18th International Conference on Space Operations, Montreal, Canada, 26 - 30 May 2025.
Copyright 2025 by DLR.
Published by the Canadian Space Agency (CSA) on behalf of SpaceOps, with permission and released to the CSA to publish in all forms.

SpaceOps-2025, ID # 458

The Way to Service-Oriented Architectures – Lessons Learned from Introducing HCC at GSOC

Hauke, Armin[a]*

[a] *Department Communication and Ground Station, German Space Operations Center, Münchner Straße 20, 82234 Weßling, Germany, armin.hauke@dlr.de*
* Corresponding Author

Abstract

With the project HCC, the German Space Operations Center (GSOC) is on its way to provide its capabilities as services and build ground segments for space operations in a service-oriented architecture. HCC has been first utilized in an operational environment in the context of sounding rockets, operated by Moraba, DLR's mobile rocket base, in 2022. After this successful prove of its proficiency, HCC is going to be introduced step by step in all the other fields of operations at GSOC: satellite operations, human space flight and experimental setups like the LUNA analogue in Cologne. In this paper, we share our experiences and findings on this journey. Overall, the change to service-oriented architecture is a great improvement. But the necessity to convert legacy systems into this new architecture, as well as guiding operations personnel to the new paradigms might be also a major effort. We present pitfalls and benefits and also ways to gradually adapt humans and systems in order to make such a transition as smooth and painless as possible.

Keywords: Service Oriented Architecture, Change Management

1. Introduction

In recent years, developments in space flight have been dominated by the trend towards becoming more agile. This means that smaller, less complex and more cost-effective satellites are being put into orbit faster and faster. At the same time, control systems on ground must also take this trend into account. On the one hand, such ground segments must be made available much more quickly, but they must also be much cheaper to operate.

The technological answer to this challenge is undoubtedly modularization and the use of service-oriented architectures. These two design features make it possible to quickly and precisely assemble ground segments from existing components. They also make it possible to significantly increase the degree of automation of the overall system.

However, service-oriented ground systems are not only essential for small missions with a low budget. It is also becoming increasingly important for large, complex missions to no longer operate from a single control center without exception. Instead, concepts such as "distributed operations" allow the various tasks to be distributed across several entities. In addition to the distribution of the various activities, this also enables new and more efficient concepts for redundancies within the ground segment and the resilience of the overall system.

All agencies and commercial providers of solutions for space flight operations are actually working on the creation of and transition to service-oriented architectures. The solution being developed at GSOC is called HCC [1,2].

2. The HCC-Infrastructure

A quick but very important realization when transferring the existing expertise for space flight operations at GSOC into a service-oriented architecture was that a suitable platform is needed as infrastructure for a service-oriented ground segment. Although there are some free or commercial products, such as cloud solutions, container orchestration and message brokers, none of these tools were able to meet all the requirements of GSOC. This is also due to the fact that we support a wide range of different types of missions at GSOC and want to continue to support them with a common platform. Two areas in particular gave rise to KO criteria: Safety aspects and the spatial location of the components.

In terms of security requirements, a system was needed in which access to data and data types can also be strictly regulated within the system. In addition, the set of rules for assigning rights must also be changeable at runtime in order to be able to adapt to special phases in mission operation. The requirements of manned space travel from the experience of operating the Columbus module as part of the ISS were used as a guideline, as well as the requirements of a critical infrastructure such as the Galileo project. Although corresponding strict safety precautions should be intrinsically anchored in the overall system, it must nevertheless be possible to deactivate these precautions, for example in order to set up a ground segment in cooperation with universities that can be used in a university environment without critical safety aspects.

The requirements for the spatial distribution of the overall system are just as wide-ranging. For more restrictive customers, a ground system must be able to be set up completely and autonomously in our control center, where it must also be physically separated. The concept for more experimental small satellites, where individual operations components or even the operations personnel can be partially located outside the control center, or access to the systems should also be possible via global Internet connections, is literally the opposite.

Figure 1 shows the schematic structure of an HCC ground segment. Logically, it can be divided into two parts: the infrastructure components of the core system and the actual services that provide the space flight-specific functionalities. It should be noted that this separation is purely logical. All services, both the basic infrastructure services and the space flight services, interact with each other in the same way. All services communicate with other components via the data transport service (HCC-DT).

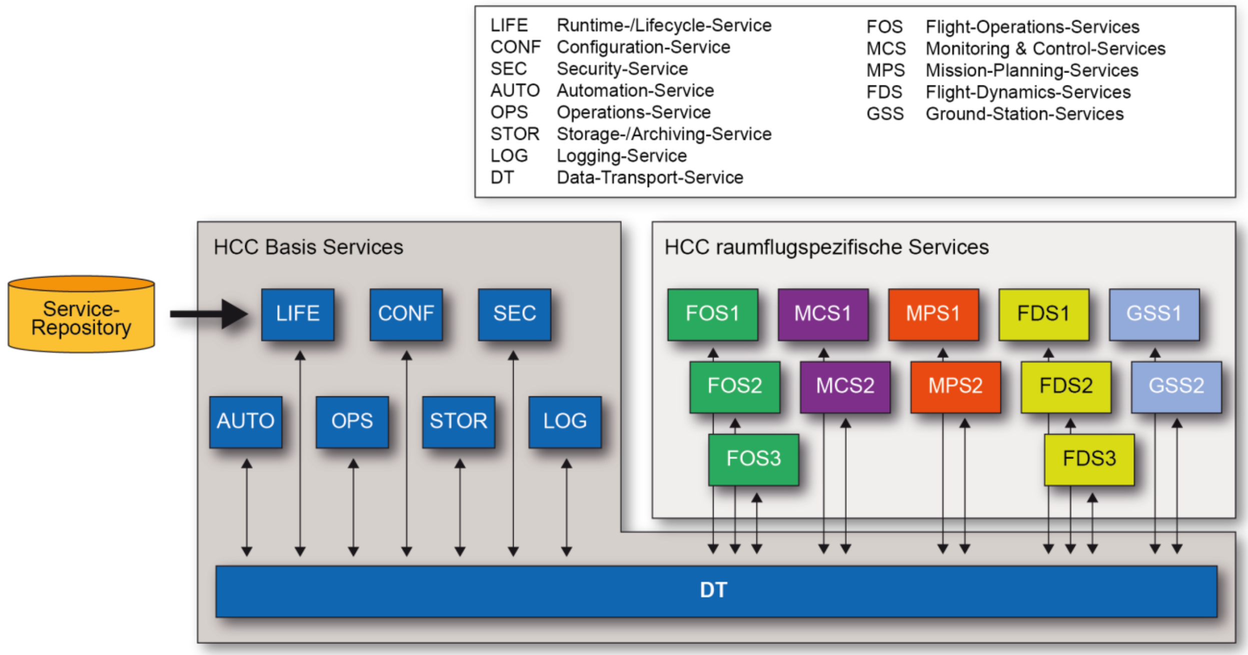
LIFE Runtime-/Lifecycle-Service
CONF Configuration-Service
SEC Security-Service
AUTO Automation-Service
OPS Operations-Service
STOR Storage-/Archiving-Service
LOG Logging-Service
DT Data-Transport-Service
FOS Flight-Operations-Services
MCS Monitoring & Control-Services
MPS Mission-Planning-Services
FDS Flight-Dynamics-Services
GSS Ground-Station-Services
HCC Basis Services
HCC raumflugspezifische Services
Service-Repository
LIFE
CONF
SEC
AUTO
OPS
STOR
LOG
FOS1
MCS1
MPS1
FDS1
GSS1
FOS2
MCS2
MPS2
FDS2
GSS2
FOS3
FDS3
DT

Fig. 1. Structure of an HCC based ground segment with its basic services to build a common infrastructure accompanied by the space flight specific services. All services are deployed from a common service repository and interact with each other through the data-transport-service.

2.1 Basic Services

As mentioned at the beginning, one of the goals of HCC is to offer an intrinsic security layer. Accordingly, there is a security service, HCC-SEC. After establishing a connection to the HCC system, all components can only communicate with HCC-SEC in order to log in. Depending on the requirements, this login process can take the form of a simple login with a password, via certificates exchanged in advance, or via an additional parallel path outside HCC as two-factor authentication.

In addition to authentication, HCC-SEC also regulates the authorization of the individual components. To this end, it uses a role concept to store the authorizations of all users and transfers these to other components as required. The extent to which these then store the information in the form of session concepts, at least over a period of time, is left to the respective components, depending on the criticality of the activities. However, HCC-SEC is also able to communicate changes in the role assignment to other applications as a notification, i.e. to actively revoke rights. An important feature of HCC is it does not differentiate between services, i.e. technical components, and human users in terms of security. Both must authenticate and authorize themselves for interactions in the same way.

The third important feature of HCC-SEC is that it manages the public keys of the individual services as a central point. Each component receives the public key from HCC-SEC when going through the login process. This can then be used so that the newly logged-in component shares its own public key with HCC-SEC. All other services can now request these public keys (depending on their authorization) via HCC-SEC and use them for communication with other services for hybrid encryption. In this way, all communication within HCC is completely end-to-end encrypted.

A second main objective of HCC is to be able to set up a complete ground segment in a decentralized manner without the individual services having to be aware of where other communication partners are physically located. The data transport service HCC-DT was developed for this feature. It acts as a single central service for the components connected to it, but is actually structured internally into individual so-called DT-nodes. This structuring makes it possible to adapt to the structure of networks and to connect different areas with each other to form a single logical unit within the framework of "distributed operation". The same structuring can also be used to reflect the internal structure of a mission. For formations or satellite fleets, for example, individual areas can exist side by side that are functionally assigned to the individual satellites, while there are components at a higher logical level that interact with the overall system as a whole.

The individual DT-nodes can be set up with multiple redundancies. However, despite all redundancies, situations can of course occur in which data packets sent through HCC-DT cannot be delivered. In such cases, the sender definitely receives feedback about this error so that it can react accordingly.

With regard to the transported data, it was a conscious decision on the part of HCC not to make any specifications. There are various standards for data descriptions in space flight operations, from purely textual variants such as orbit descriptions as TLE, to structured formats such as XML for CCSDS descriptions of ground station scheduling information, to pure binary data from space packets or TM frames [3]. All these formats have their justification and are correspondingly in use. HCC does not intend to break with these conventions. Therefore, HCC limits itself in the area of data transfer to ensuring the correct delivery of data from the sender to the recipient, but not interpreting data content in any way. This approach corresponds both with the end-to-end encryption mentioned above and with the paradigm of service orientation, according to which the service user must adhere to the service description created by the service provider.

Finally, this focus on the pure transport of data also makes it possible to keep the applications of the DT-nodes so lean that they hardly consume any CPU and are essentially limited to I/O. As a result, HCC-DT achieves a data throughput that almost matches the limitations of the network bandwidth when used sensibly in terms of packet sizes.

Another deliberate design decision for HCC-DT was to base the communication between the individual DT-nodes, and also between all HCC components and HCC-DT, on the stateful connections of the TCP/IP protocol. Ultimately, an HCC system should not resemble the volatile network of the WWW, but rather be a static ground segment with known and reliably accessible space systems. The permanent TCP connections with their handshakes, supplemented by heartbeat signals via this connection, make it possible to permanently check the presence of the connected services and to react to failures directly at system level before service requests come to nothing.

It makes sense that a distributed system, such as the one spanned by HCC-DT, requires suitable options to keep the status of the overall system consistent and to carry out analyses at system level in the event of errors. HCC also offers various services for these tasks, which are mandatory components of every HCC-based system.

The configuration service, HCC-CONF, not only contains the configuration of the overall system as such, i.e. existing services, the internal structure of HCC-DT, etc., but also offers the option of configuring the system itself. It also offers the option of centrally storing configurable descriptions for all services involved and distributing them to all relevant components. In this way, it can be ensured that all services use a consistent set of parameterizations at all times. The special requirements of space flight operations are also considered here, for example by not simply distributing a configuration change, but publishing it as an upcoming change in a step-by-step process, confirming and validating it if necessary, and only then (at a suitable moment) finally activating it synchronously for all services.

The situation is similar with the runtime/lifecycle service HCC-LIFE. It controls the individual services in an HCC system and their versioning. HCC-LIFE is able to stop running processes, start missing processes and install specific versions of services from a service repository. As with HCC-CONF, such changes are coordinated, whereby the coordination takes place both between the services involved and (if desired) with the operation itself, i.e. manual interactions are required for the activation. Since HCC-LIFE not only has to act internally within HCC, but also has to interact with the underlying operating system (especially to start processes), HCC-LIFE is also able to monitor the operating system and provide parameters such as CPU utilization etc. within the HCC system. In addition, all services also provide basic information about their status, such as their work-load or readiness to process requests. This data from both sources is collected within the monitoring service (HCC-MON), a part of HCC-LIFE, which can process and distribute it further.

In addition to this information about the current status, the logging service, HCC-LOG, can be used to analyze the development of the system in the past. All components of the HCC system have the option of sending events to be logged to HCC-LOG, where they are stored uniformly and automatically annotated with time stamp and source. In this way, errors that arise in a distributed system such as HCC from the dependencies between the services can be tracked centrally.

2.2 Space Flight-specific Application Services

The previous paragraph presented the infrastructure that HCC provides in order to build a ground segment with a service-oriented architecture. However, such a project naturally depends on how the components that perform the actual space flight operations take up and implement these possibilities. This question covers two aspects: the technical connection of a component to the overall system, but also the way in which a component fulfills its tasks, i.e. the extent to which such a component can be regarded as a service within the architecture.

The second aspect in particular becomes important when you consider that GSOC has an extensive range of operations software. Although HCC is deliberately intended to be a development towards a service-oriented architecture, HCC cannot simply discard all existing operations components and develop them from scratch. A transition is therefore necessary that makes it possible to develop the entire ground segment towards the desired structure based on the existing systems. At the same time, the systems at GSOC are undergoing continuous further development, so that the desired connection to HCC can take place in reasonable coordination with already planned developments. Just as there are a large number of operations components that may or may not fit into a service-oriented architecture in very different ways, the development paths are also very different. In the case of monitoring control systems for satellite operations, the changeover from GECCOS to EGS-CC means the integration of a completely new system. In contrast, flight dynamics systems are subject to more incremental changes. Planning systems, of which there are separate systems for mission planning and ground station planning at GSOC for historical reasons, are taking a middle course between these extremes and are gradually transferred to a joint planning tool.

In this context, it is important for a project like HCC to also allow partial implementations that can be implemented step by step - an approach that would probably be described as "agile" in the context of pure software development. HCC has already proven that this is the case, as the HCC infrastructure has been in operational use at DLR's mobile rocket base, Moraba, since October 2023 [2]. Moraba also requires a ground system that is operational at all times for continuous campaign operations, in which only individual components are improved or replaced without having to adapt the other components.

In particular, the ability of HCC-DT to transport any data format has helped here, as the data can be transferred exactly as it was before the introduction of HCC at the boundaries between the new HCC system and the old components that are still in operation.

In this sense, HCC does not force the use of its services. Applications can continue to use and manage their own configurations, write local log files or even use their own user management. Similarly, HCC cannot and will not prevent a component from maintaining further direct connections to other entities in addition to the connection to HCC via HCC-DT. These non-HCC solutions may even be justified in individual cases. On the whole, however, we are convinced that the advantages of the central HCC services outweigh the disadvantages and that developers and service managers are only too happy to access the easily available HCC services after a transition and learning phase instead of developing or maintaining their own implementations.

3. Connection to HCC

As the data transport service HCC-DT is the bracket that connects all components of a ground segment, the connection of all services to HCC-DT is essential and indispensable. In this respect, it also represents the absolute minimum that an HCC component must implement. As discussed in the previous chapter, an active connection to HCC-DT can only be established if the corresponding component successfully completes a registration process with HCC-SEC. Strictly speaking, a connection to HCC-DT alone is therefore not technically possible. However, the necessary interactions with HCC-SEC are encapsulated in the HCC API in such a way that the login process is automatically run through by the API when the connection is established, meaning that the programmer of the application does not have to explicitly create this code. This automatism justifies the following discussion, which is completely limited to the use of HCC-DT.

3.1 File Transfer

One of the most common ways in which data interfaces are implemented in applications for space flight operations is actually a file interface. One major advantage is clearly that it is easy to implement in all programming languages. In addition, there are many ways of transporting files - even across network boundaries - or accessing central file systems.

However, there are also two disadvantages: Especially when files have to be transported across different network segments, which is common in a typical setup with secured operational areas, transitions through DMZs and the like, files are written and read again at each transfer step, which is time-consuming. In addition, although files can be assigned access rights by the file system, these rights can only be synchronized across different system or even network areas with great effort. The same applies to an even greater extent to the use of content in addition to access rights. As files can in principle contain anything, in particular both data and code, there must be additional agreements on how the meta-information is transferred based on file names or similar, and how the file content is to be handled. Windows operating systems, which encode this information in the extension of the file name - and we all know how this can be manipulated - may serve as a bad example here.

Both aspects contradict the basic idea of HCC, which is to be a consistent overall system in which access rights are regulated centrally and system-wide and data is not freely available without context, but is explicitly provided by services. On the other hand, the decoupling of data transport and data usage makes it easier to replace existing transport systems without having to change the consumers, the programs and applications. Changing only the transport mechanism is therefore not a step towards a service-oriented ground system. Nevertheless, it already brings significant advantages for operations in an existing ground system, as data transfer becomes noticeably faster in many cases.

For this reason, a generic file transfer (HFT) was implemented in HCC and made available as a service, see Fig. 2. This initially replaced some file transfers for selected missions on an experimental basis. Transfers were deliberately selected that pass (in both directions) the existing network boundaries at GSOC between the operational LAN segment and the office area, as well as the DMZ in between. Such transfers were previously particularly slow due to the step-by-step copying of data from one area to the next. The tool previously used for this purpose ("Automated File Distribution", AFD) scans the source directories cyclically at intervals of several minutes. The various transfer steps result in typical transfer times of 10-15 minutes in total.

The step-by-step transfer of data is a consequence of the security requirements for the control center. Among other things, the directions in which network connections may be established are specified. As a general rule, a connection must always be established from a more secure area to a less secure area, never in the opposite direction. A file from the operational area can therefore be transferred to a DMZ, but must be picked up from the DMZ by another transfer process in the office area.

Of course, HCC must also comply with these security requirements. However, with the Data Transport Service, HCC has a functionality that was developed precisely for such cases. Although HCC-DT looks like a single, central message broker to the connected components, it still has an internal structure that is ideally suited to fulfilling the requirements described. The entire DT service consists of several instances of so-called DT-nodes, which are connected to each other and pass on the data to be transported from node to node.

Just as with the individual steps of a file transfer, the transfer within HCC-DT also takes place in individual steps from one network area to the next and the connections between the individual DT-nodes are established in accordance with the security guidelines. However, incoming data packets can be forwarded immediately via the standing connections between the DT-nodes. There is therefore no need to store and re-read the entire file on a hard disk. This reduces the duration of the entire transfer from the original source to the actual recipient from the aforementioned 10-15 minutes to just a few seconds.

In addition to proving once again that HCC is suitable for use in operational environments, this time saving has contributed significantly to the acceptance of HCC and its new technologies. However, this success also harbors a certain danger: the use of file interfaces has now become so convenient in terms of time consumption that the pressure to actually transform the underlying systems into a truly service-oriented architecture is now less great - at least for existing and established ground segments.

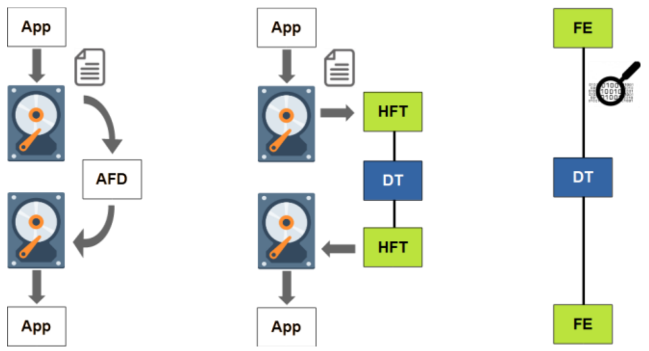

Fig. 2. Different ways for data exchange. On the left, a file exchange by a stand-alone application, AFD in this case. In the middle, replacement of the AFD by the HCC file transfer service, HFT. On the right, the desired final solution with a direct HCC based communication between the two entities. FE here denotes “functional entity”, the generic label for any service within an HCC system.

3.2 Adapter Applications

Even for applications that already exchange data with their communication partners on a network layer and do not take the detour of a file to be written, there is a very simple way to connect them to HCC and use the HCC-DT service without having to change the actual application. This can be done using an independent adapter application that is connected to HCC-DT on one side and uses the protocol of the respective application on the other. From the point of view of the HCC system, this adapter "is" the addressed service. And in the spirit of a service-oriented architecture, it does not matter to HCC how this service is provided, i.e. whether other processes do the actual work in the background, as is the case with such an adapter.

In this way, the majority of existing applications can be connected to HCC. As far as the description of the exchanged data is concerned, HCC is agnostic, as already mentioned. The encodings commonly used in the Internet world such as XML, HTML or JSON, which are often used by machine interfaces, and likewise HTTP requests are just from the point of view of HCC simple (sometimes quite long) strings, which HCC-DT can handle natively. As far as the interactions of the protocols are concerned, the commonly used protocols are compatible with HCC. In particular, the frequently used, stateless REST interfaces can be mapped in HCC without any problems.

Problems only arise with very specific protocols, such as the SLE protocol specified by CCSDS. However, such cases are extremely precisely tailored to a specific use case, so that it should be discussed very carefully how reasonable it is to "wrap" such a special protocol in HCC or to replace it with HCC.

In fact, it is an intended feature of HCC to be able to continue using such special connections "out of band". At the same time, however, it is intended to use HCC to configure such connections. To stay with the example of SLE: The standard specifies so-called "managed parameters", which the communication partners must agree on in advance. This exchange of information to ensure that both end points of the route operate their systems in a coordinated manner can and should take place via HCC.

3.3 Communication Patterns

Another quasi-standard that is often used for machine-to-machine interfaces are message brokers such as MQTT. In fact, such message brokers have also been evaluated for HCC to see whether they might represent a ready-made solution for the HCC-DT data transfer service. However, the broker itself is a very centralized component and, as already discussed in chapter 2.1, HCC requires a decentralized transport service.

Irrespective of this, ground segments are often located entirely within a single LAN segment and message brokers are used accordingly. Services that are addressed with this technology usually use the "publish/subscribe" communication pattern. Accordingly, when it comes to connecting such services to HCC, the question is often asked as to whether HCC also supports this scheme - which is basically the case at first place. However, the question that is not explicitly stated but is actually meant is almost always whether a service can simply publish a data record in HCC without having to worry about subscriptions. And here the answer is initially "no". Of course, it is possible to integrate an MQTT-style component into HCC that collects information centrally and distributes it to interested recipients. However, such an implementation contradicts the principles and ideas behind HCC in a number of respects.

First of all, the structure of HCC-DT should not only enable the connection of spatially separated areas, but also reflect the structure of a mission - especially in the case of formations or constellations. This could also be achieved by assigning a separate broker to each logical area, which would then have to be connected and synchronized with each other. And even widespread implementations of message brokers themselves point out that not all types of quality of service are completely supported in highly distributed systems.

It is also an important part of HCC to regulate access to data within a project, for which end-to-end encryption is a central component. This could also be realized with a central message broker and a little effort, but the broker is in any case a "man in the middle" for any communication, and as such a primary target for attack. Apart from that, a service that generates or manages data itself can probably judge more reliably who is allowed to receive this data and in which context, than a third party that is a hub for data products of all kinds can.

Finally, there is another point that reveals a certain paradox. On the one hand, service providers do not want to worry about the delivery of their data to all clients individually, but on the other hand, the data distributor is often required to provide guaranteed delivery. Here too, the service provider is presumably in a much better position to assess whether critical data will not reach important recipients or will not reach them on time and what measures should then be taken, rather than leaving this to a central broker.

Despite all this criticism, publish/subscribe is of course a valid communication pattern that is justifiably used in many applications. However, the arguments cited show that it should not be used centrally in a ground segment, but rather implemented directly in the services offering this pattern. This is precisely the approach chosen for the basic services at HCC. For example, the monitoring service HCC-MON mentioned above makes its collected data available according to the publish/subscribe pattern. However, this example also quickly makes it clear that it is difficult to implement this communication pattern generically for all cases. In the case of HCC-MON, it is clear from the context that the monitoring reflects the current system status. It therefore makes no sense to persistently store information from components that have been logged out of the system. In other cases, however, precisely this behavior can be useful.

Based on all these considerations, the following approach was chosen for HCC: It remains the case that the HCC API, which is used by developers to integrate the HCC protocol into their applications, provides the communication patterns "send" (without confirmation) and "request/response" (with reply) as a basis. In addition, there is a separate application that acts as a message broker and as such manages topics to which clients can subscribe. An instance of this broker can be installed "alongside" the actual service for each service that wants to offer such communication but does not want to implement it itself. These respective instances can then also be adapted in detail to the requirements if necessary: Whether, how long and how much data is buffered, how to deal with clients that are temporarily unavailable, what feedback is required to the actual service, and so on.

As with the previously discussed adapter solutions, the message broker in the HCC system can represent the entire service and the connection between it and the actual service can be outside of HCC. However, it seems to make more sense to establish both parts as components of HCC, especially in view of the fact that the HCC basic system provides more than just a pure transport layer.

4. Extension to Other HCC Basic Services

The previous chapter described how applications can be connected to an HCC system as easily as possible so that they can at least use HCC for communication with other components. However, the HCC basic services offer much more extensive functionality than just data transport.

In particular, the security layer, a unique selling point of HCC, can only be fully effective if it is also used by the connected services.

As already mentioned in chapter 3, it is not possible to use HCC-DT alone without also interacting with other basic services, in particular HCC-SEC. These absolutely necessary interactions are carried out automatically within the HCC API as required. Examples of this are the registration of the process with HCC-SEC at the start, but also the exchange of keys for encryption/decryption of messages, as well as the caching of information (e.g. role affiliation) of communication partners over shorter (configurable) periods of time. In the same way, the HCC API automatically interacts with HCC-CONF and obtains the current configuration for its service from there at start-up. In particular, the parts of the configuration that do not affect the specific service but the HCC system are kept consistent system-wide and processed immediately by the HCC API. Examples include time limits for the aforementioned heartbeat exchange, or timeouts within which an (at least provisional) response to a request is expected. The programmer of an application does not have to worry about any of these things.

However, there are also cases that cannot be processed generically within the HCC API, but really have to be handled by the actual service. Here is another example in the context of the security service: HCC ensures that a requested service knows the identity of the requester. HCC-SEC provides information on the roles to which this requester is assigned. However, the API cannot make a general assessment of whether these authorizations are sufficient for the request made; only the service itself can decide this.

This is where the disadvantage becomes apparent if the connection of existing services is implemented using the adapter solutions discussed. All relevant information (who, with which roles, wants what?) is made available to the adapter from the HCC system, but cannot be evaluated by the adapter in terms of content. This means that the adapter has to pass on all this information to the actual service application using a different protocol. The extent to which the protocol originally implemented by the service even provides for the transfer of such information is also not guaranteed.

With existing systems in particular, it often turns out that HCC offers new functionalities for which they are simply not yet designed. For example, some control systems at the ground station have no user management at all. The security and logging of access, which was of course also previously required, was adequately guaranteed by physical access control, user control of the operating systems on the computers used and the isolation of the local network against external access. As a consequence, however, physical persons are also required to go through access control and operate the system in the control room.

The transition to HCC is therefore much more than the replacement of a special implementation for data exchange. It is a paradigm shift to allow much more extensive automation of operations. And it is clear that this requires different solutions for direct communication between machines than before, involving operating personnel.

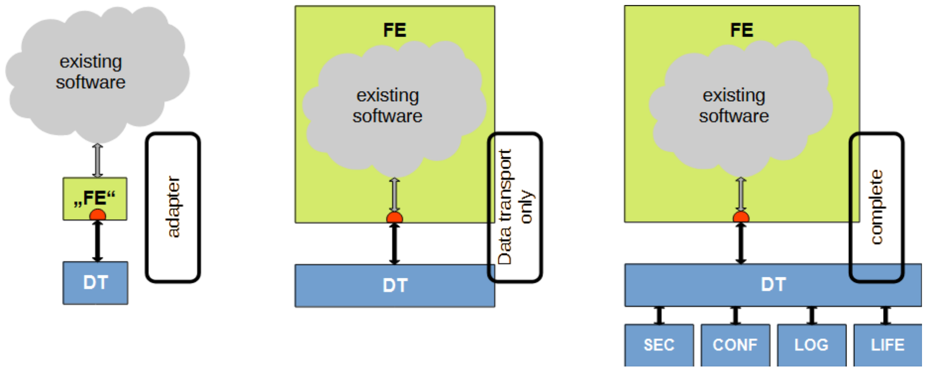

Fig. 3. Different ways to connect some service to HCC: On the left, some adapter, in the middle, utilisation of HCC-DT only, and on the right a full featured integration with interaction to all HCC basic services.

At this point, a fundamental design decision must be made for each service. Do you want to continue to adapt the existing implementation as little as possible and continue to make them do with encapsulation behind adapters (or operating personnel), or do you take the step of converting the service interface directly to the HCC format? Fig. 3 sketches the different variants.

The answer to this question depends on many factors. The current and future areas of application of the components in question play just as much a role as the availability of resources for major changes. It may also be possible to find synergies by coordinating new implementations with the original development path of the product, so that a direct HCC connection is tackled together with other features for a specific release.

From the perspective of HCC, it is important to note that this freedom of choice exists. The integration of a component into HCC does not have to take place in one fell swoop and in its entirety, but can take place step by step along other boundary conditions.

5. The HCC API and Its Use

In the discussion so far, the HCC API has often been mentioned, which makes it easy for a service manager to implement their service as part of an HCC system. This API is currently available in three variants, namely implementations in the programming languages C/C++, JAVA and python.

The API handles the establishing of the TCP/IP connection to the DT network, as well as the entire data exchange at TCP/IP level, including segmented packets, the aforementioned heartbeat exchange and similar. It also provides an interface for each HCC basic service, in which the functionality of this service is mapped in high-level functions. For example, there are login and logout methods as interaction with HCC-SEC, query and set methods for HCC-CONF or simply a routine to leave a log message with HCC-LOG. The interface to HCC-DT is also the portal that is used to communicate with all other HCC components. Here there are methods that create HCC packets according to the protocol definition, fill them with data and finally send them to an addressee.

All implementations of the API are reactive, so that various interactions can run in parallel and asynchronous incoming responses are assigned to the requests and transferred to corresponding callbacks.

As previously reported, HCC was first used by DLR's Mobile Rocket Base, Moraba, to distribute telemetry on the ground. Due to this use in a near-real-time environment, the time performance of the components was a critical condition and the components were developed in C++ using the corresponding API. The experience was so good that the HCC infrastructure has now been adopted for further developments at Moraba. This sudden increase in the number of users also led to a number of suggestions for improving the implementation of the API. In the spirit of open software development, such suggestions were and are not only taken on board but also implemented jointly by the HCC team and Moraba. The API has been considerably improved in terms of its usability, particularly through the involvement of users. Especially in a C++ environment, where developers are used to working relatively close to the operating system and hardware, it is a fine line to guarantee the user sufficient flexibility but also sufficient ease of use. This balancing takes place very fruitfully through an intensive exchange between the groups.

In this respect, the situation is simpler with a high-level language such as JAVA, which already works at a much higher level of abstraction. This variant of the API was first used by two groups.

HCC is used for the LUNA analogue built by ESA and DLR at the DLR site in Cologne [4] to control the LUNA hall itself and to configure it for the respective experiments. This includes controlling the lighting as well as the future suspension-system in the hall, which simulates the reduced gravity on the moon for astronauts. An MCS based on EGS-CC is used for the central control of this experiment environment. HCC is used as the connection between the hall hardware and the MCS. The main reason for this decision was that the modular structure of HCC with its intrinsic security layer will also allow to grant external experimenters secure access to the LUNA infrastructure or parts of it in the future. For the necessary development of HCC-compatible drivers for the hardware, only a short introduction of the responsible programmers to the concepts of HCC and the corresponding structure of the JAVA-API of about two times 2-3 hours was necessary. The programmers were then able to develop and test the necessary drivers using the API provided and a set of HCC basic services.

At the same time, the EGS-CC team from the group responsible for satellite MCS at GSOC also began to work on HCC. The first milestone in this case is to create a telemetry parameter service in EGS-CC as an HCC service. In this case too, the direct exchange between the HCC developers and the MCS experts meant that the HCC API was successfully integrated into an EGS-CC development environment in just a few hours and the implementation of the interface can now be tackled.

The third variant of the API, the version in python, has so far been used primarily by GSOC colleagues in the Flight Dynamics Services group. In fact, this implementation was also created independently by these colleagues, with the HCC team only providing the protocol description. In this respect, this version per se meets the requirements and wishes of the users and is suitable for the development environments in which it is used.

The decision to use a scripting language such as python should be understood against the background that the flight dynamics services are already available in a very modular form as individual services. For example, there are services that provide trajectory predictions on request, as well as conversions of formats or coordinate systems. The corresponding applications can be easily addressed and parameterized from a scripting language.

In addition, the calculations are often based on programs that have been hardened over many years, some of which are still programmed in variants of FORTRAN. Of course, the quality achieved should not be jeopardized by a completely new development, so that it makes much more sense to continue to use the existing routines and to simply add new scripts to call these routines for new usage scenarios.

All of these developments are taking place in the spirit of an open source community. All three variants of the API are located in a shared area in a central GitLab and can currently be used by all DLR employees.

It is the declared aim of GSOC to make the API in all variants, as well as the protocol description, open and freely available even outside DLR. There will also be freely available versions of the HCC basic services so that anyone can create HCC-compatible applications and test them in an HCC test environment. Experience with the areas of application to date shows that the entry barrier to using HCC is not particularly high.

This should explore the significance of the results of the work, not repeat them. A combined Results and Discussion section is often appropriate. Avoid extensive citations and discussion of published literature.

6. Ground Segments in Service Oriented Architecture

The previous chapters discussed how existing or newly developed components can be technically connected to an HCC system. The focus here was on data exchange with HCC-DT, or using HCC-DT with other services, as well as on the integration of the other HCC basic services. In addition to the purely technical connection, supported by the provision of the HCC API, a component should ideally also fit structurally into the service-oriented architecture of HCC.

At this point, it should be emphasized that HCC services are not necessarily to be understood as micro services. The granularity of the HCC services is rather characterized by the tasks in space flight operations. An individual HCC service itself can therefore be structured in a micro service architecture. This is the case with planning systems, for example, where the creation and optimization of a sequence can be a very extensive process. This can also be realized in the form of containers that scale dynamically during a planning run, controlled in a Kybernetes cluster, for example. Nevertheless, the creation of a consolidated timeline would then be a single HCC service.

The main conclusions of the study may be presented in a short Conclusions section, which may stand alone or form a subsection of a Discussion or Results and Discussion section.

6.1 Service Oriented Applications

In this sense, and consistent with the usual definition of service-oriented architectures, HCC services in a ground segment are characterized by the fact that they provide self-sufficient, clearly defined services that can be called via a defined interface. Existing applications that follow this concept should therefore be able to be converted into an HCC service simply by replacing the interface. However, this view is rather theoretical; in reality, such a consistent encapsulation usually cannot be achieved.

An example of this is a component that communicates with a redundant data exchange platform. In the previous system architecture with dedicated peer-to-peer connections between the individual applications, this component therefore maintains a TCP/IP connection to the data exchange system. In the event of a redundancy switch in this system - whether for a planned maintenance or in the event of a unforeseen failure - this TCP connection will be terminated. This in turn will be registered and lead to a new connection being established, this time to the redundant system.

In this special case, however, a number of activities are necessary within the application following such a switchover in order to ensure synchronization between all processes. These activities are controlled from the point where the redundancy switching is handled, i.e. in the program section that handles the TCP socket.

What does the situation look like if these components are to be integrated into an HCC system? The TCP/IP connection now exists exclusively between the application and the HCC-DT data transport service. Redundancies may also be switched here, but this only relates to the multiple instantiation of the DT-nodes and are handled automatically by the HCC API. Information that the logical communication partner, the data exchange platform, may have had to perform a switchover is of course also transmitted within an HCC system, but now reaches the application as an HCC packet, just like the rest of the data exchange. The necessary synchronizations must therefore now be triggered from the part of the program that carries out the entire data handling via the HCC API. Compared to where the TCP socket was originally implemented, this is a completely different part of the program, perhaps even a different thread - in any case a completely different context. The extent to which the triggering of the necessary processes can simply be moved there depends on many very subtle details of how the application is implemented in terms of software technology.

6.2 Operation of a Service Oriented Architecture

Finally, another problem that should not be underestimated is the user of the entire system, i.e. the operating personnel. The transition to a service-oriented architecture is also an immense step that the people involved have to take. The necessary changes run in two different directions which, paradoxically, are actually diametrically opposed.

On the one hand, as already mentioned, services should be self-sufficient and offer their services even without knowing much about the entity that uses these services - and especially for what purpose. And vice-versa, the user of a service does not need to know how the service is actually provided. This encapsulation is not always anchored in the minds of operating personnel. In discussions parallel to the introduction of HCC, the argument is often made that the supervisor of a service must have direct access to how another service is provided. Interestingly, other components should always be controlled as well, while access to one's own areas is usually denied. Typical examples are being able to actively select the prime and backup of a redundancy on a third-party system to ensure that both are regularly tested. Or fears of the kind "and what do I do if their service fails?".

It speaks for the commitment of the staff to anticipate and rule out possible errors. But it goes against the spirit of a service-oriented architecture, in which responsibilities are transferred to the service owners.

This attitude becomes particularly noticeable as soon as the entire system can no longer be fully controlled by any of the operators in distributed operations. In addition to responsibilities, liability issues must also be clearly regulated here.

Unfortunately, this extension of perceived responsibility does not lead to the entire operating staff developing comprehensive expertise for the entire ground segment. This is probably not even possible in individual cases. However, in a service-oriented architecture, the role of an overall system manager is all the more important and demanding. While the entire system can otherwise be covered relatively well by the sum of overlapping expertise, in a service-oriented architecture someone is absolutely necessary to ensure that the construct, which is made up of ready-to-use building blocks, the services, ultimately fulfills its purpose correctly. A clear distinction must be made here between in-depth knowledge in individual domains (for those responsible for the service) and comprehensive, broad but significantly less in-depth expertise (for those responsible for the system), see Fig. 4.

The paradox mentioned at the beginning is therefore that service-oriented architecture requires someone with very broad knowledge on the one hand, while it tends to restrict the individual service experts to their respective domains.

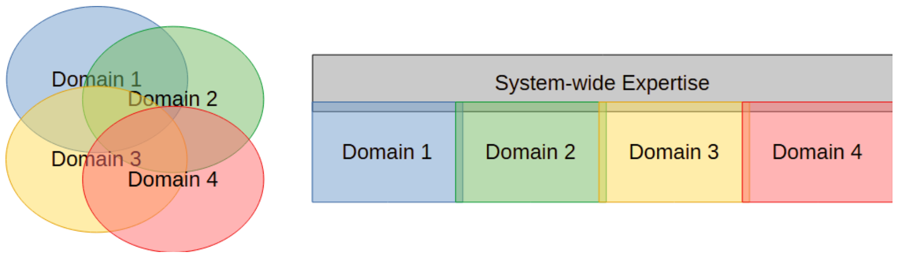

Fig. 4. Sketch of the knowledge of operations personnel of a legacy-type system (left) and a service-oriented architecture (right). See text for more details.

Of course, none of the points discussed is an insurmountable obstacle. On the contrary, the switch to service-oriented architectures is being made for good reason with regard to missions that are already making new demands now or will do so in the near future. The changes discussed are therefore not the result of an ideological decision, but are well justified by the development of the boundary conditions in space flight.

In addition to this strategic argument, however, HCC provides another aspect that should appeal to sceptical operating personnel in particular. Space flight operations are, and will continue to be, the interaction of many. Accordingly, several positions work together, such as mission control, network control and the ground stations. Until now, it has been technically difficult to display status information from all subsystems in each other. HCC's approach of viewing the ground system as a whole now enables the individual positions to receive information from all parts of the system (depending on access-rights) thanks to central services such as HCC-MON. As mentioned above, this is definitely a wish of the operating personnel. It is a nice touch that the introduction of a service-oriented architecture, which actually narrows the focus to individual services, only enables system-wide services such as HCC-MON.

7. Conclusions

Like other agencies and private space flight operators, GSOC is aiming to offer its expertise in services in future and build up ground segments in a service-oriented architecture. The necessary infrastructure components were developed with the HCC program and are constantly being extended in line with requirements. HCC has proven its suitability with the operational use at Moraba, and so HCC is set at GSOC for future missions and developments such as the LUNA analogue.

Three things are essential for transforming space flight operations into a service-oriented architecture: the technical connection of the operational components to HCC, the structuring of the tasks to be performed into services, and the adaptation of operations to the new structure.

All three points can pose major challenges, and there is a wide range of the extent to which existing components and processes already work according to the principles of service-oriented architectures. It can be observed that technical developments have been moving in this direction of their own accord for some time. An initiative such as HCC now offers a necessary common basis for all components to fully exploit the advantages of this architecture. It is precisely because of this wide range of requirements that it is important for HCC to make a step-wise transition possible for the components to be connected. As has been shown, this has been excellently achieved with HCC.

When transferring operations, it is also necessary to communicate the major goals of the development well and repeatedly, but then proceed in small steps. Operating personnel in particular have a strong tendency to stick to familiar systems that do not change. However, with the gradual integration of technical components, a step-by-step approach is almost a given in operations too. For example, it makes sense to offer high-level monitoring via HCC-MON with every component that fully uses HCC in addition to HCC-DT. This extends the view of the system for each operator step by step to the entire system, without making them immediately responsible for the entire system.

Acknowledgements

With the decision to make HCC the basis for all projects at GSOC, almost all colleagues at GSOC came in touch with HCC in one way or the other. We would like to thank all of them for their feedback and contributions to HCC as a holistic system. Special thanks to our pioneering users from the Moraba, especially Martin Stoffers, our outpost at RB-MRB, who lead his team on a save path to use HCC, and helped the HCC team a lot, to establish user friendly software releases. Special thanks also to the entire group “Flight Dynamic Services” for being eager to open-mindedly try out the HCC features, especially Martin Weigel, author of the python version of the HCC-API.

Last not least, the author would like to express his greatest gratitude to Udo Häring and Sven Kuhlmann. There would simply be no HCC at all without them.

References

[1] A. Hauke and M.P. Geyer, Towards a Modular and Flexible New Ground System, 15th International Conference on Space Operations, Marseille, France, 2018, 28 May – 01 June.

S. Gärtner, M.P. Geyer, S. Hackel, A. Hauke, C. O'Meara, and Y. Wasser, Rethinking Ground Systems: Supporting New Mission Types through Modularity and Standardization, 15th International Conference on Space Operations, Marseille, France, 2018, 28 May – 01 June.

[2] A. Hauke, GSOC's Service-Oriented Ground System "HCC" – Status and First Experiences from Sounding Rocket Missions, 17th International Conference on Space Operations, Dubai, UAE, 2023, 06 – 10 March.

[3] Kim Dismuke, Definition of Two-line Element Set Coordinate System, 29. September 1999, http://spaceflight.nasa.gov/realdata/sightings/SSapplications/Post/JavaSSOP/SSOP_Help/tle_def.html, (accessed 07.04.2025).

The Consultative Committee for Space Data Systems, CCSDS-132.0-B-3, TM Space Data Link Protocol, 2021.

The Consultative Committee for Space Data Systems, CCSDS-133.0-B-2, Space Packet Protocol, 2020.

The Consultative Committee for Space Data Systems, CCSDS 902.1-B-1, Cross Support Service Management – Simple Schedule, 2018

[4] A. Casini, et al., Lunar missions' simulations in analogue facilities: the operational concept and the first commissioning of the ESA-DLR LUNA facility, 73rd International Astronautical Congress, Paris, France, 2022, 18 – 22 September.