MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INPE-14076-PRE/9245

A MULTI-AGENT BASED GROUND-OPERATIONS AUTOMATION ARCHITECTURE

Adriana Carniello Biancho
Andréia C. De Aquino
Maurício Gonçalves Vieira Ferreira
José Demisio Simões da Silva
Luciana Seda Cardoso

Paper presented at the IAC 2006 - 56th International Astronautical Congress from 17 to 21 October 2005, Fukuoka, Japan.

INPE
São José dos Campos
2006

A Multi-agent Based Ground-Operations Automation Architecture

Adriana C. Biancho[*], Andreia C. de Aquino[†], Mauricio G. V. Ferreira[‡], José Demisio S. da Silva[§] and Luciana S. Cardoso[¶]
National Institute for Space Research, São José dos Campos, SP, 12227-010, Brazil

Reducing the costs of space operations is an increasing demand which may be achieved by automating the ground segment operations. This work proposes a *M*ulti-*A*gent *G*round-Operation *A*utomation architecture named MAGA. This architecture aggregates agents responsible for automating space operation planning and execution. MAGA architecture manages ground resource allocation for multi-satellite tracking and plans the satellite control operations. For the planning process, it considers the temporal restriction of a satellite visibility period which may have its tracking period reduced due to a time conflict with another satellite passage. MAGA architecture analyzes whether the satellite tracking period is sufficient to achieve all the tracking goals and allows the elimination of lesser priority goals in case of insufficient time. Therefore, it assings priorities to goals and allows priorities to be reconfigurable for the next satellite tracking requirements.

I. Introduction

IN the field of satellite control, there is a general interest in automating the space operation planning and execution. The automation of space operations represents a way of reducing in-orbit satellite maintenance costs.

By adopting the ideas proposed in this work, we intend to keep the number of personnel responsible for satellite control towards the project of new satellites. The aim is to be able to come up with a solution to fit Space Program budgets concernig the launching of new satellites.

The automation described in this work is strongly concerned with the task of controlling multiple satellites, dealing with temporal restrictions as well as the sharing of ground resources, and managing the conflicts that may arise. Ground resources consist of ground stations where the antennas which capture satellite signals are located. A satellite is visible to a ground station during a limited period of time (time window) called the satellite visibility period.

Due to the fact that satellites share the use of ground stations, the visibility period of one satellite may conflict with the visibility period of another. When this situation takes place, the visibility period conflicting part of the satellite with less priority must be cancelled.

A relevant issue in managing multi-satellite visibility periods is that canceling a satellite conflicting visibility period means reducing the time window in which the satellite will be tracked by the ground station. This time window reduction requires the elimination of some goals in order to fit a subset of the original goals into the new time window.

MAGA architecture identifies multi-satellite conflicting tracking periods and generates a control plan for each satellite. For plan generation, it reasons whether or not there is sufficient time to achieve all the tracking goals and allows disconsidering goals in case of insufficient time (when a time window reduction occurs).

* Doctorate student, Applied Computing Postgraduate Program (CAP), Av. dos Astronautas 1758 Jd. Granja, adcarnie@lac.inpe.br.
† Doctorate student, Applied Computing Postgraduate Program (CAP), Av. dos Astronautas 1758 Jd. Granja, ancarnie@lac.inpe.br.
‡ Doctor researcher, Satellite Control and Tracking Center (CRC), Av. dos Astronautas 1758 Jd. Granja, mauricio@ccs.inpe.br.
§ Doctor researcher, Applied Mathematics and Computing Associated Laboratory (LAC), Av. dos Astronautas 1758 Jd. Granja, demisio@lac.inpe.br.
¶ Master student, Ground System Department (DSS), Av. dos Astronautas 1758 Jd. Granja, luciana@dss.inpe.br.

Satellite control plans, named Flight Operation Plans (FOP), contain all the operations related to the control of in-orbit satellites. FOP generation requires that these operations be inserted at specific instants of time in order to achieve the satellite tracking goals. These goals comprise, for instance, calibration measure execution, telemetry reception, telecommand sending, and distance and speed measure execution.

MAGA architecture also allows automated plan execution. Nowadays satellite control plan execution is performed by human operators. In MAGA architecture, the task of human satellite operators is solely restricted to the execution monitoring.

The following section introduces some concepts about the automation of ground segment operations. Section 3 relates some fundamental concepts of the Artificial Intelligence planning area with the satellite control domain and presents an overview of MAGA architecture. Section 4 describes the MAGA architecture agents responsibilities whereas Section 5 presents the architecture general behavior. Following a discussion of related research in Section 6, the paper concludes in Section 7 with a summary of this work contributions.

II. Ground Segment Automation

Brazilian space missions are classified as Low Earth Orbiting missions. This type of mission requires an automation system with emphasis on controling and monitoring ground systems to achieve the time-critical operations during the short period the satellite is visible to the ground station.

A Low Earth Orbiting mission is based on off-line control due to the short duration of the satellite visibility period. An appropriate level of on-board autonomy guarantees the satellite survival when out of visibility, i.e. when the satellite is not visible to any ground station.

During the visibility period, the human operator uplinks operations/commands and files of pre-prepared commands named telecommands. The human operator also downlinks data stored by the satellite in the on-board memory over the non-visible period.

The real-time telecommands uplinked to a satellite are those concerning its equipment status checking and control. The uplink of a real-time telecommand is followed by the downlink of a corresponding data which informs the human operator the result of the telecommand on-board execution. The downlink of data from the in-orbit satellite is named telemetry.

Although a telecommand execution result may not be the expected one, in general the immediate reaction to anomalies is not possible due to the short duration of the visibility period. Therefore we propose an off-line replanning when anomalies take place.

In MAGA architecture there are two levels of automation – the plan generation and the operation execution. The plan generation automation means remodeling the Satellite Control Software to be capable of scheduling the tracking of many satellites and considering operation priorities for the Flight Operation Plan generation.

The operation execution automation is concerned with displaying telecommands on-line and running them automatically. This automation comprises satellite health status checking, uplink of payload and routine platform operations, and notification to remote operators if problems occur[1].

Therefore, aiming at offering more configurability to the planning task and at reducing the operational costs of in-orbit satellite control, the ground segment automation comprises the automated planning of ground resources and flight operations, and the automated ground execution of these operations.

III. Multi-agent Planning Architecture

Planning problems involve a set of initial states, a set of goals and the corresponding actions that contribute to achieve these goals. A planner agent is an agent responsible for solving planning problems. This type of agent represents a planning problem by propositional/first-order representations that allow effective heuristic derivations and the development of planning algorithms (planners) used to plan a sequence of actions whose execution will lead to the desired goals.

The representation of planning problems has been a concern since 1971 when Fikes and Nilsson developed the STRIPS language[2]. From this time on, other researchers have proposed planning problem representation languages based on STRIPS aiming at developing a more expressive language for real planning problems.

In 1998, the Artificial Intelligenge Planning groups made an attempt to standardize a language for real planning problem description proposing PDDL – Planning Domain Description Language[3, 4]. PDDL has been used as the standard language in international planning competitions allowing planning problems to be represented in a comparable notation and planner performance to be evaluated.

In its version 2.2, PDDL currently allows planning problem modellers to specify actions with duration and deterministic unconditional exogenous events, which are facts that will become true or false at time points that are

known to the planner in advance, independently of the actions that the planner chooses to execute. These two features are very important for the generation of Flight Operation Plans due to the fact that space operations have duration and must be adjusted to the well-defined time windows of the satellite tracking periods.

PDDL separates the description of parameterized actions (planning domain behavior) from the description of the initial conditions and the goals to be achieved (problem instance). Therefore, PDDL distinguishes the Planning Domain Description from the Problem Description that together represent a planning problem. By separating these definitions, PDDL allows the same Planning Domain Description to be used by several different Problem Descriptions in order to produce different planning problems for the same planning domain.

A Planning Domain Description file contains the domain types, functions, predicates and actions. An action is associated to a precondition and an effect which are conjunctions of literals that declare the environment states before and after the action execution, respectively. An action may also be assigned an execution duration.

A Problem Description file contains the objects present in the problem instance, the initial states and the goals.

Fig. 1 illustrates a planner agent that uses PDDL for specifying planning domains and problems. The planner agent uses some planning algorithm to generate plans for goal achievement.

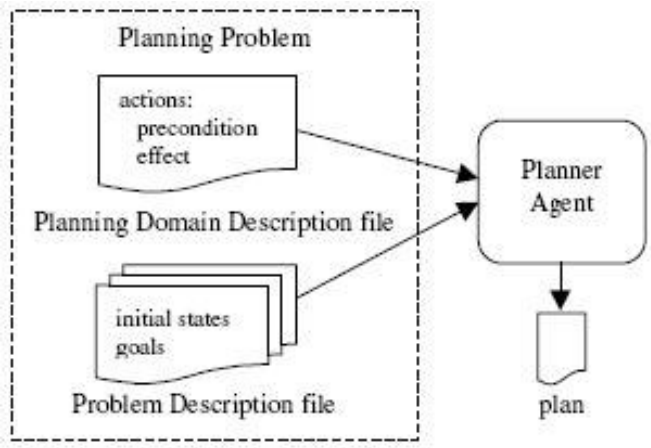

Figure 1. Planner agent

In the satellite control domain, a planning action consists of a space operation that is concerned either with the control of satellite on-board equipments status or with obtaining the satellite exploitation expected products. Planning goals are represented by tracking goals to which we propose the assignment of priorities according to their execution relevance in the domain.

MAGA architecture is a multi-agent architecture formed by planner agents[5]. In a multi-agent system, planning actions can be divided into three distinct stages – generating plans, coordinating plans, and executing them[6].

In MAGA architecture the plan generation task is distributed between two agents, forming a distributed planning architecure. Actually, the term distributed planning can refer to the plan generation process as well as to the plan execution and coordination processes[7]. MAGA architecture adopts the term distributed planning concerning solely the plan generation process once its execution and coordination are centralized.

So, the plan generation process is split between agents which cooperate with each other to reach a final plan. The planning results are integrated into a single final plan, the Flight Operation Plan, for automated execution. Fig. 2 illustrates MAGA architecture.

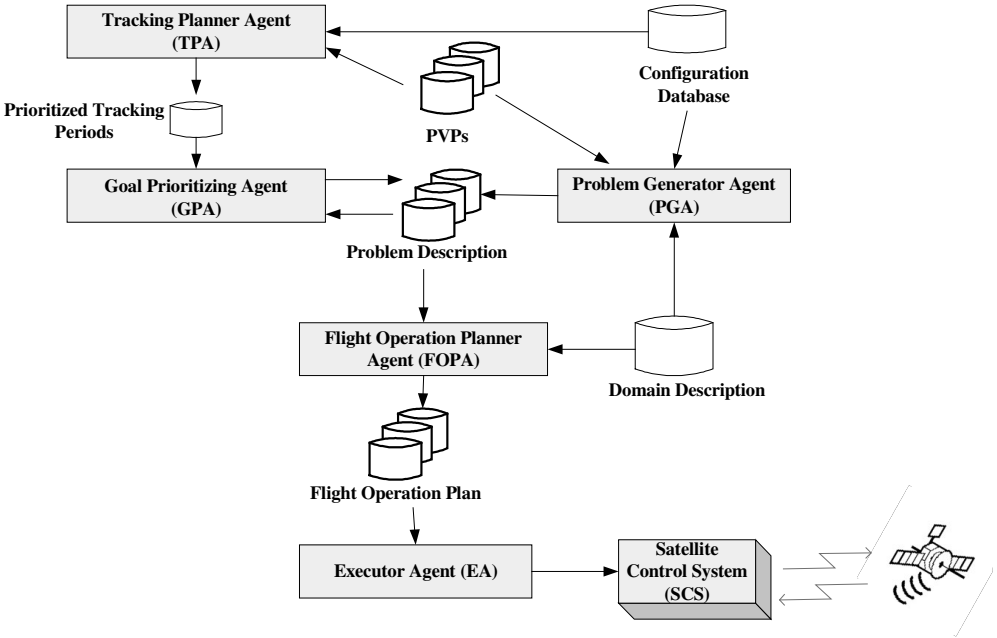

Figure 2. *M*ulti-*A*gent *G*round-operation *A*utomation architecture (MAGA architecture)

IV. MAGA Architecture Agents

The following sub-sections describe the MAGA architecture agents responsibilities.

A. Problem Generator Agent (PGA)

This agent is responsible for generating the PDDL Problem Description file for each satellite pass[8]. It senses the satellite control environment through the following perceptions: a Configuration Database, Pass Visibility Prevision files (PVP files) and the PDDL Planning Domain Description file.

The Configuration Database contains all the satellites and ground stations configuration parameters. These parameters are not specific to a satellite tracking but are applied to ground stations and satellites in general. A Pass Visibility Prevision file contains data about the future passes of one satellite to a specific ground station. This plan is generated by the Flight Dynamics System which is an external system to MASOA architecture that provides, for instance, predictions about the ground antenna pointing data.

Once having the above information, the Problem Generator Agent automatically generates the PDDL Problem Description file for each satellite tracking period. The Problem Description file contains the satellite tracking period initial states, the deterministic unconditional exogenous events and the goals that must be achieved at the end of the tracking period.

B. Tracking Planner Agent (TPA)

This agent generates Tracking Plans (TP) that define which satellites can be tracked by a ground station, the order they can be tracked and the tracking duration.

The time window a satellite can be really tracked is named the satellite tracking period which may comprise its whole visibility period or just a portion of it. A satellite tracking period comprises just a portion of its visibility period when the visibility period of a second satellite conflicts with that of the first one. In this case, the tracking period of the satellite with less priority is reduced to avoid time conflict.

So, TPA manages the problem of multi-satellite tracking with conflicting visibility periods (concerning the same ground station) by cancelling or shortening the tracking of the satellite with less priority.

TPA also applies other criteria to define the sequence of satellites to be really tracked. The restrictions imposed by these criteria are: (i) the tracking of satellites with visibility periods shorter than a pre-defined duration are not considered; (ii) when a satellite is visible to two or more ground stations simultaneously, it must be tracked by the

ground station with the highest priority; and (iii) there must be a minimal time interval between the end of a satellite tracking and the beginning of the tracking of another one.

C. Flight Operation Planner Agent (FOPA)

This agent is responsible for generating the Flight Operation Plan (FOP) which contains satellite control operations (planning actions) to be executed by an executor agent during the satellite tracking period. FOPA generates a Flight Operation Plan for each satellite to be tracked by a specific ground station antenna. The general goal is to ensure in-orbit satellites are operating accordingly and are obtaining the desired users' requests.

FOPA has as input the Planning Domain and Problem Description files, written in PDDL 2.2. For plan generation, it uses the temporal planner LPG-TD[9] as its reasoning mechanism.

D. Goal Prioritizing Agent (GPA)

This agent acts when a satellite tracking period is reduced in order to avoid time conflict with another satellite. In this case, the satellite tracking period is generally not enough to execute all the original goals previously defined in the PDDL Problem Description file which was generated by the Problem Generator Agent (PGA).

The Goal Prioritizing Agent makes possible, by attributing priorities to goals, to consider solely the most relevant goals for the Flight Operation Plan generation. The aim is to consider solely the most important goals for the planning process so that the planning actions can fit into the short-time tracking period.

In order to implement this solution, each goal must be annotated with a priority which comprises a symbolic value that might be changed from one satellite tracking to another, to better specify the need for the goal execution in the next tracking period.

E. Executor Agent (EA)

This agent is responsible for the Flight Operation Plan automated execution at the specified instants of time. Obeying the sequence of operations (planning actions) specified in the Flight Operation Plan, the Executor Agent calls the Satellite Control System functions related to each plan operation. In case of anomalies, the Executor Agent notifies the remote human satellite operator and allows his intervention.

If EA was not present in the satellite control environment, the Satellite Control System functions would be called by a human satellite operator. In MAGA architecture, the functions of human satellite operators are reduced to the monitoring of operation execution and the managing of execution failures. This work load reduction facilitates the work of these professionals besides allowing to have a short number of them to control the tracking of several satellites by a group of ground stations.

So, EA allows to reduce the need of human operator presence. Considering the increasing number of new Brazilian satellite projects, this solution allows to keep the number of human operators which is a cost-saving profit.

V. MAGA Architecture Behavior

Considering the Configuration Database, the PVP files and the PDDL Planning Domain Descriptions as input, the Problem Generator Agent (PGA) generates a PDDL Problem Description file for each specific satellite tracking. In parallel, the Tracking Planner Agent (TPA) obtains from the PVP Database all the satellites visibility periods initial and ending times for a specific ground station.

Once having this information, this agent compares the several satellites initial and ending times, reducing the least priority satellite tracking period when some intersection occurs. The satellite tracking periods that have their time window reduced are annotated with the flag *reduced* to sign it.

After performing its task, the Tracking Planner Agent (TPA) communicates with the Goal Prioritizing Agent (GPA) providing the sequence of satellites to be tracked by a specific ground station and the several satellite tracking periods annotated with the flag *reduced* when this is the case.

At first, the GPA checks for the reduced flag in each satellite tracking period. Secondly, for the satellite tracking periods annotated with this flag, the GPA selects the least priority goal to be eliminated from the PDDL Problem Description file in a third step. These three tasks are performed by the Goal Prioritizing Agent sub-components named Tracking Reduction Checker, Goal Selector, and Goal Editor, respectively.

Once having edited the PDDL Problem Description file to fit the most relevant goals into the satellite reduced tracking period restriction of time, FOPA finally generates a Flight Operation Plan for each satellite. The generated FOPs are hence ready for being automatically executed by the Executor Agent.

VI. Related Works

Growing attention has been paid to multi-agent planning systems in Artificial Intelligence. In Ref. 10 there is a proposal close to ours due to the adoption of distributed planning agents. However, the system they propose consists of a multi-agent planning system for deep space exploration.

The authors propose a formal planning model for the planning domain description. Although the model is a valid one, they justify the need for it by pointing out the STRIPS operators defficiencies, which is solely a subset of PDDL language. PDDL was not referenced.

We consider that using PDDL or extending it when it is not sufficient to model a planning domain is a good practice since the Articial Intelligence Planning groups are making a general effort to standardize PDDL and make it a practical planning language.

In Ref. 11 the author considers PDDL and proposes a planning framework that also adopts the temporal planning paradigm. Besides that, this framework also reasons about goals with priorities but does that in a distinct way from ours.

Coddington's framework edits the plan when there is insufficient time available to achieve all the goals. It removes from the plan a goal and all of its associated actions and constraints. But for doing that, there is the associated cost of maintaining the dependencies between actions and goals during the planning process.

In MASOA architecture, when there is insufficient time to achieve the goals (reduced tracking period), the plan is not generated thus avoiding plan editing after its generation. Instead of generating the plan and editing it afterwards, the Goal Prioritizing Agent (GPA) edits the PDDL Problem Description file (input for the planner) until there is sufficient time to achieve a subset of the original goals.

By solely editing the input for the planner, we avoid having to interfere on the planner activity to get the dependencies between actions and goals during the planning process. Therefore, in MASOA architecture the planner is considered as a black-box subcomponent with its input being adjusted to portray the varying context of the satellite control domain. The advantage of treating the planner as a black-box subcomponent is the possibility of replacing it as the Artificial Intelligence planning area evolves.

VII. Conclusions

In this paper a new multi-agent automation planning and execution architecture is proposed to the context of multi-satellite control domain. MAGA architecture automatically plans the space operations to be uplinked and downlinked regarding the restricted period of time that low Earth orbiting satellites are visible to ground stations.

MAGA architecture solves the problem of satellites with conflicting time tracking periods. When a set of satellites share the same ground station, the visibility period of one satellite may intersect with the visibility period of another one. When this situation takes place, the Tracking Planner Agent reduces the tracking period of the least priority satellite. This time period reduction means the MAGA architecure planner agent (FOPA) will not have sufficient time to achieve the entire set of original goals (planning goals).

At this point, the Goal Prioritizing Agent selects and removes goals in order to generate a plan that achieves the set of remainder goals within the reduced satellite tracking period restriction of time.

Goal priorities are defined by the satellite operator and are configurable in order to portrait the next satellite tracking requirements. So, goals disconsidered for the generation of a satellite control plan may have their priorities augmented in the next plan generation. This enables to configure the planning task to reflect the actual satellite tracking requirements.

The Goal Prioritizing Agent reasoning mechanism allows to fit the most relevant goals into the satellite reduced tracking period thus avoiding the risk of accidentally disconsidering crucial operations to the satellite control.

Concerning the aspect of operation execution automation, MAGA architecture facilitates the work of human satellite operators by performing the most repetitive tasks. By adopting this architecture concepts, we expect to reduce the satellite operational costs and to increase the configurability of the space operation planning task, facilitating the functions of the satellite planning and operation staffs.

Acknowledgments

This work was financially supported by the Brazilian Coordination of Postgraduate Personnel Improvement (CAPES).

References

[1]Monham, A., and Rudolph, A., "Controlled Approach to Automated Operations", The Space Operations 2004 Conference, Montreal, Canada, 2004.

[2]Gerevini, A., Saetti, A., Serina, I., and Toninelli, P., "LPG-TD – a Fully Automated Planner for PDDL2.2 Domains," American Association for Artificial Intelligence, Università degli Studi di Brescia, Brescia, Italy, 2004.

[3]Fox, M., and Long, D., "PDDL 2.1 – An Extension to PDDL for Expressing Temporal Planning Domains", Journal of Artificial Intelligence Research 20, 2003.

[4]Edelkamp, S., and Hoffmann, J., "PDDL 2.2 – The Language for the Classical Part of the 4th International Planning Competition Technical Report", Albert Ludwigs Universität, Institüt fur Informatik, Freiburg, Germany, 2004.

[5]Russell, S. and, Norvig, P., Inteligência Artificial, Translation of 2nd ed., Editora Campus, 2004.

[6]Ferber, J., Multi-Agent Systems – An Introduction to Distributed Artificial Intelligence, Addison-Wesley, London, 1999.

[7]Weiss, G., Multiagent Systems – a Modern Approach to Distributed Artificial Intelligence, The MIT Press, 1999.

[8]Cardoso, L., Ferreira, M., Orlando, V. and Biancho, A., "Aplicação da Tecnologia de Agentes de Planejamento em Operações de Satélites", Anais do VII Simpósio Brasileiro de Automação Inteligente, São Luís - Maranhão, 2005.

[9]Gerevini, A., Saetti, A., Serina, I., and Toninelli, P, "LPG-TD – a Fully Automated Planner for PDDL2.2 Domains", American Association for Artificial Intelligence, Università degli Studi di Brescia, Brescia, Italy, 2004.

[10]Xu, R., Cui, P., Xu, X., and Cui, H., "Multi-Agent Planning System for Spacecraft", Proceedings of the Second International Conference on Machine Learning and Cybernetics, IEEE Computer Society, Xi'an, November, 2003.

[11]Coddington, A., "A Continuous Planning Framework with Durative Actions", Proceedings of the Ninth International Symposium on Temporal Representation and Reasoning, IEEE Computer Society, 2002.