MULTI-AGENT PLANNING AND SCHEDULING ENVIRONMENT FOR ENHANCED SPACECRAFT AUTONOMY

Subrata Das and Paul Gonsalves
Charles River Analytics, Inc.
725 Concord Ave., Cambridge, MA 02138, USA
phone: +1 617 491 3474, fax: +1 617 868 0780, e-mail: {sdas,pgonsalves}@cra.com

Raffi Krikorian
MIT Media Laboratory
20 Ames Street, Cambridge, MA 02139, USA
raffik@mit.edu

Walt Truszkowski
NASA Goddard Space Flight Center
Code 588, Greenbelt, MD 20771, USA
Walt.Truszkowski@gsfc.nasa.gov

ABSTRACT

Spacecraft autonomy has the potential for effecting significant cost savings in mission operations by reducing the need for dedicated ground staff. In an autonomous operating mode, operators will communicate only high-level goals and deadlines directly to the spacecraft. The spacecraft will then perform its own planning and scheduling, decomposing a goal into a set of sub-goals to be achieved with onboard subsystems and/or in cooperation with other spacecraft in the environment. In this paper, we present this distributed (or equivalently, multi-agent) approach to onboard planning and scheduling that helps a spacecraft function as an autonomous agent. Such an agent's domain knowledge of tasks and their components is manifested through a hierarchical language taking into account spacecraft operational aspects and resource constraints. The task decentralization problem is solved by the use of the hierarchical knowledge structures, and the resource optimization problem is addressed by its explicit representation within the model. The reasoning performed by an agent for the required planning and scheduling tasks is based on a constraint propagation paradigm. Schedule quality is enhanced by the introduction of agent cooperation. A limited-scope Java prototype is developed and demonstrated using space-based scenarios involving onboard sensors and a satellite constellation. We are specifically targeting our effort to enhance the planning and scheduling capability of NASA's proposed Remote Agent architecture.

1 INTRODUCTION

Spacecraft autonomy has the potential for effecting significant cost savings in mission operations by reducing the need for dedicated ground staff. In an autonomous operating mode, operators will communicate only high-level goals and deadlines directly to the spacecraft. The spacecraft will then perform its own planning and scheduling, decomposing a goal into a set of sub-goals to be achieved in cooperation with other spacecraft in the environment. In this paper, we present this distributed approach to onboard planning and scheduling that helps to function a spacecraft as an autonomous agent.

The term 'planning' refers to the generation of activities that satisfy a current set of goals. For example, a planning process to satisfy the request for an image generates activities such as rolling the camera to the correct position, activating the camera shutter, and transmitting the captured image. The term 'schedule' is an association of these specific activities with particular times by satisfying constraints: for example, rolling should be performed *before* the shutter action. The onboard spacecraft subsystems must execute these time-sensitive activities autonomously to achieve the goals. If none of the subsystems of the spacecraft is capable of executing an activity then a cooperation from another spacecraft in the environment is required to get the activity executed to achieve the overall goal. For example, if a spacecraft is incapable of taking an infrared imagery of a certain swath of the planet then it has to seek cooperation from another spacecraft in the environment that can do so. In addition to serving these payload-oriented functions, planning and scheduling are also necessary to achieve goals generated to ensure safe spacecraft on-orbit operations. As described in (Pell, 1997), the onboard planner assumes a domain model containing an explicit representation of spacecraft subsystems, tasks, goals, and the norms, under which they operate. These norms are a set of flight rules and constraints that are represented in a high-level syntax.

Two major trends for task representation in the history of AI planning have been observed (Georgeff, 1987): goal achievement (GA) and hierarchical task network (HTN). The origin of GA-based planning is in STRIPS (Fikes, 1971). In this model of representation, an initial situation, a set of possible actions, and a goal that is to be achieved are given. Planning consists of finding a sequence of actions that would lead from the initial

Proc. Fifth International Symposium on Artificial Intelligence, Robotics and Automation in Space, 1-3 June 1999 (ESA SP-440)

situation to the final one. Several planners were subsequently built on the GA model including TWEAK (Chapman, 1987), and SNLP (McAllester, 1994). On the other hand, the HTN representation has its origin in NOAH (Sacerdoti, 1974). A planner based on the HTN model is presented with a *task* or *activity network,* which might contain several *non-primitive* tasks. Planning proceeds by selecting a non-primitive task, decomposing it into *subtasks* using a library of available *decomposition methods* and then detecting and resolving conflicts with other tasks. This process is repeated until no non-primitive tasks remain and all the conflicts have been resolved. Typical examples of HTN planners are FORBIN (Dean, 1988), and NONLIN (Tate, 1977). There are also planners combining features from these two such as O-Plan (Currie, 1991) and SIPE (Wilkins, 1988).

Given a representation in either GA or HTN, solving a planning problem can be viewed as a straightforward search problem, that is, find some or all possible orderings of the actions that would result in achieving the specified goal, given the rules and constraints of the environment. In general, the HTN paradigm can lead to more efficient planners because it allows the user to limit the search space by guiding the planner towards exploring only acceptable solutions. A typical implementation of the search engine of a planner operates on a temporal database such as the HSTS system (Muscettola, 1994) and Time Map Manager (Boddy, 1994). The search engine posts constraints to the database. The temporal database then constructs a constraint network and provides a *constraint propagation* (Le Pape, 1990) service to verify the global consistency of the posted constraints with the goals, rules and constraints of the spacecraft. This global consistency guarantees the existence of a schedule satisfying the constraints. Both the consistency checking and search for an optimal solution in cooperation with other agents in the environment are computationally intractable, that is, NP-hard. A distributed approach to planning and scheduling allows cooperation among agents in the environment and increases efficiency in the search for an optimal solution by partitioning the whole search space.

In recent years, there has been a growing interest in agent-oriented problem solving (CACM, 1994), which provides the basis of our proposed distributed solution (Chaib-draa, 1992) to planning and scheduling. The agent-oriented problem-solving environment increases efficiency and capability (Rosenschein, 1982) by employing a set of agents, communicating and co-operating with each other to achieve their goals, that is, to find a local solution that satisfies both its hard and soft constraints. By an *agent* we mean, an entity which operates in an environment either autonomously or semi-autonomously interacting with other agents in the environment by means of communication. Agents are sometimes software agent (Genesereth, 1994) implementing the behavior of humans, machines or hardware, etc. Agents can also be mechanical or electronic robots (Simmons, 1991) with the capability of perceiving or sensing the environment and capable of executing appropriate actions. Our assumption is that even if an environment consists of such heterogeneous agents there will be a well-defined means of communication between these agents. In other words, every agent has an interface, which understands a common communication language.

In our envisioned distributed (or equivalently, multi-agent) environment (Conry, 1988; Georgeff 1983), a set of problem solving autonomous agents (an agent is either an onboard subsystem of a spacecraft or the spacecraft itself) communicate and co-operate to achieve high-level goals through planning and scheduling. This distributed planning and scheduling emphasizes a decentralized organization, plans are generated and executed co-operatively and concurrently by the subsystem agents and spacecraft agents, taking into account system flight rules and resource constraints. In a *centralized* planning environment, goals, rules, constraints, and resources from individual agents are accumulated at a central place and a centralized planner is used to generate a global schedule. An individual agent is then provided its relevant portion of the schedule of tasks. The agent then informs the centralized planner the progress on the schedule. This centralized approach is particularly unsuitable when the problem is inherently distributed such as in a spacecraft environment where each subsystem and spacecraft functions autonomously. A centralized planner is unable to exploit fully the expertise and knowledge of each individual agent, and makes the search space unnecessarily larger. In a *distributed* environment, in contrast, each agent (i.e., an onboard subsystem or a spacecraft) generates and maintains its own plan and schedule, and therefore the whole search space is divided into a number of smaller ones to be managed by individual agents. The overall plan and schedule is obtained by combining or synchronizing plans from individual agents, resolving any conflicts that arise from the constraints on the resources.

In our envisioned distributed environment, an agent's model of the environment and tasks is manifested through a hierarchical knowledge representation language taking into account spacecraft operational aspects and resource constraints. The task decentralization problem is solved by the use of the hierarchical knowledge structures, and the resource optimization problem is addressed by its explicit representation within the model. The reasoning performed by an agent for the required planning and scheduling tasks is based on a constraint propagation paradigm. Schedule quality is enhanced by the introduction of agent cooperation. A limited-scope Java prototype is developed and demonstrated using scenarios involving onboard sensors and satellite constellations.

The rest of the paper is organized as follows. First we describe two space-based scenarios to illustrate the envisioned operating mode of a spacecraft agent, that is, to achieve high level goals through distributed planning

and scheduling. Then we present a generic architecture in section 3 that can be instantiated appropriately to implement an agent in the environment. The hierarchical syntax for modeling an agent's domain knowledge of tasks is presented in Section 4. Section 5 describes the protocol for inter-agent communication. Section 6 contains our approach to decentralization and coordination of tasks among agents. The functionality of the current Java prototype is described in section 7. Finally, we summarize our work in section 8 and lay out our future plan for extending the work.

2 EXAMPLE SCENARIOS

We present two space-based scenarios in this section to illustrate our envisioned distributed planning and scheduling. Onboard subsystems of a spacecraft are considered as agents in the first scenario, whereas individual spacecraft themselves are agents in the second scenario.

The first scenario is a modified version of the 'Spaceworld' model scenario presented in (Vere, 1983), where the goals are to send pictures of objects in deep space from the spacecraft to Earth. In the current New Millenium Remote Agent (NMRA) architecture (Pell, 1997; Chien, 1997), the executive will pass these goals to the planning and scheduling component, which recursively selects and schedules appropriate activities to achieve the goals. The component also synchronizes activities and allocates global resources over time such as power and data storage capacity. Thus, the planning and scheduling component of NMRA maintains a dynamic model for each of the subsystems to carry out its task.

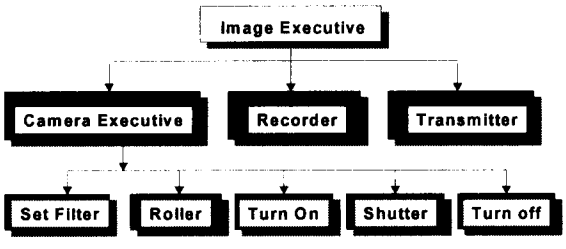

FIG 1: A Multi-Agent View of Intra-Satellite Image Request Processing Activities

In our envisioned distributed environment, the executive will delegate the task to an image executive (illustrated in figure 1), dedicated to managing goals related to obtaining, recording, and transmitting a picture to earth. This executive is only a high-level planning and scheduling agent, and it does not deal with resource allocation. The executive's plan can take one of the following two directions: 1) if the earth is in view (can be verified with the help of the camera executive) then it will send a request to the camera executive to take the picture followed by a request to the transmitter to transmit the picture; 2) if the earth is not in view then it will send a request to the camera executive followed by a request to the recorder to record for a subsequent transmission. The scheduling part of the image executive consists of specifying time intervals along with the requests to the camera executive, recorder and transmitter agent.

Upon receiving a request from the image executive, the camera executive schedules activities such as filter setting and turning on and off the camera, by taking into account its prior commitment of its own resources to other agents. Additionally, it will contact the roller agent to roll the spacecraft to position the camera for the desired picture and requests to maintain that position for a certain amount of time. A roller agent may just only be a resource manager. It will meet the request from other agents on a first-come-first-serve basis, and thus no serious scheduling activity is involved. If the camera agent fails to meet the request of the image executive then it will inform the executive with possible alternative slots for cooperation. The executive will coordinate with the camera agent to come up with an agreeable time slot. In a similar manner, the recorder and the transmitter agents will perform their own local scheduling and resource allocation in cooperation with the image executive.

To illustrate further our distributed environment where agents are individual satellites themselves, consider the scenario consisting of a constellation of satellites with different viewing capabilities (infra-red (IR), visible, or ultra-violet (UV)) orbiting a planet – and the goal is a full spectrum sweep of a certain swath of the planet. Traditionally, a human operator in the ground station would need to lookup to see which satellites with a given capability will be making a pass over the section of the planet indicated. Following this, the operator will need to address each satellite to request and organize the sweep with all relevant details down to the transmission of the data back to earth. Ideally, the operator should only need to transmit a high-level goal similar to "Take a full spectrum imagery of the area bounded by <latitude and longitude data> and transmit the picture back in two days". An executive level satellite can receive this command, decompose it, and then negotiate with the agent community (where each satellite in orbit is part of the community) to attempt to schedule a plan (as illustrated in figure 2). From there, each satellite can respond with information such as "will be passing over the site in 36 hours, I can generate in IR" or "will not be passing over the site for another 96 hours, I can not generate the image." Certain constraints may come into play also – UV and visible light sensors are only useful when that side of the planet is facing the sun. Responses to this situation may be similar to "will be passing over the site in 4 hours, but the site is currently on the dark side of the planet" or "will be passing over in 4 hours when site is on dark side, but will pass over again in 20 hours when it is local noon." Of course, there are certain requests which just cannot be fulfilled, it is the executive's job to notice these, come up with the "closest fit" to the request issued from the human operator and report back with the closest fit to ask for a go-ahead on that schedule.

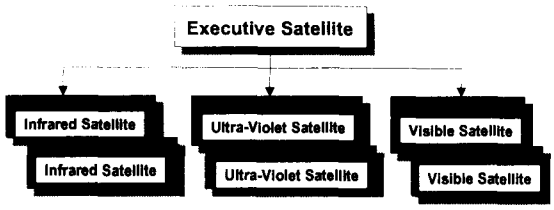

FIG 2: A Multi-Agent View of Inter-Satellite Image Request Processing Activities

When all of the planning has been performed through the negotiation, the executive satellite can issue the plan to all image gathering satellites (perhaps via the Tracking Data Retrieval Satellite System (TDRSS)). The satellites will receive their plan, and internally they will schedule their own control (perhaps via an internal agent network for subsystem control as described in the Spaceworld scenario) for setting up their imaging systems, recording the image, and then transmitting it. The satellite will pass over the section of the planet when the time is right, record the images, and transmit their image back to TDRSS. TDRSS will assemble the images when the whole spectrum has been covered, and transmit that back to the human controller.

3 AGENT ARCHITECTURE

Our architecture of an agent is essentially *deliberative*, i.e., there is an explicit symbolic representation of the model of the dynamic environment; agents make decisions via logical reasoning based on pattern matching and symbolic manipulation. Several different deliberative agent architectures have been proposed in the literature and two of them are most prominent: *horizontally layered architecture* (Ferguson, 1992) and *vertically layered architecture* (Muller, 1994). A layered approach models an agent as consisting of several hierarchical functional modules representing different requirements of an agent. Possible layers incorporate communication, reaction, inference for planning or scheduling, perception, knowledge maintenance, etc. Each layer in a horizontally layered architecture has access to both the perception and the action components whereas, in a vertical approach, only one layer has direct interface to the perception and action.

The architecture we have adopted is displayed in figure 3 and it fits into the vertically layered category. The three layers are world interface layer, inference layer, and network management layer. An agent's knowledge base is also split into three types corresponding to the three layers.

The *world interface layer* contains the agent's facilities for perception, action and communication. These activities require a detailed knowledge about the environment. An agent's world model contains information about the environment, for example, information about other agents such as their locations and capabilities. The world interface layer enables an agent to communicate with other agents in the environment to perform activities related to planning and scheduling such as sending and receiving requests, responding to a request, etc.

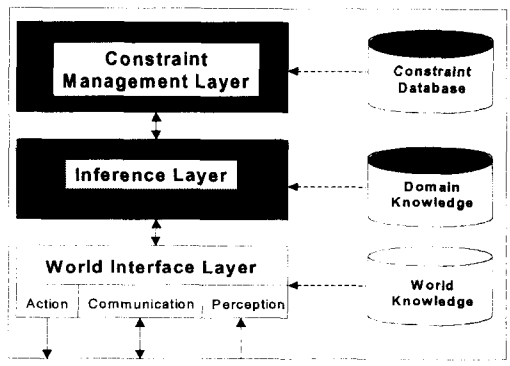

FIG 3: Vertically Layered Agent Architecture

Upon receiving a request from another agent through the world interface layer, the *inference layer* does planning or scheduling or resource allocation, depending on the type of the agent, using the available domain knowledge. The domain knowledge consists of the knowledge of the application, for example, description of different task abstractions and resources, effects of a task when it is carried out, and so on. Most part of the domain knowledge is static in nature in the sense that it remains the same for a particular application.

The job of the *network management layer* (also called the *temporal database layer*) is to manage the temporal constraint network generated during the planning and scheduling process by the inference layer. The constraint database is a persistent store for the constraint network. The layer provides the consistency checking service for the inference layer upon receiving a propagation of constraint from the inference layer.

To illustrate the interactions among the layers, we provide a small example of resource allocation activity of the recording agent described in the previous section. Suppose the maximum recording capacity at any time is 1GB (this is a resource constraint) and 700 MB of it has been scheduled for the time interval [800, 900]. This information along with the constraint is appropriately stored in the constraint database as a temporal network. The current state of the database is consistent. Now, if a request for 400MB for the interval [850, 950] arrives from the image executive, then the world interface layer will pass this request to the inference layer. The inference layer posts this request as a constraint to the network management layer. The network management layer tries to construct a consistent schedule combining the existing network with the incoming request. Upon failing, the layer informs the inference layer, which in turn informs the requesting agent through the world interface layer.

4 HIERARCHICAL MODELING

As mentioned in the introduction, a planning process based on a HTN representation first constructs a plan containing abstract high-level activities and then refines

these components in more detail. This process of refinement continues until these high-level activities themselves correspond to the physical actions in the real world. The advantage of this approach is that the feasibility of a plan can be studied incrementally. If an autonomous software agent is implementing the above refinement process then domain knowledge of the tasks and their components have to be codified in some language. We provide here a flavor of how our proposed HTN representations (similar to (Das, Fox et al., 1997)) look like. The syntactical details and expressiveness of this language are not important at this stage, as our objective is mainly to explain the concepts.

A *compound task* specification has three components: 1) a set of sub-components which specify the *subtasks* and *atomic actions* from which this compound task is built; 2) a set of *constraints* including constraints on ordering between subtasks; and 3) a set of effects when the task is carried out successfully. The two compound task specification for the image executive (figure 1) in the 'Spaceworld' example presented in section 2 is provided below:

compound-task	send-picture-to-earth(Object, Filter)@[S, E]
decomposition:	picture-object(Object, Filter)@[t1, t2]; transmit-picture(Object, Filter)@[t3, t4]
constraints:	S = t1; E = t4; t2 =< t3
effect:	received-on-earth(picture, Object, Color, S, E).
compound-task	send-picture-to-earth(Object, Filter)@[S, E]

decomposition:	picture-object(Object, Filter)@[t1, t2];
	record-picture(Object, Filter)@[t3, t4];
	transmit-picture(Object, Filter)@[t5, t6]

constraints:	S = t1; E = t6; t2 =< t3; t4 =< t5; t5 – t4 =< 100
effect:	received-on-earth(picture, Object, Color, S, E).

The first specification for the compound task send-picture-to-earth contains two subtasks: picture-object and transmit-picture. This task will be normally followed by the image executive, if the earth is in view; otherwise, the second alternative is pursued. This kind of options provides non-determinism of the unfolding process in hierarchical planning. The efficiency of the planner and the schedule quality depends on which option is chosen. If either one of these two tasks is carried out successfully then the earth will receive the picture as its effect.

The expression t2 =< t3 constrains the fact that a transmission cannot be started before taking the picture task is finished. Various types of constraints will be considered and propagated from agents to agents in a distributed planning process. *Hard constraints* represent those objective requirements and procedures that must be met to ensure a correct solution by an agent. The constraint just stated is an example of a hard constraint. On the other hand, *soft constraints* represent criteria that can be relaxed and are not essential for achieving a correct solution. For example, t5 – t4 =< 100 constrains that the time between the recording and transmitting should be less that 100. This can always be considered as a preference. The compound task specification for the camera executive is specified in a similar manner:

compound-task	picture-object(Object, Filter)@[S, E]

decomposition:	set-filter(Filter)@[t1, t2];
	roll-camera(Object)@[t3, t4];
	turn-on-camera@[t5, t6];
	shutter-camera(Object)@[t7, t8];
	turn-off-camera@[t9, t10]

constraints:	S = t1; E = t10; t2 =< t3; t4 =< t5; t6 =< t7; t8 =< t9
effect:	in-camera(picture, Object, Color, S, E).

The compound task specification for the executive satellite in our satellite constellation scenario (figure 2) is specified as follows, where an individual satellite is responsible for transmitting to the earth the image that its captures:

compound-task	full-spectrum-imagery(Object)@[S, E]

decomposition:	infra-red(Object)@[t1, t2];
	ultra-violet(Object)@[t3, t4];
	visible(Object)@[t5, t6]

constraints:	S <= t1; t2 <= E; S <= t3; t4 <= E; S <= t5; t6 <= E
effect:	received-on-earth(image, Object, S, E).

Each *primitive task* (or atomic action) in the Spaceworld scenario is specified along with its *precondition* and *effect*. The precondition of a primitive task must be satisfied before the action can be executed. The effect is the effect on the environment after the task has been executed successfully. An example representation corresponding to the shutter-camera primitive task is provided below:

primitive-task	shutter-camera(Object)@[S, E]
precondition:	locked-onto(Object); shutter-speed(Speed); camera(on)@[t1, t2]; platform(still)@[t3, t4]
constraint:	[S, E] ⊆ [t1, t2];[S, E] ⊆ [t3, t4]; Speed = E - S
effect:	in-camera(picture, Object, S, E).

The preconditions for the primitive task shutter-camera are as follows: the camera is locked onto the desired object, it is on, and the platform is still. The constraint [S, E] ⊆ [t1, t2] states that the camera is locked on at least during the interval [S, E]. Once the action is performed, the picture of the object for the interval [S, E] is in the camera. The primitive task specifications corresponding to the turning on action of the camera is simpler:

primitive-task	turn-on@[S, E]
precondition:	camera(off)
constraint:	E - S = 30
effect:	camera(on).

Our explicit representation and handling with resources usage is evident in the following example specification corresponding to the recording action:

primitive-task	record-picture(Object, Color)@[S, E]

precondition:	in-camera(picture, Object, Color, t1, t2);
	tape-recorder(on)@[t3, t4];
	data-mode(im2);
	tape-position(Start-Position);
	tape-empty(Start-Position, End-Position)

constraint:	[S, E] ⊆ [t3, t4]; E – S = 48; End-Position = Start-Position + 336;
effect:	tape-full(Start-Position, End-Position); recorded(Start-Position, End-Position, picture, Object, Color, t1, t2).

When a picture is recorded on a tape, the recorded portion of the tape resource becomes unavailable. This kind of effect on resources can be taken into account during the unfolding process of a plan construction to improve the efficiency in search for a solution. If an effect violates the

resource optimization function then this branch in the search space will not generate a potential solution.

5 DECENTRALIZAION AND COORDINATION

Decentralization or *decomposition* is the process of breaking down a problem into a set of subtasks. Since we have adopted a hierarchical modeling environment, there will be a natural way of decentralizing a task by an agent. We illustrate this process by considering the following specification of the compound task "send-picture-to-earth" already described in section 4. Suppose the image executive would like to produce a schedule for a picture. By looking at the above task decomposition it can decide to get this job done by the camera executive and the transmitter. Upon receiving a request from the image executive, the camera executive will follow the same decentralization process by using the composite task specification for picture-object.

When an agent decentralizes a task, it sends several requests to other agents. Correspondingly, the agent receives a set of replies according to the requests. It is not necessary that the agent receive the messages in the order they were sent. This is due to the fact that some agents are more efficient and some requests are harder to serve than others. Thus, every agent requires some amount of *coordination* of requests and answers. For example, messages may be tagged with priorities and an agent responds according to the priorities.

When the image executive agent sends the top-level task to its two subordinate agents then it expects two successful schedules, and their combination is the schedule of the whole task. It may so happen that one of the subordinate agents is not able to satisfy the constraint sent with the request. So the image executive may relax the constraint (e.g., by extending the interval) for that agent while imposing a constraint to another agent to compensate this. This process of relaxation and imposition of constraints is part of the coordination process.

In the above example specification, the image executive can achieve the goal in various ways. For example, it divides the interval [S, E] into two and asks two agents to plan in these two intervals so the constraint is automatically satisfied (Georgeff, 1983). The constraint need not be sent along with the request and the two agents can work simultaneously. Alternatively, it can ask the camera executive to work within the first half of the interval. If it fails then it can relax this constraint by stretching the interval. Another alternative approach is to let each agent cooperate with other agents to resolve any conflict. If an agent fails to provide a solution to a request, be it a planning or a resource request, an answer should provide reasons for failure in the constraint field.

We have argued that task decentralization by an agent and coordination is natural in a hierarchical modeling environment, which we have adopted for our distributed environment.

6 AGENT COMMUNICATION

Coherence, cooperation and conflict resolution can be improved by carefully designing the amount and type of communications among agents in the form of messages (Patil, et al., 1992). The information communicated should be relevant, timely and complete (Durfee, 1985). A message in our framework is composed of the following fields: 1) *sender*: sender of the message; 2) *receiver*: receiver of the message; 3) *identifier*: This is a unique identifier generated by the sender of the message; 4) *type*: a type describing whether a message is either a request or an answer to a request or an acknowledgement, etc.; 5) *task*: a task describes what the message is about, that is, whether it is planning (p), scheduling (s), resource allocation (r), or their combinations (p/s/r) or database related transaction such as insert, delete, update, lock and unlock; 6) *description*: in the case of a request this field describes the task requested, for example, description of a planning tasks. Similarly, in the case of an answer this field provides the answer of an earlier request or informing failure with explanations; 7) *constraint*: a constraint along with a request from a sender means that receiver meets the request by satisfying the constraint.

Following is an example of a p/s/r request message sent by the image executive agent to the camera executive agent, and its answer from the camera executive agent:

⟨m1, 'image executive', 'camera executive', request, p/s/r, image(star, color)@[800, 1100], exclude([900, 950]))⟩
⟨m1, 'camera executive', 'image executive', answer, p/s/r, image(star, color)@[950, 1050]⟩

The request is for an image of any time in the interval [800, 1100] subject to the constraint that no image in [900, 950]. The answer from the camera executive is that the task will be carried out in [950, 1050].

7 PROTOTYPE IMPLEMENTATION

The software platform used for the prototype development is Java and we used multicast sockets for inter-agent communication.

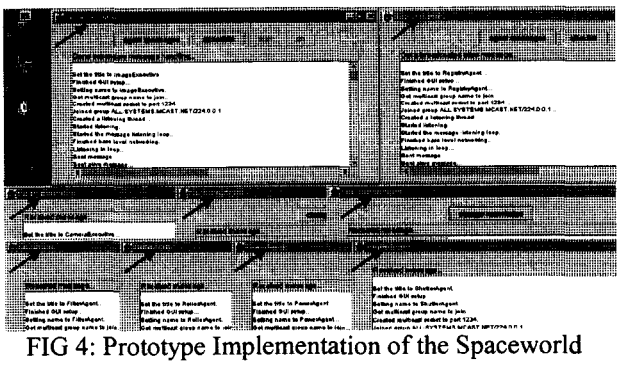

FIG 4: Prototype Implementation of the Spaceworld Multi-Agent Environment

Figure 4 shows the state prior to the start of scheduling of the implementation of the Spaceworld multi-agent environment. Each window is a separate process representing the agent pointed by a red arrow. All the agents are separate pieces of software, but we have

written a single unified program to launch the Java applications in different threads so only one Java virtual machine needs to be started up. Once all the agents are started up, they all negotiate with the Registry Agent to get confirmation that they are allowed to come online, and they negotiate with each other to allow to report ready to their executives. A newly registered agent obtains information about other agents in the environment from the Registry Agent. As shown in figure 5, the planning and scheduling process is initiated by pressing the 'schedule' button and a pop-up window will appear asking for the total time to be permitted (100 units) for the schedule. Using the hierarchical knowledge structure for 'send-picture-to-earth' specified in section 4, the executive agent then produces a schedule in cooperation with other agents.

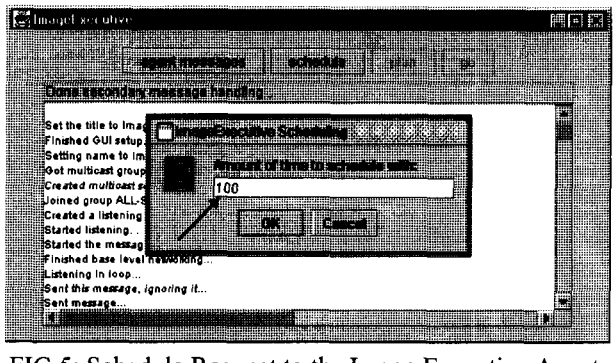

FIG 5: Schedule Request to the Image Executive Agent by Specifying the Allowed Time

The schedule produced can be viewed by pressing the 'plan' button as shown in figure 6.

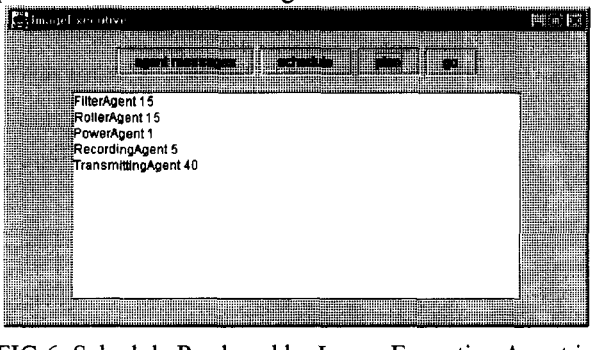

FIG 6: Schedule Produced by Image Executive Agent in Cooperation with Other Agents

Once scheduling has been finished, the actual execution begins by pressing the 'go' button. The Image Executive agent has the schedule already, so it simply sends the plan down the tree and each agent executes as needed and when needed. If a problem occurs somewhere down the line, then re-planning will be necessary. Within our Satellite Constellation scenario, there are many ways a schedule can be produced for a full spectrum sweep of a certain swath of the planet. Figure 7 is one such schedule produced by the prototype.

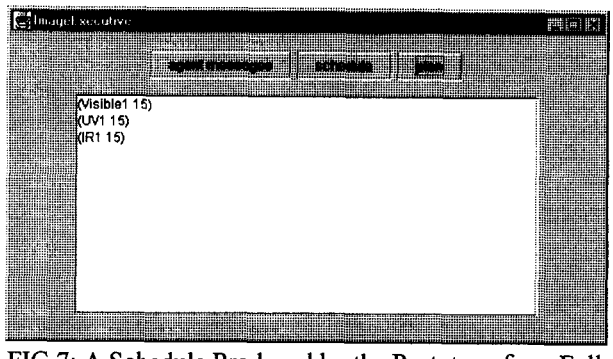

FIG 7: A Schedule Produced by the Prototype for a Full Spectrum Sweep

Our initial strategy was to accept the schedule that is found first during the search. For example, if the first satellite UV1 with ultra-violet imaging capability was not capable of carrying out the required task then the executive contacts the second satellite UV2 with the same capability and a schedule is constructed as shown in figure 8.

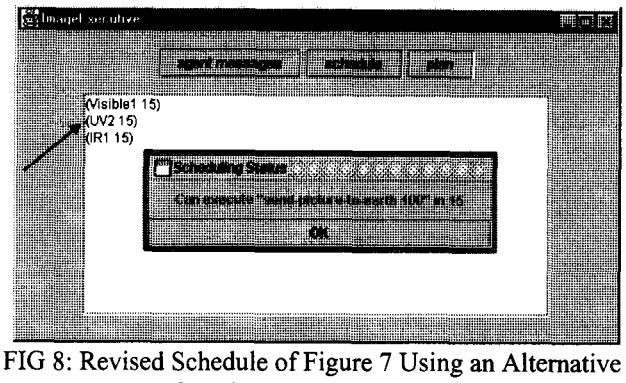

FIG 8: Revised Schedule of Figure 7 Using an Alternative for Ultra-Violet Spectrum

If both UV1 and UV2 are available then ideally the most optimized schedule among the two from the point of view of time and onboard resources should be produced. This kind of resource optimization issue will be addressed during our follow-on study. A constraint propagation paradigm usually allows encoding cost functions to produce the cost associated with each schedule.

8 CONCLUSION

In this paper, we have demonstrated how our distributed approach to planning and scheduling helps to achieve high-level goals and thereby enhances spacecraft autonomy. A hierarchical syntax has been adopted for representing domain knowledge of tasks by taking into account spacecraft operational aspects and resource constraints. The task decentralization problem has been solved by the use of the hierarchical knowledge structures. A constraint propagation paradigm has been employed for the required planning and scheduling tasks performed by an agent. The resource optimization problem has been addressed by its explicit representation within the problem domain. We have shown that a schedule can be generated (if it exists) and its quality can

be enhanced by the introduction of agent cooperation. A limited-scope prototype has been developed and demonstrated to assess overall feasibility.

This phase of the work has been carried out as part of NASA's effort on a program to develop and Remote Agents for flight software development. Various enhancements of our proposed distributed approach are planned during our follow-on effort.

Constraint propagation: Our current implementation of the constraint propagation activity is ad-hoc. For our follow-on development, we plan to use an off-the-shelf constraint-based temporal reasoning engine such as Honeywell's TMM (Time Map Manager), NASA's HSTS problem solving framework, and Prolog II software system. The advantage with a Prolog II type of declarative system is that it will allow us to perform high-level symbolic reasoning required as part of the planning and scheduling process, thus reducing the burden from the tedious development process in an imperative environment such as Java.

Resource optimization: The hierarchical representation of compound and primitive tasks of the application domain incorporates information about their resource consumption, and a database containing up-to-date resource status will be maintained. Therefore, during the hierarchical planning process, which unfolds a compound task into a set of subtasks and resolves task preconditions using the information in the resource database, the system can choose an unfolding path that consumes the least amount of resource. This process which we plan to implement guarantees an optimized plan to achieve the goal from the point of view of resource usage.

Inter-agent communication: We plan to take advantage of CORBA or KQML or ISP for enhanced cross-platform communication.

Inter-agent negotiation: Currently, we assume friendly relationship among agents and therefore no negotiations occurred between two agents. Although this is appropriate in an environment where agents represent onboard subsystems (e.g., the Spaceworld scenario), it may not be the case in a scenario involving a constellation of satellites owned by various companies, agencies, and countries. In the future, we will assume one of various types of relationships between two agents including friendly, subservient, and bargain. An agent is awarded or penalized according to its use of resources. The existence of bargain type relationship therefore introduces the possibility of negotiations between two agents without sacrificing their own interests.

Acknowledgements: This work was performed under contract NAS5-98120 with NASA's Goddard Space Flight Center.

9 REFREENCES

1. Boddy, M. (1994). "Temporal reasoning for planning and scheduling." SIGART Bull. 4(3).
2. Burke, P., and Prosser, P. (1991). "A distributed asynchronous system for predictive and reactive scheduling." Artificial Intelligence in Engineering 6: 106-124.
3. CACM (1994). Intelligent Agents - Communicuation of the ACM, ACM Press.
4. Chaib-draa, B. M., Mandiau, R., and Millot, P. (1992). "Trends in distributed artificial intelligence." AI Review 6: 35-66.
5. Chapman, D. (1987). "Planning for Conjunctive Goals." Artificial Intelligence 32: 333-378.
6. Chien, S., DeCosta, D., Doyle, R. and Stolorz, P. (1997). Making and impact: artificial intelligence at the Jet Propulsion Laboratory. AI Magazine. 18.
7. Conry, S. M., R. and Lesser, V.R. (1988). Multiagent negotiation in distributed planning. Reading in Distributed Artificial Intelligence. A. a. G. Bond, L., Morgan Kaufmann: 367-384.
8. Cooper, R. G. (1993). Winning at New Products: Accelerating the Process from Idea to Launch. New York, Addison Wesley.
9. Currie, K. a. T., A. (1991). "The Open Planning Architecture." Artificial Intelligence 52(1).
10. Das, S. K., J. Fox, et al. (1997). Decision Making and Plan Management By Autonomous Agents: Theory, Implementation, and Applications. First International Conference on Autonomous Agents, California.
11. Dean, T., Firby, R. J., and Miller, D. (1988). "Hierarchical planning involving deadlines, travel time, and resources." Computational Intelligence 4: 381-398.
12. Durfee, E. H., Lesser, V.R., and Corkill, D.D. (1985). Increasing coherence in a distributed problem solving network. 8th International Joint Conference on Artificial Intelligence.
13. Ferguson, I. A. (1992). Touring Machines: AN Architecture for Dynamic, Rational. Mobile Agents, Computer Laboratory, University of Cambridge.
14. Fikes, R. a. N., N. (1971). "STRIPS: A new approach to the application of theorem proving to problem solving." Artificial Intelligence 2: 189-208.
15. Genesereth, M. R., and Ketchpel, S.P. (1994). "Software agents." C ACM 37: 48-53.
16. Georgeff, M. (1983). Communication and interaction in multi-agent planning. AAAI.
17. Georgeff, M. P. (1987). "Planning." Annual Review in Computer Science 2: 359-400.
18. Le Pape, C. (1990). Constraint propagation in planning and scheduling.
19. McAllester, D. a. R., D. (1994). Systematic nonlinear planning. AAAI-94.
20. Muller, J. P. a. Pischel., M. (1994). Modeling interacting agents in dynamic environments. 11th European Conference on AI.
21. Muscettola, N. (1994). Integrating planning and scheduling. Intelligent Scheduling. M. a. Z. Fox, M., Morgan Kaufmann.
22. Patil, R. S., R. E. Fikes, et al. (1992). The DARPA Knowledge Sharing Effort: Proc. of Knowledge Representation and Reasoning (KR&R-92).: 777-788.
23. Pell, B., Bernard, D.E., Chien, S.A., Gat, E, Muscettola, N., Pandurang Nayak, P., Wagner, M.D., and Williams, B.C. (1997). An autonomous spacecraft agent prototype. 1st International Conference on Autonomous Agents.
24. Rosenschein, J. S. (1982). Synchronization of multi-agent plans. AAAI, pp.115-119.
25. Sacerdoti, E. (1974). "Planning in a hierarchy of abstraction spaces." Artificial Intelligence 5.
26. Simmons, R. (1991). "Coordinating planning, perception, and action for mobile robots." SIGART Bulletin 2: 156-159.
27. Tate, A. (1977). Generating project networks. 5th Int. Joint Conf. on Artificial Intelligence.
28. Truszkowski, W., Odubiyi, J., and Ruberton, E. (1995). "An agent model in a multi-agent system architecture for automating distributed system." *Proceedings of the 1st ICMAS*, pp. 463.
29. Vere, S. A. (1983). "Planning in Time: Windows and Duration for Activities and Goals." IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-5(No. 3).
30. Wilkins, D. E. (1988). Practical Planning: Extending the Classical AI Planning Paradigm, Morgan Kaufmann.