Distributed Satellite Constellation Planning and Scheduling

Robert A. Richards, Ryan T. Houlette, and John L. Mohammed

Stottler Henke Associates, Inc.
1660 South Amphlett Blvd., Suite 350
San Mateo, California 94402
richards@shai.com

Abstract

SHAI is developing a software architecture for automated, distributed planning and coordination of constellations of satellites. This architecture allows large satellite constellations to manage themselves with minimal human oversight. SHAI is utilizing an integrated approach drawing upon a broad range of AI and non-AI techniques. Advanced planning and scheduling algorithms permit the system to quickly create complex plans satisfying intricate time and other constraints. A reactive planning component deals with unexpected, time-critical local events such as new critical tasks. In addition, a knowledge base stores information about the satellites' capabilities and commitments that is used during the distributed planning process to properly allocate tasks to the satellites best suited to perform them. The resulting architecture provides the capacity for robust, scalable management of satellite constellations. The potential benefits are reduced costs, increased operational efficiency, and improved robustness. A prototype utilizing a subset of the architecture has been built and verified.

Background

Since the first American satellite was placed in orbit in 1958 scientists have recognized the tremendous potential for study of the Earth and space offered by orbiting sensors. In the past satellites worked mainly as isolated entities. The new satellite constellation model is desirable because it is more robust and flexible, providing a greater ability to adapt to changing expected and unexpected events. However, managing a constellation requires a tremendous amount of time and effort.

Given the low cost and high speed of computer processors today, it is now feasible to equip every satellite in a constellation with enough processing power to be able to direct its own activities. Control would thereby be distributed throughout the entire constellation, minimizing the reliance of the whole system on the limited and fallible resources of any single component, and minimizing human ground management. A well-designed and flexible social structure would enable the constellation to coordinate its activities as a whole, distributing tasks and allocating responsibilities to the appropriate member satellites, without losing the benefits of individual autonomy.

SHAI is developing a software architecture for automated, Distributed Spacecraft Coordination Planning and Scheduling (D-SpaCPlanS, pronounced 'D space plans). This architecture allows large satellite constellations to manage themselves with minimal human oversight. The potential benefits are reduced costs, increased operational efficiency, and improved robustness.

D-SpaCPlanS Overview

The D-SpaCPlanS architecture will support the creation and maintenance of adaptive, hierarchical social structures that will enable large satellite constellations to manage themselves with minimal human oversight. Figure 1 illustrates the context in which the D-SpaCPlanS agent is designed to function.

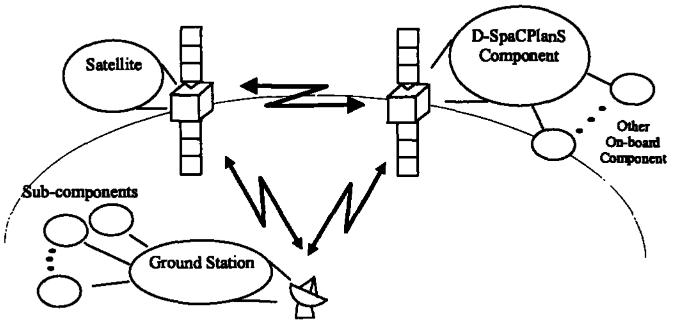

Figure 1 D-SpaCPlans Context

To accommodate multi-level organizational structures within individual satellites and satellite constellations, the system will employ hierarchical task decomposition methods allowing planning to occur at varying levels of detail. To deal with unexpected, time-critical local events (such as system failures), a reactive planning component [Bonasso et al., 96] for rapid remediation is included. In addition, a sophisticated knowledge base will store information about the capabilities and commitments of other satellites. This will be used during the distributed planning process to properly allocate tasks to the satellites best suited to perform them. The architecture will also draw on constraint satisfaction and irrelevance reasoning algorithms [Wolverton & desJardins 1998] to detect and resolve conflicts between satellites' individual plans.

Functionality

The D-SpaCPlanS advances the idea that efficient distributed planning in a densely-populated network of autonomous entities requires some form of system-wide organization to structure and guide the planning process (see discussion of organizations in [Durfee, 1993]). To ensure robustness and scalability, this organization should spread authority and responsibility as evenly as possible over all its members, thereby minimizing the impact caused by the introduction of new members or the loss of old members due to failure. In addition, the organization will be able to reorganize itself to adapt to changes to the membership of the network over time.

In D-SpaCPlanS, the organizational principles described above manifest themselves in the form of a flexible hierarchy of nested groups; every satellite in a given constellation belongs to one or more of these groups. A rough sketch of this hierarchy is presented in Figure 2, with solid circles representing the actual satellites in a constellation and different-sized ovals representing groups at different levels of the hierarchy. The groups to which a particular satellite belongs are indicated by the ovals that contain it. For example, Satellite 13 is a member of the whole constellation, of Group A, and of Subgroup Alpha. Group membership is not based on any specific properties of a satellite – such as its physical location or functional capabilities – but is rather an arbitrary, abstract division whose purpose is to provide a framework for the decomposition and distribution of the planning process across the entire constellation. These divisions are determined automatically by negotiation [Davis and Smith 1983] among the satellites' social coordinator units.

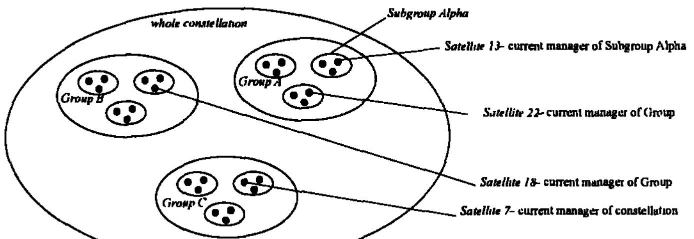
whole constellation
Subgroup Alpha
Satellite 13- current manager of Subgroup Alpha
Group A
Group B
Satellite 22- current manager of Group
Satellite 18- current manager of Group
Group C
Satellite 7- current manager of constellation

Figure 2 Sample Organizational Structure

As shown in the figure, its own autonomous manager governs each group at each level of the hierarchy. This manager receives task assignments for its group from the next higher level of the hierarchy and in turn is responsible for distributing these tasks to the most appropriate subgroups in the level of the hierarchy immediately below. The manager makes its determination of who is most appropriate for a given task based on its knowledge of the capabilities and current commitments of each of the subgroups belonging to its group. At the lowest level, managers assign tasks to the individual satellites in their subgroups, which then carry out the requested tasks. At the top of the hierarchy resides the manager for the whole constellation, who accepts orders directly from the ground controller. The management role of the ground controller is greatly simplified by this hierarchical organization, since it needs to interact with but a single entity in order to direct the activities of dozens to hundreds of satellites. Planning occurs at all levels of the hierarchy, with plans growing more and more detailed the further down one goes in the hierarchy.

It is not uncommon for a satellite to fail in orbit for a variety of reasons [Mohammed, 98]. Such a failure could be catastrophic for a constellation that relies heavily on a fixed organizational hierarchy. Imagine, for example, that the top-level manager suddenly dies without warning. If the satellites' organizational scheme is unable to compensate for this failure, the whole constellation becomes inaccessible and useless. In our architecture, granting the organizational structure the capacity to dynamically reorganize itself in the face of changes to the constellation alleviates this danger. There are two facets to this capacity.

First, group managers are designed in such a way that their existence is not tied to any single satellite. Instead, every satellite in a constellation possesses the ability, within its automated planning architecture, to take over the duties of a group manager. As part of this distribution of the managerial role, each group manager periodically broadcasts the contents of its memory to some of its subgroups, which store this information. Then, when lack of communication from a given manager indicates the death of the hosting satellite, the members of the now-managerless group cooperate to reconstruct the lost manager on a new host using their backup copies of that manager's memory.

Second, if significant imbalances in group size arise (e.g., as old satellites fail and new ones are added), the constellation's managers can also collaboratively restructure the actual groups themselves: large groups can be split into smaller subgroups, small groups can be merged, and groups can even be traded between managers.

Design

Figure 3 illustrates the high-level design; the arrows indicate the direction of information flow. As can be seen from the figure, there are four main components to the system. Each of these will be discussed in detail below.

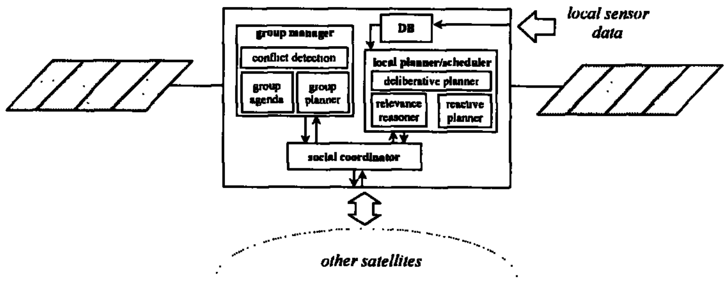

Figure 3 High-Level Design

Social Coordinator. The social coordinator acts as the interface between the rest of the architecture and the other satellites in the constellation. It keeps track of all organizational details including, among other things, which groups the satellite belongs to and where the various group managers are currently residing.

The social coordinator component draws on technologies from a number of areas. At its core, a collection of rules describing how hierarchy management should be done. Around these general rules are a number of smaller knowledge modules that "know" how to deal with specific kinds of situations. These specialized modules are activated by the rules when the appropriate situation arises.

Group Manager. The group manager is the component charged with actually directing the activities of the constellation. In essence, its job is much like that of a human manager: take orders from a higher-level manager and pass them on to the right subgroups to be carried out.

The group agenda subunit is a sophisticated knowledge base that serves as the group manager's memory, storing all relevant information about the capabilities and scheduled commitments of each of the subgroups being managed. This information is updated by reports from the subgroups as their commitments change. In turn, the group agenda sends update reports containing a summary of the agenda's information about its entire group to the manager of the level above. Thus, the group agenda subunits of all managers, taken together, can be considered a kind of distributed schedule/knowledge base for the constellation. This knowledge base will need to be able to store and retrieve multiple types of information, including complex data combining geographical and temporal parameters – for example, "satellite 3's infrared camera will be available over Honduras between 03:30 and 04:00 hours GMT on July 16."

The second subunit is the group planner, which takes the tasks given to the group manager, breaks them up into smaller subtasks, and assigns them to specific subgroups. At the core of this subunit sits an automated partial-order planner that, using hierarchical task decomposition, is capable of generating plans at different levels of detail depending on the group manager's position in the hierarchy. For example, a plan generated by a high-level manager might contain an abstract task like "Observe ocean currents along west coast of U.S.," while a lower-level manager might break this very general task down into a number of more specific tasks: "Observe 50 miles from California shore with infrared cameras," "Observe 30 miles from Oregon shore with infrared cameras," and so on. This hierarchical decomposition of tasks continues until "primitive" tasks that can actually be directly acted upon are reached. This planner relies heavily on information provided by the group agenda to determine which subgroups are best suited to perform a particular task.

The conflict detector is the group manager's third subunit. It ensures that the plans devised by different subgroups do not conflict with one another (e.g., by trying to use the same communications channel at the same time). When conflicts are found, this subunit sends the plans back to the subgroups that devised them with instructions about how to revise them to avoid the conflict.

Note that the group manager component of the architecture is only active when a manager is actually being hosted on the satellite. Thus, for most satellites in a constellation this component lies dormant most of the time, though it always remains ready to take over hosting duties if necessary.

Local Planner/Scheduler. Whereas the group manager component of the architecture handles the coordination of groups of satellites, the local planner/scheduler component is concerned with controlling the activities of a single satellite, specifically the one on which the architecture is running. The local planner receives its assignments from the satellite's group manager, then produces a complete executable plan for completing those assignments. This plan is sent to the group manager to be checked for conflicts with the local plans of other satellites. If conflicts are found, the local planner alters the plan according to the manager's instructions; otherwise, the plan is carried out as is. The local planner/scheduler comprises three main subunits: deliberative planner, reactive planner, and relevance reasoner.

The *deliberative planner* is an automated partial-order planner (using SHAI's intelligent entities concept) combined with a constraint-satisfaction-based scheduler. It is this planner that, given the manager's task assignments, produces the detailed low-level plan to carry them out. For example, the local planner might receive the assignment, "Observe San Francisco at 15:35:27 on August 10 with radar." It would then, using its built-in knowledge about the prerequisites for different kinds of actions, determine that in order to observe San Francisco at the desired time, it will need to first point its radar at the correct location and then actually take the observations by activating the radar. At this point, the intelligent entity for the "pointing radar" task would note that in order to point the radar, the satellite must fire an attitude control jet; the entities representing the jets negotiate amongst themselves to determine which jet is the correct one to fire. This continues until a precise plan of action is constructed. These plans generally are for events scheduled hours or days in advance and thus the deliberative planner does not need to be able to plan in real time.

On the other hand, the *reactive planner* , is a fast special-purpose planner designed solely to allow the satellite to react immediately in case of an emergency. If a solar panel is struck by a micrometeorite, for example, the satellite may immediately inform another satellite of the situation and its commitments (which will now be abandoned) and go into a safe-hold mode until an operator can intervene. This planner does not consult with the group manager before acting. Instead, it acts first and then informs the manager afterwards of its actions so that the manager can resolve any conflicts those actions might have caused. Because it must react in seconds, and because the remediation plans it generates are not very long, the reactive planner does not do an in-depth, knowledge-intensive analysis of its situation, goals, and resources to generate a plan; rather, it relies on a relatively small set of rules, precomputed plans, and scripts to quickly produce an effective, appropriate response.

The final component of the local planner is the *relevance reasoner*. This subunit uses irrelevance reasoning [Wolverton & desJardins 1998] to determine which parts of the plan generated by the local planner could possibly generate conflicts with other satellites. To do this it uses models of the dependencies between tasks and resources in combination with information about the goals of the subgroup's other members transmitted by the group manager. Only these relevant parts of the plan, attached to a skeletal plan framework, are then sent to the group manager to be checked for conflicts. This subunit is necessary to keep from wasting the limited communications bandwidth available for intra-constellation communication on the transmission of unimportant data.

Local Database. The local database is the repository for all local information stored by the satellite: status logs for internal systems, history of past activities, and so on. It supports the local planning and scheduling [Muscettola, 94] module by providing the data necessary to make informed planning decisions. It also stores scientific observation data until it is ready to be downlinked.

Initial Prototype

The initial D-SpaCPlanS prototype (D^{pr}) implements portions of the *local planner/scheduler*. In the prototype each satellite has its own D^{pr} component to perform scheduling specific to that satellite. The output from D^{pr} is a plan and schedule for a particular satellite. D^{pr} monitors the execution and replans when required. The *relevance reasoner* is not implemented in the prototype.

Structure

The D-SpaCPlanS prototype consists of an overall, high-level planning/integration engine, that provides the *deliberate* and *reactive planner* functionality, along with various modules that provide the basic building blocks that the integration engine manipulates and uses in the production of the final plan. Not all modules are necessarily of the same order, in that the integration engine calls upon some modules, while others will act as tools for these modules.

The overall goal of the system is the placement of a new observation into an existing schedule (the existing schedule may be blank). This is the most basic of planning and scheduling functionality and allows for the incremental construction of a full observation schedule. Figure 4 diagrams the major components of D^{pr}. The main large box in the figure 4 corresponds to the *local planner/scheduler* box in figure 3.

Plan Integration Engine. The Integration Engine is the main planning component of the D^{pr} system. The primary task of the Integration Engine is to call upon the experts to give information concerning when and how they would "prefer" observations to be scheduled, and then to mesh this together to provide a heuristically good plan/schedule. Observation and task scheduling requests are received from ground controllers or other satellites.

Observation Geometry Expert. An important factor in deciding which observation to perform at which time will be when the observation is even possible. This expert utilizes the Orbital Geometry Expert to determine whether or not the desired target is observable and it utilizes sensor knowledge to determine when the observation is possible.

Orbital Geometry Expert. A major component of the Observation Geometry Expert is the computation of the satellite's orbital path. The Orbital Geometry Expert accomplishes this. In the prototype, this module accesses data generated by the Satellite Tool Kit (STK), from Analytical Graphics, Inc.

Scheduling Utilities. The Scheduling Utilities perform detailed scheduling operations as requested by the Plan Integration Engine. Additional utilities to define, modify, and access schedule data definitions are also provided. Many of the experts within the Scheduling Utilities use these additional utilities to carry out their function.

Observation Priority Monitor. The Priority Monitor is tasked with providing information concerning the specified importance and timeliness of observations. This information is used to order observations when there is no clear advantage of one over the other.

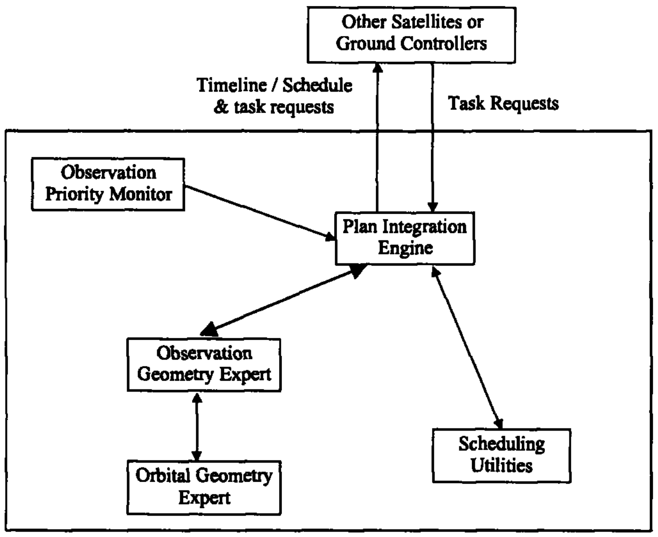
Other Satellites or Ground Controllers
Timeline / Schedule & task requests
Task Requests
Observation Priority Monitor
Plan Integration Engine
Observation Geometry Expert
Scheduling Utilities
Orbital Geometry Expert

Figure 4 Prototype Components

Prototype Test and Verification

The D^{pr} includes the ability to initialize a schedule; and accept observation schedule requests with varying priorities and varying constraints. The D^{pr} schedules observations while satisfying satellite and target site accessibility constraints, download availability constraints and daylight constraints.

D^{pr} takes tasks, priorities, and ephemeris data as input and generates schedules for the individual satellites and a history of the process.

Tests have been conducted utilizing a simulated constellation consisting of six low-earth-orbiting remote sensing spacecraft and associated ground infrastructure. The spacecraft fly equally spaced in the same sun-synchronous 653km-altitude orbit plane. There are two groundstations: one located in Fairbanks, Alaska and the other in Kiruna, Sweden.

Simplifying assumptions include: 1) a task request can be uploaded to any satellite whenever any satellite can be reached from the ground station because task requests are small and can be forwarded via satellite crosslinks; 2) observation tasks can only be accomplished when the target is in daylight; and 3) all payload memory can be downloaded with one pass over a ground station. In the D^{pr} the satellites work virtually independently of each other: there is no master planner.

When a satellite receives a task request (either from the ground or another satellite) the satellite attempts to schedule the task (on itself). If a satellite fails to accommodate the task, it will then pass the task on to the satellite that has not already attempted to schedule the task, which will pass over the target first. The passed task request includes a list of the satellites that have tried and failed to schedule the task. When the list contains all the other satellites if the last satellite to receive the task can not schedule it, then the scheduling of the task will fail and the ground station will be notified of the failure.

For the first part of the test 24 tasks are distributed to the 6 satellites. The ground assigns each of the tasks to the first satellite to pass over the respective targets. The result is that all tasks are successfully scheduled on the satellites to which they were distributed.

The CPU time is recorded for major operations because as the complexity of the scheduling task and constellations grow the computational overhead may become a limiting factor since the computations are being performed on the satellites.

For the second part of the test, 8 new higher-priority tasks are added. In this case, 5 of the 8 new tasks were scheduled by swapping out already scheduled (lower-priority) tasks. The swapped out tasks are passed to other satellites and eventually successfully scheduled. For example, one of the swapped out tasks is passed to another satellite where it fails to schedule. The task is then passed to a third satellite where it fails again. Finally it is passed to a fourth satellite where it is successfully scheduled. This portion of the test verified that the system can reactively update schedules due to new tasks being added and it can properly handle priorities, while showing the distributed nature of the scheduling process.

For the third part of the test, 2 of the tasks already scheduled fail to execute properly (for instance due to cloud cover) and 2 more tasks are added. In this case, both of the failed tasks are rescheduled, one on the same satellite on which it was originally attempted, and the other by being passed once. The new tasks are also successfully scheduled via some swapping and passing. In addition, all the tasks that were swapped out to accommodate the new tasks were also eventually successfully scheduled.

This portion of the test verified how the system can reactively update schedules due to failed tasks, as well as again showing how distributed reactive planning occurs when new tasks are added while properly handle priorities.

Statistics are captured as the system runs, the overall statistics include:

34 observations attempted.
0 rejected because the target was not recognized or out of range during the scheduled period.
13 passed from one satellite to another.
6 unscheduled to schedule another of higher priority.
2 rescheduled because the observation failed.
0 could not be scheduled by any satellite.

Conclusion

The design for automated distributed planning and coordination of constellations of satellites that SHAI is developing will allow large satellite constellations to manage themselves with minimal human oversight. The prototype already verifies some of the concepts while hinting at the full potential of the complete design. The potential is for robust, scalable management of satellite constellations while reducing costs and increasing operational efficiency over present constellation management paradigms.

References

[Bonasso et al., 96] Bonasso, R.P., Kortenkamp, D., Miller, D.P., and Slack, M., "Experiences with an Architecture for Intelligent, Reactive Agents," in Intelligent Agents II, Eds. M. Woolridge, J.P. Muller, and M.Taube, pp. 187–202, Springer Verlag, New York, NY, 1996.

[Davis and Smith, 1983] Davis, R., and Smith, R.G. "Negotiation as a Metaphor for Distributed Problem Solving." Artificial Intelligence 20, 1983. pp. 63-100.

[Durfee, 1993] Durfee, Edmund H. "Organisations, Plans, and Schedules: An Interdisciplinary Perspective on Coordinating AI Systems." In Journal of Intelligent Systems 3(2-4), 1993.

[Mohammed, 98] Mohammed, John L., "Spaceborne Satellite Anomaly Resolution using Model-Based and Case-Based Reasoning and Knowledge-Based Planning — A Preliminary Analysis," Report to AFRL, Phillips Research Site under contract F29601-97-C-0050, SHAI, Jun 1998.

[Muscettola, 94] Muscettola, N., "HSTS: Integrating planning and scheduling," in Fox, M. and Zweben, M. (Eds.) Intelligent Scheduling. Morgan Kaufman, 1994.

[Wolverton & desJardins, 1998] Wolverton, Michael, and desJardins, Marie. "Controlling Communication in Distributed Planning Using Irrelevance Reasoning." In Proceedings of the Fifteenth National Conference on Artificial Intelligence, 1998.