Decentralised Avionics and Software Architecture for Sounding Rocket Missions

Jasminka Matevska
Hochschule Bremen
Bremen, Germany
jasminka.matevska@hs-bremen.de

Enrico Noack
Airbus Defence and Space GmbH
Bremen, Germany
enrico.noack@airbus.com

Manuel Reinhold
Hochschule Bremen
Bremen, Germany
mreinhold@stud.hs-bremen.de

Eike-Kristian Diekmann
Hochschule Bremen
Bremen, Germany
ediekmann@stud.hs-bremen.de

***Abstract* — This paper describes our ongoing work in the context of the TEXUS/MAXUS sounding rocket program. Based on analysis of requirements, technologies and tools, we propose a solution to cope with increasing number of software applications and hardware components due to decentralisation of the communication system based on the OPC UA communication standard for distributed services. Our main goal is to provide an efficient avionics and software architecture configuration for both the initial development and the maintenance while assuring consistency and increasing availability and reliability of the system for different experiment and mission scenarios.**

Keywords* — *decentralised avionics and software architecture, sounding rockets, hardware / software interfaces, sensor data, experiment control, decentralised / distributed services, "Industrie 4.0", OPC UA, configuration, monitoring, error handling, availability, and reliability

I. Introduction

Since April 2017, the flight of the MAXUS 9 rocket, a framework from the „Industrie 4.0"agenda [1] is used to control the spacecraft experiments. This agenda provides a platform for automation and data exchange in industrial context. Similar tasks are required for spacecraft operation. For example, on-board each TEXUS/MAXUS sounding rocket, a control of three to five experiments is performed. The responsible scientists from various disciplines and experiment engineers monitor these experiments. Each experiment has to be connected to the system and its sensor data has to be collected in order to establish the appropriate control. That is why it is obvious that an „Industrie 4.0"platform is a good candidate as a reference system for spacecraft control [2]. The transition to the new system enables features that are very useful and reasonable, but it is challenging since it requires new concepts as shown in this paper.

In the conventional TEXUS/MAXUS sounding rocket system (since December 1977), a purely centralised data exchange between space and ground was in use. There was no network connection between the flight and ground computer. For the data exchange, a proprietary communication protocol was in use, which required specialised hardware. It was not possible to operate a single experiment without the specialised hardware.

The replacement of the proprietary communication with the standardised Ethernet/IP (Internet Protocol)-based interface and OPC UA (Open Platform Communications Unified Architecture) [3], [4] enables the operation of an experiment with just one standard laptop (or PC). The decision

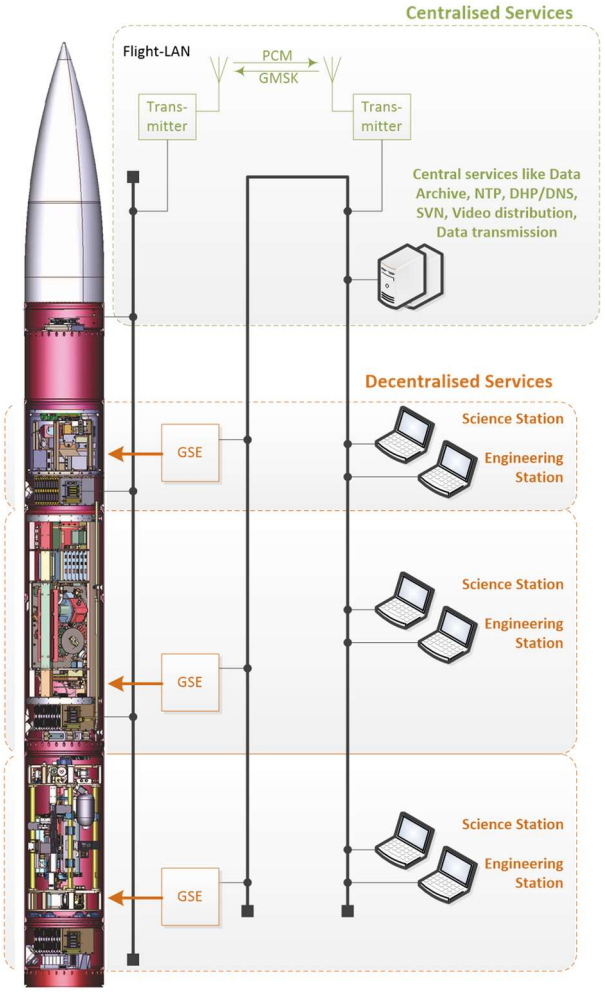

Fig. 1. TEXUS/MAXUS Avionics and Software Architecture

for implementing OPC UA is based on different trade-offs performed by experts and students documented within the master thesis [5]. The reference avionics and software architecture is presented in Fig. 1.

Furthermore, now it is possible to perform the experiment on various execution platforms such as parabolic flight, drop tower and even in the laboratory from the scientists themselves. However, this decentralisation is leading to two basic challenges [6], [7], and [8].

The first challenge arises when the experiment goes on a campaign on its own. Some of the "home" services must be also available during this campaign. That can be as well

simple services like DHCP/DNS (Dynamic Host Configuration Protocol / Domain Name System), NTP (Network Time Protocol), as also some more sophisticated services like backup services or telemetry data storage. Therefore, parts of the services must join the campaign or must be globally available. Another important point at this scenario is the consistency of the overall system. The telemetry data / software produced during the particular campaign must be re-integrated into the system that stayed at home.

The second challenge is the growing number of computers. Due to distribution of services, the different functions are deployed on different hardware platforms. Even the Ground System Equipment has today a separate data interface. While in the past, only two computers where used (one on-board and one on ground) to control the experiment, today more than six computers are common. Three are in use in the flight system (experiment control, data services, video services) and three on ground (Ground Support Equipment with an own data interface, separate control stations for scientists and engineers). The overall network is hosting today more than 30 computers, whereby each single is important for the mission success. We need to keep track of each single service. Additionally, the system configuration is changing quickly. Every two years three sounding rockets with different experiments are launched, each having its own configuration adapted to the specific scientific needs. The growth of hardware and software hast to be managed without decreasing availability and at the same time without increasing the effort to maintain such a system.

This paper presents our ongoing work on appropriate concepts in order to answer these requests.

II. Requirements on the Reference Architecture

In order to meet the increased requirements on the ground system a reference decentralised avionics and software system architecture is developed. By keeping the effort for configuration, maintenance and update of the system as low as possible, we can guarantee a permanent stability and consistency thus providing high availability and reliability of the system. .

The requirements include several criteria that shall be met by the reference architecture. These are listed as follows.

A. Controlled Environment / Scenarios

The system functions shall stay stable and comprehensible in the following scenarios:

1) *Ground testing with an experiment in the laboratory*
2) *Scientific tests with an experiment on a parabolic flight*
3) *System tests with different experiments*
4) *TEXUS/MAXUS Flight Operations*
5) *Post-Flight Evaluation*

B. Modular Architecture

Changes to a configuration in a particular scenario (e.g. software updates) shall not affect the correctness, functionality and executability of other configurations.

C. Recovery

"An error is that part of the system state that can cause a subsequent failure. An error is detected if its presence is indicated by an error message or error signal" [9]. If an error occurs in the system, this shall be recognized and a corresponding action should be proposed or executed in order to prevent any failure. Furthermore, if certain software is no longer functional, it shall be possible to easily and quickly recover from the failure and set up a new system. This system shall be identical to the initial system before its failure.

D. Transparent Interface

After recovery, the user shall have an unmodified interface (hardware & software configuration). Windows 10 is used as the operating system. The reason for this is that users can operate and manage Windows machines themselves. In addition, software is used that is only available for Windows.

E. Mobile Systems

It shall be possible for the ground system to be used on different locations (for different scenarios) with a full functionality. The system and also the required services must be available offline during a mission

F. Effort

The effort for the configuration and maintenance of new system shall be as low as possible. The resulting costs can be considered secondary. Here the trade-off between costs and effort has to be considered. The reduced effort can save working hours, which the employees can use efficient for other engineering work. This finally reduces the overall costs.

G. Availability / Reliability

"Availability is a system's readiness for correct service. Reliability is a system's ability to continuously deliver correct service" [9]. In order to carry out any space and thus a sounding rocket mission, many different sub-systems have to be available and properly work together. Starting with an appropriate mission, spacecraft and sub-system design to the space mission operation, the avionics and software system components are the link between the spacecraft and the Ground Utilities. Therefore, we have to ensure that they are available and reliable operating as specified.

III. Proposed Concept

We performed an analysis of the requirements, suitable technologies and tools in order to find an appropriate solution. We decided that a DevOps (Development/IT-Operations) toolchain/pipeline is suitable for fulfilling the criteria, as it is capable of automating the setup of user instances as far as possible. In a trade-off, we compared several concepts. We analysed the advantages and disadvantages of the considered concepts and their suitability for meeting the requirements. Subsequently, a decision was made in favour of the proposed concept. For the TEXUS/MAXUS specific environment, a pipeline built of the tools mainly from HashiCorp (https://www.hashicorp.com/) is considered suitable. HashiCorp provides products for the provisioning and configuration of individual systems up to system landscapes. An optimal solution can be achieved with the tools Packer, Vagrant and Ansible. Ansible was not developed by HashiCorp, but is an important part of the pipeline.

A. Tools

- *Packer* is used to create machine images. These images can be created from a single source configuration for multiple platforms such as Amazon Machine Images (AMI) for Amazon Elastic Compute Cloud (EC2), VMDK (virtual discs) and VMX (configuration files) for VMware or OVF (Open Virtualization Format) exports for VirtualBox.
- *Vagrant* is an application for creating and managing virtual machines (VM). With Vagrant it is possible to create and manage complete virtual machine environments with a single workflow. This drastically reduces the setup time of the development environment.
- *Ansible* is a tool that automates the configuration and administration of systems. This ranges from simple to highly complex tasks. Only SSH (secure shell) access is required to access remote systems and the system can be managed without any additional software.

B. Architecture

For the implementation, a server is set up for configuration and deployment. The server has the tools for provisioning, configuration, execution, and testing of VM images as mentioned in the previous section. The required software and the fully set up VM images are made available to the servers, laptops and computers in the network, using file share service. The provisioning of the VM images is handled by the tool Packer. Packer uses files in JSON (JavaScript Object Notation) and XML (Extensible Markup Language) format for the description. The subsequent configuration of the provisioned VM images is done with Ansible.

Depending on the component, different playbooks are used, which support the required software installations and execute them. After completion of the VM Images, this can be tested with Vagrant. Using the command line, the Vagrant tool can start and run a virtual machine in Virtualbox in minutes. This allows the engineer to test the functionality of the virtual machine. Since no automated tests are available for Infrastructure as Code, the only way to do this is to manually review the built virtual machine. The effort for this is limited to a minimum. Once the virtual machine has been tested, it can be made available to the other engineers and scientists. For this purpose the image is released in the file share. The written code is also committed and pushed into the Git repository for versioning. The underlying concept of this architecture is also called Immutable Infrastructure. A schematic procedure is presented in presented in Fig. 2.

The chosen concept ensures that the requirements are met very well. The controlled environment can be guaranteed by using Infrastructure as Code, because the state of the system is always identical, as it is never modified after deployment, thus ensuring the transparent interface.

Since configurations are encapsulated in a virtual machine, it can be ensured that other configurations are not affected. This also makes it easier to recover failed systems and configurations. In addition, online services have been avoided as far as possible for the use of the concept, so that offline operation is possible.

Due to a high degree of automation and the use of open source tools, the resulting effort can be kept low.

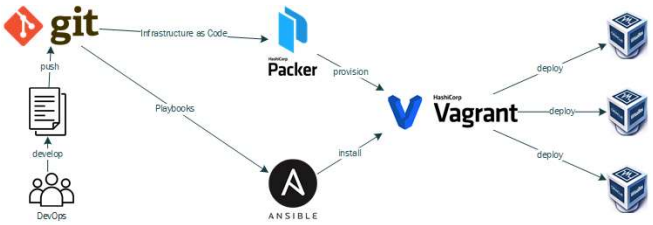

Fig. 2. Concept Immutable Infrastructure

C. Other considered Architectures

Another approach was to outsource all systems and services to a public cloud. From a technological point of view it would be a good approach. Provisioning and configuration would also be a lot better and easier. The shortcoming of this option is that the system would no longer function or be accessible in offline mode.

The On Premise Configuration was also considered. Only the tool Ansible would be necessary. The disadvantage of this approach is that the actually consistent setup is disturbed by manual actions of users (installation of additional software, misconfigurations). Correcting these actions individually would be very time-consuming.

IV. Improving Availability / Reliability

A high level of availability of all systems is required to operate the ground station. System errors and lack of resources must be recognized in a short time or even predicted in order to be able to take countermeasures and prevent a system failure. To assess the system status, information about the systems have to be recorded and evaluated according to predefined rules.

The collected information has to be integrated into the system communication interface concept in use. The TEXUS/MAXUS project uses the open “Industrie 4.0” standard OPC UA to provide, for example, telemetry data and the data from scientific experiments. We are working on extending the existing monitoring system, in order to integrate it into the OPC UA infrastructure and include monitoring data analysis. Fig. 3 shows the proposed components extending the existing system.

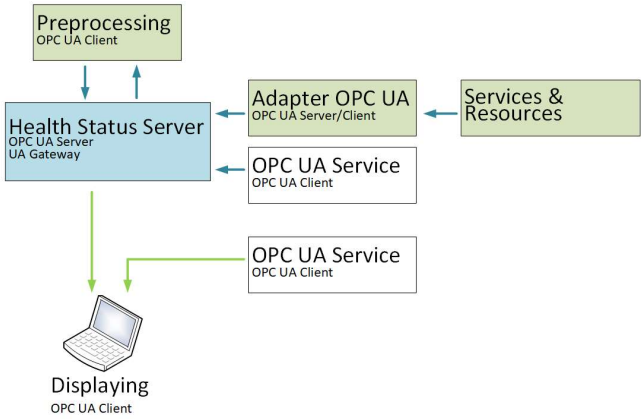

Fig. 3. TEXUS/MAXUS Error Detection Architecture

A. Information Collection

The telemetry data and data from the experiments are already provided as OPC UA nodes. Additional necessary information shall be collected from infrastructure, development stations and other PC systems. At TEXUS/MAXUS, these systems are mainly operated with Windows operating systems. On these systems, so-called agents, that implement Windows Management Instrumentation (WMI), are used to collect necessary information. Furthermore, our systems collects information on processes, services, resources such as CPU, RAM, hard disk space, network status, etc. They aremade available as OPC UA nodes shown as "Services & Resources" as well as "Adapter OPC UA" in Fig. 3.

B. Information Rating

The information provided can be fetched centrally from the Health Status Server via the OPC UA gateway. This information is then called up by pre-processing, where error detection and error evaluation is performed. If necessary, measures for problem solving are proposed or carried out. Furthermore, a distinction hast to be made between different application scenarios in order to select the corresponding method.

C. Relevant Scenarios

We consider the application scenarios 1) Ground testing with an experiment in the laboratory, 3) System tests with different experiments and 4) TEXUS/MAXUS Flight Operations from section II.A for the monitoring and analysis of the system.

V. Error Handling

According to the requirements, we propose the following rule based error handling approach.

A. Error Detection

The approach of an expert system based on a knowledge database filled by specialists was chosen for error detection. Since a rocket launch is a rare event, AI (Artificial Intelligence) approaches such as neural networks or deep learning are only suitable to a limited extent, since many training data is required there. In addition, operators and developers are required to ensure that errors are under the control of specialists. A distinction is made between error scenarios because the different scenarios run different systems and processes and intentionally show different behaviour.

B. Error Rating

An error evaluation takes place depending on the application scenario. The criticality differs depending on the application scenario and is divided into the following categories according to VDMA (Verband Deutscher Maschinen- und Anlagenbau e. V.) standard sheet 24582 [10]:

- Defect / error
- Critical condition
- Warning
- Good
- No status statement

C. Error / Fault / Failure Occurrence, Scenario Definition and Classification

TEXUS/MAXUS developers store error, fault and failure occurrences, identified scenarios and their classification using common tools such as Microsoft Excel. The assessment of the monitoring and analysis system is based on these entries.

D. Recovery Actions

If an error announces itself by a fault or a failure has already occurred, the monitoring system reports this event with the corresponding criticality and recommends actions to remedy the problem. In addition, the failure will be traceable hierarchically to the fault as the origin of the error. This helps to narrow down the errors and correct them. Measures and rules for detecting and correction errors are provided by experts in the knowledge and rule set database.

By continuous monitoring of all systems with appropriate recovery actions in the case of errors and failures, we can achievehigh availability and reliability of the system.

Summary

This paper shows a work in progress within the TEXUS/MAXUS sounding rocket program. A standardised reference system based on "Industrie 4.0" OPC UA communication platform is facing new challenges due to distribution of services, and increasing number of software applications and hardware components (mainly PCs and laptops). The configuration and maintenance of the systems for different experiments and mission scenarios shall be provided in an efficient and consistent way, monitoring, information collection and error handling including recovery mechanisms shall be implemented in order to improve the availability of the systems. Based on requirements, technology and tool analysis we propose an appropriate avionics and software architecture for sounding rocket systems and missions. Currently we are working on implementation of the proposed solutions.

References

[1] Bundesministerium für Wirtschaft und Energie, Bundesministerium für Bildung und Forschung. Plattform Industrie 4.0. https://www.plattform-i40.de.

[2] E. Diekmann, M. Reinhold, J. Matevska, E. Noack. „Idustrie 4.0 in der Raumfahrt". Deutscher Luft und –Raumfahrt Kongress 2018. „Luft- und Raumfahrt – Digitalisierung und Vernetzung". Sep. 2018.

[3] OPC Unifed Architecture Foundation. url: https://opcfoundation.org/

[4] Freeopcua Project. OPCUA Server and Client implementation. Aug. 2017. url: http://freeopcua.github.io/.

[5] P. Kathmann, J. Scheichel. Masterthesis: „IP-Kommunikation auf Forschungsraketen". Hochschule Bremen. Sep. 2017

[6] J. Matevska, "ibacus (IP-BAsed CommUnication in Space)", Presentation ESC Kiruna, Mai 2018.

[7] P. Grashorn, A. Stein, E. Noack, „TEXUS 2.0: Neue Konzepte für Raketenexperimente in der Zukunft", Presentation ESC Kiruna, Mai 2018.

[8] E. Noack, J. Matevska. "TEXUS made in Bremen - Neues Datensystem für TEXUS kommt aus Bremen", Presentation, Sternstunden 2018.

[9] A. Avizienis, J.-C. Laprie, B. Randell, C. Landwehr. Basic Concepts and Taxonomy of Dependable and Secure Computing. In: IEEE Transactions on Dependable and Secure Computing 1, 2004, Nr. 1, S. 11 – 33

[10] VDMA Verband Deutscher maschinen- und Anlagenbauer e. V. Einheitsblatt 24582: Feldbusneutrale Referenzarchitektur für Condition Monitoring in Fabrikautomation, Berlin: Beuth Verlag GmbH, April 2014