Simulating On-Board Autonomy in a Multi-Agent System with Planning and Scheduling

Amedeo Cesta[†*], **Jorge Ocon**[‡], **Riccardo Rasconi**[†] and **Ana María Sánchez Montero**[‡]

[†] ISTC-CNR, Italian National Research Council, Rome, Italy
[‡] GMV, Advanced Space Systems and Technologies, Tres Cantos, Spain

Abstract

The paper describes an experiment for simulating on-board autonomy capabilities within a larger multi-agent system that provides product delivery for a distributed Earth observation system. It is shown how robust state of the art technology for constraint-based scheduling and execution has been customized for endowing agents with a basic autonomy capability.

Introduction

The Distributed Agents For Autonomy (DAFA) study supported by the European Space Agency (ESA) aimed at demonstrating the advantages of using distributed agents in a space system. During the study, a number of suitable case studies were evaluated as candidates for implementation, for example complex missions like Exomars, multi-satellite missions like Swarm, formation flying missions like Darwin, as well as GMES (Global Monitoring for Environment and Security). Finally, a software demonstrator for DAFA was implemented on the basis of a GMES-like scenario, and Agent-Based Programming was experimented with the twofold purpose of demonstrating the high level of realism in the obtainable GMES services within such a software paradigm, and obtaining a platform for experimenting new technologies at work in a realistic framework. In this second line of work the initial design of the demonstrator (see (Ocon et al. 2008)) has been recently integrated with a rather simple on-board re-planning capability in order to show such features within the Earth observation scenario.[1]

As shown by the first complete experiments of on-board space autonomy (e.g., (Muscettola et al. 1998) and (Chien et al. 2005)) adding goal-oriented capabilities on-board a space system opens interesting new scenarios and allows to foresee new challenges. In particular we were interested in showing, even in small scale, how a limited plan adaptation ability may have an impact on the overall space mission quality, by increasing its reliability, flexibility and responsiveness to science opportunities. In addition, having the ability of on-board re-planning enables the possibility to respond to possible failures. In the simpler cases, spacecraft can enact predefined emergency plans when needed, while in the more complex cases spacecraft can enable alternative plans synthesized on occasion. Mission flexibility is a byproduct of the previous features: re-planning capabilities directly translate into the capability of "adapting" to unforeseen situations, which comprise the ability to detect and take advantage of science opportunities, seen as "positive" contingencies. Lastly, the capability to make decisions directly on board generally allows to enable timely reactive behaviors, thus increasing the science product timeliness, which results in a significant increase of the return of mission investment, not to mention the capability of early detection of disastrous events, which is often invaluable.

This paper describes a specific activity within DAFA to endow the on-board segment of basic autonomy abilities to perform local plan adaptation. In particular we show how off-the-shelf constraint-based scheduling technology has been integrated in the agent-based system allowing more complex nominal behavior on board. Advantages of such an integration are demonstrated through a complete running example.

The GMES Scenario and the DAFA Demonstrator

GMES (Global Monitoring for Environment and Security - http://www.gmes.info/) is a major European Earth Observation Programme initiative for the establishment of All-European Earth observation capabilities. The GMES services are grounded on the coordination of a number of Earth Observation (EO) satellites and on an effective use of different ground segment facilities. In general we can say that our choice of GMES as a test case has been driven by the extreme relevance of the services that this scenario offers; such services can be classified in three major categories:

– **Mapping**, including topography or road maps but also land-use and harvest, forestry monitoring, mineral and water resources that do contribute to short and long-term management of territories and natural resources. This service generally requires exhaustive coverage of the Earth surface, as well as the archiving and periodic updating of data;

– **Support** to civil protection institutions responsible for the security of people and property for emergency management in case of natural hazards. This service focuses on

[*]For correspondence: amedeo.cesta@istc.cnr.it

[1]A complete description of the latest version of the DAFA demonstrator can be found in (Ocon 2009).

the provision of the latest possible data before intervening;

- **Forecasting**, to be applied for marine zones, air quality or crop fields. This service systematically provides data on extended areas permitting the prediction of short, medium or long-term events, including their modelling and evolution.

Different Software Agents. Within the DAFA study, agent autonomy has been ranked in three categories according to the employed decision-making capabilities of the spacecraft. In particular, *Type I Spacecraft* are those that manage on-board pre-defined plans that cannot be modified, and the users work by subscription to the corresponding spacecraft; *Type II Spacecraft* manage plans that can be modified to adapt to near real time (NRT) requirements, and rescheduling is possible in *emergency mode* situations only; finally, *Type III Spacecraft* have the capability to detect scientific events over specific areas, where the pre-programmed scientific event detections can trigger a direct modification of the spacecraft's on-board plan (autonomous re-planning).

Autonomy can be generally achieved by means of different technologies, and multi-agent systems (MAS) offer one particularly valid alternative. They represent a software engineering paradigm where systems are built-up out of individual problem-solving agents pursuing high-level goals; moreover, due to the MAS-based homogeneous and powerful design environments available nowadays, the development process of MASs is facilitated, as is the deployment of simulation systems. The main objective of the DAFA study is to prove that autonomous MASs are able to meet the necessary requirements in order to foster their possible use in real-world operational systems.

The DAFA Multi-Agent Service Provider. We have developed a quite realistic demonstrator of a GMES service by devising a system which is completely agent-programmed. The main contribution we intend to provide within the DAFA distributed architecture is the capability to autonomously schedule observations on behalf of EO Satellite fleets.

This issue entails large search spaces, complex constraint checking, such as power, thermal data, and limited time over the target, and complex bottlenecks related to planning, non-operational times, downlink windows and limited capacities.

The selected GMES scenario deploys a set of Type I, Type II, and Type III spacecraft that cooperate for the fulfillment of EO operations. Within our DAFA GMES demonstrator, two different basic use cases are considered:

- **Provision of data products.** Different spacecraft compete to provide the best possible data product, taking into account the restrictions that apply for each spacecraft (e.g., next fly-over, current planning, status of the spacecraft and instruments, deadline for delivery, etc.);
- **Detection of a scientific events.** The capabilities of Type III spacecraft is used to detect a given condition. This is a collaborative scenario in which the different spacecraft of Type III work in conjunction to improve the detection of a scientific event. In this scenario agents of different missions collaborate in order to provide early detection. Agent cooperation is implemented through a synchronization mechanism between the ground and the on-board planning. Each time a plan is changed on board, the modification is communicated to ground as soon as possible, making it possible to trigger new observations on behalf of the other spacecraft (both of type II and type III) that will be flying over the interested area.

Figure 1 presents a high level view of the agents that exist in our demonstrator. Note that the agents in white are agents that are Earth-based and that are mission-independent, meanwhile the agents in light gray are agents that are Earth-based and are specific for each mission. In addition, agents in dark grey belong to the space segment. Both agents in light gray and agents in dark grey (with the exception of the *coordinated ground station* agent, which is a single agent for the whole demonstrator) are instantiated for each mission (i.e., a proxy agent is launched for the ENVISAT mission, another proxy agent is instantiated for the ALOS mission, etc.). Note that at the mission and space segment level, we instantiate the corresponding agents for a mission (except the coordinated ground station agent), which means that for a standard simulation with n different missions we have a total of:

- 3 multi-mission agents (orderer, Product broker and centralized planner);
- 5 mission agents per mission (mission planning, mission proxy, flight dynamics, monitoring, and data handling), which yields $5n$ different agents;
- 1 Coordinated ground stations agent;
- 3 agents for the space segment, which yields $3n$ agents.

Overall, we have a total of $4 + 8n$ agents for a simulation where n missions are deployed, each new mission therefore involving 8 more agents.

Among all the types of agents, the most important agent for the purpose of this paper is the **Science Agent** (SA). In fact, the SA retains all the necessary autonomy to perform the detection of scientific events and, more importantly, on-line planning, scheduling and execution of goal-based plans. The detailed description of the SA features will be the object of Section .

On-board Autonomy Through Automated P&S Technology

As previously stated, Type I autonomy only entails the capability to run a pre-specified set of actions without intervention from ground; in this case, the analogy with the well-known binomial program-process is very strong: the executing sequences of actions cannot be modified while running. This limitation is overcome with autonomy of Type II, which entails the possibility to change the line of action on the fly with pre-specified plans, depending on the occurring emergency. The Type III spacecraft represent the most advanced concept in terms of on-board autonomy; they have the capability to autonomously detect some pre-defined events, upon

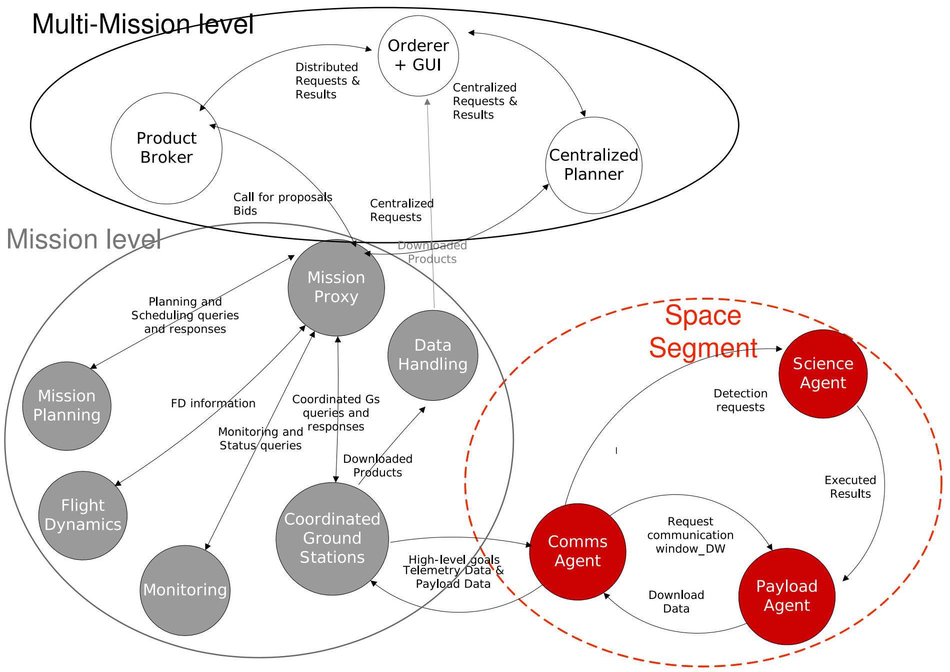


Figure 1: The agents in the DAFA software

which to reason and extract new goals. In fact, they implement a goal-based behavior, where the activity sequence to be executed is synthesized on-board on the base of the objective that has to be reached.

The agent structure of the Type III spacecraft as developed in our DAFA demonstrator, as well as the relationships among the involved agents are shown in Figure 1. In particular, the agents involved in the space segment are the following:

– The **On-board science analysis agent**: this agent is in charge of observing the Earth for those particular areas of interest using the on-board instruments, detecting the presence of interesting events (e.g., fires, oil spills). It also manages the on-board planning and the scheduling of activities to be executed by the spacecraft. This agent also guarantees the proper execution of commands at the spacecraft level, based on the on-board planning mechanisms and the ground commands that are being sent;

– The **Communication agent**: this agent acts as an interface with the ground system agent, it optimizes the bandwidth and the sending/receiving of data to the ground stations;

– The **Payload agent**: this agent manages and stores the payload data. Uses the X-band communication window available with the Ground stations to download payload data whenever there is an opportunity to do so.

Spacecrafts with On-Board Planning Capabilities

Figure 2 depicts the structure that defines the goal-based behavior of the Type III SA that has been implemented in the DAFA architecture. As previously stated, one of the most appealing aspects of autonomy is the capability to synthesize plans based on the notion of high level objectives; our SA is designed to accept such high level goals, prioritize them and use such information to extract from a *plan library* the most suitable line of action that fulfills the selected goal.

All relevant information about the plan currently under execution is stored in the Current Plan Database (CPD); the data stored in the CPD concern the execution status of the activities (terminated, under execution, to be executed), and, more importantly, all the temporal information related to the constraints that might be imposed among the activities. Such information is maintained, for the current purposes, by means of a Simple Temporal Network (STN) (Dechter, Meiri, and Pearl 1991), which underlies the current plan. The STN maintains the information about the distances between any pair of time points of the network, which gets properly updated each time a new temporal constraint is inserted. Through the STN it is therefore possible to create and maintain a structure of temporal relationships that generally involve all the plan activities; such relationships are continuously enforced in the plan, and this feature is useful to control how unexpected events occurring on a single activity (i.e., delays, duration changes, etc.) propagate to the rest of the plan.

The SA is also able to control its execution by dispatching the activities currently present in the executing plan in due time, by simulating the passing of time in the CPD, and triggering the tasks whose start time meets the current time of execution.

Moreover, the autonomous *Event Detection* capability of the SA allows to simulate the detection of particular events as a result of the execution of a detection task. This ability is of primary importance, as it allows to generate new goals to be inserted in the Goal List at execution time, therefore closing the *sense-plan-act* loop with the real world, and ultimately enabling a continuous goal-based replanning be-

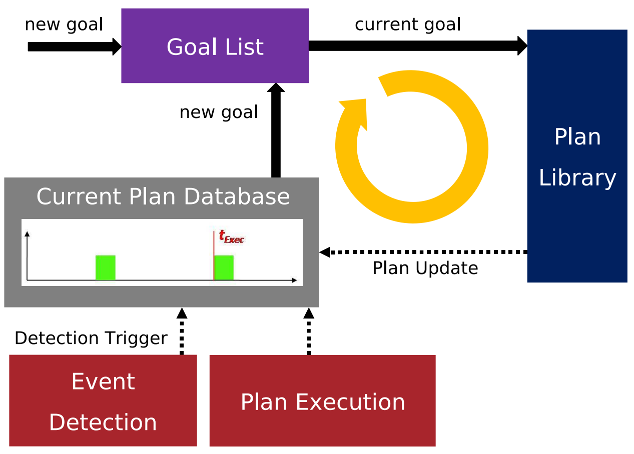


Figure 2: Our Type III Science Agent architecture

havior.

The Algorithm 1 presents the scheme of the execution steps performed in the SA, that were described above. The SA enables two *behaviors* in two concurrent threads, the *Listener* behavior and the *Executor* behavior; in the algorithm, both behaviors are shown as a single execution thread, for the sake of simplicity. The *Listener* behavior accepts all the incoming goals from Ground and actually performs the re-planning by updating the current plan. At each execution cycle the Listener behavior receives the incoming goals and updates the Goal List; then, a goal is selected and the current plan is updated by choosing the most appropriate sub-plan from the Plan Library and by scheduling it on the current global plan. Scheduling entails the removal of the possible resource conflicts between concurrent tasks, while enforcing the feasibility of all the necessary previously imposed temporal constraints. Moreover, since the already planned activities are periodical in nature (i.e., all observation tasks must be repeated at each orbit until a positive observation occurs), at the end of orbit n the Listener behavior is also in charge of replanning the activities for orbit $n + 1$.

The *Executor* behavior is in charge of dispatching the scheduled tasks in due time. As the execution time advances, the Executor behavior checks whether some activity has completed its task; in case a *detection* task has been completed and the related detection is positive (i.e., a fire has been observed), then a new goal is added to the Goal List for future re-planning; in case a generic "science production" task has been completed, the `informPayload()` function is launched, in order to inform the Payload Agent (see Figure 1) that there is a science product ready to be dumped to Ground.

The JOSCAR Timeline-based Package

The central idea in enforcing autonomy in satellite operations is the exploitation of on-board planning and scheduling capabilities. More specifically, the current DAFA SA can autonomously decide to insert/remove tasks to/from the on-board executing plan; it has the capabilities to decide feasible insertion times by checking out the onset of possible

Algorithm 1: Executing and dynamically updating the Plan through a goal-oriented approach

```
Input: GoalList, Plan, PlanLibrary
Output: Executed Plan
while (TRUE) do
    // Receiving new goals
    if (New Goal has been received) then
        GoalList ← UpdateGoalList (Goal, GoalList)
    // Re-planning step
    if (GoalList NOT empty) then
        Plan ← UpdatePlan (Goal, PlanLibrary, Plan)
    advanceExecTime ()
    // Activity Execution step
    if (isExecuted (Activity)) then
        if (Activity == Detection) ∧ (Event is detected) then
            GoalList ← UpdateGoalList (Goal, GoalList)
        else if (Activity == ScienceProduction) then
            informPayload ()
```

resource conflicts that may be raised during operations because of the oversubscription of on-board resources. Such features have been implemented by exploiting the services offered by the JOSCAR (Java Object-oriented SCheduling ARchitecture) library, purposely devised for DAFA to provide all the necessary APIs to create, manage and reason upon plans and/or schedules, intended as sequences of temporal events.

In the DAFA SA, plans and/or schedules are reasoned upon as a special type of Constraint Satisfaction Problem (CSP); the scheduling problem we deal with is well-known in the Operations Research (OR) literature, and derives from a project management environment in which activities represent steps that must be performed to achieve project completion (see (Cesta, Oddi, and Smith 2002)). These activities are subject to partial order constraints that reflect dependencies on project progression. Such problem represents a specific formulation of the basic scheduling issues which underlie a number of real-world applications and is considered particularly difficult, due to the presence of temporal separation constraints (in particular maximum time lags) between project activities.

The JOSCAR library is implemented so as to capture all the modelling features that are strictly connected with the scheduling of resource-consuming activities in a *Timeline*. A Timeline can be informally described as an ordered sequence of temporal events. The JOSCAR library has been designed to exploit the efficacy of the Timeline-based Planning & Scheduling centered around state-of-the-art Constraint Reasoning techniques. By exploiting this constraint reasoning technology, it is possible to represent all the significant aspects of a planning and scheduling problem in the Timeline (i.e., both the temporal and resource aspects). The capability to efficiently manage a Timeline object therefore corresponds to being in control of the most important modelling and reasoning aspects of the Planning & Scheduling problem. The basic class the JOSCAR library provides is the *TimelineManager* class; by instantiating a *TimelineManager* object it is possible to represent the activities in the plan, control their precise timing data as well as their mutual temporal relationships, check the schedule's resource feasibility through profile-based resource conflict detection routines, as well as requesting initial scheduling solutions for further refinement by means of a built-in scheduling algorithm.

Using JOSCAR for Onboard Replanning

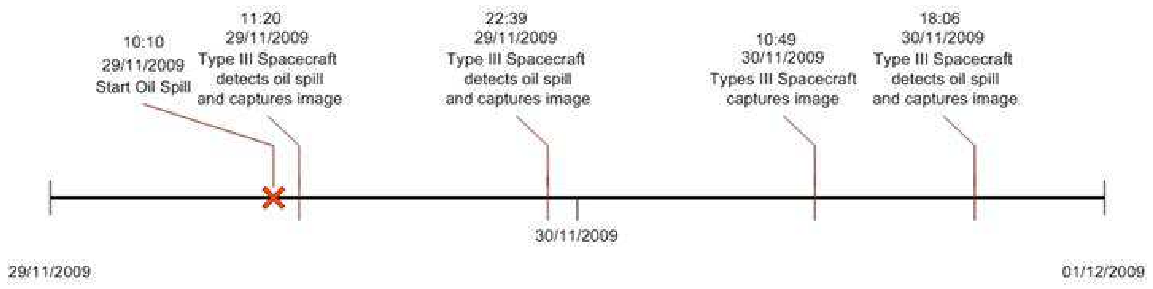
We here briefly describe some of the JOSCAR methods that have been used to implement the DAFA SA's execution and re-planning features described in the algorithm 1. The *UpdatePlan()* function is implemented through the use of the `createActivity()`, `addActivity()` and `retractActivity()` primitives, which allow the dynamic management of a network of tasks; through these methods it is possible to respectively create, add and retract activity from the executing timeline on the fly. Moreover, the dynamic management of the constraint network is guaranteed by the `addTemporalConstraints()` and `retractTemporalConstraints()`; as their names suggest, these methods allow for the easy insertion and retraction of various types of temporal constraint that might be present in a plan. Both the previous primitives entails the execution of state-of-the-art temporal propagation algorithms, whose efficiency is therefore of primary importance. The conflict detection capability is guaranteed by the `isConflictFree()` and the `showResourceProfiles()` primitives, which return the information about the possibly conflicting activities in the executing plan. In particular, the joint utilization of both the previous primitives allows to assess which particular resources are currently oversubscribed, as well as the oversubscription intervals in the general case of plans with activities that use multiple multicapacitated resources. This information is extremely useful to immediately detect all problematic areas in the current solutions, as well as to assess the severity of the conflict. The library also provides a built-in scheduling capability by means of the `solve()` primitive, which can be used to attempt a resource conflict resolution, in order to maintain continuous resource feasibility of the current solution, while the time feasibility is guaranteed by the above mentioned temporal propagation algorithms.

Running the Demonstrator

The following is a description of the results obtained when running the demonstrator, simulating a particular GMES scenario. This simulation includes a constellation of nine spacecraft, three of which are of type III (i.e., endowed with additional autonomy). The configuration of each spacecraft is different, and matches the characteristics of real spacecraft that are currently being used for Earth observation (instrument, instrument parameters, orbits, restrictions of use, communication bandwith, as well as visibility windows). The simulation is centered upon detecting oil spills (i.e., oil spill detection activities have the highest priority), and illustrates the advantages of exploiting autonomous on-board re-planning capabilities in a multi-agent system. To this aim, two approaches to this emergency scenario have been reproduced: a first approach, where onboard autonomy is disregarded and the spacecraft are not capable of autonomously detecting the event, and the products are requested from ground (when an oil spill is detected, a team of experts will request the spacecraft that overfly the area for image products to analyze the situation and solve the emergency according to the obtained images); a second approach where autonomy is exploited for early event detections; in this approach, due to the high probability of oil spills, the type III spacecraft will be requested to check for an oil spill on a certain area. Upon event detection on behalf of any of the spacecraft, an image of the event will be automatically taken and downloaded to the first available ground station; moreover, the spacecraft will reschedule its plan so as to keep capturing images of the area of interest until the emergency is resolved.

Figure 3 shows two timelines comparing spacecraft performances, respectively with and without on board autonomy. The first time line (shown on Figure 3) shows the results when several spacecraft of type III are requested to observe an area for a possible disaster. In this particular scenario, an oil spill occurs at 10:10 am close to the Spanish coast. At 11:20h a type III spacecraft detects the oil spill, takes an image of the area and reschedules its plan so that a further image will be taken at the next over flight of the same area (on the following day, at 10:49h). Similarly, at 22:39, another type III spacecraft overflies the area detecting the oil spill, taking an image and rescheduling a new product for the next over-flight. At 18:06 of the next day, an additional Type III spacecraft detects the same oil spill and takes the corresponding image. As a result, since the onset of the event at 10:10 of the first day, four different images of the area have been taken at different times (within 18:06 of the next day) without the need of human intervention.

The second timeline in Figure 3 shows the results when using type II spacecraft only. Following the same pattern of the previous execution, the oil spill starts at 10:10 am. As type II spacecraft capabilities do not allow for an immediate notification, an extra time inevitably elapses before the authorities acknowledge the event and an emergency plan is activated. Such plan entails the involvement of ground operator to re-program the interested spacecraft to obtain the corresponding images of the area with the proper characteristics. Eventually, the first image available will be the one taken at 21:30, followed by other images at 05:50, 14:09 and 18:43 taken by different spacecraft, i.e., the first captured image of the area is taken more than 17 hours later than in the previous case. In large emergencies like oil spills and fires, fast responsiveness is a key factor: therefore, the possibility to obtain immediate evidence of the events represents an invaluable asset for the user. Conversely, the second timeline describes a situation where it is possible to rely on a large number of spacecraft to provide more images, but much later and at a much higher costs. Although this is



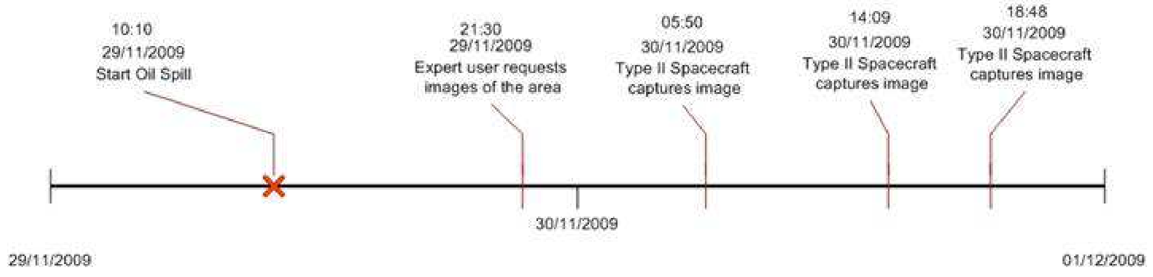


Figure 3: Timeliness when using several spacecraft with on-board re-planning (upper timeline) and without on-board re-planning (lower timeline) capabilities

a particular scenario, different executions for other similar scenarios have consolidated these results: in all cases that a Type III spacecraft overflies the area in the next 12 hours after the event occurs, the results are much better than when using non-autonomous spacecraft.

Conclusions

In this work we have described a software system based on multi-agent technology that was developed within the Distributed Agent For Autonomy (DAFA) study as a demonstrator for the Global Monitoring for Environment and Security (GMES), a relevant initiative jointly promoted by the European Space Agency and the European Commission.

The major objective of the DAFA demonstrator was to show the benefits of agent-oriented programming in space missions, and also to allow the demonstration of advanced techniques, like on-board autonomy, by deploying multi-agent based spacecrafts capable of diversified autonomous decision-making skills. We have described the overall system, focusing our attention on its multi-agent inner structure combined with the state-of-the-art Planning & Scheduling technology that was employed to introduce an autonomous planning capability.

The main objective of this work is twofold: on one hand, we have described an example of reusability of off-the-shelf P&S components currently available from the P&S research areas to solve real-world problems; on the other hand, we have provided a glimpse of how multi-agent systems can be endowed with advanced AI-based problem solving capabilities.

Experiments carried out with the DAFA demonstrator show how the ability to perform on-board planning decisions by autonomously deciding the current line of action depending on the detected events and/or opportunities, can represent a significant advantage in terms of mission cost reduction, increased quality of science products, and response timeliness (see (Ocon 2009) for a more complete account in this respect).

Acknowledgments. Authors have been partially supported by ESA under project DAFA (AO/1-5389/07/NL/HE). Thanks to Quirien Wijnands and Joachim Fuchs for their continuous supervision on our work.

References

Cesta, A.; Oddi, A.; and Smith, S. F. 2002. A Constraint-based Method for Project Scheduling with Time Windows. *Journal of Heuristics* 8(1):109–136.

Chien, S.; Sherwood, R.; Tran, D.; Cichy, B.; Rabideau, G.; Castano, R.; Davies, A.; Mandl, D.; Frye, S.; Trout, B.; Shulman, S.; and Boyer, D. 2005. Using Autonomy Flight Software to Improve Science Return on Earth Observing One. *Journal of Aerospace Computing, Information, and Communication* 2(4):196–216.

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal Constraint Networks. *Artificial Intelligence* 49:61–95.

Muscettola, N.; Nayak, P.; Pell, B.; and Williams, B. 1998. Remote Agent: To Boldly Go Where No AI System Has Gone Before. *Artificial Intelligence* 103(1-2):5–48.

Ocon, J.; Rivero, E.; Strippoli, L.; and Molina, M. 2008. Agents for Space Operations. In *Proceedings of the SpaceOps Conference, AIAA*.

Ocon, J. 2009. DAFA - Distributed Agents For Autonomy - Final Report. Technical report, GMV Aerospace and Defence S.A.