Robust Commanding

Tobias Göttfert* Christoph Lenzen* Maria Theresia Wörle* Falk Mrowka†
Martin Wickler‡

Deutsches Zentrum für Luft- und Raumfahrt e. V., German Aerospace Center
Münchener Straße 20, 82234 Weßling, Germany

In this paper we present Robust Commanding as a new method that mission planning systems can implement to improve the reaction of the planning system to unsuccessful commanding. This method can be used to improve upon flexibility and reaction time of the mission planning while maintaining a safe commanding concept that avoids gaps in the mission timeline. The prerequisites are highlighted and the method is presented and exemplified on the basis of commanding low-earth orbiting satellites. The necessary commanding interfaces are discussed and an outlook for application of this method to future satellite missions is given.

I. Introduction

Current and future satellite missions, especially for low-earth orbiting satellites, strive to improve on the planning and commanding flexibility to satisfy increasingly challenging user requirements. For low-earth orbiting satellites, the time windows when commands can be sent to the spacecraft are restricted to the times of visibility with respect to the used ground station(s). Typical requirements on a newly developed mission planning system (MPS) include the desire to adapt the planning rhythm to changing patterns of command uplinks for increased flexibility of booking ground station contacts, the possibility to place a planning request on short notice directly before an upcoming uplink opportunity, and the inclusion of near-real-time (NRT) features, such as commanding, executing and downlinking a data take in the very same contact while still using the contact for uplinking the commands of the nominal mission. These requirements have to be fulfilled without compromising the safety of the spacecraft and the consistency of the mission timeline. On the MPS side, the latter translates into two direct requirements: The MPS shall always create consistent conflict-free timelines that can be executed on the spacecraft. On the other hand, the MPS shall be robust enough to cope with errors during commanding, e. g. an early loss or a late acquisition of signal (early LOS, late AOS), without creating too much outage in the desired timeline.

We present a new approach of fulfilling these requirements, called *Robust Commanding*, that enables the MPS to reduce time buffers and safety margins and thus implement increased flexibility in the planning process. This will be used in conjunction with the *Incremental Planning* functionality[1] currently under development at GSOC.

II. Goal and Prerequisites

The goal of Robust Commanding is to make typical safety mechanisms superfluous, that are presently introduced to fulfill the two aforementioned requirements at the cost of planning flexibility. A current example is the combined TerraSAR-X/TanDEM-X mission,[2,3] where routinely two adjacent uplink contacts of the same ground stations are used as a so-called *uplink session* to ensure a complete uplink of the planned commands. In this scheme, the *earliest commanding* time—the point in time from which the on-board timeline may be modified—lies after the second uplink contact, while the time frame in between the two

*Mission Planning System Engineer, Mission Operations department, German Space Operations Center
†Head of Mission Planning Team, Mission Operations department, German Space Operations Center
‡Deputy Head of Mission Operations Department, Mission Operations Department, German Space Operations Center

contacts is unmodifiable. The second mechanism is that not only the so-called *critical* time interval to the next uplink session is commanded, but also the *desired* time interval up to the succeeding uplink session. Therefore, a dropout in one uplink session allows to still execute an older version of the mission timeline that was commanded one uplink session before. However, the MPS needs to be able to modify a timeline on the spacecraft that was already commanded as part of the desired time interval of the preceding uplink session. Secondly, a failure in commanding the critical time interval also leads to an inability to command the desired time interval, since the commands in the desirable interval depend on the state of the timeline on-board that was created by the commands of the critical interval.

Robust Commanding gets rid of these limitations and allows a much more agile planning scenario, which is described in the following. However, the presented method only works if two key prerequisites are fulfilled: First, the system to be commanded must support some sort of *commit concept*, i. e. a set of instructions is either applied as a whole or not at all, similar to the atomicity of a database transaction. Second, a project-specific mission planning software has to provide a repair/timeline correction algorithm which covers situations where one or multiple command uplinks failed. The application implementing Robust Commanding—we call it *Timeline Manager*—then uses this distinct interface and in return prepares the best possible target timelines for all possible failure scenarios that are to be used in the respective situation. From these target timelines and the currently commanded timeline, a project specific command export functionality derives the command sets to be used for uplink in the corresponding uplink scenario.

Although the project-specific effort of writing a repair/timeline correction algorithm is not negligible, the benefit for the mission may be immense: From the mission planning point of view, all uplink failures are then part of the nominal mission scenario, so no manual interaction is required when an uplink failure occurs and the mission will not be interrupted. Although the timeline which will be executed after an uplink failure may not include the latest modifications of the on-ground calculated timeline, it still consists of the consistent, latest timeline available on board. Furthermore, due to the fact that such an uplink failure is a well prepared nominal scenario, the mission may define challenging uplink scenarios, such as commanding modifications of the timeline that shall already apply during the very same ground station passage in which they are uplinked. In such a case late AOS would be supported by automatically sending the timeline that would have been used without support for in-pass modifications. Another challenging scenario would be the use of not fully reliable ground stations, e. g. ground stations which shall be used for command uplink on short notice without proper testing.

III. Robust Commanding Method

The main idea behind Robust Commanding is that every command uplink may fail and that the mission planning system can prepare for this case ahead of time. The atomic commit mechanism on the spacecraft ensures that a safe and consistent version of the timeline is executed, whether uplinking one set of commands was successful or not. The task of the MPS is then to ensure that all uplinked command sets create consistent and executable timelines, and that commanding may stop after each set. However, the order of command sets has to be preserved, since they do not commute. As an example, consider the switch-off of an instrument at the end of one command set, to bring it into a safe state if commanding stops after this set, and the deletion of this switch-off command at the beginning of the next set, to allow for continued operation across command set boundaries. Obviously, commuting these two commands, i. e. first deleting the (non-existent) switch-off and afterwards commanding the switch-off, does not have the desired effect.

The interface that the project-specific repair/timeline correction functionality shall present is the following: Given an *initial timeline*, i. e. the timeline which reflects the commands on board the satellite(s), the *master timeline*, i. e. the on-ground timeline covering the whole planning horizon, and the *export horizon*, the repair algorithm creates the *target timeline*, based upon the initial timeline by incorporating the activities of the master timeline during the export horizon. After the export horizon, activities that are required to make the timeline consistent and safe are also taken over into the target timeline from the master timeline. Further additional activities that are not part of either of the two input timelines may also be included for the same reason, e. g. switch-off commands for on-board instruments after the end of the export horizon. During nominal operations, these additional commands are deleted by the succeeding command upload, but they ensure a safe state of the instrument if the succeeding upload fails.

This timeline correction interface is used twofold in the Robust Commanding method: First, for “forward-correcting” the master timeline into the target timeline, and second, for “backward-correcting” the on-board timeline into the master timeline in case a desired command uplink failed.

III.A. Example with one satellite

III.A.1. Separated uplink contacts

The method is best explained by starting with a simple example with one satellite and a set of clearly separated uplink contacts, as depicted in Fig. 1.

The mission planning system has the knowledge of the complete on-ground timeline prepared by the planning process, called master timeline, and the state of the timeline on-board, called *commanded timeline*, which is usually only uplinked with a limited time horizon. The Timeline Manager begins by using the timeline correction functionality to "blend" the master timeline into the commanded timeline during the interval between the earliest commanding of the first and second uplink contact. The result is the first target timeline. Next, the TimelineManager may create a target timeline for the time in between the second and the third uplink contacts' earliest commanding times. For this purpose, it calls the timeline correction functionality again, however this time the commanded timeline is replaced by the first target timeline, which is expected to be the commanded timeline when this second uplink takes place. Depending on the number of uplink contacts that shall be commanded in advance, this procedure may be repeated in order to generate further target timelines.

The project specific command generation functionality is then used to generate the actual command sets to the spacecraft by comparing the resulting timelines pairwise, i. e. commanded and first target timeline, first and second target timeline, second and third target timeline, . . . , so that the n-th command sets transforms the on-board state of the timeline from the previous state to the n-th target timeline, when sent sequentially. The result is that uplinking the command sets can be stopped at any time in between or during the command sets, while due to the commit concept and the set of individually consistent target timelines the on-board timeline remains consistent.

The new commanded timeline after this uplink contact is then simply the target timeline that belongs to the last command set that succeeded during uplinking. If not even the first command set could be uplinked, the master timeline on ground needs to be corrected a posteriori with the contents of the on-board commanded timeline within the timeframe EC1–EC2. However, in the case of separated uplink contacts, the time until the upcoming contact is assumed to be sufficient for this.

III.A.2. Combined uplink sessions and in-pass updates

In many cases, the uplink contacts are not separated by a sufficiently large duration, so that re-running scheduling algorithms and performing command set generation cannot be performed when an uplink failure occured. The uplink contacts are then grouped into an *uplink session*, and traditionally the combined earliest commanding of the session would be put at the end of the last uplink contact of the session, to allow for a safe commanding should some of the contacts fail. The downside of this solution is that late modifications after the first uplink contact cannot be incorporated into the timeline anymore and no modifications of the timeline are possible until the combined earliest commanding is reached.

A special case of this situation are in-pass updates of the timeline. One may split up contacts into two parts: During the first part only commands are uplinked that update the on-board timeline during the rest of the contact. The second part of the contact is used for uplinking all timeline updates taking place after the contact, while in parallel the execution of the in-pass timeline is already ongoing. For this, it is necessary that commanding starts very early during the contact to the ground station, which is error-prone. At the same time, commanding the rest of the timeline should still be possible if uplink of the in-pass modifications has failed or could not be started in time due to a late AOS event.

Robust Commanding handles both these situations by preparing for all possible failure scenarios beforehand and then choosing "fallback" command files when the need arises. Let's consider the case of in-pass updates of the timeline: the contact is split into two phases that are grouped into one uplink session: The in-pass uplink contact and the uplink contact for the rest of the timeline. In Fig. 2, these would be the part of pass 1 before EC1, where the timeline updates within the time interval EC1–EC2 shall be uplinked, and the part of pass 1 after EC1, where the timeline updates within the time interval EC2–EC3 shall be considered. The Timeline Manager prepares three command files in total, where the first command set 1 is for the uplink of the in-pass modifications only. For the second set, there are two versions prepared: a nominal one in case the

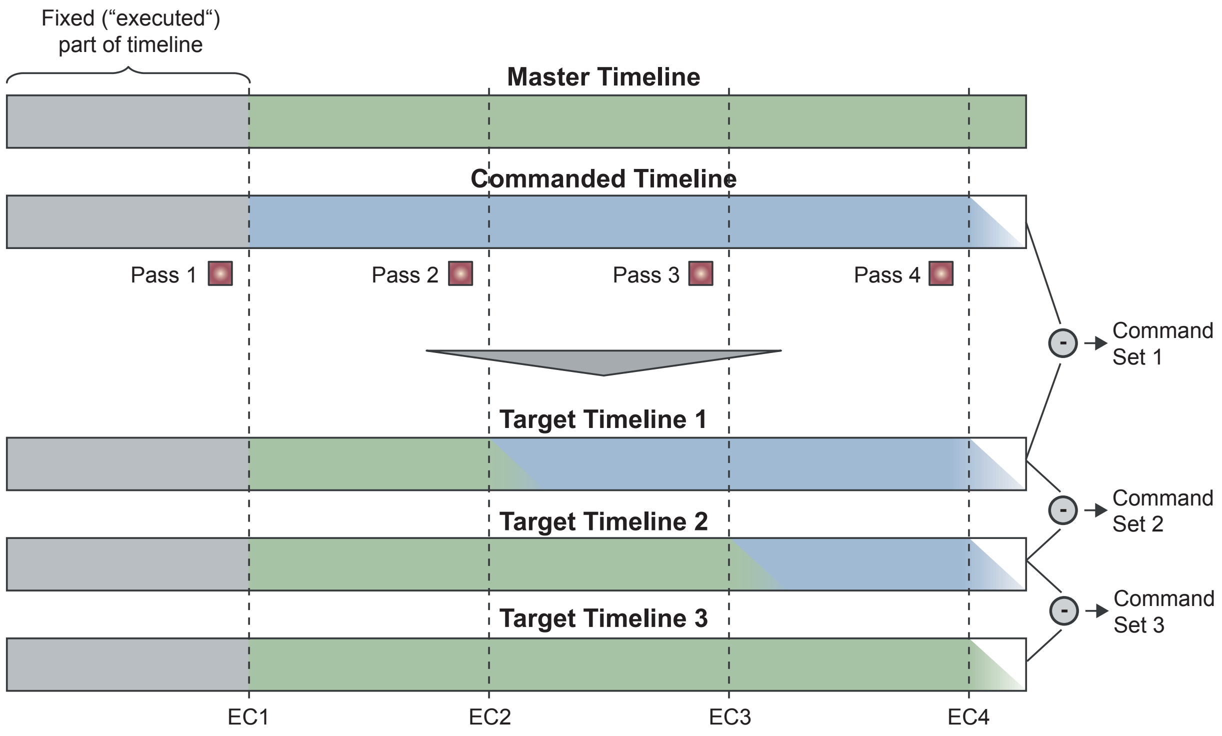

Figure 1. Creation of the target timelines and command sets for a group of separated uplink contacts ("passes"). Each contact defines its own earliest commanding time (EC) and the target timelines span an increasing number of ECs. The command sets are generated by pairwise comparison of subsequent target timelines.

in-pass uplink succeeded (command set 2), and a fallback set (command set 2′) in case the in-pass uplink failed.

As before, target timeline 1 is created by integrating the master timeline between EC1 and EC2 into the commanded timeline, possibly adding further commands after EC2 for consistency and safety reasons. Target timeline 2 takes over the master timeline, restricted to the timeframe EC2–EC3, provided the commands of target timeline 1 have been uplinked. For the fallback timeline 2′, one needs to take into account that the uplink of command set 1 is not successful; i. e. before using the master timeline, it has to be updated and corrected by integrating the commanded timeline into the master timeline between EC1 and EC2. Afterwards, the *corrected master timeline* can be used to generate the target timeline 2′ by integrating it into the commanded timeline between EC2 and EC3. The alternative command set 2′ then has to be created by taking the difference between the commanded timeline and target timeline 2′, since in this scenario, the upload of command set 1 has failed.

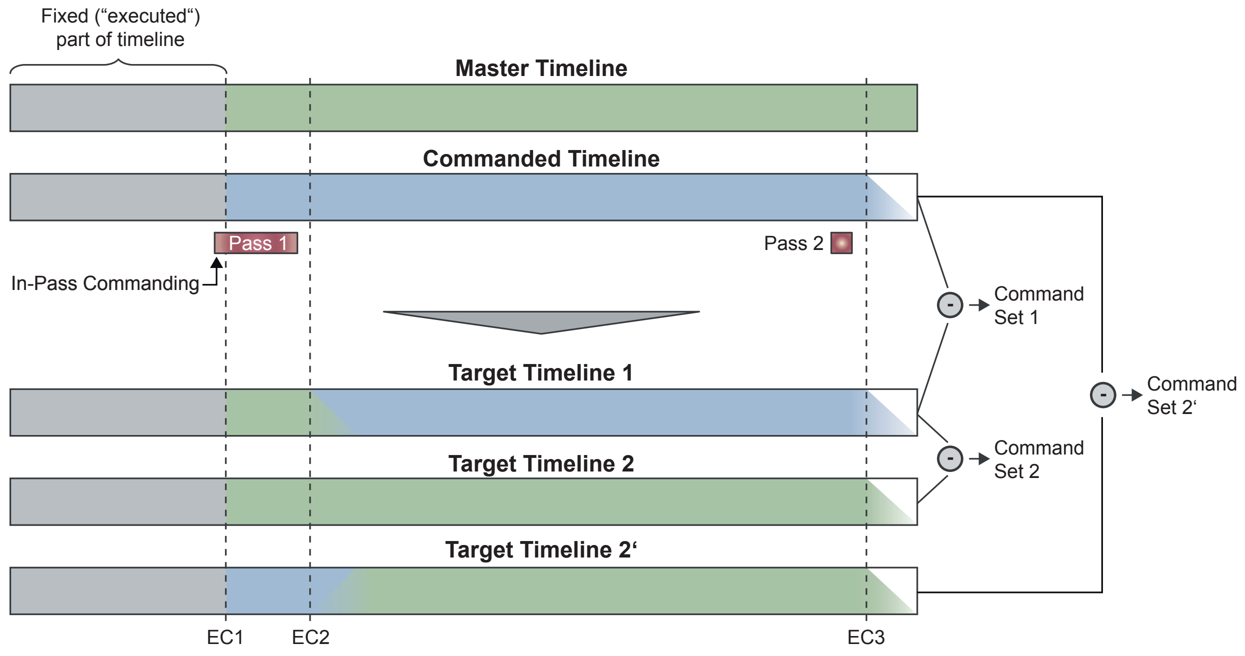

Figure 2. Creation of alternative command sets for in-pass updates. Command set 1 only applies in-pass updates, while command set 2 updates the timeline until the next uplink contact. The target timeline $2'$, which serves as a fallback timeline if uplinking command set 1 fails, is created by correcting the commanded timeline between EC1 and EC2 into an updated master timeline and then performing the nominal target timeline creation between EC2 and EC3. The command set $2'$ then has to reflect the differences between the commanded timeline without in-pass modifications and target timeline $2'$.

III.B. Example with two satellites

Modern satellite missions often incorporate more than one spacecraft flying in a constellation or even close formation. The MPS for such a satellite system might then use a single combined planning model in order to e. g. distribute the load on the two or more spacecraft evenly or to incorporate dependencies between the spacecrafts. As an example, the TerraSAR-X/TanDEM-X mission needs a combined planning model, since the two satellites fly in very close formation and in particular their radar instruments need to be synchronized for taking interferometric radar images.

Fig. 3 shows an example scenario where two satellites are to be commanded by a single MPS. The uplink session that is considered comprises three consecutive uplink passes that do not overlap. Let's assume that in between the uplink passes there is enough time to decide whether the previous command uplink was successful. The passes and their earliest commanding are in the following written with an index composed of satellite and pass, e. g. P_{12} denotes the second pass of satellite 1. The Timeline Manager would then perform the following steps and create seven command sets for all possible uplink success scenarios:

1. **Create target timeline 1 for pass P_{11}**
   by incorporating the activities of satellite 1 from master timeline into commanded timeline between EC_{11} and EC_{12}.

2. **Create target timeline 2 for pass P_{21}**
   by additionally incorporating the activities of satellite 2 from master timeline into commanded timeline between EC_{21} and EC_{22}.

   (a) **Create target timeline 3 for pass P_{12}**
   by additionally incorporating the activities of satellite 1 from master timeline into commanded timeline between EC_{12} and EC_{13}.

   (b) **Create target timeline $3'$ for pass P_{12}, assuming that P_{21} has failed**
   Therefore, the master timeline that is used has to be corrected with the commanded timeline of satellite 2 between EC_{21} and EC_{22} before generating the target timeline. The project-specific

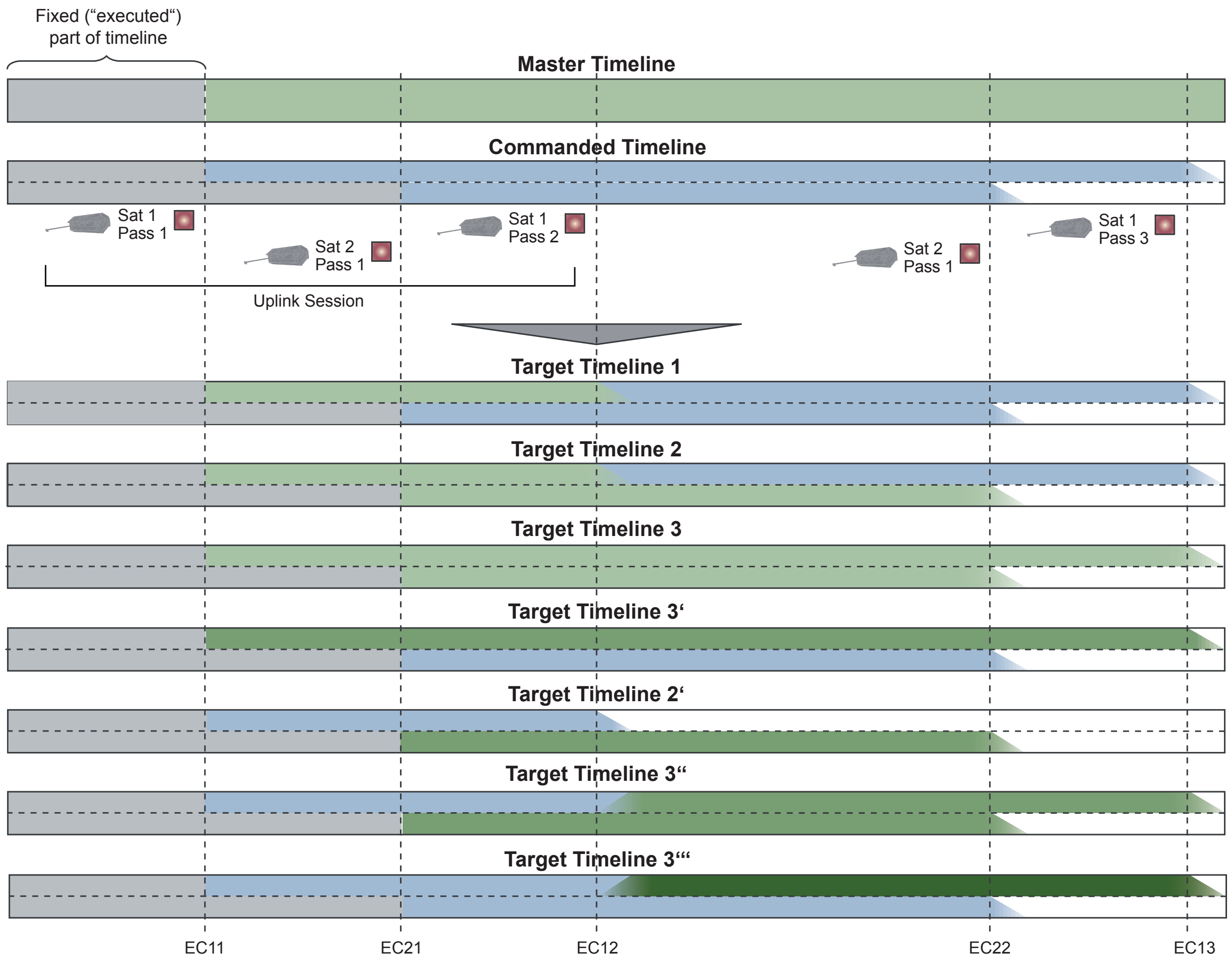

Figure 3. Creation of the seven target timelines for three uplink passes of two different satellites within one uplink session. Subsequent target timelines are created by forward-correcting increasing parts of the master timeline into the commanded timeline, uplink failures are considered by backward-correcting the parts of the commanded timeline that remained on-board into the master timeline.

timeline correction algorithm is now in charge to correct the master timeline in the best possible way for this situation. It may happen that conflicts remain on activities which need synchronization with satellite 1. Nevertheless conflicts may be resolved beginning with EC_{12}.

3. **Create target timeline $2'$ for pass P_{21}, assuming that P_{11} has failed**
by taking over the commanded timeline of satellite 1 into the master timeline up to EC_{12}. Again, a project-specific timeline correction algorithm needs to correct the adapted master timeline in order to remove conflicts in between activities on the different satellites which have to be synchronized. The result is then used to generate the target timeline $2'$ as in step 2. Note that the previously commanded timeline is conflict free, therefore the correction algorithm should be able to remove all conflicts on the adapted master timeline.

   (a) **Create target timeline $3''$ for pass P_{12}**
   by additionally incorporating the activities of satellite 1 from master timeline into commanded timeline between EC_{21} and EC_{22}.

   (b) **Create target timeline $3'''$ for pass P_{12}, assuming that also P_{21} has failed in addition to P_{11}**
   by correcting the master timeline with the commanded timeline for satellite 2 up to EC_{22}, too. The corrected master timeline can then be used to generate the target timeline, as in step 2a.

The seven command sets would then be generated by comparing the appropriate pairs of target timelines, concretely:

- commanded timeline – target timeline 1 for command set 1
- target timeline 1 – target timeline 2 for command set 2
- target timeline 2 – target timeline 3 for command set 3
- target timeline 1 – target timeline $3'$ for command set $3'$
- commanded timeline – target timeline $2'$ for command set $2'$
- target timeline $2'$ – target timeline $3''$ for command set $3''$
- commanded timeline – target timeline $3'''$ for command set $3'''$

Additional complication arises when one does not have sufficient time to decide whether the previous uplink was successful or not. This can easily happen, an example would be two spacecraft with overlapping uplink contacts, as it is the case for TerraSAR-X/TanDEM-X. The two are flying in a close formation and are commanded via ground stations that are located close to each other. During the uplink for one satellite one cannot yet know whether the command uplink of the other satellite that is happening in parallel will be successful. In this case, a successful command uplink of the other satellite must be assumed. For preparing the case that only one of the uplinks fails, the Timeline Manager once again calls the project-specific timeline correction algorithm, which adapts the master timeline in the best possible way, assuming that the next uplinks succeed. For the case that both uplinks fail, e. g. in case the failure can be detected in time to break-off the commanding of the other satellite, the timeline correction algorithm will be able to return a conflict free timeline, as long as the commanded timeline has been consistent.

III.C. General formulation

As it can be seen from the given examples, a general formulation of the tasks of the Timeline Manager can be given that covers all possible failure scenarios and the possible knowledge of the MPS about the success or failure of the command uplinks. It is then for the specific mission to decide, which cases the MPS shall support, and to provide the necessary repair/timeline correction mechanism, since this is strongly dependent on the mission planning model. Depending on the mission, also not every combinatorial case might need to be supported by the Timeline Manager, which may greatly reduce the computational effort the MPS has to fulfill.

The general formulation of the Timeline Manager's algorithms shall be given now. Using the mission-specific repair algorithm that is needed as a prerequisite, the task of the Timeline Manager is to prepare corrected versions of the timeline that cover all possible failure scenarios and can be applied without the need for additional scheduling runs.

For the first uplink pass the target timeline is created by

- using the current master timeline,
- using the commanded timeline of the satellite to be commanded, and
- calculating the first target timeline from this input via the project-specific timeline correction algorithm (in order to assure the commanded timeline on-board the satellite is complete and conflict-free when no further commanding takes place).

For the (n+1)th uplink pass, the Timeline Manager has to consider all possible uplink scenarios of the n-th uplink pass. For each of these scenarios, the target timeline is created by

- using the master timeline as generated by the assumed uplink scenario of the n-th uplink pass,
- using the target timeline of the assumed uplink scenario of the n-th pass as new commanded timeline, and
- calculating the (n+1)th target timeline from these two as before.

Unfortunately the uplink scenarios do not restrict to *i-th pass has passed/failed*. Instead, one has to determine what commands have been sent, which depends on the knowledge about failure of preceding uplinks at the point in time of the command uplink. The Timeline Manager therefore has to determine:

- which preceding uplinks are considered to have succeeded
- what commands have been part of the command set of the successful uplinks, which in turn depends on the same criteria, restricted to the knowledge when performing that preceding uplink

The set of uplink scenarios—i. e. all possible uplink success assumptions for every step during creation of the uplink passes—can be noted in a triangular matrix form with possible entries 0 and 1. Consider the following example for an uplink scenario in the case of four uplink passes within a session:

$$\begin{array}{ccc} 1 & & \\ 1 & 1 & \\ 0 & 1 & 0 \end{array}$$

The first line denotes that in this scenario, the command set for the second uplink pass is prepared assuming that uplink pass 1 has succeeded. Analogously, the second line tells that the command set for pass 3 is prepared assuming that the first and second uplink passes have succeeded. Only before uplink pass 4 (third line in matrix), it becomes known that uplink pass 1 has indeed failed (while pass 2 was commanded unaware of this fact) and that uplink pass 3 has failed as well. So the command set for uplink pass 4 is prepared using the commanded timeline before uplink pass 1, modified by the commands of the second pass, which have been generated under the assumption that pass 1 has succeeded. However the modifications of pass 1 have not succeeded, so they are not part of the commanded timeline to be considered by this case. Before deriving the target timeline, the master timeline is adapted to the current assumption, i. e. the commanded timeline generated this way is taken over into the master timeline up to the earliest commanding of the currently considered uplink pass and the timeline correction algorithm is called in order to make the best out of this inconsistent state.

In general, for n uplink passes, the triangular matrix has $n-1$ rows and columns, yielding $2^{(n^2-n)/2}$ possible uplink scenarios. For the given example of four uplink passes, this would already mean that the Timeline Manager has to calculate $\sum_{i=1}^{4} 2^{(i^2-i)/2} = 75$ target timelines and command sets, calling the timeline correction function more than a hundred times in the process.

However, there exist some simplifications that can be made to reduce the amount of calculations that need to be done to some extent without loss of generality. For example, consider the last line of the matrix consisting of only zeros, i. e. *all* uplinks are considered to have failed before the last one. It can be seen immediately that the content of the other lines is irrelevant, since no commands were uploaded to the spacecraft at all, making the history of assumptions about command success meaningless. Therefore only one common command set has to be prepared for those scenarios.

If the resulting computational effort is still too high, several restrictions can be applied to reduce the number of cases that need to be considered:

Assuming reports about uplink failures are correct: If the report about a failed command uplink is never revised, which normally should be the case for a satellite control system, all matrices containing ones below zeroes in the same column can be ignored. The number of cases that needs to be considered in this case reduces to $n!$.

No preparation of alternative command sets: If the previous simplification still results in too much calculation effort, one can resort to stopping the commanding once the first failure becomes known. Even if no alternative command sets for these cases are then generated, the state of the satellites will be known and the MPS can consider it for future scheduling. The number of cases that needs to be prepared with this restriction reduces to 2^n.

Uplink success of previous passes is known: For special cases it may also be possible to restrict to cases in which the uplink success of certain previous passes is known. An example would be the splitting of uplink passes into in-pass and nominal commanding. Here the nominal command set would only be uplinked if the in-pass commanding would be successful, otherwise an alternative command set would be used. The decision whether the in-pass commanding is successful would be taken on-line and not revised afterwards. Also in case of a multi-satellite system, one might require that the uplink success is known for subsequent passes of the same satellite. For these simplifications, the reduction of combinatorial cases depends on the actual commanding concept.

IV. Interface to the Command System

To make use of the uplink scenarios created by the Timeline Manager, two interfaces between the commanding system and the MPS are needed: First, the commanding system has to be able to receive all command sets for the various uplink scenarios. A Robust Commanding logic has to be implemented such that the commanding system is able to choose the correct alternative command sets in case of uplink failures. Second, after the complete uplink session is handled, the command system has to give feedback to the MPS about which of the uplink scenarios has occurred in reality. The MPS can then use the target timeline of the realized scenario as the new commanded timeline for future scheduling.

These functionalities will be provided as one component of the GSOC Incremental Planning system, so that automatic commanding systems and command operators can interact with the MPS via a well-defined (user) interface.[1]

V. Conclusion and Outlook

Robust Commanding is a method of preparing for possible uplink failures already within the mission planning software. This reduces the time that is lost for the mission, should an uplink failure occur, and at the same time allows for challenging and dynamic uplink scenarios. Robust Commanding is able to handle a variety of use cases, e. g. NRT applications with in-pass updates, constellation and/or formation geometries of more than one satellite with or without overlapping ground station contacts, or the consistent continuation of planning after failed uplink contacts, even if the failure becomes known after additional uplinks were conducted already. However, the timeline can only be guaranteed to remain conflict-free with respect to the activities of each satellite individually. Since Robust Commanding relies on the ability of the spacecraft for atomic command set uplink, it is highly desirable that as many spacecraft as possible implement this interface in the future, so that on-ground planning and commanding systems can exploit the additional agility and safety.

Robust Commanding is developed at GSOC in the frame of the Incremental Planning[1] project and therefore will be part of upcoming mission planning systems that use the Incremental Planning concept, as long as the spacecraft support it. However, the Robust Commanding method does not depend on the Incremental Planning system and could also be used in other mission planning systems. The first mission to use Incremental Planning, including a Timeline Manager that implements Robust Commanding, is foreseen to be EnMAP.[4]

Glossary

uplink contact One possibility to command the spacecraft/executing unit, e. g. one ground station contact.

uplink session A set of associated uplink contacts that is characterized by the fact that their time distance is not sufficient for performing additional scheduling and command set generation in between, so that they shall be handled during a single command set generation.

master timeline The set of activities that is the outcome of the planning process, i. e. the full state of the timeline on ground.

commanded timeline The set of activities that was already commanded to the satellite(s) during previous uplink contacts, i. e. the state of the timeline(s) on board.

target timeline A modification of the on-board timeline that includes activities from the on-ground timeline that shall be commanded to the spacecraft during the next uplink session.

uplink scenario A set of assumptions about the success of all uplink contacts within one uplink session.

earliest commanding The earliest point in time from which on modifications of the on-board timeline of a spacecraft are allowed.

References

[1]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”, *SpaceOps 2014 13th International Conference on Space Operations*, 5–9 May 2014, Pasadena, California

[2]Krieger, G., Moreira, A., Fiedler, H., Haynsek, I., Werner, M., Younis, M. and Zink, M., “TanDEM-X: A Satellite Formation For High-Resolution SAR Interferometry”, *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 45, Number 11, pp. 3317–3341, November 2007

[3]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

[4]Storch, T., Habermeyer, M., Eberle, S., Mühle, H., and Müller, R., “Towards a critical design of an operational ground segment for an Earth observation mission”, *J. Appl. Remote Sens.*, 7(1), 073581, Mar 07, 2013, http://dx.doi.org/10.1117/1.JRS.7.073581