Adjustable Synchronization: A Proposal for Next Generation Space Telescope Operations

Gary Welter[1], Jim Legg[2], and Glenn Cammarata[3]

1. Computer Sciences Corporation, 7700 Hubble Drive, Lanham/Seabrook MD 20706, gary.welter@gsfc.nasa.gov
2. Raytheon, Code 582, Goddard Space Flight Center, Greenbelt MD 20771, jim.legg@gsfc.nasa.gov
3. National Aeronautics and Space Administration, Code 582, Goddard Space Flight Center, Greenbelt MD 20771, glenn.cammarata@gsfc.nasa.gov

Abstract

The Next Generation Space Telescope (NGST), planned for launch in 2007, poses a number of interesting engineering challenges. This article addresses one such challenge, specifically that of maximizing observatory efficiency through an appropriate balance between non-real-time construction of the near-term observing schedule and onboard adaptation to real-time conditions during actual observation execution. The appeal of onboard "adaptive scheduling" is particularly strong for NGST because of its expected placement near the second Earth/Sun Lagrange point, a comparatively benign location vis-à-vis scheduling constraints. This is expected to make event-driven, in contrast to time-tagged, execution the norm for most observations. However, experience with previous missions has demonstrated to science operations teams that it is very useful for the team to be able to have substantial control of schedule construction, particularly when the mission is yet young and the team is first learning how to effectively use the satellite and science instruments. In this article, we explore a hybrid scheme designed to provide the advantages of real-time adaptive response under most circumstances, while at the same time giving the operations team the level of control perceived as appropriate for current mission conditions.

Introduction

The Next Generation Space Telescope (NGST), projected for launch in 2007, is a key component of NASA's origins program (cf. Stockman 1997). Its principal purpose is to enable studies of the cosmological "dark ages" at times and distances from just beyond that probed by the Hubble Space Telescope to near the "recombination" epoch studied by the Cosmic Background Explorer.

Current expectations are that NGST will be a large aperture (~ 8 m) infrared observatory located near the second Earth/Sun Lagrange point (L2). The technological challenges for creating the required hardware systems are substantial, particularly in areas such as assembly and figure control of the segmented primary mirror, cryogenic operations of complex space systems, and construction and rigidification of large, low weight space systems.

Placement of the observatory near L2 offers certain very appealing features with respect to operations. In particular, it eliminates all of the moderate frequency orbit-related constraints that plague missions in low Earth orbit, including frequent target occultation by the Earth, increased electronics interference during passage through the South Atlantic Anomaly, and thermal and power stresses related to regular passage through the Earth's shadow. There does remain a solar avoidance constraint prohibiting observations of targets too close to the sun, but this occurs with a yearly, rather than near hourly, time scale. Additional advantages related to L2 placement, at least with most of the proposed NGST designs when science attitude constraints are met, include (1) abundant and continuous solar power, and (2) continuous viewing of the Earth by the antenna communications system.

As part of the general engineering effort towards creating NGST, a number of integrated product teams (IPTs) have been established to investigate various aspects of the challenge. One of the mandates of the *Operability* IPT has been to investigate strategies whereby general observatory operations can be made easier and more cost effective. Leveraging off the low-constraint nature of the L2 environment, a number of groups have proposed the development of an onboard *adaptive scheduler* that, at a minimum, would apply an event-driven strategy for execution of most observations (cf. GSFC & TRW NGST pre-phase A study teams, 1996). We, the authors of this paper, are part of a team supporting the Operability IPT through studying various possible realizations of adaptive scheduling for NGST.

As part of our exploration, one line of investigation (Welter 1998a, b, c) attempted to define an algorithm appropriate for supporting full onboard near-term scheduling of science and general "house-keeping" activities. A second line (Welter and Legg, 1998; hereafter, Ref. 1) began with the goal of placing most near-term scheduling in a ground-based system, the principal driver for this track being the realization that on previous missions science operations teams (SOTs) have found it desirable and important to retain fairly direct control of mission planning and scheduling. This need is particularly clear when the mission is young and the SOT is first learning how to use the satellite and science instruments effectively. The team then needs considerable flexibility in trying to find good procedures, in both a planning and scheduling sense, for

using instruments whose functions and operations are only understood at a design and ground testing level.

While developing the ideas presented in Ref. 1, we realized that they allow for a considerably greater degree of flexibility in both ground and flight control than we originally anticipated. In this article, we summarize the ideas presented in Ref. 1, with some additional ideas developed in Legg and Welter, 1999. Section I defines certain adaptive scheduler fundamentals. Section II explores procedures to be applied to the flight and ground systems to take into consideration both absolute and relative time constraints within a primarily event-driven system. Section III examines strategies for exploiting time gaps that may arise during schedule execution as a consequence of the imposed time constraints. The generality arises from realizing that such "gaps" can be made arbitrarily large, in principle extending over the entire schedule. Finally, section IV proposes functionality to support real-time operations within the context of our hybrid event/time-based scheme. We call our approach *Adjustable Synchronization.*

I. Adaptive Scheduler Fundamentals

This section presents certain fundamentals for an onboard adaptive scheduler (AS) that we will be assuming for the remainder of the paper. These fundamentals are taken in part from proposals sketched out by other groups, specifically the GSFC & TRW NGST pre-phase A study teams, as well as ideas developed for an AS prototype study by the larger team of which we are members (cf. Cammarata et al, 1998, hereafter "the Prototype").

As noted in the introduction, the minimum requirement for an AS is that it execute basic observatory activities in an event-driven manner, at least primarily. For a space observatory such as NGST, these activities will usually be executed as part of an astronomical observation. An observation request is the most common basic scheduling unit (SU); each SU consists of a sequence of activities such as [slew, acquire guide star(s), observe with science instrument A]. In addition to science, SUs can support engineering "house-keeping" that may be required periodically, such as using thrusters to dump excess angular momentum. The ultimate goal is efficient execution of SUs to maximize total observatory science output. The house-keeping SUs can be either explicitly provided by the ground or created as needed by the onboard system.

In the Prototype, the AS coordinates the execution of SUs in sequence order as specified in a list provided by the ground. The AS consists of two pieces. The first piece coordinates validation of individual SUs as a whole and their component activities individually prior to execution; the second coordinates actual execution of the activities. At the SU-level, component activities can be flagged as required, e.g., slewing to specific target coordinates or acquiring guide stars upon arrival. If an SU-required activity is either found to be invalid during validation (e.g., target direction too close to the sun) or fails during execution (e.g., failure to acquire guide star), the entire SU is dropped.

Validation and execution of activities within the Prototype are performed by intelligent managers within the subsystems that receive commands from the AS. The Prototype supports three such subsystems, one each for attitude control, science instrument control, and data recorder control. The intelligent managers report to the AS on the validity of commands received and on the success or failure of commands at the completion of execution. The ability to validate commands prior to execution requires that each subsystem manager have a model defining the requirements of each acceptable command type and also sufficient perception of the subsystem state to perform evaluation for command validation and execution.

For efficiency, space observatories with multiple science instruments often allow their parallel use. NGST is expected to be similarly designed. The Prototype takes this expectation into consideration, providing a mechanism to allow parallel activity execution in certain circumstances. We will be relying upon the existence of some such mechanism in the discussion in section IV pertaining to real-time operations.

The Prototype uses management of angular momentum as an archetype for onboard-controlled house-keeping activities. Upon receiving notification from the attitude control system that momentum exceeds a specified level, the AS initiates a smooth suspension of any science in progress, commands the attitude control system to dump momentum, and then instructs the science instrument module to resume the suspended science. The significance is not the specific house-keeping activity selected, but rather the illustration of the smooth insertion of a necessary activity on an as-needed basis.

II. Applying Time Constraints

Adjustable Synchronization is intended as an extension of the ideas explored in the Prototype. It is designed to enable the onboard AS to (1) execute SUs in the order provided by the ground if possible, (2) force schedule synchronization with ground expectations at certain time-critical events, and (3) respond flexibly as an event-driven system for enhanced science efficiency to the extent allowed by the SOT. We envision it as being most applicable in situations where the following conditions pertain:

1. Timing constraints for at least some important SUs are fairly tight and/or can produce complex scheduling interaction between SUs. These constraints can be either absolute time constraints for specific SUs or relative time constraints for linked sets of SUs.
2. Most, or at least many, SUs need not be tightly constrained in time.
3. The ground system can accurately model optimistic, conservative execution of SUs over the scheduling time period of interest. "Optimistic" means all SUs proceeding successfully; "conservative" means execution times padded for activities with uncertain duration.

Condition 1 is critical for the ideas being developed here and in fact may not be true for the type of cosmological science of primary concern for the NGST mission. If condition 1 is not true, it would be more efficient to revert to a purely event-driven system. The purpose of stating condition 2 is to recognize that if most SUs were tightly time constrained, one would do as well to use a fully time-driven system. Condition 3 basically asserts that it is not a lack of understanding by system engineers that will lead to deviations during schedule execution, but rather not fully predicable events (e.g., the time required for a guide star acquisition attempt and whether the attempt succeeds).

Let's assume now that, as a consequence of the first and third conditions, a ground-based system has been created for production of short-term (a few days to a week) schedules. At this point, the details of how schedules are generated are not important – other than that the algorithm applied be optimistic and conservative. Because the schedule is conservative, most real-time deviations (e.g., due to occasional failure to acquire guide stars) will be such that observations end earlier than predicted by the ground system. A purely event-driven AS would respond by starting each SU as soon as the previous one ends. This makes schedule execution more efficient, but can lead to problems if SUs with time constraints are encountered.

As noted in condition 1, there are two types of time constraints of potential concern: those applied as absolute constraints on individual SUs, and those applied as relative constraints forming links between SUs. (Strictly speaking, such constraints could apply between subactivities within SUs as well. The ideas presented in this paper could be generalized to cover that situation, but we will ignore that possibility in what follows.) We will first consider a mechanism for handling absolute time requirements, and thereafter discuss how the procedure can be augmented to allow for relative time constraints.

Absolute Time Constraints

To account for absolute time constraints, Adjustable Synchronization extends purely event-driven logic as follows. Let each SU have an additional set of three parameters. These parameters, created by the ground-based scheduling system and uplinked with the SU, are the earliest permitted begin time (t_{B1}), the latest permitted begin time (t_{B2}), and the latest permitted end time (t_E). As each SU is evaluated by the AS for possible execution, which includes verification that the current spacecraft state is compatible with the needs of the SU, the AS also compares current time (t) to each of t_{B1} and t_{B2}. If $t < t_{B1}$, the AS delays execution of the SU until $t = t_{B1}$; if $t_{B1} < t < t_{B2}$, the AS proceeds with the SU; if $t_{B2} < t$, the AS rejects the SU. During SU execution, the AS monitors t to determine if it exceeds t_E; if $t > t_E$, the AS terminates the SU. To produce purely event-driven behavior in cases where there are no time-critical SUs, the ground system can have set $(t_{B1}, t_{B2}, t_E) = (0, \infty, \infty)$ for all SUs, with ∞ any sufficiently distant future time.

We now consider how to specify (t_{B1}, t_{B2}, t_E) values in the case where there are one or more time-critical SUs. Suppose that somewhere in the middle of the schedule there is a time-critical SU, C, with a constraint that it must start within the time window $[t_{C1}, t_{C2}]$. As noted previously, most real-time departures from the schedule will be such as to make the SUs require less time then predicted. It would therefore *usually* be safe to set $(t_{B1}, t_{B2}, t_E) = (0, \infty, \infty)$ for all SUs except C, while for C one could set $(t_{B1}, t_{B2}, t_E) = (t_{C1}, \infty, \infty)$. The usual result would be that the SUs prior to C would finish sometime prior to t_{C1}, after which the spacecraft would suspend science operations until t_{C1}.

A "usually acceptable" approach is not adequate for the general case. There may occasionally be circumstances that cause unexpected delays in some SUs, e.g., recurrent guide star losses of lock and associated exposure reinitializations. To prevent C from being shoved to a future time outside its acceptable execution window, one must set $(t_{B2})_C = t_{C2}$. If C is of no higher priority than any preceding SU, no further adjustments are needed; if the preceding SUs are delayed to the point that C can no longer start before t_{C2}, C is simply dropped. However, if there exists an SU, A, sometime prior to C that is of lower priority than C, specification of the (t_{B2}, t_E) pair for A must be adjusted as follows:

$$t_E(A) = t_{C2} - \Sigma_k (d_k + \Delta t_{k>k+1}) \quad (1)$$

$$t_{B2}(A) = t_E(A) - (d_A + \Delta t_{A>A+1}) \quad (2)$$

where the sum in equation (1) pertains to all SUs between A and C with priority greater than A, d_k is the estimated duration of SU k, $\Delta t_{k>k+1}$ is the time to maneuver between SU k and the next high priority SU (with C as the last "k+1"), d_A is the estimated duration of A, and $\Delta t_{A>A+1}$ is the time to maneuver from A to the next high priority SU. The term "maneuver" here should be taken as a generalized concept that includes not only change of attitude, but also post-slew settling, any required instrument reconfiguration, and any required house-keeping activities (e.g., momentum dumping) that ground modeling indicates will probably be inserted by the spacecraft between the two SUs. If there are more than two priority levels, the procedure can be applied in a nested structure, working downward towards SUs of progressively lower priority.

In the case of multiple time-constrained SUs, the ground system specifying (t_{B2}, t_E) pairs must work backwards through the schedule, applying equations (1) and (2) to all low priority SUs relative to each time constrained SU. Let C_N be the Nth high-priority time-constrained SU in the schedule. Equations (1) and (2) applied relative to C_N set upper limits on (t_{B2}, t_E) for all earlier low priority SUs, even intrinsically constrained ones – i.e., if the intrinsic value of $t_{C2}(C_{N-i})$ is greater than t_{B2} imposed on C_{N-i} by C_N for any $i < N$, then $t_{B2}(C_{N-i})$ based on C_N replaces $t_{C2}(C_{N-i})$ as the last permitted start time of C_{N-i}. This backwards progressing procedure ultimately finds the smallest upper limit for each of t_{B2} and t_E for each SU.

If nothing further is done than inclusion of the (t_{B1}, t_{B2}, t_E) triplets and associated rules, one would tend to lose the benefit of schedule compression whenever a time-critical SU is added to the schedule; the usual result would be the creation of an unused time gap just before the earliest

permitted start time for each time-critical SU. For the time being let's just consider those gaps to be opportunities to be used advantageously by the AS. We will return to this issue in section III, where a number of schemes for using these opportunities will be discussed.

Relative Time Constraints

One approach for handling relative time constraints would be to have the ground system convert all relative time constraints to absolute constraints after the schedule has been constructed. The flight system could then operate exactly as described in the preceding subsection. Although this approach would probably be the easiest to implement onboard, it sacrifices flexibility for the sake of simplicity. We therefore propose the following ideas instead.

Constraints on the timing of an SU A_j that is a member of a set {A} of linked SUs can arise from links to set members from within four distinct time periods: (1) before t_H, the last time for which the ground-based scheduler has knowledge of actual execution history, (2) within $[t_H, t_j]$, where t_j is the start time of A_j (i.e., current time when A_j is being considered by the AS for execution), (3) within $[t_j, t_F]$, where t_F is the final time of the ground-generated schedule (not known exactly until the schedule has executed), and (4) after t_F.

Any link back to an SU that occurred before t_H can legitimately be handled in the ground system by converting the relative constraint into an absolute constraint. A link from A_j back to an SU, say A_i, scheduled in the range $[t_H, t_j]$ must be handled at least in part by the onboard AS; the actual execution time of A_i, or the failure of A_i to execute, can influence the timing of A_j in a way only determinable after A_i is complete.

A link from A_j to a later SU, A_k, can constrain A_j only if (1) A_k is absolutely constrained, (2) there exists an SU, C, between A_j and A_k that is absolutely constrained, thereby effectively constraining A_k, or (3) A_k is itself constrained via a link, either forward or backward, to some other constraining SU. Point 3 makes constraint tracing recursive, but in a way that either allows a linked set to move as a block, or ultimately anchors to an absolutely constrained SU. In principle, points 1, 2, and 3 pertain whether or not A_k is part of the current schedule, i.e., exists before t_F. This point could be important if the ground system works not merely with the immediate schedule for the near-term period, but also with an approximate schedule extending even further in time.

The principles outlined above apply generally, irrespective of the selected convention for specification of links between SUs. For simplicity, let's assume that links may be specified via the following convention. We will then indicate how the constraint information can be provided to the onboard AS for easy use. Each linked SU set {A} has an associated link set $\{L\}_A$. Each SU within {A} explicitly specifies as part of its definition the identities of the elements of $\{L\}_A$ for which it is a node, e.g., SU A_1 is a node for L_{12}, A_2 is a node for L_{12} and L_{23}, etc. We assume now that each SU can be linked only to its two immediate chronological neighbors within {A}. Each link, L_{jk}, is defined by its two nodes (i.e., the predecessor A_j, and the successor A_k), time parameters Δt_{MIN} and Δt_{MAX} specifying the minimum and maximum separation of the nodes, and a binary flag, f_{kj}, indicating whether A_k can execute if A_j does not. Finally, we impose a rule that if (1) f_{kj} is set to *TRUE*, (2) A_j is linked backwards to an earlier SU (A_i), and (3) A_j fails to execute, then a new link is formed between A_i and A_k based on the Δt_{MIN} and Δt_{MAX} values from L_{ij} and L_{jk}. (We leave it as an exercise for the reader to select equations for generation of the new link.)

Let's now imagine this system applied to schedule construction in the ground system and subsequent execution by the onboard system. We pick up shortly after time t_H. The ground system has received a history log from the spacecraft indicating which SUs have actually been executed through t_H and what the state of the spacecraft is at t_H. The ground also has available the specification of the currently executing schedule and knowledge of the rules that the onboard AS applies. The first thing that the ground system does is construct an optimistic, conservative model of how the remainder of the current schedule will play out based on execution history through t_H and knowledge of how the AS will respond to any anomalies or SU failures that occurred before t_H. This may imply the elimination of some SUs by the AS in the range $[t_H, t_F']$, where t_F' is the revised expected end time for the currently executing schedule. It also implies that any relative time constraint linking an SU in the range $[t_H, t_F']$ to an SU prior to t_H can be transformed to an absolute constraint.

Using this revised version of the current schedule, together with the set of SUs that have been submitted for scheduling, the ground-based scheduler constructs a nominal schedule for the time period $[t_F', t_F]$. This schedule construction can be based on any convenient scheduling algorithm; our only requirement is that the new schedule be internally self-consistent. This latter point implies that the SUs as placed into the nominal schedule satisfy all required constraints, including absolute and relative timing constraints. The details of the ground-based scheduling system may be quite complex, but fortunately are of no consequence to the design of the onboard AS; we may view the ground action for this phase as magic. Alternatively, we could imagine using a version of the short-term scheduling system currently used for the Hubble Space Telescope (cf. Samson 1998), appropriately scaled back for an L2-based mission.

After constructing the nominal schedule, the ground system assigns (t_{B1}, t_{B2}, t_E) triplets to each SU, with equations (1) and (2) used for t_E and t_{B2} as needed. If there are either no absolutely constrained SUs or no relative links, construction of the nominal schedule is complete. Otherwise, the ground system must refine the t_{B1} values based on implicit limitations imposed on the AS's ability to move any given time-linked SU, A_j, to earlier times as a consequence of links from A_j to later SUs. The simplest example occurs with a schedule fragment like [... A_j, C, A_k ...], with C absolutely constrained. C prevents A_k from moving

forward, which can prevent A_j from moving forward. Such future-imposed constraints can be much more complicated, possibly involving tangled webs of linked sets and anchor points beyond the last SU in {A}.

Resolving such future-imposed constraints can be done up-front, i.e., as part of the ground-based process, if one is willing to apply an approximation that event-driven execution will only result in SUs moving to earlier time – true usually, though not always. Using this approximation, together with assumptions on the nature of permitted links, one can construct an algorithm for defining how much each SU will be allowed to move earlier in time during event-driven execution. Appendix A provides pseudo-code for one such algorithm based on the link convention specified earlier in this section. Multiple forward and backward passes through the SU sequence following A_j may be required to trace through complex link entanglements. Furthermore, if one or more SUs after A_j require more time than allotted for their execution, the procedure may fail to prevent a relative time constraint violation linking back to A_j. For such cases, we rely upon the t_{B2} and t_E values specified for each SU to prevent low priority SUs from encroaching upon high priority SUs, although this will protect A_j's successor only if the ground-based scheduling system specified it as having high priority.

The discussion in the preceding paragraph, with its goal of resolving all web entanglement within the ground system, implicitly assumes that no real-time anomaly occurs before A_j that will render SUs after A_j unexecutable. This could happen, for example, if a science instrument required by a later SU fails, or if an SU linked to and required by a later SU fails. Because of the possible intricacies of link entanglements, the loss of any SU (say, B) prior to the last linked SU in the schedule (say, D_k) can lead to a relaxation of the future-imposed constraint on A_j's earliest permitted start time – even if neither B nor D_k is part of {A}. Two possibilities exist to handle this situation. The simplest would be to ignore it, in which case at least two schedule gaps could develop – one immediately before A_j, and the other at the time of or sometime after B's nominal location. The AS could then use the strategies to be discussed in Section III to fill those gaps.

The second possibility would be to provide the AS with a list of anomaly types that could imply loss of future SUs. Upon encountering such an anomaly, the AS could purge the future schedule of unexecutable SUs and then recompute the (t_{B1}, t_{B2}, t_E) triplets for all remaining SUs. To support these computations, the AS would have to be provided with the intrinsic (t_{C1}, t_{C2}) time limits for all absolutely time-constrained SUs as well as the relative priorities of all of the SUs in the schedule. The computations are lengthy and not clearly worth imposing on the onboard system, particularly if a good set of gap-filler strategies have been provided to the AS. The possible advantage of having the AS recompute the time triplets is that it could allow the creation of a smaller number of larger gaps. Large gaps are typically more efficiently used by scheduling systems; the AS is likely to do better with them as well. A decision regarding which approach to take should be based on how often relative time constraints are expected to be used.

After the ground system has finished determination of the time triplets, the schedule (including the triplet and link specifications) is sent to the spacecraft. It would be the responsibility of the AS to apply the relative time constraints on SU links. A reasonable approach would be to have the AS transform a relative link between two SUs into an absolute constraint on the trailing node immediately upon execution completion of the leading node, i.e., if/when A_j with L_{jk} succeeds, the AS traces L_{jk} to A_k and applies the MIN/MAX time parameters as constraints to $(t_{B1}, t_{B2})_k$. Thus when the time comes to execute A_k, all of its constraints will already have been made absolute.

III. Filling Schedule Gaps

As was noted earlier, simply overlaying the (t_{B1}, t_{B2}, t_E) time triplets and associated rules on top of an optimistic, conservative ground-constructed schedule would tend to result in unused time gaps just before each SU with a t_{B1}-constraint. In this section we discuss three possible strategies whereby the AS can take advantage of these time gaps. The strategies being considered are: (1) augmentation of the schedule using in-line gap fillers, (2) selection of fillers from an auxiliary SU pool, and (3) autonomous onboard creation of SUs for gap filling. This list is not intended to be complete; some additional possibilities are suggested in Ref. 1. The purpose, rather, is to illustrate the general possibility with a few variations.

Schedule augmentation using in-line fillers

The simplest strategy for gap filling would have the ground system pad the schedule with additional low priority "filler" SUs at the points where the gaps are likely to form, i.e., just before each SU with a t_{B1}-constraint, and perhaps before members of linked sets that are constrained by future SUs. The (t_{B2}, t_E) time-tags for such filler SUs would be constructed as previously specified in equations (1) and (2) for low priority SUs before a time-critical activity. The fillers would have to be inserted into the schedule as a final phase of ground processing after relative time constraint evaluation has been completed, placement there being to prevent interference with the latter process. The number of such filler SUs inserted at any point in the schedule could be based on a statistical estimate for the maximum size of a gap likely to form at that location, e.g., on a three-sigma estimate for worst case number of SUs or science time lost due to typical anomalies. As far as the AS is concerned, there need be no distinction between SUs that constitute the nominal schedule and those that are inserted as in-line fillers; the AS could apply the same rules for both types, at least during normal execution. An exception would be required if one chooses to include onboard purging of future unexecutable SUs followed by (t_{B1}, t_{B2}, t_E) reconstruction, as discussed near the end of section II.

Selection from an auxiliary SU pool

The second strategy for gap filling would require the ground system to create and uplink a pool of unsequenced auxiliary SUs distinct from the nominal schedule. As part of evaluation for each SU A in the nominal schedule, the AS would compute the time period $\Delta T = (t_{B1,A} - t)$, where t is current time. If ΔT exceeds a database-specified duration, D_{MIN}, the AS would select an appropriate SU from the auxiliary pool for insertion into the gap. All acceptable auxiliary SUs would have to meet the constraint $(d_I + \Delta t_{>I} + \Delta t_{I>A}) < (t_{B2,A} - t)$, where d_I is the expected duration of the SU to be inserted, $\Delta t_{>I}$ is the time to maneuver from the current attitude and configuration to the SU to be inserted, and $\Delta t_{I>A}$ is the time to maneuver from the inserted SU to the attitude and configuration required for A. The AS must also create a latest permitted end time for the inserted SU: $t_{E,I} = t_{B2,A} - \Delta t_{I>A}$, and verify that the expected duration of the candidate SU is such that it will end before $t_{E,I}$. The SOT may wish to apply additional constraints, e.g., that any acceptable candidate must have an attitude within a database-specified angle of the attitude of the last SU executed from the nominal schedule.

If multiple SUs within the pool are acceptable candidates for local insertion, the AS would require a procedure for selecting the best. The selection rule may be as simple as picking the SU with target direction closest to the telescope's current pointing, or perhaps the SU that would provide the greatest amount of observing time while still being able to fit within the gap. Rules such as these would be appropriate if the typical size of a gap corresponds approximately to the expected duration of auxiliary SUs. If the gap could be substantially larger, then it may be desirable to introduce an onboard scheduling algorithm capable of more sophisticated optimization across the entire gap; in the limit of a very extended gap, one may wish to host a simplified version of the ground-based near-term scheduler within the flight system.

If the complexity of the selection process is high, either because of the number of auxiliary SUs or the length of the gap to be filled, it may desirable to expedite processing by applying much of the selection/scheduling process while execution of the preceding SU is still in progress. This would require some additional complexity within the AS; see Ref. 1 for some speculation in this area.

Autonomous SU creation

The third strategy for insertion of filler observations would be to have the AS generate such fillers as may be needed based on some internal rules. For example, after computing $\Delta T = (t_{B1,A} - t)$ and finding $\Delta T > D_{MIN}$, the AS could invoke a template for creation and insertion of an SU at or near the current pointing, e.g., $(\Delta T/D_{MIN})$ exposures using science instrument A and a standard filter. Any such onboard-generated SUs would be subject to the same constraints indicated in the preceding discussion for filler SUs drawn from an auxiliary pool. A possible use for autonomously created SUs would be for quasi-regular accumulation of data for science instrument calibration. Another possibility would be to support core NGST mission survey work, e.g., early universe supernova frequency statistics, or early universe galaxy size and morphology statistics (cf. Appendix C of Stockman, 1997). The proposal to use autonomously created SUs is essentially the same as an idea suggested by Hallock and Love, 1998 – the only difference being that our proposal restricts its use to a gap filling strategy, whereas theirs suggested it as a strategy employable by an AS independently of the existence of schedule gaps.

IV. Real-time Operations

A common flight operations team (FOT) concern with event-driven systems is that it may not be possible to know in advance when commands intended for immediate execution can be sent to the spacecraft. It would, of course, always be possible for the FOT to simply deactivate the onboard AS whenever real-time commanding is required, but this seems needlessly disruptive. In this section we describe three approaches to real-time operations appropriate for various different circumstances; the approaches are: (1) forced command insertion, (2) interleaved commanding, and (3) dedicated time block scheduling. In all cases, the real-time commanding being done can be in the form of either individual commands or extended command macro sequences.

Forced command insertion

First and foremost, the AS must be a servant to the SOT and FOT, facilitating rather than hindering spacecraft operations. Under some circumstances, it may be necessary for the FOT to issue commands that may not be compatible with current schedule execution. Under such circumstances, and depending upon the urgency of the command, it would be reasonable for the AS to follow one of three procedures: (1) complete the current SU and then execute the FOT-issued command, (2) suspend the current SU soon (e.g., at the end of the current exposure) and then execute the FOT-issued command, or (3) immediately suspend the current SU and execute the FOT-issued command. The action of suspending an SU envisioned in procedures 2 and 3 would allow the AS to more-or-less smoothly resume execution of the SU after the FOT interaction is complete. Depending upon the complexity involved with SU suspension, option 3 may in fact be further divided into two suboptions, one allowing the AS the time required to complete the suspension process, and another that simply forces immediate command execution irrespective of the implications for the SU in progress, e.g., the possible loss of all further exposures. Which, if any, of these approaches are used depends upon the engineering details of the mission. The important point, as emphasized in the lead sentence, is that the AS should be designed to support the FOT's needs in urgent situations.

Interleaved commanding

As noted in section I, we assume that the AS is designed to support parallel execution of activities. For this section, we further assume that this support is such that SUs with parallel activities are in the form of non-overlapping blocks of parallel activity sequences; see Doxsey et al, 1998, for details on one variation of such an approach. At the time of command uplink, some particular block of activities will be being executed.

For interleaved commanding, we envision that each incoming command macro will be stored in a buffer for evaluation. The AS then examines each command in the macro, comparing it against each activity request in the thread sequences comprising the block currently being executed. Rules associated with each permitted command type allow the AS to determine whether a command can be executed in parallel with any particular high-level activity that the observatory can be requested to execute.

If every command in the macro is compatible with every activity in the current SU block of parallel threads, the AS generates a new thread associated with the block and launches the macro as that thread. Being a thread of the current block implies that the macro will be allowed to run to completion before the current block terminates even if all other threads are complete, thus preventing any subsequent activities from possibly being in conflict with any macro elements. If any commands within the macro are incompatible with any of the activity requests within the block, the AS examines each remaining block within the current SU until it finds one compatible with the macro. If the macro commands are not compatible with any block of the SU, the AS postpones execution of the macro until after the SU has run to completion.

Dedicated time block scheduling

Dedicated time commanding would actually be rather simpler than interleaved commanding. The FOT obtains a block of dedicated spacecraft time for real-time commanding by creating a "dummy" SU (DSU) containing the earliest and latest permitted start times, estimated duration, priority, and other parameters such as spacecraft attitude or required science instrument configuration. Notice that this provides optional autonomous attitude slew and onboard equipment reconfiguration at the beginning of the dedicated time block if desired. Similarly, commands to return the spacecraft to a nominal state at the end of the time block could also be included in the DSU if appropriate.

If the time constraints do not specify a specific time, other than that the DSU occur during a time when the ground station can see the spacecraft, the scheduling system determines an optimal time for the DSU relative to the other SUs in the schedule. The DSU is scheduled and executed like any other SU. The spacecraft configures itself appropriately (e.g., slews to a specified attitude), activates a timer to measure the reserved time block, and awaits FOT commands.

The timer triggers two events. Prior to full completion of the scheduled time period, the AS sends a message to the FOT stating that DSU expiration is imminent. The FOT may optionally extend the time limit if desired. Upon expiration of the timer, the AS executes any appropriate clean-up activities associated with the DSU and proceeds to the next SU. The FOT can optionally send a "timer expired" command to trigger early termination of the dedicated time period and immediate transition to the next SU.

Summary

This paper provides a description of an approach, herein designated Adjustable Synchronization, whereby the science operations team can control the flexibility provided to an onboard Adaptive Scheduler for responding to real-time deviations between the ground's prediction of schedule execution and that which actually occurs. The approach is designed as an augmentation of the event-driven logic used in the NGST AS Prototype described by Cammarata et al, 1998. Section I provides an overview of the basic features of that Prototype. Section II presents an approach for incorporating absolute and relative time constraints into a basically event-driven system. Introduction of time constraints leads to the possibility of time gaps prior to time constrained activities; section III presents some options to allow an adaptive system to fill those gaps with useful science or calibration work. Finally, section IV presents some AS functionality designed to facilitate real-time commanding and thereby ameliorate FOT concerns associated with the basic event-driven process.

Acknowledgements

The investigation reported in this article was performed for the Goddard Space Flight Center Flight Software Branch, Code 582, in support of the NGST Operability Integrated Product Team under contracts GS-35F-4381G (task # S-2331-G) and NAS5-32350. We thank Lou Hallock, Tom Pfarr, Merle Reinhart, Roberto Samson, and Kevin Stewart for valuable discussions.

Acronym List

AS	adaptive scheduler
DSU	"dummy" scheduling unit
FOT	flight operations team
IPT	integrated product team
L2	second Earth/Sun Lagrange point
NASA	National Aeronautics and Space Administration
NGST	Next Generation Space Telescope
SOT	science operations team
SU	scheduling unit

References

Cammarata, G., Hallock, L., Duran, S., Bruno, C., Myers, P., Welter, G., Stewart, K., & Legg, J., 1998. Next Generation Space Telescope Adaptive Scheduler Prototype – Lessons Learned, Goddard Space Flight Center.

Doxsey, R., Balzano, V., Henry, R., Isaacs, J., Johnson, C., & Kutina, R., 1998. NGST Integrated Ground/Flight Software Operations Concept.

GSFC & TRW NGST pre-phase A study teams, 1996. The Next Generation Space Telescope – Concepts and Technology, presentations to the NGST Study Office, Goddard Space Flight Center.

Hallock, L., & Love, B., 1998. NGST Autonomy Candidates, NASA/GSFC Branch 582 Technical Report.

Legg, J., & Welter, G., 1999. Next Generation Study Space Telescope Adaptive Scheduling Study Report, Prepared for NASA/GSFC Branch 582 by Raytheon and Computer Sciences Corporation. (in preparation)

Samson, R., 1998. Greedy Search Algorithm Used in the Automated Scheduling of Hubble Space Telescope Activities. In Proceedings of an SPIE Conference: Observatory Operations to Optimize Scientific Returns, vol. 3349, 282-290.

Stockman, H. (ed.), 1997. Next Generation Space Telescope – Visiting a Time When Galaxies Were Young, Space Telescope Science Institute.

Welter, G. 1998a. Scheduling NGST – I; AXAF Scheduling & Single Chain Scheduling, Technical memorandum, Prepared for NASA/GSFC Branch 582 by Computer Sciences Corporation.

Welter, G. 1998b. Scheduling NGST – II; Scheduling with HST SPSS, Technical memorandum, Prepared for NASA/ GSFC Branch 582 by Computer Sciences Corporation.

Welter, G. 1998c. Scheduling NGST – III; SPIKE/Min-Conflicts and Simulated Annealing, Technical memorandum, Prepared for NASA/ GSFC Branch 582 by Computer Sciences Corporation.

Welter, G., & Legg, J., 1998. Scheduling NGST – IV; Adjustable Synchronization, Technical memorandum, Prepared for NASA/GSFC Branch 582 by Computer Sciences Corporation and Raytheon.

Appendix A - Resolving Future-Imposed Time Constraints

This appendix provides pseudo-code for an algorithm specifying how the ground-system can determine to what degree each forward-linked SU within a linked set is constrained by future SUs. It assumes the link model specified in section I, i.e., whereby each SU in a linked set can have links only to its immediate neighbors within the set. For the following pseudo-code, an "intrinsic" absolute time constraint is one defined as part of the SU, whereas an "implicit" absolute constraint is one derived from another SU. Memory of constraints designated as "implicit" is retained only over the duration of the loop over SU A.

```
Do for each SU (A) in the schedule in forward direction (except last)
  If A has a forward link to a scheduled SU (A+)
    Initialize any previously constructed implicit absolute constraints for A and subsequent SUs to null.
    Do until no further constraints are imposed
      Do for each SU (B) after A in forward direction (except last)
        If B is absolutely constrained (either intrinsically or implicitly)
          Mark B's successor (C) as implicitly absolutely constrained.
          Determine the implied earliest permitted start time for C based on nominal execution of B with B starting as early as possible. (Note, if C was already constrained, the earliest permitted start time for C is the later of its previously determined earliest start time and that derived from B.)
        Endif
      Enddo
      Do for each SU (B) after A in reverse direction
        If B is absolutely constrained and B has a backwards link to an SU (B-) not earlier than A
          Mark B- as implicitly absolutely constrained.
          Determine the earliest permitted start time of B- based on B's earliest permitted start time and the relative constraints between B- and B. (Note, if B- is already constrained, the earliest permitted start time for B- is the later of its previously determined earliest start time or that derived from B. Note further that within this block, linked set {B} could be {A}, and SU B- could be A.)
        Endif
      Enddo
    Enddo
  Endif
Enddo
```