Operating and Evolving the EDRS Payload and Link Management System

Tobias Göttfert*, Maria Theresia Wörle†, Sven Prüfer‡, Christoph Lenzen§
Deutsches Zentrum für Luft- und Raumfahrt e. V., German Aerospace Center
Münchener Straße 20, 82234 Weßling, Germany

Based on its Plato scheduling library, GSOC has developed the scheduling components for controlling the EDRS-A and EDRS-C payloads, called Link Management System. This paper reports on the operational experience gained from almost two years of EDRS-A operations and the improvements that have been made to the codebase. The necessary changes for EDRS-C are highlighted. Finally, thoughts and recommendations for future ground segment engineering are given.

I. Introduction

The Link Management System (LMS) is the mission planning component of the two DLR-hosted control centers that operate the European Data Relay System (EDRS) mission [1]. It participates in the automated scheduling and commanding process as well as autonomous operations the two control centers employ [2]. To do so, the LMS is the first mission planning system at the German Space Operations Center (GSOC) that performs event-based scheduling and real-time timeline updates. The LMS incorporates external inputs and planning requests from the Mission Operations Center (MOC) of Airbus and in addition schedules necessary activities on its own in order to keep the timeline consistent and fulfill all requirements, e. g. regarding automated instrument and platform maintenance.

Since the start of EDRS-A commissioning in April 2016, the LMS has been running continuously and, at the time of writing, supported more than 10 000 optical inter-satellite communication links. Valuable operational experience has been gained and several extensions were implemented since then, not only for EDRS-A alone, but also for the upcoming launch of the EDRS-C node. The codebase for initial EDRS-C operations is currently undergoing finalization with a projected launch in 2019.

A. The EDRS mission

The EDRS mission is an ESA–Airbus cooperation to place two geostationary relay nodes in space that can be used to transfer data to European groundstations via laser-optical and Ka-band inter-satellite communication [3]. The first node, EDRS-A, comprises hosted payloads aboard a Eutelsat platform and was launched early 2016. Since late 2016, EDRS-A is in routine operations and provides laser-optical transmissions from low-earth-orbit (LEO) to ground via a laser communication terminal (LCT) and Ka-band transmissions via a dedicated inter-satellite steerable Ka-band antenna. The second node of the mission, EDRS-C, will be launched in 2019 on a dedicated platform, operated by GSOC, and host another LCT. Both nodes are placed over Central Europe and offer transfer rates of 600 or 1800 MBit/s (LCT) or 300 MBit/s (Ka-ISL). The main goal of EDRS is to increase the orbit fraction during which a LEO satellite can transmit payload data to ground, as well as offering a downlink opportunity geographically close to European customers. Currently, the Sentinel satellites of the ESA Copernicus program are the main customers of EDRS, performing in total around 35 optical inter-satellite transmissions—hereafter called inter-satellite *links* (ISL)—per day.

For this mission, the ground segment was set up by Airbus and DLR GSOC, comprising a Mission Operations Center, four ground stations and two spacecraft/payload control centers, as depicted in Fig. 1. The MOC plans and coordinates all link activity with the customers, whereas the EDRS-A Devolved Payload Control Center (DPCC) and the EDRS-C Spacecraft Control Center (SCC), located in GSOC, schedule the planned links together with platform and maintenance activities for automated execution. The DPCC interfaces with Eutelsat to uplink the commands to the EDRS-A payload aboard a Eutelsat platform, whereas the SCC directly operates the EDRS-C satellite.

*Mission Planning System Engineer, Mission Operations department, German Space Operations Center, Tobias.Goettfert@dlr.de
†Deputy Head of Mission Planning Team, Mission Operations department, German Space Operations Center, Maria.Woerle@dlr.de
‡Mission Planning System Engineer, Mission Operations department, German Space Operations Center, Sven.Pruefer@dlr.de
§Mission Planning System Engineer, Mission Operations department, German Space Operations Center, Christoph.Lenzen@dlr.de

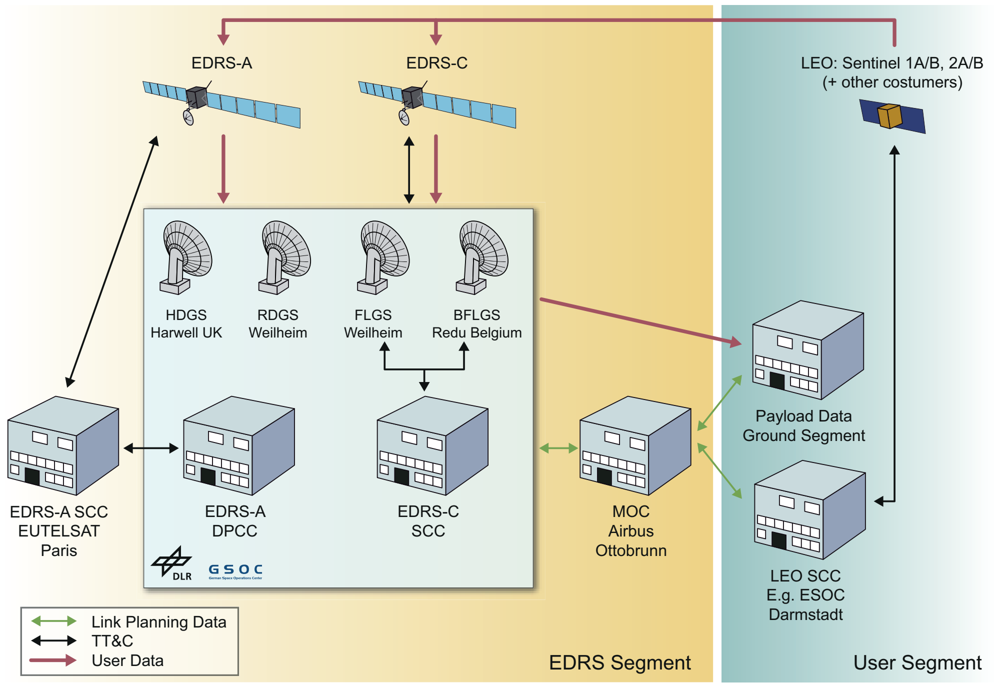

Fig. 1 Overview of the EDRS system. The four GSOC-built ground stations receive user data. Two of them also perform commanding of EDRS-C, whereas commanding of EDRS-A is performed via the Eutelsat control center.

B. Tasks of the Link Management System

The design of EDRS foresees the two payload control centers—EDRS-A DPCC and EDRS-C SCC—to operate in a fully automated way under nominal conditions, in order to achieve high throughput and short order deadlines for ISL operations. The responsibilities of DPCC/SCC comprise

- uplinking of the planned activities received from MOC,
- managing the limited on-board resources (mainly time-tagged telecommand buffer size and uplink rate),
- receiving and analyzing telemetry as well as automating operations using telemetry feedback,
- and performing routine maintenance procedures in an automated way for keeping the ISL payloads in a nominal state.

Therefore, the LMS as the scheduling component of DPCC/SCC, was implemented with many design goals of the future GSOC Reactive Planning System [4] already in mind, most notably the immediate reaction upon new input from MOC or the automated monitoring and command system (MCS).

The LMS keeps a single timeline which is updated on each new input following all scheduling rules. The constant connection to the geostationary EDRS platforms then allows for an immediate commanding of the differences to the previous state of the timeline, in order to maximize the time budget for recovery in case of connection loss. Another novel feature in comparison to traditional automated mission planning systems at GSOC is the incorporation of feedback from the automated MCS. This happens via confirmations about the success or failure of commanded flight operations procedures (FOPs) and via reports obtained from telemetry about the state of the payloads.

In detail, the LMS has to accomplish the following tasks:

- Scheduling all activities received from MOC and commanding them to the EDRS payloads as early as possible to maximize the length of the uploaded timeline, while respecting the capacity of the on-board buffer for time-tagged telecommands (TTCs). Excess planning requests are kept unscheduled until enough slots have become available. A certain amount of slots is kept reserved for the invocation of planning requests on short notice. The activities received from the MOC are:
  - Scheduling/commanding Optical ISL (O-ISL) activities
  - *EDRS-A only*: Scheduling/commanding Ka-ISL activities, including on-board management of the Ka-ISL antenna trajectory

    - Scheduling/commanding routine activities, like preparation for spacecraft station keeping maneuvers, when the payloads have to be "parked"
    - Uploading data in advance that shall be transferred from ground to the customer LEO satellite during an ISL.
- Autonomously performing LCT routine payload maintenance, i. e. updating the LCT time (*EDRS-A only*), LCT on-board orbit propagation (OOP) parameters (*EDRS-A only*) and the LCT alignment (*both missions*)
- *EDRS-C only*: Autonomously performing routine bus maintenance, i. e. updating the platform time and platform OOP parameters
- Allowing co-existence of operator-based commanding with automated commanding
- Detailed reporting towards both MOC and GSOC-internal operator personnel.

The LMS was first implemented for EDRS-A, went into initial operations April 2016, and has been running since, with few very short maintenance periods. The software was already designed with the upcoming EDRS-C mission in mind, which will use a separate instance of the LMS with an extended codebase. This extended codebase serves as the base for further development and will eventually also be used by EDRS-A. The LMS is based on the GSOC Planning and Scheduling library Plato [5], therefore all scheduling algorithms and objects in the planning model are expressed in the GSOC modelling language [6]. On top of this foundation, a continuously running software system was built that integrates into the operating system as a service daemon and communicates with a separate GUI program for operator interaction. For a detailed description of LMS components and architecture, see [1].

II. Operational experience from the EDRS-A mission

During the two year operation of the LMS at the GSOC EDRS-A DPCC, valuable experience has been gained, both on the automated system and the operations involving operations personnel.

A critical piece in terms of spacecraft safety and also for achieving the required level of DPCC availability is the stability and robustness of the software. It must be ensured that errors, be them from inputs from other systems, or from internal malfunctions, do not stop automated operations. On the other hand, unforeseen problems shall not compromise the internal consistency of the planning model, possibly leading to corruption of the LMS outputs and to tedious and hard to perform manual interventions, which might also result in severe downtime of the LMS. In addition, the state of the LMS and its internal error handling mechanisms shall not be hidden from the operators, so that manual action can be taken as early as possible, should it become necessary.

Therefore, the balance between “keeping the LMS software running at all costs” and “stopping LMS operations to prevent further problems down the road” was carefully chosen and had to be readjusted also during the initial operations phase. For example, during one occasion a problem occurred where telemetry feedback was considered multiple times, leading to an overbooking of the resource which models the used TTC slots, and therefore to a conflicting state of the timeline. The LMS stopped at this point to avoid corruption of the planning model. After solving this anomaly, a new software version was rolled out, where updated scheduling algorithms prevent this kind of conflict in the first place.

Another topic is operator awareness, which is very important for LMS operations. Due to the automated, in-the-background operations and the complex event-based and time-based scheduling processes going on, the LMS is not always as intuitive and as easily predictable as other DPCC components. Therefore, being able to inspect its state and analyze the reasons why it behaves in a particular way are crucial for judging whether the LMS operates correctly.

One example where the LMS did at first not provide the necessary awareness shall now be given: The LMS has an interface to the DPCC monitoring system, to which it not only reports its current status, but is also able to send out alarm messages to the operators. During one occasion it was noticed that not all errors logged in the LMS internal log file also trigger an alarm message, which caused a specific problem to go unnoticed for almost one day. Some time after the original problem occurred, the LMS stopped commanding new FOPs to the spacecraft, so that the anomaly was finally realized. The problem could be resolved in time before the already commanded on-board timeline ran out, so that the EDRS service was not interrupted. Since it cannot be required from the operators to regularly browse through all log files of all DPCC components, the whole LMS codebase was then reviewed for similar cases and the appropriate operator alarm messages were added to the software where missing .

Another issue that affected operator awareness was a problem in the LMS GUI that led to constantly increasing memory consumption of the GUI. While the problem persisted in the software, the LMS GUI could still be used well enough for performing normal operations (e. g. monitoring the state of the LMS or sending commands to the LMS) for a certain amount of time. The GUI could just be left closed while the LMS was running in the background and be restarted whenever necessary, thus the bug did not affect nominal operation of the system. However, such a situation should not prevail for a longer period of time, as the operations personnel quickly gets accustomed to the situation. After getting used to the workaround of having the LMS GUI not opened all the time, operators tend to keep this habit even after the problem is resolved, at the cost of reduced operator awareness, which shall be avoided.

The three problems mentioned were so far the only software bugs affecting routine operations of the LMS, and none of them led to an outage of the DPCC service. The rest of the time, the LMS has been running very stable and reliable, so that a very positive conclusion on the operational properties of the LMS can be drawn.

A. Achieving long-term performance

From the beginning of development it was clear that the projected number of total links the platform would perform during its lifetime of more than a decade and the number of all additional entities in the planning model would exceed the capabilities of the Plato data model in terms of memory and performance. This is because the scheduling algorithms will loop over many activities from the past when searching for conflicting opponents, and any update on the resource usages has to perform many computations when having to take into account all past activities. Thus the time for an LMS scheduling run increases and is at some point above the threshold for acceptable performance. The single timeline the LMS keeps and constantly modifies can therefore not go back in time indefinitely.

After a couple of months of LMS operations, the experience that was gained showed that the performance of the LMS would no longer be sufficient somewhen during the second half of 2017. With more and more objects in the internal planning model—about 30 000 activities had accumulated in the planning model by June 2017—every scheduling run slowed down noticeably.

By July 2017, the maximum duration of scheduling runs was approaching 10 minutes (the duration depends on the type of input that needs to be scheduled), reaching the performance limit that was set in the requirements. Most often, new input was received during those 10 minutes of scheduling, leading to constant LMS operations without any idle time in between. The duration was also increasing faster than linearly, since the addition of more Sentinel satellites on the LEO customer side meant that the rate of incoming O-ISL planning requests was increasing. In addition, the size of the planning model also manifested in a large storage size required on the database server, where regular database backups of the planning model were taking more than one gigabyte of space (uncompressed) and took several minutes, during which the LMS also was unresponsive. The frequency of such backups therefore had to be reduced drastically.

In order to guarantee good constant performance also in the future, the planning model content and therefore the timeline had to be restricted to a certain size. The implemented solution relies on a three-step "cleanup" process, which limits the amount of objects that are stored in the Plato planning model, ensuring the duration of the scheduling algorithms does not exceed the thresholds the LMS is allowed to use.

As a first step, unused activities residing in the past are removed from the model as a whole, since they can no longer be scheduled. These are mainly OOP parameter updates that were not scheduled, as the LMS receives many times more OOP products than necessary for platform operations in order to allow the scheduling algorithm to choose the best available parameters for a given time.

The second step limits the timeline to a given length in the past with a tunable parameter. This is currently set to 90 days and can be reduced (or enlarged), should the LMS become too unresponsive again with increasing EDRS-A usage. For this to work, the activities on the timeline can not simply be dropped, since they influence the modeled resources not only during their scheduled time frame, but usually also in the future, potentially until infinity. Therefore, a clean-up algorithm was implemented that ensures that effects of scheduled activities persist on the timeline. In the planning model of the EDRS-A LMS, fortunately there are no constraints on activities that require other activities to exist on the timeline, so the choice of activities to remove can be freely made—in our case all activities ending before a selected time T. The clean-up algorithm aggregates the effects on all modeled resources, which are caused by activities which end before T. These aggregated effects are restricted to times greater than T and added to a new placeholder object *sum task*, which is scheduled at T. Now the past activities are cleared, which erases their effects on the resources. This means that before T, no modifications on the resource profiles persist, whereas after T, the *sum task*'s resource modifications yield the same modification as the deleted activities did before (see Fig. 2). By repeatedly applying the clean-up algorithm as part of normal scheduling operations, the history of the timeline is therefore kept within a configurable gliding window in the past. This method allows for non-stop operations of the LMS for an indefinite amount of time and also has the benefit that the individual clean-up steps are comparatively short (however very frequent). During the phase-in of the clean-up feature in July 2017, the gliding window had to be chosen very large and then be reduced in several small steps, in order to not render the LMS unresponsive for a longer time during initial model clean-up, and therefore impact the service availability.

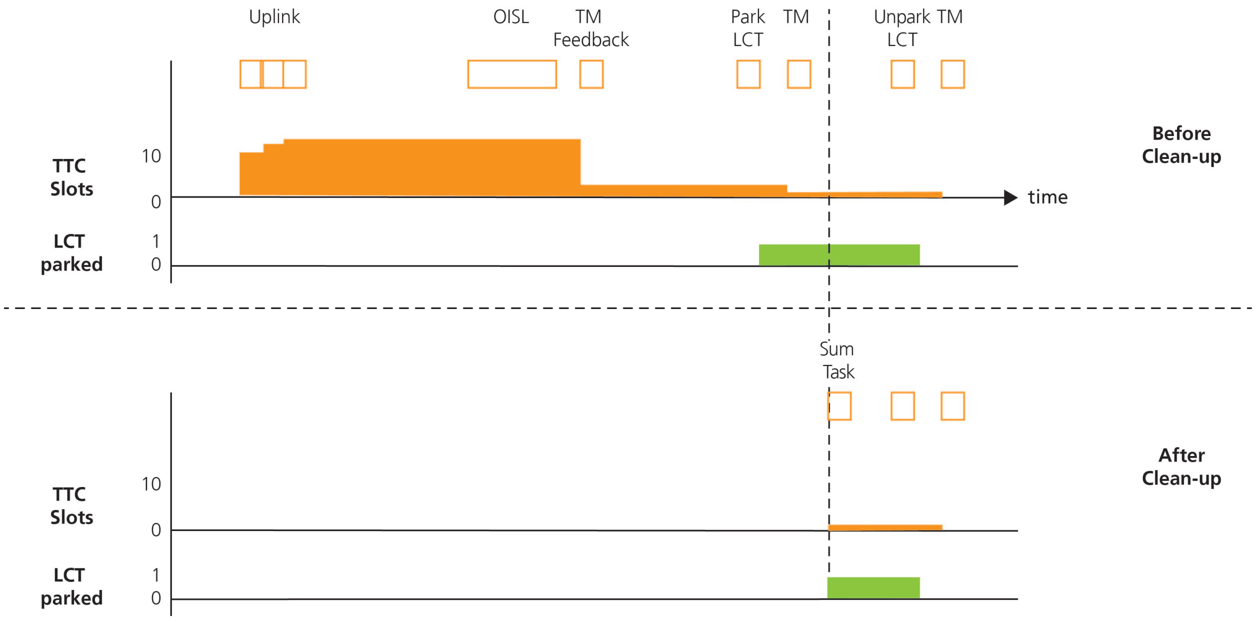

Fig. 2 Schematic example of model clean-up: all activities before a certain time *T* (depicted as a dashed vertical line) are removed and their resource modification profiles are attached to a replacement activity (*sum task*) in a joined form. Past resource modifications are lost in the process, whereas the future profiles are retained for each resource individually, such that the shape of each resource profile after the chosen time *T* is unchanged.

The third step of the clean-up process does not impact the memory footprint or scheduling performance of the planning model, but only the on-disk space usage. Here, the clean-up makes use of the model versioning scheme from the Plato model persistence, which saves the model state in a relational database: all objects, which have been deleted before a given model version, may be deleted from the database. This means that older versions of the persisted planning model are dropped, hence saving disk space and improving the database server performance, which keeps the LMS start-up time short.

After these measures, the LMS scheduling runs, including processing of new inputs, performing scheduling algorithms and the clean-up algorithm, and creation of commanding and report files, now take well below five minutes under all conditions. The size of the database backups stays constant at about 250 MB, and the number of activities in the planning model is constant at around 16 000. These performance numbers are well within the performance requirements and allow the LMS to handle a higher load in the future. Further reduction of scheduling run duration will be achieved once the optimized version of one resource-hungry scheduling algorithm is rolled out, which will save another 30–60 s per LMS scheduling run.

B. Summary of operational experience

After these two years of continuous 24/7 operations, with only three incidents that affected the running LMS, none of which caused a service outage, it is fair to say that the extensive work that was invested to create an extremely reliable and low-maintenance automated system has paid off. During the last year, the LMS was regularly running continuously for several months before being restarted (mostly for operational reasons). The main conclusions that can be drawn from LMS operations, from the software engineering and mission planning standpoint shall now be given.

First, a well chosen, capable and flexible system and software architecture are key for successfully implementing and modifying a reliable planning system fulfilling the requirements. Functional and quality requirements are of equal importance in achieving the required operational properties. The GSOC Plato library constitutes a solid base to implement an event-based, reliable, continuously running service and also offers a way to easily implement new and changed scheduling functionality upon changes in requirements or mission parameters. The split between the core scheduling service and the GUI for human interaction has proven to be beneficial, since their non-functional requirements and their way of operating are quite different. This way, they can also be e. g. developed by different people, or advanced at different rates or different phases of the project.

Second, for maintenance it is essential to have a software integration and deployment infrastructure that allows to roll out an updated version of the software easily, safely and fast. A semi-agile development scheme with several branches (for maintenance and feature updates) in the version control system, an automated build process directly creating installation packages, and a continuous integration process build the basis for LMS development. An extensive suite of automated test cases ensures that every build of the LMS contains the promised features and does not introduce regressions. Here, not only the basic software functionality, like processing data correctly and supporting interfaces in the right way, but also the required scheduling functionality is tested via running small scheduling scenarios. Updated versions of the LMS can be installed with minimal LMS downtime, not affecting the DPCC service, due to a solid persistence layer that allows the updated LMS to start in the state in which the previous version was stopped. In addition, the software can be rolled out in incremental stages, where important base functionality is implemented first and already used, whereas features that have lower priority or are scheduled later can be added afterwards. After being released, new LMS versions also undergo extensive integration testing, before being rolled out to the operational system. To this date, there were nine version upgrades of the LMS rolled out to the operational system, where three of them added new or missing functionality and the other ones were maintenance updates for fixing bugs or adapting to small changes in the operational concept.

Third, as the scheduling functionality and automation is almost fully implemented (also for the EDRS-C version of the LMS), still a fair amount of work needs to be invested in other areas of the LMS, mainly the LMS GUI. The focus is now shifting towards increased interaction possibilities for the operations personnel, allowing for better co-existence of manual operations with the automated scheduling and commanding process, increased possibilities of intervention and recovery in the case of anomalies, and improved operator awareness and information.

III. Incorporating the EDRS-C mission

A. Link Management System changes and additions for EDRS-C

The EDRS-A mission will be accompanied by a second geostationary platform named EDRS-C in 2019, also located over central Europe. The OHB-built platform hosts another LCT, thus doubling the capacity of EDRS for optical transmissions. While the MOC serves both missions and is able to schedule user requests on both platforms, the GSOC builds up another control center for EDRS-C and additional ground station antennas. For EDRS-C, also the platform is in the responsibility of GSOC, whereas the main difference on the payload side are the omission of the Ka-ISL antenna and the addition of an external hosted payload.

Since EDRS-C operates on a considerably newer and different platform, many of the scheduling requirements are changed in comparison to EDRS-A. Most notably, the EDRS-C bus has vastly more resources for operating ISL payloads, in particular a large TTC memory, and operates on TM/TC standards common to other modern missions at GSOC. But also the fact that the EDRS-C satellite is controlled directly by GSOC, as opposed to EDRS-A, where EDRS operations have to be routed via the Eutelsat control center, strongly influences e. g. timing requirements on the scheduling side.

The LMS implementation for EDRS-C needs to handle a new set of flight operations procedures and different scheduling algorithms and constraints for the routine maintenance activities of the payload. Due to the different communication between the platform and the LCT payload, it is possible to command the on-board orbit propagation updates and the clock timing updates directly to the platform, which in turn distributes the information to the LCT payload. The LMS schedules those maintenance activities on the platform side for automated execution, as opposed to the usual manual process on standard communication satellites, since they are now needed far more frequently due to the increased pointing and timing requirements the LCT payload imposes.

In detail, the main changes for EDRS-C are:

- Ka-ISL functionality shall not be offered, since this payload does not exist on EDRS-C. The respective scheduling functionality is removed from the LMS for EDRS-C.
- The updates for the on-board orbit propagator are commanded directly to the spacecraft bus. Since the propagation method implemented on board is superior to the one found on-board EDRS-A, OOP updates are only needed once per 24 h, as opposed to every 4 h. This accuracy is sufficient for both platform and LCT operations. In addition, the activation of updated parameters does not interfere with LCT operations. Therefore the LMS scheduling constraints are adapted, which also allows for simplified scheduling algorithms.
- Since maneuver planning is no longer done by Eutelsat but by GSOC instead, further cross-checks for consistency of the timeline—i. e. no forbidden activities take place within the maneuver boundaries—are possible and are

implemented in the LMS.

- The LMS can make use of a large on-board time-tag schedule buffer, which it also does not have to share with other entities. For EDRS-A, the same buffer has to be shared with Eutelsat platform operations and EDRS manual operations. For EDRS-C, separate sub-schedules are used independently.

Without Eutelsat in the loop and with a more capable platform, also the round-trip time for commanding the satellite and receiving its telemetry data will drastically reduce. This means additional budget for fast and time-critical automated operations. On the other hand, LMS performance becomes even more important than for EDRS-A in order not to defeat the possible performance gain from the quicker TM/TC chain.

B. Development strategy for LMS-C

It became clear very quickly that the changes required for EDRS-C should be implemented not as a separate planning system, but rather as a second incarnation of the same codebase. The LMS software is tailored to each of the two missions by configuration and dedicated scheduling algorithms and constraint definitions. The GSOC Plato library allows this by offering well-tested basic building blocks that can be combined into larger scheduling algorithms using a dedicated XML syntax, which makes the scheduling algorithms rather a part of configuration than code. The similarities in the two missions—most notably they use the exact same LCT payload—allows for efficient implementation of new features and fixes, benefiting both missions at the same time. In addition, almost all interfaces, both to the MOC and also GSOC-internal, are still present and common to both missions, further encouraging code re-use.

To allow for a common codebase, the LMS was set up to be built in two configurations, which toggle the code features and algorithms that shall be active in each flavor of the scheduling system. Also the unit tests were evaluated and extended, allowing for common (identical behavior for both missions) or mission-specific test cases. The build server was configured to always integrate the most recent code contributions into both LMS flavors and run the full test suite comprising the common automated tests plus the ones specific to the respective flavor. Also safeguards against processing input that is meant for the other mission were implemented to exclude mistakes during testing and setting up the operational processing chains. As opposed to many of today's public web services or smartphone apps, the LMS feature toggles are implemented such that by design they cannot be changed once the system is compiled. Figure 3 shows the current LMS architecture, which was extended from previous EDRS-A-only versions of the LMS.

IV. Lessons learnt and recommendations

As stated in Section II, developing a reliable, event-based and continuously operating scheduling system for the EDRS DPCC was very successful. Many of the design concepts of the LMS will also become manifest in the upcoming GSOC Reactive Planning Framework, which is currently under development and will first be used within the Mission Planning System of the EnMAP mission [8]. The experience gained from LMS operations serves as valuable input for the implementation of the Reactive Planning framework, e. g. regarding questions of timeline size and long-term performance, or how operators interact with and inspect a highly automated scheduling system.

Moreover, the LMS is being developed since 2014 in the context of the whole EDRS ground segment. Its main interface partners are the DPCC components dealing with automated commanding and the MOC [7]. From this experience, a number of best practices or recommendations can be drawn that go beyond the scope of software development of the individual LMS component and reach out to the system engineering process for the whole ground segment and also for planning and scheduling systems of other missions.

The first observation is that the splitting of planning and scheduling functionality between different components—here the MOC and the LMS—leads to some challenges. If this system architecture is required, needed or desired, one has to ensure a clear boundary between the component responsibilities and tight interfaces. Usually, a single planning instance will hold all information needed for the complete scheduling process more efficiently, sparing also the need for additional interfaces. With two planning instances, inconsistencies between them need to be avoided and potential conflicts need a defined way of resolving. An upcoming CCSDS standard will be of high value for the exchange of scheduling information between two planning instances.

One concrete example of this is the scheduling of O-ISL activities in concurrency with OOP parameter uploads, which may not happen simultaneously on EDRS-A. The knowledge of the O-ISL requests from the customers lies at MOC and is only transferred 8 h before link start to DPCC, in order to allow the LMS an upload on time, while the OOP uplink schedule is exclusively maintained by the LMS. Therefore, overlaps of those two activities often occur and have to be removed by re-scheduling the OOP upload, which regularly needs to also be propagated to the satellite by removing telecommands and uploading different ones. On the other hand, informing the DPCC much earlier about customer

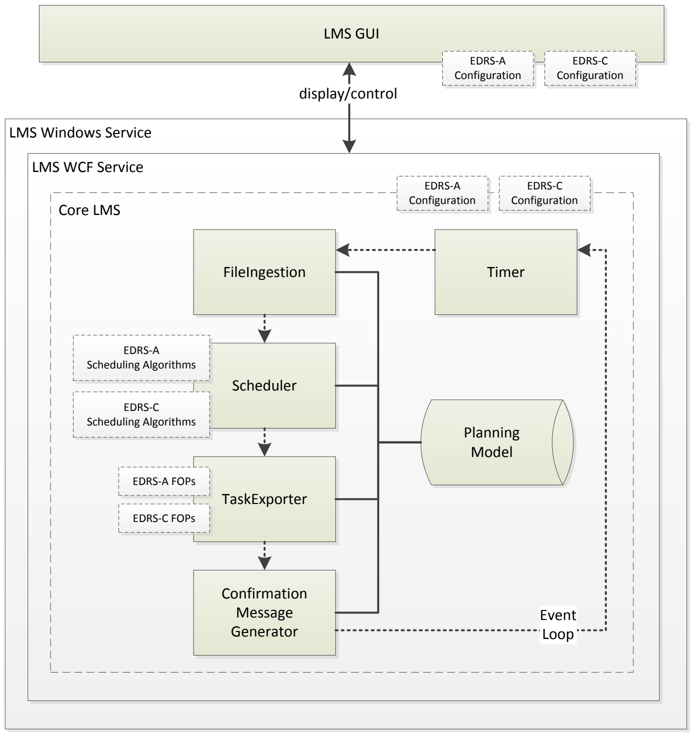

Fig. 3 LMS architecture from [1], extended for EDRS-C: During compilation, the target mission (EDRS-A or EDRS-C) is set. This determines which features and configurations are active and which scheduling algorithms, constraint definitions, and flight operations procedures the Scheduler and TaskExporter components use.

O-ISL requests would mean that potentially many modifications of the O-ISL timeline would need to be transmitted from MOC to DPCC. The solution is to carefully balance the involved timings so that the DPCC scheduling process considers only the timeframe which was already transferred to it by the MOC in a way conforming to the mission requirements.

Another observation at EDRS and also many other missions is that cooperation between the engineering processes for on-board and on-ground hardware and software can potentially benefit the whole system by increasing the operability. The interfaces and telecommands the spacecraft offers directly affect the complexity and maintainability of the on-ground software.

For example, the functionality to control the steering of the Ka-band inter-satellite antenna of EDRS-A dictates that the uploaded telecommands also depend on the current state of the payload. This leads to a complex modeling that is needed on-ground to fulfill this stateful interface. Additionally, the state of the payload is only visible in very limited form within the telemetry, making it difficult to align the on-ground model to the full on-board state in case of software problems or anomalies.

V. Conclusion

The Link Management System for EDRS-A has been in operation for two years and performs automated continuous scheduling functionality for the control center of the EDRS-A payload. The developed software has proven to be very reliable and robust and also to be modifiable in a safe and efficient manner. The GSOC Plato library and its basis, the GSOC modeling language, together with the chosen software architecture allowed for the needed flexibility to handle changing requirements and shows the necessary performance to cope with the expected loads of the mission. Valuable experience has been gained from the first years of operations and points that still need development work were identified, especially in order to improve interaction with operations personnel and their situational awareness.

Several new LMS software versions were released, tested and rolled out during the initial years of operations and a few anomalies stemming from LMS problems could be handled without affecting the operational mission. The required availability and robustness was achieved and contributes to the success of fulfilling the required DPCC availability of more than 99.5%. Also the codebase was upgraded to support continuous uninterrupted operations over the whole mission lifetime while keeping the desired performance. The LMS is now regularly kept running continuously for several months in the operational system without being restarted.

Meanwhile, also functionality needed for EDRS-C was incorporated into the LMS codebase. The chosen implementation approach is the use of a common codebase for both scheduling systems, where the necessary differences are switched via feature toggles. Therefore, both flavors of the LMS benefit from bugfixes and improvements alike. For safety, the feature toggles cannot be configured during run-time, but are fixed at compile time. The LMS codebase is now ready for being rolled out to the EDRS-C spacecraft control center and equally successful operations.

Abbreviations

DPCC	Devolved Payload Control Center (of EDRS-A)
EDRS	European Data Relay System
FOP	Flight Operations Procedure
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
O-ISL	Optical Inter-Satellite Link
SCC	Spacecraft Control Center (of EDRS-C)
TM/TC	Telemetry / Telecommand
TTC	Time-tagged Tele Command

References

[1] Göttfert, T., and Grishechkin, B., and Wörle, M. T., and Lenzen, C., “The Link Management System of the European Data Relay System”, *SpaceOps 2016 14th International Conference on Space Operations*, 16–20 May 2016, Daejeon, South Korea

[2] Scharringhausen, J.-C., and Beck, T., “Automated Procedure Based Operations for the European Data Relay System”, *68th International Astronautical Congress (IAC)*, 25–29 September 2017, Adelaide, Australia

[3] Hauschildt, H., et al., “European Data Relay System - one year to go!”, *Proc. International Conference on Space Optical Systems and Applications (ICSOS)*, 7–9 May 2014, Kobe, Japan

[4] Wörle, M. T., et al., “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

[5] 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

[6] “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)

[7] Beck, T., Schmidhuber, M., and Scharringhausen, J.-C., “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] Fruth, T., Lenzen, C., Gross, E., and Mrowka, F., “The EnMAP Mission Planning System”, *SpaceOps 2018 15th International Conference on Space Operations*, 28 May–1 June 2018, Marseille, France (submitted for publication)