceOps-2020,4,14,x281, International Astronautical Federation (IAF).