YAMCS - A Mission Control System

Alejandro Fernández Sela[1], Nicolae Mihalache [2]
Space Applications Services, Belgium

Didier Moreau [3]
Belgian Institute for Space Aeronomy (IASB/BIRA), Belgium

In mission control centers, a major part of the ground segment is the front-end and back-end software which is used to assist payload and system operators in their daily tasks. YAMCS (which stands for "Yet Another Mission Control System" and is pronounced as yams, the sweet potato) is a software package which has been created for long-term payload operations. YAMCS is currently implemented at two User Support and Operations Centres (USOCs) as an extension to ESA's standard Mission Control System (MCS) for Columbus. This lightweight YAMCS integrates into the existing USOC architecture, fulfilling specific mission related needs that the standard MCS is not capable of.

During the last three years, YAMCS has quickly grown and it has been tested in several conditions supporting multiple parallel commanding/monitoring chains. It provides secure access to telemetry/telecommanding by remote users, and it supports the operators in archiving tasks.

This paper outlines why YAMCS is optimally suited as a solution when designing a "modern Mission Control System" and it describes YAMCS' strengths as middleware, archiving, and display technology. In May 2011, Space Applications Services has released YAMCS in conjunction with AISB/BIRA as open source software.

I. INTRODUCTION

THE Columbus Decentralised Mission Control System (CD-MCS) is the control system delivered by ESA to the different User and Support Operations Centres (USOCs). The Columbus Ground Support (CGS) software is the basis for CD-MCS.

However, none of the payloads which the USOCs are operating have been developed using the CGS software, thus CD-MCS is not perfectly suited for the operation of these payloads.

Originally, YAMCS was designed to complement CD-MCS, rather than to become a complete MCS. However, during the development, in order to satisfy all the operations needs, it has evolved into a complete Mission Control System. During the last four years, YAMCS has been used to support operations at the B.USOC (based at IASB/BIRA in Brussels, Belgium) and the Erasmus USOC (based at ESTEC, Noordwijk, the Netherlands).

II. RATIONALE FOR YET ANOTHER MISSION CONTROL SYSTEM

This section presents the factors and reasons that have driven the development of YAMCS. The problems encountered with the traditional MCS are explained from the point of view of both the Operators (Ops) and the Ground Controllers (GCs). Even though each of these positions has different needs and objectives, the final aim is

[1] Operations Engineer, Space Applications Services, Leuvensesteenweg 325, B-1932 Zaventem, Belgium, not an AIAA member.
[2] Ground Controller, Space Applications Services, Leuvensesteenweg 325, B-1932 Zaventem, Belgium, not an AIAA member.
[3] B.USOC Manager, IASB/BIRA, Avenue Circulaire 3, B-1180 Bruxelles, Belgium, not an AIAA member.

the same: successfully operating a mission. Therefore, good communication and support tools are indispensable for achieving this goal. YAMCS has been developed taking into account both perspectives, and this is the secret of its success: it satisfies the Ops needs and simplifies the GC activities.

A. Benefits of YAMCS for the Operators

1. Parallel operations

There are several reasons which encouraged the development of YAMCS, of which one of the most important was the need to run multiple commanding/monitoring chains in parallel. The USOCs can have several payloads to control, and each payload usually requires a simulation/development environment as well as an Engineering Model. Given that CD-MCS comes by default in two hardware "instances" (duplicated software), this becomes insufficient for real time operations. YAMCS was therefore conceived as scalable software, able to deal with multiple commanding/monitoring chains.

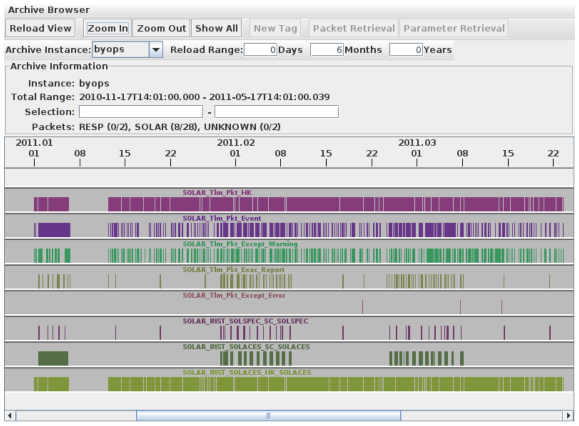
2. Archiving

The second reason is related to archiving. Although CD-MCS does contain an archiving system, it does not provide the ability to easily inspect or retrieve data from the archive, or to assess the archive completeness. Archive completeness is one of the problems faced by the USOCs (even more than by the usual Mission Control Centers): there are multiple places on the data path from the payload to the USOC where data can be temporarily stored and lost. As such, a mechanism for efficiently detecting and retrieving the missing data is required.

YAMCS provides a smart solution for this issue: the Archive Browser.

This Graphical User Interface is used to inspect the content of the Archive and to retrieve telemetry (TM) packets or parameters in dump (bulk) mode. The base for the archive browser display is a database containing a list of start time/stop time records for different TM packets. This allows a quick overview of when specific experiments have been turned on/off, as well as assessing the archive completeness.

Figure 1 shows the different packets received from the SOLAR[4] payload on a graphical view. By placing the mouse over each block of packets, YAMCS provides the start and end time of that specific packet.



Archive Instance: byops
Reload Range: 0 Days 6 Months 0 Years

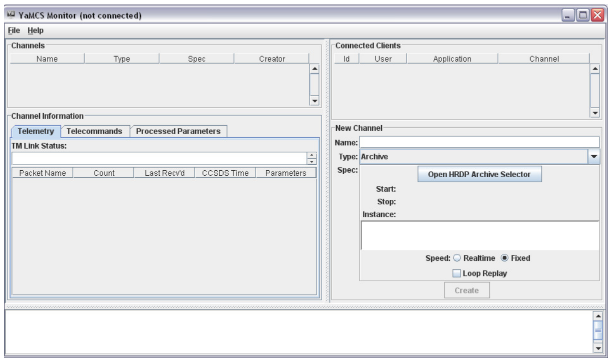
Instance: byops
Total Range: 2010-11-17T14:01:00.000 - 2011-05-17T14:01:00.039
Selection: -
Packets: RESP (0/2), SOLAR (8/28), UNKNOWN (0/2)

Figure 1. Archive Browser interface

3. Accessibility for external users

Another important role of the USOC is to provide external partners with access to the scientific and real time data, in this case the Principal Investigators. For specific payload operations dedicated User Home Bases (UHBs) can be set up. UHBs, which are the investigator's location at the "home" institutes, are typically national institutions (e.g. universities, national centers), which need to obtain an adequate communication and data processing infrastructure, allowing real-time data monitoring and control (in some cases) of their experiment (e.g. for remote operations).

The external partner only needs a Virtual Private Network (VPN) connection to the USOC and the YAMCS-client to be able to browse the data. Commanding capability can also be enabled with YAMCS. The YAMCS-Monitor interfaces easily with the user, who is able to connect locally running clients to the different available channels with a simple click. On top of that, the YAMCS-Monitor also shows statistics about the received/sent TM/TC (telecommand).



Type: Archive
Speed: Realtime [✓ Fixed]

Figure 2. YAMCS Monitor interface

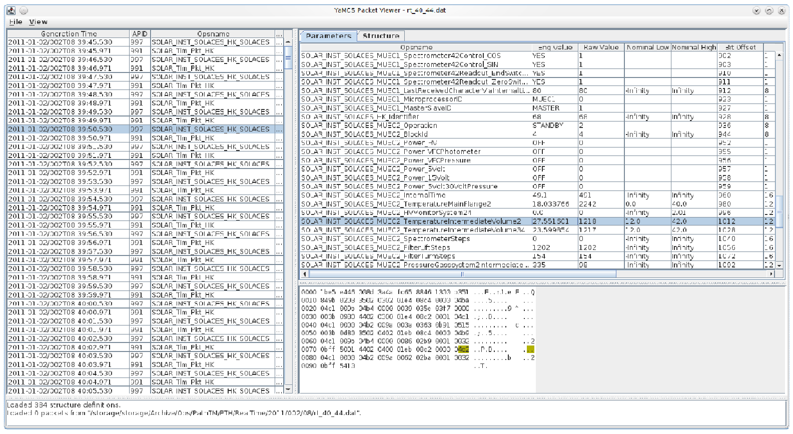
[4] SOLAR is a payload installed on the Columbus External Payload Facility (CEPF) on ISS which measures the total and spectral solar irradiance with three instruments.

4. Replay

During Real Time operations, another important capability is the replay. Replays are critical in case of anomalies in order to identify the root cause and to assess real-time actions if needed. If the anomaly occurred during Loss Of Signal (LOS), the replay can only be performed after receipt of this data and the completion of the merging process. YAMCS is capable of performing this task within seconds of the signal being restored, which can be vital for successful operations. Additionally, the replay channel can be launched in parallel with the operations display, allowing the operator to follow the current payload status whilst performing the replay.

5. Packet and parameter inspection

Troubleshooting of ground segment related problems or monitoring specific TM items that might not be available on the current display is a concern for the Ops team. The complete set of parameters and packets being received can be listed thanks to the Packet-Viewer tool. The intuitive interface is shown in Figure 3.



Generation Time	APID	Opsname	...
2011-01-02/002T08 39:45.530	997	SOLAR_INST_SOLACES_HK_SOLACES	...
2011-01-02/002T08 39:45.971	991	SOLAR_Tlm_Pkt_HK	...
2011-01-02/002T08 39:46.530	997	SOLAR_INST_SOLACES_HK_SOLACES	...
2011-01-02/002T08 39:46.971	991	SOLAR_Tlm_Pkt_HK	...
2011-01-02/002T08 39:47.530	997	SOLAR_INST_SOLACES_HK_SOLACES	...
2011-01-02/002T08 39:47.971	991	SOLAR_Tlm_Pkt_HK	...
2011-01-02/002T08 39:48.530	997	SOLAR_INST_SOLACES_HK_SOLACES	...
2011-01-02/002T08 39:48.971	991	SOLAR_Tlm_Pkt_HK	...
2011-01-02/002T08 39:49.530	997	SOLAR_INST_SOLACES_HK_SOLACES	...
2011-01-02/002T08 39:49.971	991	SOLAR_Tlm_Pkt_HK	...
2011-01-02/002T08 39:50.530	997	SOLAR_INST_SOLACES_HK_SOLACES	...
2011-01-02/002T08 39:50.971	991	SOLAR_Tlm_Pkt_HK	...
2011-01-02/002T08 39:51.530	997	SOLAR_INST_SOLACES_HK_SOLACES	...
2011-01-02/002T08 39:51.971	991	SOLAR_Tlm_Pkt_HK	...
2011-01-02/002T08 39:52.530	997	SOLAR_INST_SOLACES_HK_SOLACES	...
2011-01-02/002T08 39:52.971	991	SOLAR_Tlm_Pkt_HK	...
2011-01-02/002T08 39:53.530	997	SOLAR_INST_SOLACES_HK_SOLACES	...
2011-01-02/002T08 39:53.971	991	SOLAR_Tlm_Pkt_HK	...
2011-01-02/002T08 39:54.530	997	SOLAR_INST_SOLACES_HK_SOLACES	...
2011-01-02/002T08 39:54.971	991	SOLAR_Tlm_Pkt_HK	...
2011-01-02/002T08 39:55.530	997	SOLAR_INST_SOLACES_HK_SOLACES	...
2011-01-02/002T08 39:55.971	991	SOLAR_Tlm_Pkt_HK	...
2011-01-02/002T08 39:56.530	997	SOLAR_INST_SOLACES_HK_SOLACES	...
2011-01-02/002T08 39:56.971	991	SOLAR_Tlm_Pkt_HK	...
2011-01-02/002T08 39:57.530	997	SOLAR_INST_SOLACES_HK_SOLACES	...
2011-01-02/002T08 39:57.971	991	SOLAR_Tlm_Pkt_HK	...
2011-01-02/002T08 39:58.530	997	SOLAR_INST_SOLACES_HK_SOLACES	...
2011-01-02/002T08 39:58.971	991	SOLAR_Tlm_Pkt_HK	...
2011-01-02/002T08 39:59.530	997	SOLAR_INST_SOLACES_HK_SOLACES	...
2011-01-02/002T08 39:59.971	991	SOLAR_Tlm_Pkt_HK	...
2011-01-02/002T08 40:00.530	997	SOLAR_INST_SOLACES_HK_SOLACES	...
2011-01-02/002T08 40:00.971	991	SOLAR_Tlm_Pkt_HK	...
2011-01-02/002T08 40:01.530	997	SOLAR_INST_SOLACES_HK_SOLACES	...
2011-01-02/002T08 40:01.971	991	SOLAR_Tlm_Pkt_HK	...
2011-01-02/002T08 40:02.530	997	SOLAR_INST_SOLACES_HK_SOLACES	...
2011-01-02/002T08 40:02.971	991	SOLAR_Tlm_Pkt_HK	...
2011-01-02/002T08 40:03.530	997	SOLAR_INST_SOLACES_HK_SOLACES	...
2011-01-02/002T08 40:03.971	991	SOLAR_Tlm_Pkt_HK	...
2011-01-02/002T08 40:04.530	997	SOLAR_INST_SOLACES_HK_SOLACES	...
2011-01-02/002T08 40:04.971	991	SOLAR_Tlm_Pkt_HK	...
2011-01-02/002T08 40:05.530	997	SOLAR_INST_SOLACES_HK_SOLACES	...

Opsname	Eng Value	Raw Value	Nominal Low	Nominal High	Bit Offset
SOLAR_INST_SOLACES_MUEC1_Spectrometer42Control_COS	YES	1			902
SOLAR_INST_SOLACES_MUEC1_Spectrometer42Control_SIN	YES	1			903
SOLAR_INST_SOLACES_MUEC1_Spectrometer42Readout_EndSwitc...	YES	1			910
SOLAR_INST_SOLACES_MUEC1_Spectrometer42Readout_ZeroSwit...	YES	1			911
SOLAR_INST_SOLACES_MUEC1_LastReceivedCharacterValInternalLi...	80	80	-Infinity	Infinity	912
SOLAR_INST_SOLACES_MUEC1_MicroprocessorID	MUEC1	0			923
SOLAR_INST_SOLACES_MUEC1_MasterSlaveID	MASTER	1			927
SOLAR_INST_SOLACES_HK_Identifier	68	68	-Infinity	Infinity	928
SOLAR_INST_SOLACES_MUEC2_Operation	STANDBY	2			936
SOLAR_INST_SOLACES_MUEC2_BlockId	4	4	-Infinity	Infinity	944
SOLAR_INST_SOLACES_MUEC2_Power_HV	OFF	0			952
SOLAR_INST_SOLACES_MUEC2_Power_VFCPhotometer	OFF	0			955
SOLAR_INST_SOLACES_MUEC2_Power_VFCPressure	OFF	0			956
SOLAR_INST_SOLACES_MUEC2_Power_5Volt	OFF	0			957
SOLAR_INST_SOLACES_MUEC2_Power_15Volt	OFF	0			958
SOLAR_INST_SOLACES_MUEC2_Power_5Volt30VoltPressure	OFF	0			959
SOLAR_INST_SOLACES_MUEC2_InternalTime	49.1	491	-Infinity	Infinity	960
SOLAR_INST_SOLACES_MUEC2_TemperatureMainFlange2	18.033766	2242	0.0	40.0	980
SOLAR_INST_SOLACES_MUEC2_HVMonitorSystem24	0.0	0	-Infinity	2.01	996
SOLAR_INST_SOLACES_MUEC2_TemperatureIntermediateVolume2	27.551501	1218	12.0	42.0	1012
SOLAR_INST_SOLACES_MUEC2_TemperatureIntermediateVolume34	23.599854	1217	12.0	42.0	1028
SOLAR_INST_SOLACES_MUEC2_SpectrometerSteps	0	0	-Infinity	Infinity	1040
SOLAR_INST_SOLACES_MUEC2_FilterLiftSteps	1202	1202	-Infinity	Infinity	1056
SOLAR_INST_SOLACES_MUEC2_FilterTurnSteps	154	154	-Infinity	Infinity	1072
SOLAR_INST_SOLACES_MUEC2_PressureGassystem2Intermediate...	335	98	-Infinity	Infinity	1092

```
0000 1be5 e445 008d 3a4a fc65 8846 1300 0351  ...F...l.e F ..Q
0010 849b 0203 3502 0302 01e4 08c4 0000 04ba  ....5....    ...
0020 04c1 000b 04b4 0000 0039 035e 03f7 0000  .........9 ^ ...
0030 003b 0900 4402 0300 01e4 08c2 0001 04c1  .;..D....    ...
0040 04c1 0000 04b2 009a 003a 0363 0b91 0515  .........   c ...
0050 003b 0d80 3502 0402 01eb 08c4 0000 04b9  .;..5....    ...
0060 04c1 0009 04b4 0000 0086 02b9 0001 0032  .........    ..2
0070 0bff 5001 4402 0400 01eb 00c2 0000 04c2  ..P.D....    ...
0080 04c1 0000 04b2 009a 0062 02ba 0001 0032  .........b   ..2
0090 0bff 5410                                ..T.
```

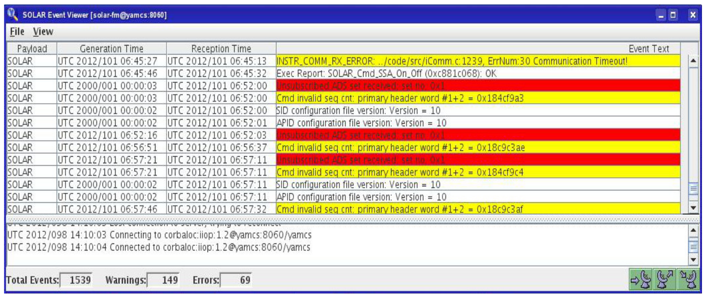
```
Loaded 334 structure definitions.
Loaded 0 packets from "/storage/storage/Archive/Ops/PathTM/PTH/RealTime/2011/002/08/rt_40_44.dat".
```

Figure 3. Packet-Viewer interface

6. Events from the payload

Besides the housekeeping telemetry, payloads often send packets upon the occurrence of an event. These event packets require a tool for them to be decoded and visualized, and YAMCS caters for this by providing the Event Viewer.

The Event Viewer is extremely useful for anomaly identification during real-time operations. For a complex, long-term mission as SOLAR, it has become an indispensable tool. Figure 4 is an example from SOLAR operations.



Payload	Generation Time	Reception Time	Event Text
SOLAR	UTC 2012/101 06:45:27	UTC 2012/101 06:45:13	INSTR_COMM_RX_ERROR: ../code/src/iComm.c:1239, ErrNum:30 Communication Timeout!
SOLAR	UTC 2012/101 06:45:46	UTC 2012/101 06:45:32	Exec Report: SOLAR_Cmd_SSA_On_Off (0xc881c068): OK
SOLAR	UTC 2000/001 00:00:03	UTC 2012/101 06:52:00	Unsubscribed ADS set received: set no. 0x1
SOLAR	UTC 2000/001 00:00:03	UTC 2012/101 06:52:00	Cmd invalid seq cnt: primary header word #1+2 = 0x184cf9a3
SOLAR	UTC 2000/001 00:00:02	UTC 2012/101 06:52:00	SID configuration file version: Version = 10
SOLAR	UTC 2000/001 00:00:02	UTC 2012/101 06:52:01	APID configuration file version: Version = 10
SOLAR	UTC 2012/101 06:52:16	UTC 2012/101 06:52:03	Unsubscribed ADS set received: set no. 0x1
SOLAR	UTC 2012/101 06:56:51	UTC 2012/101 06:56:37	Cmd invalid seq cnt: primary header word #1+2 = 0x18c9c3ae
SOLAR	UTC 2012/101 06:57:21	UTC 2012/101 06:57:11	Unsubscribed ADS set received: set no. 0x1
SOLAR	UTC 2012/101 06:57:21	UTC 2012/101 06:57:11	Cmd invalid seq cnt: primary header word #1+2 = 0x184cf9c4
SOLAR	UTC 2000/001 00:00:02	UTC 2012/101 06:57:11	SID configuration file version: Version = 10
SOLAR	UTC 2000/001 00:00:02	UTC 2012/101 06:57:11	APID configuration file version: Version = 10
SOLAR	UTC 2012/101 06:57:46	UTC 2012/101 06:57:32	Cmd invalid seq cnt: primary header word #1+2 = 0x18c9c3af

```
UTC 2012/098 14:10:03 Connecting to corbaloc:iiop:1.2@yamcs:8060/yamcs
UTC 2012/098 14:10:04 Connected to corbaloc:iiop:1.2@yamcs:8060/yamcs
```

Total Events: 1539 Warnings: 149 Errors: 69

Figure 4. Event viewer display during SOLAR operations.

B. Benefits of YAMCS for the Ground Controllers

1. Complex architectures

The MCS is a key element of the Ground Segment infrastructure. The Ground Controllers (GCs) are responsible for its maintenance and well-functioning. The risk of failures is directly dependent on

-The amount of services running on a server

-The amount of hardware to be maintained (servers, switches, routers, workstations)

-The complexity of the Ground Segment (GS) architecture.

The use of virtual machines is an option to reduce hardware risks, but it is not a fail-safe solution.

The typical YAMCS architecture is composed (hardware-wise) of a server and the clients that are connected to it. Software–wise, YAMCS developed with a message based architecture with a main bus where all the data is received and broadcast to the different clients. Figure 5 describes this structure.

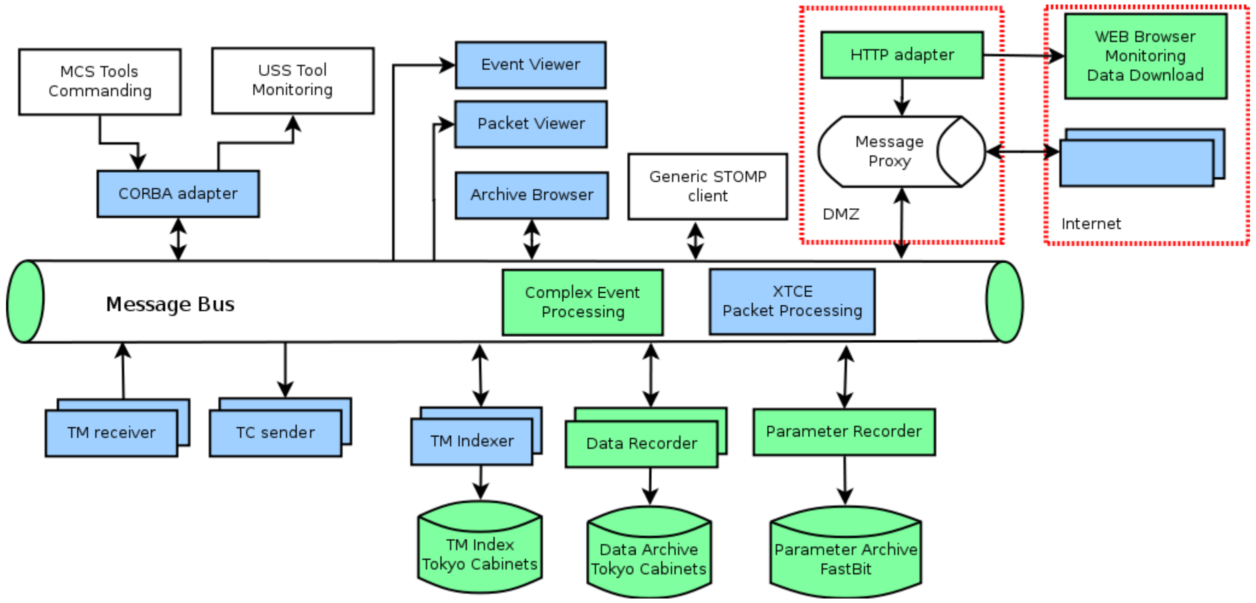


Figure 5. YAMCS current architecture. YAMCS parts under development are in red frames.

In addition to that, there is a single service running on the server and the configuration is quite straight forward. With the appropriate installation packages, the YAMCS Server can be installed in a matter of minutes. This is certainly an asset for the GC team.

All the YAMCS components are implemented in Java and the software is delivered as rpm or deb packages on Linux or as simple zip file on Windows. One of the main goals when implementing YAMCS has been to keep it as light as possible.

2. Different protocols, compatibility problems

The USOC MCS is delivered with different software for acquisition of telemetry and dispatching of commands using different protocols: CD-MCS multicast, raw packets over TCP, Data Services Subsystem: DaSS (Columbus Ground Segment specific protocol), PaCTS (EGSE software used by some of the Columbus payloads). YAMCS is able to interface with all of them. Furthermore, custom protocols can be easily added.

3. Packet definition

Some instruments use data structures (e.g. multi-level sub-commutated packets) and encodings (e.g. little endian) that are not supported by CD-MCS. There was a need for software able to decode those packets. The parallel development of the XML Telemetric & Command Exchange (XTCE) standard has been most welcome. The concepts of XTCE are used throughout the monitoring chain of YAMCS. XTCE has been found very well suited to encode the data structures of all the payloads and experiments the USOCs have to deal with.

4. Access to data, privileges

Another important challenge is the need to have different levels of secured access to TM/TC by remote users. This feature is implemented in YAMCS together with the possibility of smartcard based authentication and encryption for all the commanding activities. Authorization at parameter, packet, and command level is available. The authorization data are stored in an LDAP database.

5. Middleware selection: RPC versus Messaging

An important decision taken during the development of YAMCS has been to drop CORBA (and RPC in general) in favour of messaging based communication between the different YAMCS components. Other options, like HTTP based RESTful architectures, have been considered but discarded because they are more suitable for Internet wide communications rather than intra-MCS communications.

To justify the decision to drop CORBA, three communication (or interaction) patterns that appear when implementing an MCS have been identified: Request-Reply, Publish-Subscribe and Large Data Transfer.

Request-Reply is a typical pattern used when sending a command (telecomand, control command for some service, etc.). It is also used to retrieve small information packets. Naturally, all RPC systems support this pattern.

However (and this is considered as a main flaw of the RPC concept), it is very easy to forget that remote calls exhibit different characteristics from local calls: they can timeout, they can fail partially and they are essentially executed in a different call stack (meaning unexpected concurrency problems). In a messaging system, one has to set up two queues: one for sending the request, and the other for receiving the reply. Both the server and the client are made aware that network messages are being sent, and error conditions have to be taken care of.

The Publish-Subscribe pattern is typically used when retrieving real-time data or replay of historic data if multiple end points require the same historic data. Implementing the pattern using an RPC system means that clients are registering callback objects, and the server is sending data to them. One has to take care that one client will not block the server for sending data to the other clients. To overcome this problem, the YAMCS server creates a thread for each client and uses internal bounded queues to receive data from the producer. The producer will not add data into the queue if the queue is full (which means that slow clients will not receive all data). The usage of callbacks poses firewall problems and requires usage of an additional protocol (bidirectional IIOP) to support communication between CORBA type objects, which adds to the complexity of the system. Essentially, the functionality described above is built into all the messaging systems, which significantly simplifies development. The messaging infrastructure can use optimizations like reliable multicast (implemented in DDS) which brings extreme scalability for systems that requires many subscribers.

Large Data Transfer is mentioned as a separate pattern because it does not need to be sensitive to latency (a scenario for which Request-Reply would typically be used), but it does require flow control to deal with slow clients (a scenario for which Publish-Subscribe is normally used). Typically, to implement large data transfer (using whatever technology, even raw TCP), the data has to be truncated and sent in chunks (like one telemetry packet at a time or a set of parameters at a time). In YAMCS this has been implemented using one way CORBA invocations. Unfortunately, the invocation semantics of one way operations are poorly specified: there is no guaranteed delivery (although in practice the delivery is guaranteed since TCP is used), and worse, there is no guaranteed order of delivery. In JacORB (the CORBA implementation used by YAMCS), to achieve the required ordering of data, a single threaded Portable Object Adaptor (POA) is used on the client side. In a direct server-client connection, the flow control is relayed to the TCP stack. This causes *head of line* problems if the client wants to perform a call in parallel with the ongoing data transfer because the reply ends up in the server queue after all the data. In addition, when passing many one way invocations over the CORBA proxy, the proxy buffers the data, crashing after consuming all the memory. The solution implemented on the YAMCS server to deal with this problem is to synchronize with the client regularly.

6. Archiving: Traditional SQL database versus Key-Value Datastores and Column Oriented Database

Another decision when implementing archiving, is where and how to store the data. MICONYS has moved from an embedded database (C-tree) based storage to a standard network database based storage (MySQL or Oracle). OCTAVE seems also to use a traditional SQL database to store the data.

An MCS has to process time ordered streams of data (and not individual records like a traditional database). There are two types of utilization scenarios for these streams:

1) Retrieval from a time interval - this is used for data replay to displays or for data extraction for post processing
2) Searching for individual parameters values (usually the engineering values).

The two scenarios are somehow conflicting: while the first one calls for storing entire data packets or parameters in time ordered manner, the second one calls for storing a reversed view which maps parameter values to time when they have been generated (in traditional database terms indexing on all parameters). This can be daunting for a traditional database when the number of parameters to be indexed is as high as a few hundreds.

For stream based data (TM, TC, Events), a key-value datastore that sorts based on the key is a good choice. The key can be chosen such to achieve proper ordering and uniqueness.

For the parameter database a different architecture has been selected: the column oriented database (as opposed to row oriented databases).

These databases partition data vertically: each column (or group of columns) being stored in sequence. This approach has a few advantages:

1) Accesses are more efficient because the software needs to read only the columns required without having to read the entire row, which contains unnecessary columns
2) Because all data of one column is of the same type, the database can compress much better, and can use bitmap and run length encoding algorithms which do not even require a decompression to read the data values.

After analyzing the available open-source options, the following products have been found most suitable to be used by YAMCS:

1) Tokyo Cabinets - a key-value datastore for storing the telemetry, events, telecommands and other "raw" stream data. The B+Tree type of database is used to ensure sorting.
2) Fastbit - An Efficient Compressed Bitmap Index Technology for the parameter archive.

III. YAMCS – The next version

This section describes how YAMCS is evolving in order to be applied in future projects. An important project where YAMCS will be used is the ongoing ACES (Atomic Clock Ensemble in Space) Ground Segment development. The main addition required by this project is an archiving component. The following additions/changes are currently in development:

1) An HTTP adapter is being developed allowing for real-time monitoring using SVG+Javascript based web pages.
   The same adapter allows for retrieving data in bulk mode from the archive.

2) Complex Event Processing

According to Wikipedia:
Complex event processing (CEP) consists of processing many events happening across all the layers of an organization, identifying the most meaningful events within the event cloud, analyzing their impact, and taking subsequent action in real time.

If ”organization” is replaced by ”spacecraft” this sounds very much like what a spacecraft operator has to do. The current MCSs do not allow easy implementation of sophisticated monitoring alarm conditions. XTCE allows raising an alarm if a parameter passes some predefined thresholds (Out of Limits) or if the parameter value’s rate-of-change is either too fast or too slow. Even simple things, like monitoring an average of the last x values of a parameter, are not straightforward to implement. Correlation between different events is also problematic. For example, it is not easy to express a condition that a parameter passing a threshold should raise an alarm only if other events (like other parameter out of limits) have happened in a certain timespan. CEP introduces some interesting concepts:

1) Moving windows - a window represents a part of the data stream of a certain length (either based on time or number of samples). As time passes, new data is added on one side of the window and removed from the other side. At any given moment, a window is very much like a table in a traditional database. One can define various queries or selecting, averaging, aggregation, etc. The queries are automatically executed each time the data in the window changes.
2) Patterns - patterns are useful to perform temporal event correlation within a window. Using patterns one can address the problem described above of raising alarms when a set of parameters pass some thresholds simultaneously. Negative patterns are also possible: for example a negative pattern could be that a heater does not turn on in a certain time window after a temperature has reached a low threshold.

Although the examples and the concepts presented above are very credible, the current trends in MCS development do not seem to consider implementing this kind of functionality. The current commercial CEP systems have been mainly designed for the finance industry (monitoring the stock exchange) and thus are very expensive and not particularly suitable for spacecraft monitoring. In YAMCS a certain level of window processing has been implemented to aggregate over moving windows. Pattern detection is currently not implemented; however, this as an area which would greatly benefit the Mission Operations and this functionality will be included in the future.

IV. CONCLUSIONS

This paper gives an overview of a YAMCS - a lightweight Mission Control System. It shows how YAMCS integrates into existing USOCs architecture, fulfilling specific mission related needs that the standard setup/configuration is not capable of. YAMCS has been developed with a message based architecture allowing a logical flow of the data from the servers to the clients. It is mission independent and can handle multiple protocols and clients. Access control is assured and can be handled by LDAP if needed. The required hardware architecture, and the software design of YAMCS with a single server that can be accessed by any client, makes this tool a great choice as a Mission Control System. All its components have been developed in JAVA and they are compatible with the common operating systems being delivered as rpm or deb packages on Linux or as simple zip file on Windows. YAMCS has been successfully tested and validated during the last four years in two USOCs and is being improved for further missions with HTTP and complex event processing features. In addition to that YAMCS is released as open source software, available at www.yamcs.org.

Appendix A
Acronym List

ACES	Atomic Clock Ensemble in Space
CD-MCS	(Columbus) Decentralised Mission Control System
CEP	Complex Event Processing
CGS	Columbus Ground Support
CORBA	Common Object Request Broker Architecture
DaSS	Data Services Subsystem
DDS	Data Distribution Service
deb	Debian software package
EGSE	Electrical Ground Support Equipment
GC	Ground Controller
GS	Ground Segment
HTTP	Hypertext Transfer Protocol
JacORB	Free CORBA 2.x compliant ORB written in Java. [Open Source, LGPL]
JAVA	A general purpose, high-level, object-oriented, cross-platform programming language developed by Sun Microsystems
LDAP	Lightweight Directory Access Protocol
MCS	Mission Control System
OCTAVE	High-level interpreted language, primarily intended for numerical computations
OPS	Operator
PaCTS	EGSE software used by some of the Columbus payloads
RESTful	Representational state transfer (REST) constraints
RPC	Remote Procedure Call
rpm	Red Hat software package
SQL	Structured Query Language
SVG	Scalable Vector Graphics, a portable graphics format defined in XML for vector graphics.
TC	Telecommand
TCP	Transmission Control Protocol
TM	Telemetry
UHB	User Home Base
USOC	User Support and Operations Centre.
VPN	Virtual Private Network
XML	eXtensible Markup Language
XTCE	XML Telemetric & Command Exchange
YAMCS	Yet Another Mission Control System

Appendix B

Acknowledgments

The authors wish to thank the European Space Agency (ESA) Directorate of Human Spaceflight (D/HSF) (ESA Exploitation Program) and the Belgian Science Policy Office (BELSPO) (ESA Prodex and other programs) for their funding. B.USOC is part of the Services & Operations Division of the Belgian Institute for Space Aeronomy (BISA).

Also a special word of thanks to the BUSOC management and Space Applications Services management for their support and for providing the budget and personnel for the activities.