The Link Management System for the European Data Relay System

Tobias Göttfert* Boris Grishechkin* Maria Theresia Wörle†

Christoph Lenzen*

Deutsches Zentrum für Luft- und Raumfahrt e. V., German Aerospace Center

Münchener Straße 20, 82234 Weßling, Germany

A new scheduling system is under implementation at the German Space Operations Center to support the EDRS mission with its two payloads in the geostationary orbit. The tasks of this Link Management System involve receiving and checking the operational plan from the Mission Operations Center, scheduling additional payload-related maintenance activities, and relaying the resulting timeline incrementally towards the automated control system, while respecting all spacecraft constraints. By its design, the Link Management System already incorporates some ideas and functionality of the upcoming GSOC Reactive Planning framework.

Within this paper, the scheduling tasks and implemented algorithms are described in the context of being as generic as possible for both the EDRS-A and EDRS-C payloads and already including concepts and some code base of the upcoming GSOC Reactive Planning framework.

I. Introduction

The Link Management System (LMS) is the Mission Planning component of the control centers operated by the German Space Operations Center (GSOC) for the European Data Relay System (EDRS) mission. EDRS-A comprises an optical laser communication terminal (LCT) as well as a Ka-band antenna on the Eutelsat EB9B satellite to provide data relay services to the ESA Copernicus program and other low-earth-orbit (LEO) customers. EDRS-C will host an LCT on a dedicated platform operated by the GSOC. The LMS is developed in-house by the Mission Planning team specifically for EDRS, but based on the generic GSOC scheduling library Plato[1] and its domain specific planning modeling language.[2] Its technological basis represents one step further towards the next generation planning system of GSOC, called "Reactive Planning" (formerly known as "Incremental Planning Framework").[3]

I.A. The EDRS mission

The European Data Relay System[4] will eventually comprise payloads on two geostationary satellites. EDRS-A is hosted on-board a Eutelsat telecommunication platform as two payloads—a Ka-band inter-satellite antenna and an LCT[5]—and the Ka-band feeder antenna for data transfer to ground. The payloads allow for 300 MBit/s (Ka-Band) and 600 or 1800 MBit/s (LCT) LEO-to-ground data transfer. This satellite was launched end of January 2016. The second part of the EDRS system will be the EDRS-C satellite, scheduled for launch end of 2017. The primary payload will be the LCT for EDRS, while other hosted payloads will be mounted on the platform as well. The EDRS service also covers the possibility of ground-to-LEO data upload.

The ground segment of EDRS is composed by the Mission Operations Center (MOC), two satellite/payload control centers, and four ground stations for the EDRS antennas. While the MOC schedules and coordinates all inter-satellite links (ISLs) between the LEO customers and the EDRS satellites, the Devolved Payload Control Center (DPCC) of EDRS-A and the Satellite Control Center (SCC) of EDRS-C—both hosted at

*Mission Planning System Engineer, Mission Operations department, German Space Operations Center

†Deputy Head of Mission Planning Team, Mission Operations department, German Space Operations Center

GSOC—have the duty to execute the scheduled links and perform the necessary routine tasks to keep the EDRS payloads in working order. The actual commanding of the satellites and their bus operations happens at the SCCs for EDRS-A and EDRS-C, at Eutelsat and GSOC, respectively.

The DPCC—and also the payload operations of SCC—operate in a fully autonomous mode, starting from April 2016, in order to support a load of up to 200 programmed inter-satellite links per day and a very short order deadline of less than one hour. The operations concept foresees manual interaction into the link programming process only in contingency cases. For this required autonomy, a sophisticated system has been designed and implemented that includes components for fully automated scheduling and commanding, coupled via telemetry feedback from the satellite.

I.B. Tasks of the Link Management System

The LMS is a fully automated software component that receives planning requests from the Mission Operations Center and allows for scheduling optical and Ka-band inter-satellite communication (hereafter called *links*), the potential upload of data from ground to be transferred during the links, as well as other payload configuration requests to control the state of the EDRS payload. The LMS consists of several components which implement ingesting requests, scheduling them together with additional tasks for the management of the EDRS payloads, exporting sequences of Flight Operations Procedures (FOPs), and reporting the status of the requests to various ground segment components and the DPCC operators. Live telemetry is ingested to consider the actual state of the payload when modelling scheduling-relevant resources. The LMS runs continually, incorporating each new input into the planning process as it arrives.

In particular, the LMS receives the following three types of planning requests from the MOC: requests for regular inter-satellite links, additional requests for automated execution of particular pre-defined FOPs (e. g. for supporting spacecraft maneuver operations on the payloads), and requests for manual execution of any non-standard FOP.

These inputs are pre-scheduled by the MOC and are requested from DPCC/SCC for uplink several hours in advance. For changes to the planning on short notice, the MOC is allowed to add/remove planning requests until less than one hour before execution. The LMS populates its internal timeline with the MOC planning requests as far as all constraints are met. Constraint can e. g. be non-overlap rules between certain types of links, or between links and spacecraft maneuvers, or the fill level of the on-board command buffer. In case the request format or an overlap with other requests does not allow for execution of the request, the LMS treats it as unfeasible and reports to the MOC and operators, whereas upon a violation of the command buffer fill level the requests are kept unscheduled until a later time, when enough free slots are available in the buffer. Special care has been taken to ensure that the required pre-processing by the flight dynamics system, which is needed for optical link sessions, does not allow later Ka-band link sessions to completely fill up the command buffer while the optical links can not yet be scheduled. Otherwise, the uploaded Ka-band links would need to be removed again from the satellite to make room for earlier optical links. The aim of the LMS is to always fill the on-board command buffer as full as the received amount of requests permits, so that the time of autonomous on-board operations is maximised for the case of temporary loss of commanding capability. At the same time, a configurable amount of TTC slots is kept free onboard the satellite, in order to allow for out-of-order planning requests from the MOC, termed “late requests”, that can still be commanded without the need to remove TTCs onboard.

In addition to the external requests, the LMS is in charge of scheduling activities for the maintenance and calibration of the EDRS payloads to ensure a constant level of pointing accuracy. These are mainly the upload of operational products that the LCT needs in regular intervals, namely orbit parameters for the on-board orbit propagator, time synchronisation to the ground clock, and the matrix elements of the alignment of the LCT. Their scheduling has to fulfill several rules in order not to disturb onboard link operations on the one hand, while also not placing additional restrictions on MOC scheduling either. Therefore, sophisticated re-scheduling algorithms were implemented that modify the timeline upon reception of new requests from MOC that would be in conflict with these maintenance activities.

Finally, the LMS has to use remaining time of the TC uplink budget for the possible upload of data that is buffered onboard the GEO satellite for optical transmission towards a customer’s LEO spacecraft (*forward data*). After the forward data has been fully stored on the EDRS payload, it can afterwards be transferred as part of the optical link session towards the LEO satellite at a much faster bandwidth (up to 1.8 Gbit/s). Since the forward data needs to be uploaded—due to the construction of the payload—via S-band TC uplink, which usually lasts several hours (up to 1.5 days), the scheduling algorithm needs to model the amount of uplink time already used by the rest of the telecommand uploads and throttle the forward data upload rate in order to not block the uplink for link programming. Therefore, once the MOC has requested such a data upload, the LMS starts uploading it in small segments by modelling their uplink duration and the uplink duration of the link and routine FOPs, and allowing for only a certain amount of time in the near future being blocked. Once the LMS performs its next scheduling run, this time horizon has moved a little and probably some further segments fit into the uplink timeline. This is repeated until the whole forward data has been uplinked.

To fulfill the requirement that the scheduling process shall be based on telemetry feedback, the amount of used slots for time-tagged telecommands (TTCs) is fed back into the LMS and its internal model of used TTC slots is updated with actual values. This allows to e. g. detect TTCs injected without the knowledge of LMS, or TTCs that were not being uploaded properly. The time delay it takes for a TTC requested by LMS to reach the satellite and the delivery of telemetry feedback about the TTC buffer contents needs to be taken into account by an elaborate algorithm in order to align the LMS resource model with the feedback received.

In addition to its scheduling duties, the LMS has to fulfill also extensive reporting requirements that allow the various parties within the ground segment to keep track of the processing status of the requests within the LMS and its overall internal status.

II. Architecture of the Link Management System

The software architecture of the LMS marks a step forward from the traditional cycle-based autonomous planning systems of GSOC (e. g. the one used for the TerraSAR-X/TanDEM-X[6] mission), towards the future "reactive" planning systems, that incorporate each new input directly into their timeline. In addition, in order to support unattended operations while being usable by non-expert operations personnel (i. e. without strong background in the GSOC mission planning systems), a strict separation of the core functionality from the user interface (GUI) has been implemented, allowing the GUI to be started and shut down independently of the core LMS. The core LMS is wrapped in an operating system service and can receive commands and report status from/to the GUI via a communication framework.

Apart from that the core LMS permanently listens for incoming requests on its file interfaces and triggers new scheduling and telecommand export for every new input it receives.

Since the underlying Plato library is implemented in Microsoft's .NET C# programming language, the technologies and platforms chosen for the LMS are a Windows server operating system with .NET, WCF (Windows Communication Foundation) and WPF (Windows Presentation Foundation).

II.A. Components of the Link Management System

The decomposition of the LMS software is depicted in Fig. 1. Within the core LMS, five main components deal with different tasks of the planning service:

- The *Timer* handles the main event loop and calls every other component in turn once new input has been received (incl. commands issued by the operator). In addition, it can also trigger certain actions, including scheduling runs, based on time. For instance, every night the export of summary reports is triggered; missing telemetry input triggers an alarm message after some timeout; ...

- The *FileIngestion* is responsible for translating all received input items according to their interface definitions into the GSOC modeling language, in order to be processable by the GSOC Plato library. The requests are examined for consistency criteria and stored into the planning model.

- The *Scheduler* is the main planning component and applies an EDRS-specific configuration and composition of generic algorithms available in the Plato library to perform all scheduling functionality of the LMS. Especially, planning requests are scheduled to create a conflict-free updated version of the timeline, and in case of violation are either held back until a later time or flagged as unfeasible, providing the reason of unfeasibility.

- The *TaskExporter* takes care of exporting the scheduled tasks in the format required by the automatic commanding system of DPCC/SCC and the satellite. In addition, it manages the on-ground model of the so-called Ka-band on-board pointing table, from which it deducts the correct telecommand parameters to upload the Ka-band antenna pointing vectors.

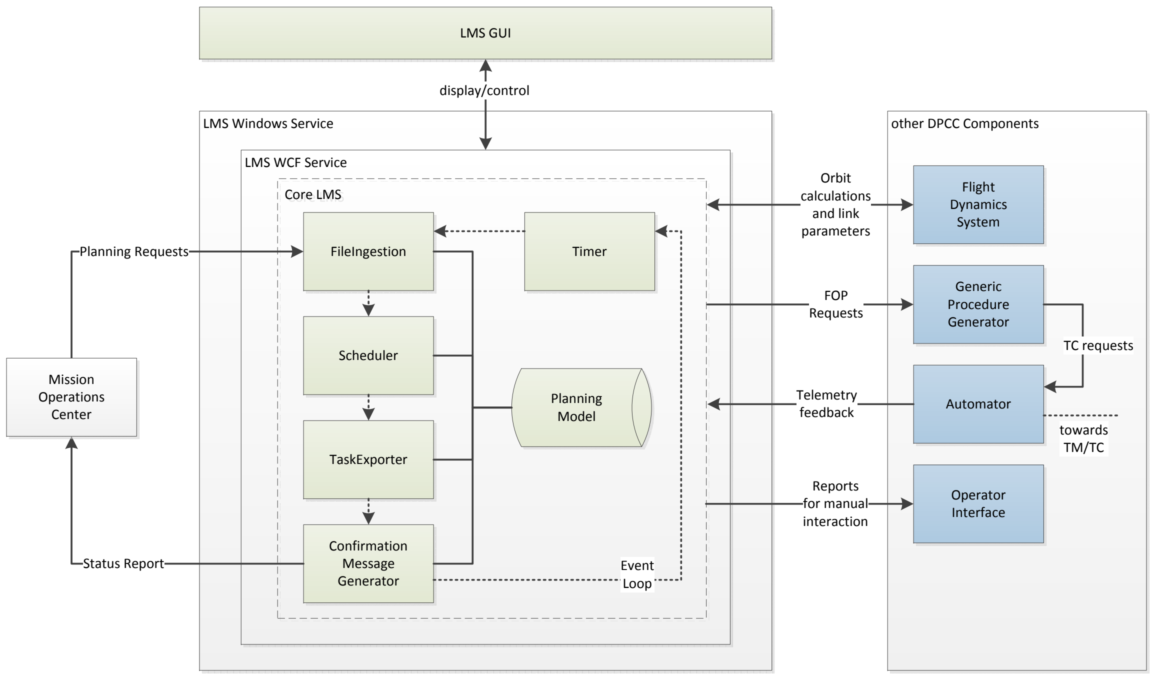


Figure 1. Components and main interfaces of the Link Management System (green). Only the most relevant DPCC components from the LMS point of view are shown (blue).

- The *ConfirmationMessageGenerator* takes care of creating all necessary reporting messages to support the various status reporting interfaces of the EDRS ground segment, both internal (within DPCC/SCC) and external (to MOC).

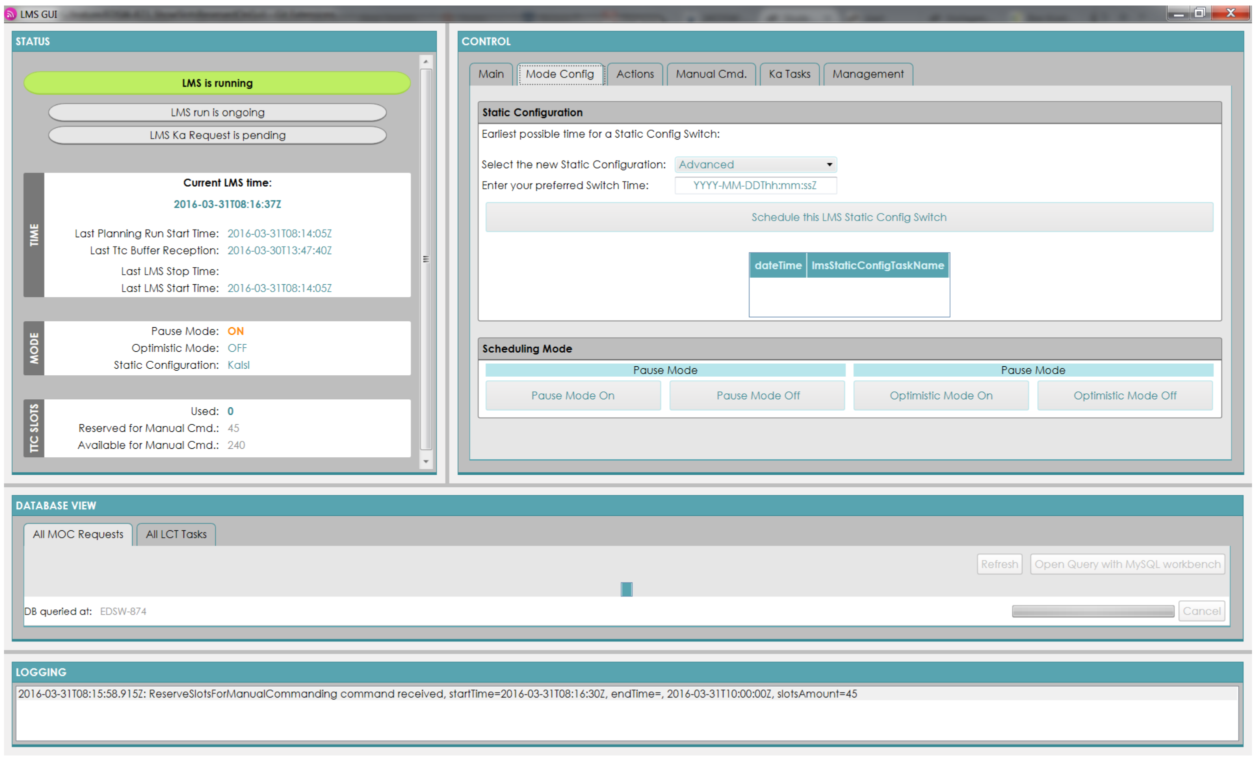
In addition to the core components that form the autonomous mission planning functionality, the LMS features components for wrapping them into an operating system service (for being able to run without a logged-in user), a communication layer to interface with the LMS GUI, and the GUI itself. The usage of the communication layer allows for an easy change or exchange of the GUI, should it become necessary. Fig. 2 shows a screenshot of the LMS GUI, where the operator has several possibilities to monitor the core LMS and interact with it. For example, the LMS can be put into one of different scheduling modes (c. f. Section II.C), or a reservation for TTC slots that the LMS may not use because they are needed for manual operations can be placed, etc.

II.B. LMS scheduling functionality

Since the LMS uses the GSOC Plato (Planning Tool) scheduling library at its core, within the LMS planning model the incoming requests are represented as *groups* and *tasks* with associated parameters, as defined in the GSOC modeling language. As part of the planning model, the LMS Scheduler always maintains a single latest and valid version of the timeline, to which it adds timelineentries (TLEs) for tasks newly to be scheduled and from which it removes TLEs for tasks that are to be deleted. The persistency layer of Plato keeps track of past versions of the planning model, including the timelines, for debugging and archival purposes. The scheduled tasks interact with the modeled resources via modification profiles; the main resources that are modeled are the fill level of the on-board TTC buffer, the uplink time budget already used, and the configuration the EDRS payload is currently in.

The sequence of scheduling algorithms is executed for each scheduling run, i. e. after any new input has been received, and performs both simple and more complex scheduling functions. The most important algorithms shall be outlined in the following:

- Scheduling of a special resource correction task brings the TTC resource to the correct value when



LMS is running

LMS run is ongoing

LMS Ka Request is pending

Current LMS time: 2016-03-31T08:16:37Z

Last Planning Run Start Time: 2016-03-31T08:14:05Z
Last Ttc Buffer Reception: 2016-03-30T13:47:40Z
Last LMS Stop Time:
Last LMS Start Time: 2016-03-31T08:14:05Z

Pause Mode: ON
Optimistic Mode: OFF
Static Configuration: Kalsl

Used: 0
Reserved for Manual Cmd.: 45
Available for Manual Cmd.: 240

Earliest possible time for a Static Config Switch:
Select the new Static Configuration: Advanced
Enter your preferred Switch Time: YYYY-MM-DDThh:mm:ssZ

DB queried at: EDSW-874

```
2016-03-31T08:15:58.915Z: ReserveSlotsForManualCommanding command received, startTime=2016-03-31T08:16:30Z, endTime=, 2016-03-31T10:00:00Z, slotsAmount=45
```

Figure 2. Screenshot of the LMS GUI while the LMS is running. The operator may monitor several parameters of the current planning (e. g. the number of used TTC slots) and issue commands to the LMS (e. g. switching to a different scheduling mode).

incorporating the TTC buffer fill level information that was received via telemetry input, while taking the time delay of FOP uploads and TM delivery into account.

- Scheduling of link requests has to take into account the varying amount of TTC slots that are used by different requested link types, as well as the current configuration of the EDRS payload. Depending on it, different FOPs need to be scheduled and a different amount of TTC slots is used.
- Scheduling of parameter uploads for updating the on-board orbit propagator (OOP) must be performed in order to upload fresh parameters before the currently active parameters have expired. However, the time-tag for an OOP upload must not overlap with an optical link, unless it is performed within the initial pointing phase of the link. Therefore the Scheduler must be able to un-schedule OOP uploads from the timeline, should a newly received link request overlap with it, and schedule other OOP tasks instead to completely cover the resulting interval without OOP validity.
- Scheduling of planning requests for automated FOP execution can also trigger resource modifications, e. g. to model the parking state of the LCT. With this knowledge, conflicts and inconsistencies have to be avoided (e. g. rejection of double-parking of the LCT).
- Scheduling of requested deletions for both links and automated FOP executions is handled using special additional tasks for two reasons: First, it must be ensured that the original request has either been uplinked completely or not been scheduled at all, i. e. there must be no residual TTCs in transit. Only then the proper parameters for the deletion telecommands can be obtained, since they depend on the integer on-board time of the original telecommand, which is only known to the LMS once the telemetry contains it. Second, the LMS has reporting duties also on the execution state of the deletion request and it is handled within the MOC-DPCC interface as a separate object. If the first condition cannot be met, the deletion request stays unscheduled for the moment, until all TTCs are visible in the telemetry feedback (which may take up to 30 minutes).
- The execution of the on-board processing functionality that moves the Ka-band antenna to track the LEO satellite during a link is handled via an involved on-ground model that generates the needed

telecommands. The amount of required TTCs is only known after all Ka-ISLs are scheduled and then fed back into the model.

- Additional payload maintenance tasks for the LCT instrument are scheduled without a planning request from MOC, in order to update the optical alignment of the LCT and the LCT clock. Those tasks are scheduled either while the LCT is parked, or outside of requested LCT links and after the order deadline, in order to not pose any limitation on the links the MOC can request.
- All scheduling algorithms take the required deadlines and spare TTC slots into account, i. e. the latest time for receiving new planning requests from MOC, the amount of TTC slots that have to remain free in favor of requests received on short notice, and the latest time when the Eutelsat SCC accepts TTCs for uplink to EDRS-A.
- The times and constraints stemming from spacecraft maneuvers are taken into account and reflected in the scheduling algorithms; e. g. alignment matrix updates are scheduled while the LCT is parked during a maneuver, and a fresh OOP product is scheduled at the end of each maneuver since the orbit has changed then.

II.C. Telemetry feedback into the LMS

Special care has been taken to properly incorporate telemetry feedback into the LMS. As such, the LMS is the first planning system at GSOC that has a direct interface from the telemetry/telecommand (TM/TC) chain to the scheduling process. The requirements of the EDRS mission state that the fill level of the TTC buffer must be monitored and that every scheduling and reporting activity shall be based on the actual state of the satellite. The customary assumption that a requested TTC will be successfully uploaded and be removed from the TTC buffer upon its excecution time may not be made for EDRS. In addition, the TTC buffer of EDRS-A can hold only a very limited amount of TTCs, which usually does not allow that all requested links from the MOC can be directly uploaded to the satellite, so that the TTC buffer is constantly kept as full as possible by the LMS. To complicate the situation even more, the LMS shall be able to fall back to a modeling-only approach in case no telemetry is available anymore, but the command uplink is considered to be safe and operations shall continue.

The DPCC and SCC have a dedicated component handling satellite telemetry, automated commanding, and reporting based on telemetry analysis, called *Automator*, which is presented at the same conference.[7] The LMS receives information about the current state of the TTC buffer from the Automator on a regular basis, as depicted in Fig. 1. However, the number of TTC slots reported by telemetry cannot directly be used and must be compensated for TTCs that were requested for uplink by the LMS, but have not yet appeared in the telemetry feedback. This problem is pronounced for EDRS-A for several reasons: First, since the satellite is operated at the Eutelsat control center, in addition to the processing delay inside the DPCC, also transfer and processing times to/inside the Eutelsat control center occur. Second, the EDRS-A satellite takes a significant time until the full content of the TTC buffer is reported in telemetry (usually 2×8 minutes). The LMS therefore internally stores each TTC that it requests for uplink. The Automator processes the telemetry and correlates the TTCs from the satellite with the requests from the LMS. With this information, the LMS can determine which TTCs are still "in transit" and which are already contained in the telemetry information, and correct the planning model accordingly.

The telemetry feedback was included since the LMS is required to catch several problems that an optimistic resource modeling of the TTC usage would not be able to handle. First, problems of the satellite would be caught, e. g. TTCs that unexpectedly did not drop out of the TTC buffer, but also problems within the TM/TC processing chain on ground. Second, and possibly more relevant, the LMS also gets notified about TTCs that were uploaded outside the LMS modeling, e. g. by operator interaction, which is an off-nominal case that needs to be caught.

The nominal mode of telemetry feedback can also be deactivated in favor of falling back to a traditional modeling-only mode, called "optimistic mode", which is useful if for a limited time the telemetry is not available, while reliable commanding can still be ensured. For this, a combined modeling approach was implemented that continuously tracks the TTC slots resource in both manners, i. e. using pure modeling and a block-by-scheduling/release-by-telemetry approach. Depending on the settings, one of the two resources is used for scheduling, while seamless switching between the two modes is possible any time by a command from the operator.

III. The LMS as technology preview of the Reactive Planning Framework

III.A. Overview of the GSOC Reactive Planning Framework

While not being a direct implementation of the upcoming GSOC Reactive Planning framework, the LMS already incorporates concepts and some base libraries that will be used in future GSOC mission planning systems.

The Reactive Planning framework is a project at GSOC to develop the next generation of mission planning systems that will be used for the missions in house. The EnMAP mission[8] will be the first mission to be completely based upon this new framework. The aims of Reactive Planning are four-fold:

- To allow for better code re-use by clear separation between generic and mission-specific components, interfaces and functionalities.
- To improve on reaction time (e. g. shorter order deadline, incorporation of last-minute inputs, updated or removed ground station contacts, . . .) by maintaining an up-to-date timeline; i. e. each input triggers an incremental planning run immediately after reception.
- To be robust against uplink anomalies by taking into account different outcomes of the commanding process and preemptively preparing corrective timelines and command files, should a command uplink fail.
- To improve on customer feedback and control by instant status updates whenever the planning status changes and the possibility to get a preview of the effects that a desired planning request has on the timeline. The planning system may calculate possible outcomes of the request and the user may select which of the offered alternative solutions shall be submitted (e. g. which other requests to cancel to be able to submit the current request without conflicts).

To satisfy these requirements, a set of components, interfaces and their interactions has been designed: A central scheduling component operates continuously on the latest and authoritative version of the timeline, directly incorporating each new input. Multiple independent customer interfaces can present the user with previews of the desired planning requests and also the effects on the timeline that are to be expected, allowing him to order the best fitting alternative. The inputs, including planning requests, are queued and sorted according to urgency of scheduling. A timer component can trigger actions based on upcoming events, e. g. the export of portions of the timeline to the command system in time before the next ground station contact. Telemetry information is incorporated into the system by an interface to the monitoring and control system, allowing to react to external events. Finally, alternative versions of the timeline may be generated by one component in anticipation of the success/failure of the commanding process, in order to resume the nominal mission as quickly as possible by uploading prepared corrected timelines.

In addition to the generic components, mission-specific components then need to be added to adapt the Reactive Planning framework to a specific satellite mission: The external input needs to be converted into the GSOC modeling language, possibly using knowledge about the payload and/or sensor characteristics. The specific algorithms for the scheduling problem at hand must be composed and configured. Finally, the actual commanding interfaces need to be added, i. e. concrete flight procedures and their parameters.

III.B. The LMS as a Reactive Planning precursor

As mentioned before, the LMS is a precursor in spirit for the GSOC Reactive Planning, since its requirements and design share some of the goals of Reactive Planning.

First, the LMS shall run continuously and autonomously and react on every new input from MOC and the other DPCC components, providing instant feedback about the changes in the planning model—i. e. be responsive—and also provide a very short order deadline. Second, the LMS, for some resources, shall not only rely on modeling, but also wait for telemetry feedback, which is then incorporated and corrects the modeled resource fill levels.

While most of the code of the Reactive Planning framework is not yet available, some basic parts already made it into the LMS, namely the persistency layer of the Plato model. This new persistency allows for serialisation of the Plato model into a MySQL database with a one-to-one correspondence between objects in memory and in the database. In addition, the Plato model is versioned, i. e. tracks all previous content of the model by storing all states of objects in a differential way. Modifications on the Plato in-memory model are performed using transactions (Software Transactional Memory[9]) and written to the database at the end of each transaction. During each scheduling run, the LMS components do not retrieve their information from the database or the overall model, but pass around and evaluate so-called Change Sets, i. e. the set of modifications applied to the model during the previous transaction. For instance, the ConfirmationMessageGenerator filters the obtained Change Set in order to detect status modifications of model objects, which it then announces via reports.

IV. Current state of implementation and outlook towards EDRS-C

As of March 2016, the LMS is ready to support the commissioning and first operational phases of the EDRS-A mission, which will start in April. Currently, the code is extended to add some functionality needed in the future (e. g. the capability to upload forward data), and to support also the EDRS-C environment within the SCC. EDRS-C will bring much more bandwidth available to upload forward data, but sacrifice the Ka-ISL antenna. In addition, EDRS-C is based on a completely different satellite bus, making it necessary to command different FOPs and adjust some parameters of the scheduling (e. g. the size of the TTC buffer). This has the most drastic effect on scheduling the OOP products, which needs to be handled by an all-new algorithm, since they now act on the spacecraft bus directly, instead of only the LCT payload. Therefore, the requirements with respect to timing and spacecraft maneuvers considerably change. Other parts of the system, most notably the interfaces to MOC and most other DPCC components remain unchanged. The Plato library and the chosen component architecture allows to keep most of the LMS code generic and only implement EDRS-C-specific versions of some scheduling algorithms and the FOP export functionality, while handling other differences between EDRS-A and EDRS-C via configuration updates. In the end, separate instances of the common LMS codebase will run that support EDRS-A and EDRS-C, respectively.

V. Conclusion

The Link Management System is a fully automated mission planning system tightly integrated into the ground segment of the EDRS mission. It already supports EDRS-A operations and will evolve to also support EDRS-C payload operations with a second instance. From a planning point of view, the scheduling capabilities required are not more sophisticated than for typical LEO satellite missions, but far more involved than for classical GEO missions. In addition, tight requirements on timing and reporting behavior of the system have been set up.

The pre-plan that is received from the mission operations center via planning requests is checked for consistency, complemented with payload maintenance activities, and scheduled for uplink bit-by-bit, respecting the limited size of the on-board telecommand buffer.

The main novel properties for a GSOC planning system are the continuous mode of operation in order to process any new input directly and with a very short order deadline, and the usage of the new persistency layer of the Plato planning library, which both lead the way towards the upcoming GSOC Reactive Planning system. The third large novel feature is the automated incorporation of telemetry feedback into the scheduling, used to correct the resource fill levels and partly replacing modeling where required by the mission.

Abbreviations

DPCC	Devolved Payload Control Center (of EDRS-A)
EDRS	European Data Relay System
FOP	Flight Operations Procedure
GEO	geostationary orbit
GSOC	German Space Operations Center
ISL	Inter-Satellite Link
Ka-ISL	Ka-band inter-satellite link

LCT Laser Communication Terminal

LEO low earth orbit

LMS Link Management System (of the EDRS-A DPCC or EDRS-C SCC)

MOC Mission Operations Center

MPS Mission Planning System

O-ISL optical inter-satellite link

SCC Spacecraft Control Center l

TLE TimeLine Entry

TM/TC telemetry / telecommand

TTC Time-tagged Tele Command

References

[1]Lenzen, C., Wörle, M. T., Mrowka, F., Spörl, A., and Klaehn, R., "The Algorithm Assembly Set of Plato", *SpaceOps 2012 12th International Conference on Space Operations* 11–15 June 2012, Stockholm, Sweden

[2]"Planning Modelling Language", URL:
http://www.dlr.de/rb/en/Portaldata/38/Resources/dokumente/GSOC_dokumente/RB-MIB/GSOC_Modelling_Language.pdf
(cited 22 March 2016)

[3]Wörle, M. T., Lenzen, C., Mrowka, F., Göttfert, T., Spörl, A., Grishechkin, B., and Wickler, M., "The Incremental Planning System - GSOC's Next Generation Mission Planning Framework", in: *Space Operations: Innovations, Inventions, and Discoveries*, edited by C. Cruzen, M. Schmidhuber and L. Dubon, *Progress in Astronautics and Aeronautics*, Vol. 249, AIAA, Reston, VA, 2015, Chap. 13, pp. 285–307

[4]Witting, M., et al., "Status of the European Data Relay Satellite System", *Proc. International Conference on Space Optical Systems and Applications (ICSOS)*, 9–12 October 2012, Ajaccio, Corsica, France,

[5]Heine, F., Mühlnikel, G., Zech, H., Philipp-May, S., and Meyer, R., "The European Data Relay System, High Speed Laser Based Data Links", *7th Advanced Satellite Multimedia Systems Conference and the 13th Signal Processing for Space Communications Workshop (ASMS/SPSC)*, 2014, Livorno, pp. 284–286, doi: 10.1109/ASMS-SPSC.2014.6934556

[6]Mrowka, F., Geyer, M., Lenzen, C., Spörl, A., Göttfert, T., Maurer, E., Wickler, M., and Schättler, B., "The Joint TerraSAR-X/TanDEM-X Mission Planning System", Symposium Proceedings, pp. 3971–3974, *IGARSS 2011*, July 24–29, 2011, Vancouver, Canada, ISBN 978-1-4577-1004-9

[7]Beck, T., Schmidhuber, M., and Scharringhausen, J., "Automation of Complex Operational Scenarios - Providing 24/7 Inter-Satellite Links with EDRS", *SpaceOps 2016 14th International Conference on Space Operations*, 16–20 May 2016, Daejeon, South Korea

[8]Guanter, L., et al., "The EnMAP Spaceborne Imaging Spectroscopy Mission for Earth Observation", *Remote Sens.*, Number 7, pp. 8830–8857, 2015, doi:10.3390/rs70708830

[9]Shavit, N., and Touitou, D., "Software transactional memory", *Distrib. Comput.*, Feb 1997, Vol. 10, issue 2, pp. 99–116