Implementation of CCSDS Mission Operations Services at the German Space Operations Center

Stefan A. Gärtner[1], Jens H. Hartung[2], and Michael Wendler[1]
German Space Operations Center GSOC, DLR Oberpfaffenhofen, 82234 Weßling, Germany

The CCSDS Mission Operations framework defined by the CCSDS Spacecraft Monitor & Control working group aims to provide a reusable, interoperable and distributed mission operations system based on a service-oriented architecture paradigm. Services are defined in terms of an abstract model called the "Message Abstraction Layer", which is cast to a concrete technology by separately defined bindings. This work is concerned with prototype implementation of binding to the Space Packet Protocol. Challenges and solutions are outlined as well as next steps in terms of testing to prove interoperability. A second prototype implementing a core set of services – the Monitor and Control services – is discussed briefly, as well as a possible way of transition towards a Mission Operations framework based infrastructure at the German Space Operations Center GSOC. The work is put into context by pointing out how benefits for missions arise from the framework and its design.

Nomenclature

AMQP	=	Advanced Message Queuing Protocol
API	=	Application Programming Interface
APID	=	Application Process Identifier
CCSDS	=	Consultative Committee of Space Data Systems
CNES	=	Centre National d'Etudes Spatiales
COM	=	Common Object Model
DLR	=	Germany Aerospace Center / Deutsches Zentrum für Luft- und Raumfahrt
ECSS	=	European Cooperation for Space Standardization
ESA	=	European Space Agency
GSOC	=	German Space Operations Center
ICD	=	Interface Control Document
M&C	=	Monitor and Control
MAL	=	Message Abstraction Layer
MO	=	Mission Operations
PUS	=	Packet Utilization Standard
SLE	=	Space Link Extension
SM&C	=	Spacecraft Monitor and Control
SOA	=	Service-oriented Architecture
SPP	=	Space Packet Protocol
TCP	=	Transmission Control Protocol
URI	=	Uniform Resource Identifier
XML	=	Extensible Markup Language

I. Present Day Situation

ONE of the major concerns when operating a space mission is the definition of interfaces between the parties involved – be it the ground-to-space interfaces involving control center(s) and spacecraft(s), ground-to-ground between different space agencies or ground-to-ground within one agency. Usually these interfaces are mission- and/or agency-specific. While standards exist such as CCSDS Space Link Extension (SLE), CCSDS Packet Telecommand and Packet Telemetry to name just a few, their scope is focused on the transport or lower

[1] Mission Control and Data Systems Engineer, Mission Operations, DLR German Aerospace Center, Münchner Straße 20, 82234 Weßling, Germany.
[2] Mission Planning Systems Engineer, Mission Operations, DLR German Aerospace Center, Münchner Straße 20, 82234 Weßling, Germany.

layers. Protocols targeting the application layer tend to have either their own special field of application (e.g. CCSDS File Delivery Protocol) or have strong coupling to specific underlying technologies (e.g. ECSS Packet Utilization Standard) and thus are not suitable for a more general description of interfaces between possibly very different systems using very different underlying technologies.

A. Ground-to-Ground Interfaces

Detailed Interface Control Documents (ICDs) are necessary to describe the interfaces of different systems or subsystems interacting with each other. They describe external interfaces between control centers, spacecraft manufacturers, customers and agencies. It is by nature that setting up these documents is a tedious process with limited reuse for other missions. In principle the same process applies to internal interfaces inside a single agency or a single control center: Every involved group has to agree on a set of interface parameters. Depending on the history of missions and their demands interfaces grow with time, generating heritage that is not always easy to adapt to new missions. ICDs tend to get outdated the longer missions are operated. This amounts to increased cost when changing external partners or internal structures.

B. Ground-to-Space Interfaces

Interface definitions between ground and space segments are crucial for spacecraft commanding and reception of telemetry. In contrast to ground-to-ground interfaces it is usually not easily possible to change both partners' interface parameters when new needs arise as this implies changes to the space segment that are not possible at all in many cases. Naturally, the ground-segment for a mission is tailored to fit the space segment. This means that interoperability of different ground-segment providers is limited if not explicitly planned beforehand or expensive to achieve. Furthermore, a ground-segment providing support for several missions needs to maintain possibly very different interfaces for all these missions. Synergy effects between missions are possible and exploited, but by nature of these deployments they do not arise in a systematic fashion.

II. The CCSDS Mission Operations Framework

One possible solution addressing the shortcomings of present day mission operations system deployments is the Mission Operations (MO) framework[1,2] defined by the Spacecraft Monitor and Control (SM&C) working group of the Consultative Committee for Space Data Systems (CCSDS), an international standardization organization in the realm of space data systems. The MO framework provides a common and standardized way

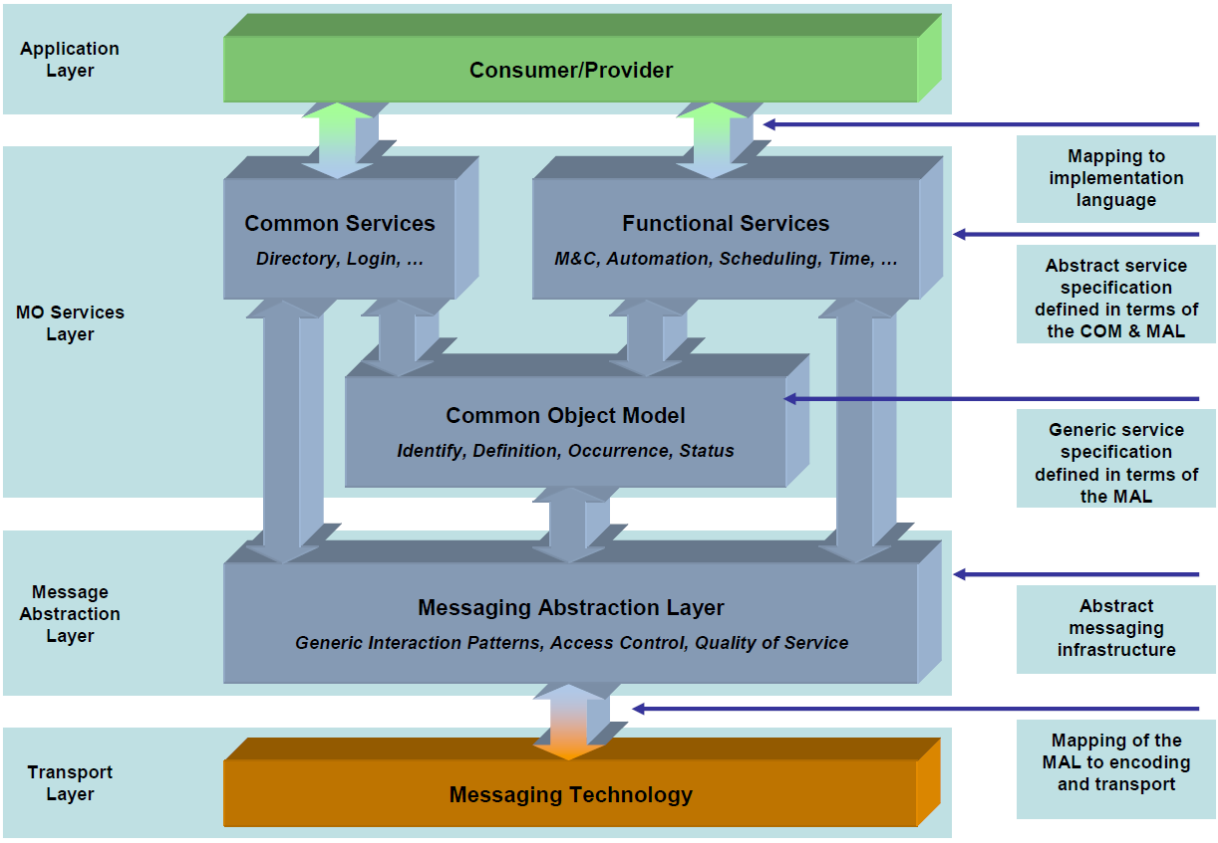

Figure 1. General structure of the CCSDS Mission Operations framework (from Ref. 1).

of performing high-level mission operations functions on distributed systems. It uses a service-oriented architecture (SOA) with different services representing the mission operations objectives and a message-based communications layer for allowing the services to exchange data and invoke operations. Per se, the framework is not tied to a specific technology, but is expressed in abstract terms. By standardizing both the services and the service interactions mission operations functions may be shared across missions and agencies. The aim of the MO framework is to provide an interoperable, reusable and thus cost-effective mission operations system. More information on MO framework benefits can be found in M. Merri's paper[3]. Providing only "least-common denominator services" targeting possibly very different missions is avoided by definition of "capability sets". More capable missions might choose to implement more demanding capability sets. Missions not needing all functionality or restricted by a limited environment (e.g. processing power) can choose to implement a simpler capability set. Thus tailoring of the MO services is not needed, leading to interoperable implementations. Because not only services are standardized but also service interactions and the way how services are defined, missions are still free to define own services or extend existing services, if their demand is not met by standardized services.

A. Structure of the Mission Operations Framework

The general structure of the MO framework can be seen in Fig. 1. The framework is made up of four layers, with the application layer at the very top. The application layer provides implementations of mission operations services that are defined in the services layer. This is the layer where users interact with the framework and where the framework interacts with underlying layers. The services layer defines several services in an abstract way using data types and message exchange patterns provided by the next lower layer, the Message Abstraction Layer[4] (MAL). Finally, these basic building blocks offered by the MAL are mapped to a concrete on-the-wire representation by the transport layer. The transport layer connects to remote MO framework stacks adhering to the same layered structure. The application layer of a remote stack can then consume the services provided by the local stack and vice versa.

1. The Services Layer

Services contained in the service layer are either standardized services defined by the CCSDS SM&C working group or user-defined services for a particular mission or purpose. A number of commonly needed services has been identified by the working group and is subject to standardization. Adhering to these standardized services is one of the keys to interoperability. The user is still free to define own services for specific needs. Service specifications themselves do not depend on any particular programming language binding. They are formulated in an abstract, technology-independent way instead. For this purpose the MAL provides basic building blocks that allow construction of services. Specifications may be expressed as XML (extensible markup language) documents adhering to an XML Schema enforcing the MAL building blocks and composition rules. In fact all CCSDS defined services are available as XML documents.

The services layer consists of three characteristic service types: Functional services, common services and the Common Object Model[5] (COM). Functional services provide the high-level mission operations services the user expects from the framework, while common services are concerned with more basic and administration tasks such as providing a service directory or proper authentication and authorization. Services may use the COM as their data model, which builds upon the MAL and introduces the notion of objects, which possess certain characteristics. Using this model gives services the ability to archive and retrieve objects, generate and react to certain events or track progress of activities without the need to specify anything of that themselves.

2. The Message Abstraction Layer (MAL)

The MAL provides a common abstract language that is used by services definitions. It provides primitive data types such as integers and strings without referring to any particular implementation or data representation. The MAL imposes rules on how to use these primitive types to define more complex types such as compositions or lists and already provides a number of useful complex types. The MAL not only defines rules for data type definitions, it also defines a message format and message exchange rules, allowing services to define an interface suited to their objectives. The message exchange rules are called "interaction patterns", where there are six types of: SEND, SUBMIT, REQUEST, INVOKE, PROGRESS, PUBLISH-SUBSCRIBE. A service defines the operations it offers using these patterns and the data it exchanges using the data type building blocks. An XML Schema is available to validate a service description in XML against.

3. The Transport Layer

Because the MAL describes messages and data types in an abstract way without reference to any particular representation, it is not directly usable on the wire. Therefore the transport layer is needed, casting the building blocks of the MAL into a concrete technology representation. Every transport technology to be used with the MAL needs to be implemented and is called a "binding" of this technology to the MAL. Usually bindings are

composed of two parts: the actual data encoding and the transport of the encoded data, though this is not enforced by the framework. A second type of binding should not be confused with this transport layer binding: In order to use the abstract interfaces and data types defined by the MAL a mapping to some concrete programming language is required. This language binding is independent of the transport layer binding.

III. This Work in Context of the MO Framework

The work presented here is split into two parts, each addressing a different layer in the MO framework. The first part is concerned with a technology binding casting the MAL concepts into a concrete representation of Space Packets. The second part is concerned with implementation of the Monitor & Control (M&C) services on top of the framework.

IV. MAL/SPP Binding

As a first technology binding to be standardized the CCSDS SM&C working group has selected the binding of the MAL to the Space Packet Protocol (SPP), or "MAL/SPP binding" in short.

A. The Problem

Space Packets are widely used by many satellite missions. The Space Packet Protocol[6] describes a mechanism of transferring packetized data over space links, i.e. from ground to space or from space to space. Space Packets are transported across several subnetworks from the control center via the ground station to the satellite and vice versa. Transport on each subnetwork is not handled by the SPP but is relied on. The SPP defines a structure consisting of a packet body containing the payload data and a packet header containing routing and control information. Usage of the payload data is not part of the Space Packet Protocol and as such it is up to the mission to define the meaning of the payload data. This leads to individual solutions for different satellite platforms, rendering reuse of existing software on board and especially on ground difficult and costly. The ECSS Packet Utilization Standard[7] (PUS) tries to mitigate this by defining a set of services common to many satellite operations needs and an unambiguous mapping to Space Packets. However, it is subject to tailoring, possibly leading to non-interoperable implementations. The set of services is rather fixed and strongly bound to the usage of Space Packets as service data units. Furthermore, the effort is purely European.

B. The Solution

The MO framework is independent of any concrete technology representation, which means it can be used anywhere where a binding to a concrete technology exists, thus enhancing interoperability. While PUS per se is restricted to the space link using SPP, the MO framework can be used on ground, in space and on space links without the need to bridge concepts as these are the same for all domains. The only need is to bridge technologies, which is made possible by the abstract MAL formalism. As such a binding of the MAL to the SPP is needed to expose the service interface to space links. The peculiarities of space links are taken into account by the MAL/SPP binding[8]. A size-efficient binary format for data encoding is employed optimizing data throughput across limited bandwidth space links. At the same time computational complexity is kept low in order to be used in the limited environment of a spacecraft. Space Packets produced by the MAL/SPP binding may have fixed length primary and secondary headers allowing efficient introspection. Packets may be routed on board a spacecraft and exchanged between services, decoding the body only on demand.

C. Obstacles encountered during Prototyping

While mapping of MAL messages to a representation in Space Packets sounds straight-forward at first glance, some major difficulties needed to be solved in order to fulfill this goal:

1. Interpretation of the Space Packet Protocol

The interpretation of the Space Packet Protocol itself posed quite an obstacle as the standard not only defines a packet structure but also hints at a concrete high-level infrastructure in which Space Packets are to be deployed. Stemming from the consolidation of the older Packet Telemetry and Packet Telecommand standards the asymmetry between telemetry and telecommands is still present in SPP to a certain extent. The notion of a space segment containing several numbered applications and a ground segment commanding these applications or receiving telemetry from them is replaced by the notion of numbered paths connecting these applications in space with applications on ground. One of the goals of the MAL/SPP binding is not to put any restrictions on the mission deployment if not absolutely necessary as this would impair acceptance of the binding. This means missions should be free to choose the kind and number of applications they wish to deploy, if at least one of them allows access to the MO framework on board and on ground. A scheme uniquely addressing service implementations needs to be compatible with every conceivable deployment. The MAL uses endpoint identification of service implementations by uniform resource identifiers (URIs). This addressing scheme is not

compatible with the path identification scheme offered by the SPP. This scheme neglects the possibility of having conceptually same applications on both sides of the path. Instead it promotes the idea of having a telecommand/telemetry source and a telecommand/telemetry sink on either end. This kind of interpretation is further enforced by allowing only unidirectional data flow along one path. All interaction patterns of the MO framework except one (SEND) rely on bidirectional communication instead. This problem of SPP and MAL concept incompatibility can be overcome by two ways: One can put in major configuration efforts reflecting the actual deployment of the systems and making the MAL/SPP binding aware of all connection paths between all service instances. This puts a high burden on the user and adds a possible source of errors. Furthermore, this configuration step needs to be carried out before any communication across the space link takes place. It results in major efforts if new needs require change of configuration. The flexible nature of the MO framework is put in a static "corset" of configuration eliminating one of the advantages of the framework. The second way for overcoming the incompatibility is not to use the path concept at all but just take the packet structure defined in the SPP and get rid of the conceptual burden only introduced by Packet Telemetry and Packet Telecommand consolidation. This means applications are numbered instead of paths, reconciling the endpoint identification of the MAL with SPP. Configuration effort is kept low and the flexibility of the service-oriented approach is maintained.

2. Message Routing

Interpretation of the Space Packet Protocol is directly related to the realization of message routing. While the SPP transparently routes messages through underlying subnetworks, routing capabilities of the protocol itself are very limited. Given the above interpretation of using numbered applications instead of numbered paths, one application (on ground or in space) is identified by an 11 bit Application Identifier (APID). Actually, the *unique* application identification is only accomplished by adding the so-called APID Qualifier; however this identifier is usually carried by the subnetworks and defines a unique naming domain, usually on the scope of a spacecraft. The APID address space is limited to $2^{11} - 1$ addresses (2047 is reserved for so-called idle packets), and is therefore a scarce resource. Depending on the deployment there might only be one application having an APID assigned running an instance of the MO framework which in turn needs to provide several services. Therefore service instances provided by one instance of the MO framework get assigned a so-called "instance number". Thus, the combination of APID Qualifier, APID, and instance identifier uniquely identifies a service instance and makes it accessible to consumers. These three parts form the URI of the MAL/SPP Binding in the following way:

```
malspp:<APID Qualifier>/<APID>/<instance identifier>
```

As long as two MO framework instances are connected by SPP their consumers and providers can exchange messages using this URI scheme. The SPP properly routes the packets through the subnetworks using APID and APID Qualifier, the MO framework routes the packets to the proper service using the instance identifier. In case technologies have to be bridged, a proxy or relay needs to act on behalf of the service. If for example a consumer application is connected to the control center infrastructure by AMQP[9] (advanced message queuing protocol) and needs to invoke a service operation on a spacecraft connected to the control center via SPP, it has to do so by a proxy or relay bridging the two technologies, replacing the source and destination URIs according to the used protocol.

3. Oversized Messages

The maximum size of the Space Packet Data Field is 65536 bytes, possibly less depending on the size of the secondary header. Because MAL messages can contain fields of variable length, encoded message sizes larger than this maximum might be encountered. Furthermore, a mission-dependent maximum size of Space Packets might be effective, rendering this limitation even stricter. In those cases the MAL/SPP binding uses the sequence control fields offered by the SPP to segment oversized packages. Reassembling the segmented packets on the receiving end is a non-trivial task due to the nature of the SPP: packages can be transmitted out of order, they can be lost, or they can arrive multiple times. The SPP offers a counter field, however this is not granular down to the service instance level. At the time of writing, the detailed specification of oversized message handling is subject to discussion.

4. Efficient Data Encoding

Another field of concern is the efficient encoding of data. Header fields are ordered in a way that variable fields come last, right before the encoded message body data. This enables routing messages to the correct service instances, examining interaction pattern and state information, all without decoding the entire packet and with header fields at fixed positions in the byte stream as long as non-variable header fields are concerned. By declaring some of the fields optional by configuration, size can be kept to a minimum, allowing the packets to be sent over slow space links. The encoding of payload data is also done in an efficient way: Type information

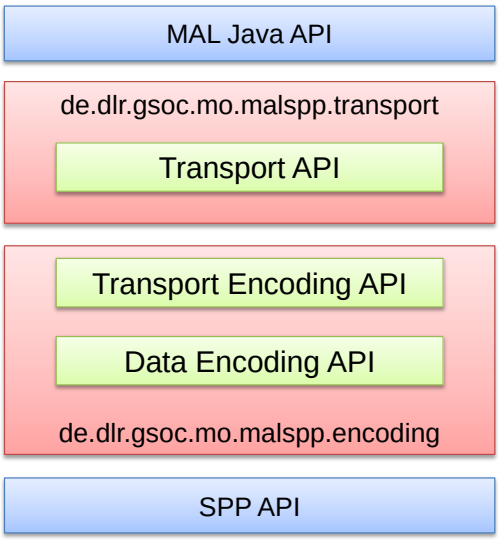

Figure 2. Structure of the DLR MAL/SPP binding implementation and how it fits into the MO framework.

is only encoded where absolutely necessary, i.e. when polymorphism is employed. Otherwise, type information is directly taken from the service specification. Updates contained in a PUBLISH message of the PUBLISH-SUBSCRIBE interaction pattern need not to be decoded by a broker that is just relaying these updates to subscribers. Integers are encoded with variable size, occupying less space for smaller numbers than for large ones. Encoding of time fields is controlled by configuration and can use any (sensible) binary CCSDS time code format[10]. Thus message sizes of less than 20 bytes can be achieved.

D. Benefits for Missions

Many control centers have existing infrastructure for generation, processing, and transport of Space Packets. A standardized binding of the MAL to Space Packets as a protocol crossing the space link allows reusing this infrastructure. Adapting the infrastructure to the MO framework technology comes with manageable costs. Mission-specific solutions of mapping MAL messages to Space Packets are avoided, thus allowing easier cross-support, reuse of components for several missions and less personnel training. Problems arising during technology mapping have already been solved by the standardizing process, ensuring a consistent and usable mapping. The hard work has been done by the SM&C working group such that missions can focus on their objectives.

E. Prototype Implementation

The purpose of a prototype implementation of the upcoming MAL/SPP binding standard is to evaluate consistency, interoperability and unambiguousness of the standard. During this work many comments have been raised, some of the most severe ones being presented in section C. The prototype implementation presented in this work has been performed using the MAL Java API[11] and its implementation by ESA. The MAL Java API consists of a set of several APIs representing the layer structure of the MAL presented in Fig. 1. It also details the interface to the transport layer. The prototype implementation has been developed with this structure in mind. The MAL Java API allows a monolithic transport layer, but encourages making use of the optional

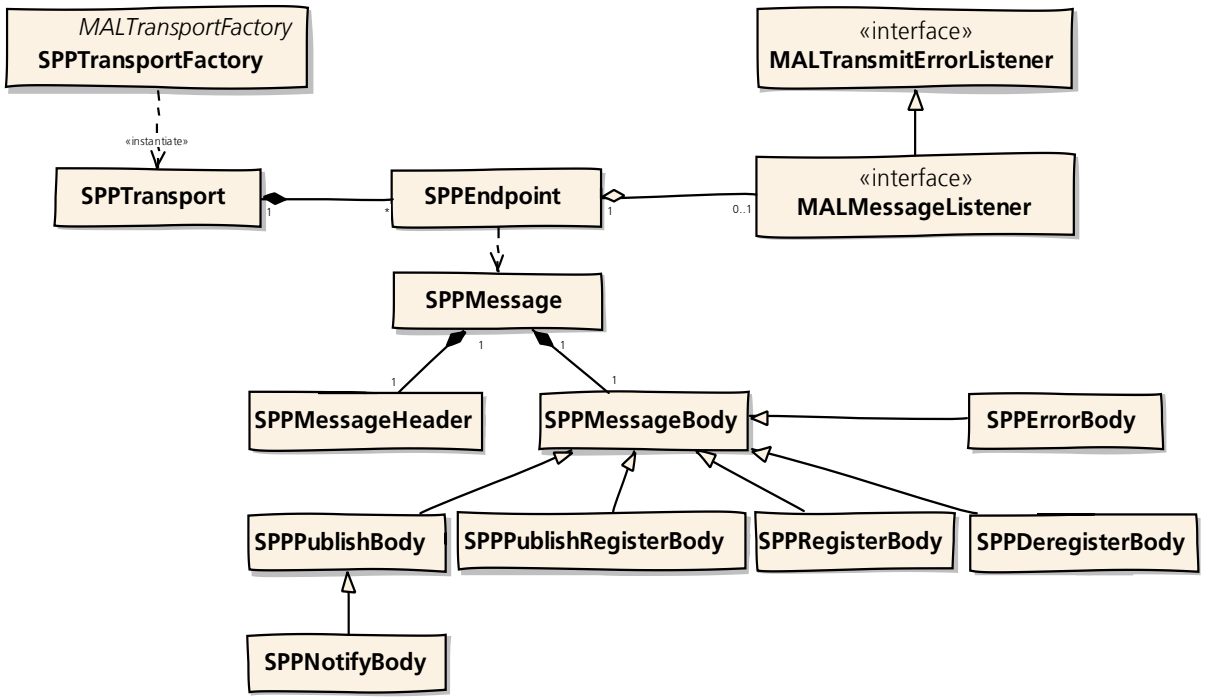

Figure 3. UML class diagram showing the transport module of the MAL/SPP binding prototype implementation.

Encoding API, thus separating data encoding from actual transport. This prototype makes use of this separation, making it possible to swap either part or reuse them in different places.

The general structure of the binding implementation and how it fits into the MO framework can be seen in Figure 2. On top is the MAL Java API with all interfaces needed to connect the MAL building blocks to the transport layer. All other boxes except the bottom one make up the transport layer. The bottom box represents a Java API to the Space Packet Protocol. The SPP implementation is not part of the prototyping work, and a simple implementation by CNES using TCP sockets is used. Separation of the transport layer in encoding and transport is depicted by the two big center boxes named with their Java package names. The encoding module again is conceptually split in two: The data encoding part is only concerned with encoding of basic MAL data types, while the transport encoding part is concerned with encoding the whole message body, making use of the data encoding part. The enclosed boxes depict which of the many APIs provided by the MAL Java API are actually used. So to summarize: The data encoding part is responsible for encoding MAL data types, the transport encoding part is responsible for composing all data into the encoded body of a message. Together they form the encoding module. The transport module is responsible for making the connection to the rest of the MAL and the underlying transport.

The class diagram for the transport module can be seen in Fig. 3, which is an implementation of the respective API defined in the MAL and as such looks very similar to its structure. Interfaces `MALMessageListener` and `MALTransmitErrorListener` are not part of this module, but are provided by the MAL as interfaces in order to react to received messages. The other classes implement or extend their counterparts in the MAL (package `org.ccsds.moims.mo.mal.transport`) with the same name, but prefixed with `MAL` instead of `SPP`. These are the interfaces used by the MAL to bind to a certain technology.

The transport encoding class diagram is presented in Figure 4. The interfaces and classes prefixed with `MAL` belong to the package `org.ccsds.moims.mo.mal.encoding` of the MAL Java API. Here element-wise encoding and decoding of whole message bodies takes place.

In Figure 5 the data encoding class diagram can be seen. The interfaces prefixed with `MAL` are contained in `org.ccsds.moims.mo.mal`. Encoding and decoding of single elements is handled by these classes.

A few helper classes have been defined, such as `CCSDSTime`, handling time code representation in CCSDS

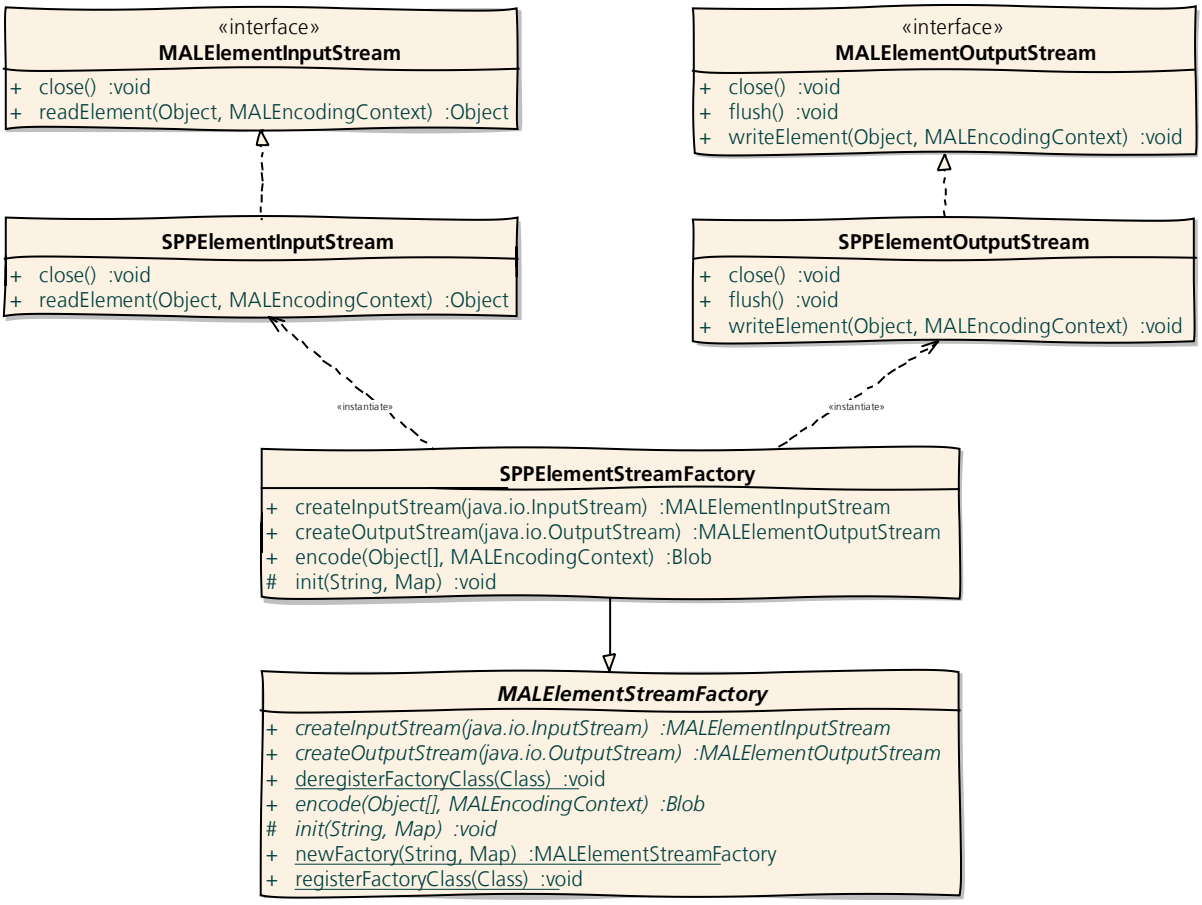

Figure 4. UML class diagram showing the transport encoding of the MAL/SPP binding prototype implementation.

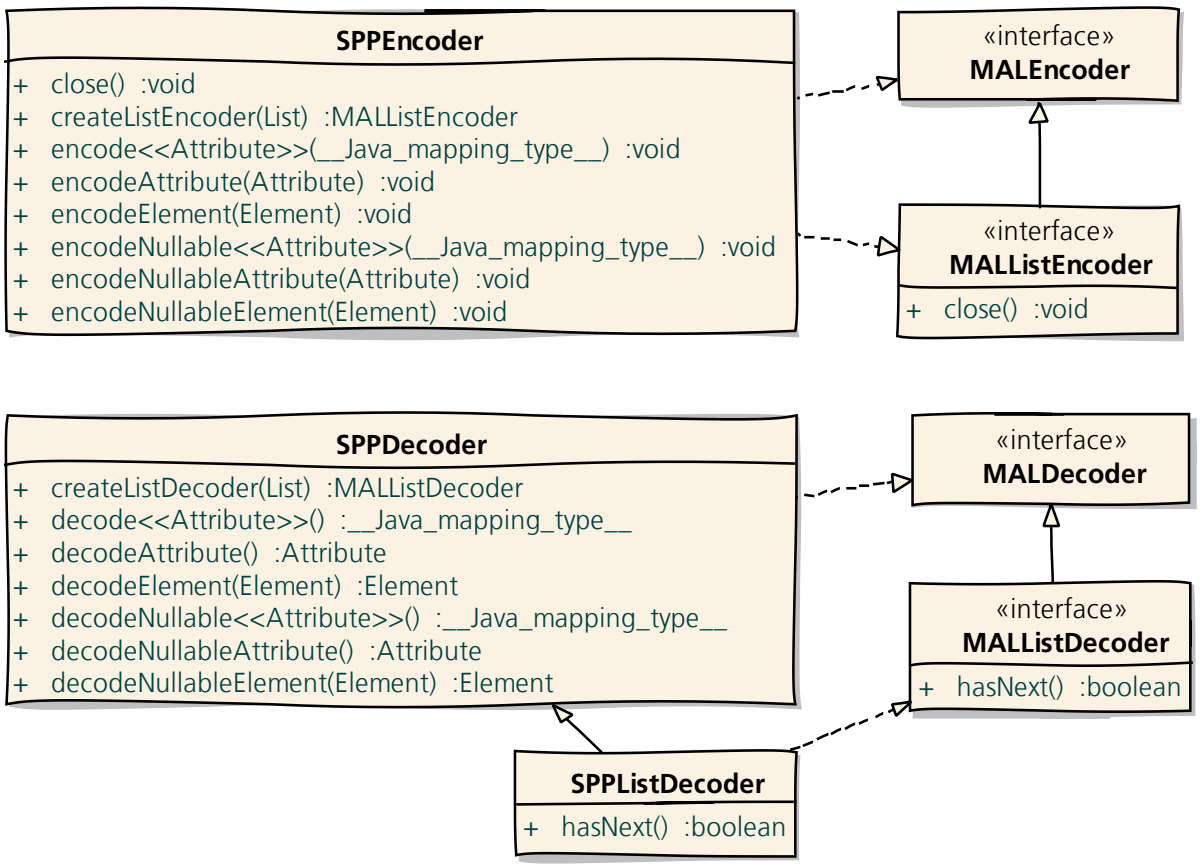

Figure 5. UML class diagram showing the data encoding of the MAL/SPP binding prototype implementation.

formats, or `SPPSegmenter`, handling the sequencing and reassembly of oversized messages.

As a useful side effect of the prototyping work, we used the MAL Java API specification and its implementation by ESA without previous knowledge and thus were able to provide a hands-on tutorial lowering the entry barrier for using the MO framework in Java. It is planned to make the prototype implementation and its tests available as open-source software.

F. Next Steps

The concerns raised on the MAL/SPP binding, mainly the interpretation of the Space Packet Protocol and its effects on message routing, have proven to be quite serious. It is expected to have these issues resolved soon, which results in the finalization of the standards book. Once interoperability has been demonstrated successfully the standard is ready for publication. Therefore test cases are defined and carried out with a prototype implementation by CNES as counterpart. The current test status is summarized in Fig. 6. Two MO framework stacks are shown, connected by an underlying transport layer responsible for exchanging Space Packets over TCP connections between the two stacks. These two stacks can be run on a single machine as well as on different machines. Up to this point test efforts have concentrated on unit tests, mainly regarding data encoding, resulting in 514 test cases. Interoperability tests testing through the whole stack will be performed once the standard has been finalized. The test environment has already been defined using FitNesse[*] as test tool, which allows fully automated testing. Two MO framework stacks are set up in different processes. Each stack consists of a MAL implementation and a MAL/SPP binding implementation. One stack hosts a test service provider implementation, while the other stack hosts a test service consumer implementation. Both stacks are initialized and sets of automated tests covering every single statement of the MAL/SPP binding specification are carried out. At the time of writing this "test bed" is used with the DLR prototype implementation for both stacks and with reused tests for the MAL that were originally written to test interoperability and unambiguousness of the MAL standard. They are not specific to the MAL/SPP binding, but still use many of the transport features and thus help eliminating possible bugs. Definition of specific tests is under way and together with usage of the two independently developed prototypes is prerequisite for the successful publication of the standard.

[*] http://www.fitnesse.org

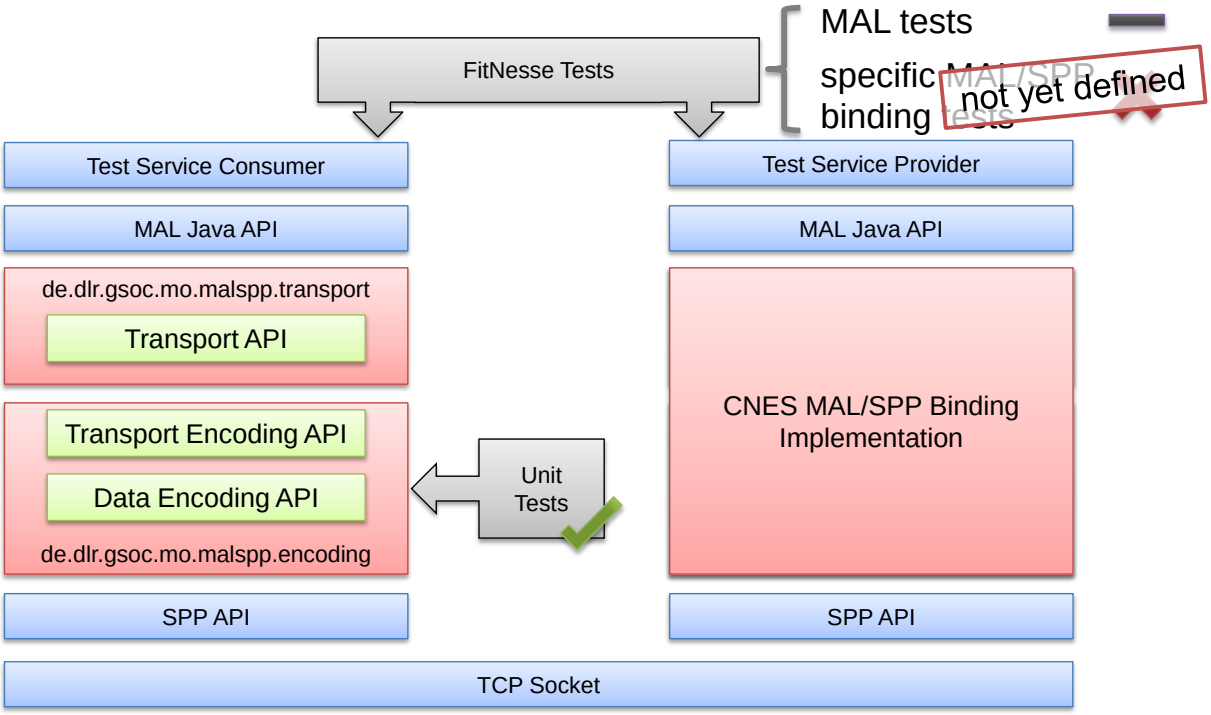

Figure 6. Setup of the test framework for MAL/SPP binding testing.

V. Monitor and Control Services

A. The Problem

A set of mission operations functions common to almost all spacecraft is the capability to monitor spacecraft parameters by telemetry and to control the spacecraft's subsystems by telecommands. Packet Telecommand and Packet Telemetry protocols, which are now consolidated into the Space Packet Protocol[6], were designed exactly for these requirements. However, they only provide the means of sending and receiving packetized data. It is still up to the mission to exactly define how packet telemetry is reported and how packets are used for telecommanding. The European Packet Utilization Standard PUS[7] provides a possible solution as it offers services designed for monitor and control. As already outlined in section IV-A this standard is strongly tied to the Space Packet Protocol and is thus not usable by the MO framework, which has a more encompassing claim: SPP is only one of many protocols and it should be possible to switch to a different technology without redefining the service. Furthermore, it is conceivable that monitor and control needs are not limited to the space segment, but can also arise on ground. For example, monitor and control of ground stations could also be handled by this service.

B. The Solution

The CCSDS SM&C working group has chosen to standardize monitor and control (M&C) services[12] as a first set of services, as these are at the core demanded by almost every mission. They can be mapped to the Space Packet Protocol by the MAL/SPP binding[8], but are not limited to that. If a different binding for some other technology is employed, they can also be used on ground or in ways nobody currently thinks about. Component reuse, interoperability, and cross-support thus become easily possible.

The M&C services provide a consistent set of services that allow action execution and tracking, parameter reporting, alert raising, parameter checking, statistics generation, and aggregation monitoring. The services rely on the Common Object Model[5] as their data model, and use its archiving and event services. Definition of some helper services completes the picture. The services will be briefly described in the following:

1. Action Service

This service is concerned with definition, invocation and execution tracking of actions, such as the execution of a telecommand. In principle actions could also be defined as operations of a custom service, however using the action service and thus its dynamic definition of actions ("dynamic" relates to the fact that the action service itself does not fix the exact kind of actions, it does not mean, that the system has to support dynamic definitions) reduces overhead and provides a common interface without the need of implementing custom services. Actions

can be checked, forwarded, their execution can be tracked, and the relationship between actions triggered by other actions can be monitored.

2. Parameter Service

This service allows consumers to subscribe to parameter updates. Parameter information is dynamically defined (in the same sense as dynamic definition of actions). The updates can be generated periodically, when the parameter value changed more than a user-defined threshold, or ad-hoc in a user-defined way.

3. Alert Service

This service is responsible for asynchronously publishing alerts to subscribers. Alerts are dynamically defined in the same sense as actions.

4. Check Service

This service provides parameter checking capabilities. The supported check types are "limit check", "expected value check", and "delta check". Additional types may be supported by individual implementations.

5. Statistics Service

This service allows association of parameters to statistic functions. The available functions are deployment-dependent. Supported are "maximum", "minimum", "mean value", and "standard deviation". Intervals for parameter sampling, report generation, and collection time between resets can be set independently.

6. Aggregation Service

The aggregation service is similar to the parameter service, allowing definitions of parameter aggregations to retrieve updates for in a single message. Updates can be generated periodically, ad-hoc triggered by implementation-specific measures, or when the values in the aggregation change more than a defined threshold (or after a user-defined time-out).

7. Helper Services

The helper services consist of a conversion and a group service. The conversion service offers structures for converting raw values to engineering values and vice versa. The minimum set of conversions includes discrete value mappings, linear conversions, polynomial conversions, and conversions from ranges to discrete values. Implementations may offer support for more conversions. The group service allows definition of groups of objects of the same kind, simplifying the usage of the other services, when they need to perform several operations on the same group of objects.

C. Benefits for Missions

Benefits for missions include the ability to monitor and control spacecraft implementing these services without the need to exchange or rework existing ground infrastructure if a mission using the MO framework is already supported. This reduces implementation costs of new missions and allows cross-support by other agencies, because they do not need to implement a mission-specific interface, but can use the standard monitor and control interfaces described here. However, these services explicitly allow for extensions that are mission-specific, like custom conversion functions or custom criteria for report generation. Still, a basic set of functionalities is guaranteed to be available, which can be used by any consumer adhering to the standard.

D. Current Status and Next Steps

Currently the M&C standard is under review and prototype implementation is pending. Similar to the MAL/SPP binding prototype, the implementation will be thoroughly tested in conjunction with an independently developed prototype by ESA in order to ensure interoperability. The same test bed will be used as for the MAL/SPP binding with M&C-specific tests covering every single clause of the specification. The unambiguousness and interoperability as shown by the test report is the final step to publication of the standard.

VI. Mission Operations Framework Usage at GSOC

Considering the benefits of the MO framework this section argues why the framework is not yet deemed ready to be implemented at the German Space Operations Center GSOC and how a possible transition could look like in future.

A. Why GSOC does not use the Framework (yet)

While the benefits of the MO framework sound promising there are several reasons the MO framework is not in use at GSOC right now. Despite much work has been put into the framework, it is still quite new and thus

unproven in mission-critical systems. Only now we see a complete MO stack from technology binding to MAL up to the services to reach a mature state that allows implementing systems based on that. A space segment relying on MO services and requiring MO support from ground still is not foreseeable in the near future of GSOC missions. As long as demand is low, it is hardly justifiable to implement the MO framework for space segment support. In future some technology demonstrator project might change that.

The same holds true for introducing the MO framework for ground-based infrastructure albeit to a lesser extent. Existing systems work well, people are trained and experienced using them and much effort has been put in to make use of them across several missions, avoiding individual solutions for each project. Switching to the MO framework was out of scope until now, because it still was incomplete at core components. The core has evolved by now, yet many of the services are still missing or are in the process of being defined. Rolling out the framework now would mean more work in the future in order to replace self-defined services with services standardized by then.

By exploring the framework and committing work to prototyping we assess the chances offered by it and generate knowledge in-house on how to use it and where to possibly apply it. The principal feasibility and fitness of the MO framework concept to accomplish its ambitious goals seem to be supported by our prototyping activities.

B. Possible Way of Transition to an MO Framework based Infrastructure

While the MO framework will not be introduced at GSOC in near future, it is still worth to think about a possible introduction process nonetheless. Although the first specified technology binding is to the Space Packet Protocol, the MO framework would likely to be introduced for ground-based systems first, allowing easier access and less cost. A very viable way of transition is by introducing the framework in phases, starting with a single system, running in parallel with existing systems for a while, eliminating deficiencies and proving its feasibility for the task. At GSOC we identified our system responsible for data exchange at the interface between in-house groups to be a good candidate for evaluating the MO framework in a real-life context and eventually replace the existing legacy system. Both systems would be able to run in parallel, thus allowing a smooth transition. The missing piece from the MO framework is a standardized technology binding for ground-to-ground transport, which is under discussion in the SM&C working group at the time of writing. A possible transition to the MO framework at GSOC would probably involve implementing a technology binding compatible with the current file-based approach having the possibility to slowly switch to more robust and performant technologies in future. Thus, the modular and abstract nature of the MO framework would allow gradually migrating systems without introducing disruptive changes and to then change underlying technologies over time without introducing any changes in higher-level interfaces. This makes it easier to argue in favor of the MO framework and to safely gain operational knowledge.

VII. Conclusion and Outlook

The CCSDS Mission Operations framework has been introduced briefly and the contributions of GSOC have been pointed out: The prototyping work for the technology binding to the Space Packet Protocol has shown some major obstacles like different interpretations of the concepts behind the Space Packet Protocol and the MO framework. These obstacles are in the process of being resolved. Next steps are finalization of the standard, definition and execution of interoperability tests and publication of the standard. The second prototype is concerned with one of the core sets of MO services, the Monitor and Control services. The standard is under review and prototype implementation is about to start. Interoperability tests will show the readiness of the standard. It has been outlined what the possible future for MO services at GSOC are and how they could be introduced unobtrusively. The next step for the SM&C working group is the definition of a ground-to-ground technology binding and the identification of service needs.

References

[1]"Mission Operations Services Concept", Informational Report (Green Book), CCSDS 520.0-G-3, December 2010, http://public.ccsds.org/publications/archive/520x0g3.pdf

[2]"Mission Operations Reference Model", Recommended Practice (Magenta Book), CCSDS 520.1-M-1, July 2010, http://public.ccsds.org/publications/archive/520x1m1.pdf

[3]Merri, M., Sarkarati, M., and CCSDS SM&C Working Group, "Get more Science out of your Missions with the CCSDS Mission Operations Services", SpaceOps 2014 Conference, AIAA, Reston, VA (submitted for publication)

[4]"Mission Operations Message Abstraction Layer", Recommended Standard (Blue Book), CCSDS 521.0-B-2, March 2013, http://public.ccsds.org/publications/archive/521x0b2e1.pdf

[5]"Mission Operations Common Object Model", Recommended Standard (Blue Book), CCSDS 521.1-B-1, February 2014, http://public.ccsds.org/publications/archive/521x1b1.pdf

[6]"Space Packet Protocol", Recommended Standard (Blue Book), CCSDS 133.0-B-1, September 2003, http://public.ccsds.org/publications/archive/133x0b1c2.pdf

[7]"Space Engineering: Ground Systems and Operations – Telemetry and Telecommand Packet Utilization" (PUS), ECSS-E-70-41A, 30 January 2003

[8]"Mission Operations Message Abstraction Layer – Space Packet Binding", Draft Recommended Standard (Red Book), CCSDS 524.1-R-1 (to be published)

[9]"OASIS Advanced Message Queuing Protocol (AMQP)", Version 1.0, OASIS Standard, 29 October 2012, http://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-complete-v1.0-os.pdf

[10]"Time Code Formats", Recommended Standard (Blue Book), CCSDS 301.0-B-4, November 2010, http://public.ccsds.org/publications/archive/301x0b4e1.pdf

[11]"Mission Operations Message Abstraction Layer – Java API", Recommended Practice (Magenta Book), CCSDS 523.1-M-1, April 2013, http://public.ccsds.org/publications/archive/523x1m1.pdf

[12]"Mission Operations Monitor & Control Services", Draft Recommended Standard (Red Book), CCSDS 522.1-R-3 (to be published)