Autonomous Agents in Space Missions

Stefan Buettcher

<stefan at buettcher dot org>

Course Project

Multi-Agent Systems for Real-World Applications

Prof. Robin Cohen

University of Waterloo, Spring 2004

Contents

Outline

In the first section, an introduction to the general topic of autonomous agents in space missions is given. The following two sections deal with the necessity of adjustable autonomy in the context of autonomous agents in space and with the dangers the use of autonomous agents involve. The fourth section presents proposed solutions to three different problems in either manned or unmanned space missions. After that, one particular architecture, the Personal Satellite Assistant, is discussed; suggestions for future agent cooperation within the scenario are made. Finally, a conclusion is given.

Autonomous Agents in Space – Why?

Space missions, such as NASA's Mars missions or the International Space Station (ISS), are natural domains for the application of autonomous, agent-based control systems. Among the main reasons are saving money and protecting human lives:

- Current spacecraft control systems need about 3 persons per spacecraft that are monitoring and sending new commands at every time. This leads, for instance, to a control staff size of 200 persons for the Iridium satellite phone service [Rouf2002]. This large number is one reason for the continuous financial problems of satellite phone systems.
- Having software agents control spacecrafts allows for unmanned space missions which can avoid the loss of human lives in disasters such as the 1986 Challenger space shuttle accident.
- In manned space missions, the presence of agents can increase a mission's efficiency significantly. In a 90-day test run by NASA in 1998, a crew of 4 persons spent 90 minutes per day for general maintenance and repair [Lewi1998]. Agents (robots or software agents) accompanying the crew on their mission could reduce the time spent for these activities.

So, good reasons for using agent-based solutions instead of human mission control and for agents supporting the human crew on manned missions exist. But do such agents necessarily need to be autonomous? There are some good reasons why autonomy is strongly required under the special circumstances of agent-controlled space missions:

- *Very high communication latency*
  While a message sent to any satellite in the earth's orbit can be received by the satellite within a few milliseconds, the great distances that are involved in the exploration of Mars or planets that are even farther away make any communication difficult: light needs between 3 and 22 minutes to travel from Earth to Mars, resulting in a round-trip time between 6 and 45 minutes (depending on positions of Mars and Earth in their orbits around the sun) [Nasa2004a]. An Earth-based control center that sends a short command to a satellite in Mars' orbit, waits until it receives the confirmation that the command has been executed by the satellite and then sends the next command seems ridiculous under these circumstances.
- *Restricted communication timeframes*
  Even if the communication latency was not a major problem to the traditional approach to spacecraft control, a rover on Mars' surface would only have short periods of possible communication with its control center on Earth because both planets rotate, and therefore direct communication is not always possible. In fact, a direct communication channel can only be established during 3 hours per day for a rover, and 16 hours per day for one of the Mars

orbiters. [FN2004]
The same problem arises for spacecrafts in low earth orbit, which – despite their proximity to the control center – only have an effective communication time of 80 minutes per day (8 windows of 10 minutes each). [CC2004]

- *Limited communication bandwidth*
  Assuming, the communication latency is not a problem, and the spacecraft can communicate with its control center on earth at every time using relay satellites, the issue of limited bandwidth remains. If the spacecraft encounters a problem that it cannot solve alone but which could be solved by the control center on Earth, the control center might need large amounts of data in order to recognize the problem and to come up with a solution. The information is available in the spacecraft's memory, but it has to be transferred to the control center.
  When the Galileo probe was unable to unfold its main antenna in 1991 (two years after the start of the mission), it had to use its low-gain antenna, which was theoretically able to transmit up to 1600 bits per second [Nasa1991]. Other sources report that the effective communication bandwidth of the low-gain antenna was between 10 and 160 bits per second ([CWRU], [RESA]). Obviously, the possible information exchange at these speeds does is far too low for remote control of a spacecraft or remote failure analysis.

While these issues do not arise in manned space missions, in which the accompanying agents probably do not perform the mission control but only play a supportive role, there are still good reasons for agent autonomy in this case:

- An agent that is not autonomous has to ask a human crew member before it starts a certain action. Overhead caused by these confirmation requests could possibly consume more time than the presence of the agent saves.
- A truly autonomous agent, which is able to override human commands, can improve overall mission safety. After all, most problems are not caused by hardware failure but by human mistake. A support agent that checks human actions for sanity can decrease the risk for the humans involved in the mission.

Autonomous agents, either controlling a spacecraft or supporting the human crew on their mission, can make a space mission more safe for the astronauts involved and also less expensive for the operating space agency, allowing for more frequent and more successful missions.

The Need for Adjustable Autonomy

In the previous section, we have seen that there is need for autonomous agents in both manned and unmanned space missions. A truly autonomous agent could take control of a spacecraft shortly after its detachment from the carrier and maintain it until the end of the mission. Even if today's agent technologies were able to execute such a task in principle, which is clearly not the case, this scenario is unfavorable for a number of reasons:

- Mission objectives may change over time. If, for example, an organization that has ordered the execution of a certain experiment but becomes bankrupt after the launch of the probe, the control center could decide to drop this experiment and give more time to a different task, which a client will for.

- It can become necessary to repeat certain experiments in order to obtain the desired information. If a satellite takes pictures, and the quality of these pictures is not good enough to extract the desired information, the control center could decide to let the satellite move closer to the object and take more pictures.
- Space is an unpredictable environment. It is impossible to foresee all possible events. In a case of an unexpected event that the remote agent on the spacecraft does not know how to react to, it could ask the ground control center for help. In particular, this applies to problems within the spacecraft itself, i.e. the breakdown of one of its components.

All these are reasons to equip an agent with the ability to dynamically adjust its own autonomy or to allow the control center to adjust the agent's autonomy. The need for adjustable autonomy becomes even greater if a space mission does not only consist of a single spacecraft but entails a number of spacecrafts conducting experiments both jointly and independently. Dorais et al. [Dora1998] mention the Deep Space Three project in which three independent probes are used to form an optical inferometer. If one of them becomes short of fuel, individual goals should be reassigned (giving less fuel-consuming activities to the probe that is short on propellant) in order to achieve a longer total lifetime.

The Hazards of Autonomous Agents

Despite the urgent need of agent autonomy, mission operations managers at NASA are reluctant to use new, unsufficiently tested technology on their spacecrafts because it makes an already very dangerous operation even more dangerous. [Rouf2002]

In fact, every bit of autonomy given to a remote agent controlling a spacecraft can turn out to be a wrong decision. When inserting a space probe into the orbit around a planet or landing a rover on Mars, even a single mistake can lead to the destruction of the vehicle and thus the failure of the entire mission. Therefore, agent autonomy has to be adjusted in a very conservative, very risk-averse fashion. As soon as the possibility of a dangerous situation is detected that the agent could potentially not be able to deal with, the agent has to immediately reach a safe state in which information with the control center can be exchanged in order minimize the risk or even avoid the dangerous situation at all.

Similarly, when overriding human commands, the agent has to be very careful not to refuse to carry out a command that seems dangerous and might in fact be dangerous but is the only possibility to eventually avoid a situation which would destroy the spacecraft. Moreover, it must be possible to completely turn off agent autonomy in case of severe software bugs in the autonomous systems.

Existing Research

Research in autonomous space agents has been done in several sub-areas. Some research has led to the use of agent-based systems in actual space missions, while other work has remained purely theoretical so far. Some of the research done in both categories is summarized in this section.

Autonomous Spacecraft Control

The independent control of a spacecraft by an autonomous agent is certainly the most challenging and most dangerous (or potentially dangerous) application of autonomous agents in space missions. The first successful implementation of such an agent system is the New Millennium Remote Agent (NMRA) architecture, which was used to control the Deep Space One spacecraft during its mission (1998-2001).

In the NMRA architecture, high-level scientific mission goals are sent to the spacecraft by ground control. The planning process that is necessary to achieve these goals is then performed independently by the control agent in the spacecraft.

The NMRA architecture, consists of 3 major components: [Pell1997]

Planning & Scheduling The spacecraft starts with an initial set of scientific goals, but goals can be added to or removed from the database throughout the entire mission. In addition to purely scientific goals, there are also goals that are necessary for the control of the spacecraft, e.g. taking pictures of star constellations in order to determine the own position.
From the goals in the goal database and the current status of the spacecraft's subsystems, a schedule of abstract actions is computed with respect to a certain scheduling horizon, i.e. the time until which a current schedule will be valid.

Executive The executive translates the abstract actions it receives from the planner into a sequence of low-level, system-specific commands. If the execution of one of the abstract actions fails, the executive retries the corresponding low-level command sequence or tries another low-level sequence that results in the same high-level state. If, for example, the spacecraft's camera has to be turned towards a particular star constellation, this can either be done by using the camera's motors or by rotating the entire spacecraft. If a low-level command fails, this is an indication of a hardware error, in which case the executive changes the schedule and tries to reach a *safe state*. The rationale behind this is that any hardware failure poses a risk to the entire system. Therefore, as a safety measure, a safe state has to be reached before the next action may be performed. Once that safe state has been reached, the planner is asked to create a new schedule, since the old schedule will be incompatible with the new situation.

Monitoring & Mode Identification The monitoring and mode identification subsystem takes low-level spacecraft commands and reads sensor values in order to determine if a given command has been executed successfully and if the observed state of the spacecraft is consistent with its expected state after the execution of a certain command. If an anomaly is detected, the mode identification unit can in some cases identify the specific component of the spacecraft that is responsible for the failure. This information about defective components is then shared with the executive so that it can be taken into account for future command sequences.

NMRA is an example of adjustable autonomy in spacecraft control: The autonomy of the planner/scheduler as well as the executive can be changed by ground control in the case of change in the mission objectives or unforeseen events, e.g. if the failure of an on-board component is noticed, the executive may be told exactly what to do in order to recover from that failure. So, their autonomy can be restricted by human intervention. Furthermore, the executive agent can adjust its own autonomy (and the autonomy of the planning agent); in normal operation mode, the executive simply does what it is told by the planner. If, however, an error in one of the spaceship's components is detected, the executive changes its behavior, performs a task (reaching a safe state) that is completely different from the current schedule and tells the planner to compute a new schedule, effectively exchanging the mutual roles.

After the implementation had been finished, the system was tested in a simulated insertion of a spacecraft into a Saturn orbit. The successful simulation finally led to the use of the NMRA architecture as the control system of DS-1. During the actual mission, the autonomous agent system caused no problems and will therefore be used again in future space missions.

Mars Rover Control

In contrast to the NMRA system, an agent-based control system of a Mars rover is not mission-critical in the sense that an error in the control software can make the entire mission fail. However, a rover is still a multi-million-dollar device whose loss is usually inacceptable.

Traditionally, Mars rovers were operated by sending manually created sequences of actions to the rover, having it perform these actions, inform about its resulting state and wait for the next sequence of commands. Obviously, this is a very time consuming process, considering the average round-trip-time of 20 minutes for communication between Earth and Mars. Moreover, it requires lots of human work in the ground control center, since the control sequences are created manually.

While, despite all problems, this was possible when the Pathfinder *Sojourner* rover was travelling on Mars in 1997/1998 (the total distance covered by the rover was about 100m) or even for the current Mars missions (in which a rover can go up to 100m per day), the necessity for autonomous operation will become more urgent for future missions, in which a rover will be able to travel much longer distances, possibly several kilometers a day.

The main goal of Mars rovers so far was the geological examination of Mars' surface, which is essentially taking pictures of rocks. Problems that occur within this context are:

- The rover acquires huge amounts of image data through its cameras. Not all these data can be transferred to Earth for closer examination by scientists (bandwidth limitations). Priority values must be assigned to the pictures gathered.
- Scenarios like this should be avoided: The Rover encounters a rock that could possibly lead to new scientific insights, but the image it takes arrives 10 minutes later in the ground control center, has to be analyzed. When the rover receives the command to go back to the rock in order to take additional pictures, several hours have elapsed and the rover has to go all way back to the rock.

Estlin et al. [Estl2003] have proposed the incorporation of machine learning techniques into the rover software that can solve the problems described above. The rover autonomously analyzes the pictures it takes from the surrounding area. Different feature values are assigned to the objects encountered, including the color, the size, and the visual texture of the object. Pictures and measurement results are then transmitted according to the priorities assigned. If this process gives a certain object a very high priority, the rover may independently change its original schedule in order to take more pictures or conduct additional measurements on the object.

Under certain circumstances, the rover may interrupt its travel on Mars in order to wait for new instructions from the control center, for instance if a small green object of humanoid shape has been detected.

The described techniques have not been employed in actual Mars rovers so far, but have performed very well in tests conducted in Mars-like regions of Arizona. In one test, the rover was able to detect a piece of petrified wood, which was then selected for closer examination by the rover's scheduling algorithm.

A few additional ideas on the topic of autonomous Mars rovers are given by Dorais et al. [Dora1998]. They address, for example, the problem of synchronization between the autonomous remote agent and its control center on Earth, when new mission objectives are inserted into the rover's list of goals.

Personal Satellite Assistants (PSAs)

Much work has been done in the area of personal satellite assistants that can assist crew members in their regular maintenance/monitoring duties on manned space missions. There are different types of PSAs suggested in the literature. The prevalent type is a softball-sized spherical flying robot that is able to operate autonomously on either a manned or an unmanned spacecraft (in the latter case, the correct term is *portable* instead of *personal* satellite assistant, since there is no person on the spacecraft).

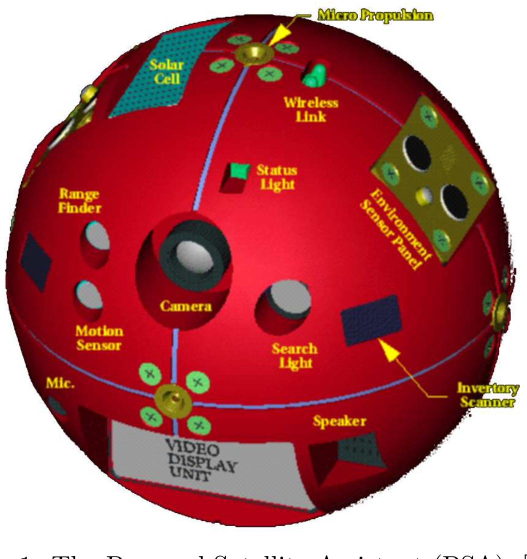

Figure 1. The Personal Satellite Assistant (PSA). [Nasa]

The necessity for having robotic agents supporting human crew members on manned space missions is clear from the mission description of a shuttle mission in 1998: [Gawd2000]

> *One astronaut, Andy Thomas, will undertake several hundred research runs involving 26 different science projects in five disciplines. The projects are provided by 33 principal investigators from the U.S., Canada, Germany and the U.K.*

The amount of work to be done by a single astronaut is overwhelming. It is therefore desirable that autonomous robots perform general monitoring and maintenance activities so that the crew can focus on the scientific goals on the mission. NASA's vision of the PSA's future is best described by [PSA]:

> *The PSA is an astronaut support device designed to move and operate independently in the microgravity environment of space-based vehicles. The PSA will assist astronauts who are living and working aboard the Space Shuttle, Space Station, and during future space exploration missions to the Moon and Mars.*

Among the proposed capabilities of PSAs are: [Gawd2000], [PSA]

Environmental Health Monitoring The agents keep track of gas levels, such as oxygene and carbone dioxide, and of air pressure in the spacecraft.

Communications PSAs are multi-purpose communication devices for the crew members. Equipped with a microphone, speakers, a camera and a visual display, they can be used for video conferencing with ground control or for reading the status of spacecraft components.

Crew Worksite Support Having a wireless connection to the onboard computer systems, a PSA can keep track of inventory that is used for a certain experiment and warn a crew member if the spacecraft is running short of a special item needed for the experiment. Or it can visually search for an item that an astronau needs but cannot find.

Beyond there ability to carry out work that would otherwise have to be done by an astronaut, they can even solve problems that would be unsolvable for a human because she is too large to enter the region of the spacecraft were a particular problem is located.

Hexmoor and Vaughn [HV2002] describe the PSAs' capability to cooperate in order to achieve a certain goal. In the example scenario, an agent, while on its routine tour through the space station. detects an anomaly in a room of a space station, e.g. the room temperature that is lower than it should be. Because finding the exact location of the problem is easier when more than one PSA is present (triangulation techniques can be used), two more robots should enter the site. For this purpose, the agent broadcasts an alert message M to all PSAs on the space station. This message includes the identity of the agent that has found the problem, the type of the problem, its approximate location (given by the room it is located in), and the time since when its existence has been known:

$$M = (Sender, ProblemType, Room, Time).$$

Initially, $Time = 1$. But if no other PSA responds to the message, or no other PSA feels like helping, then the message is resent, and $Time$ is increased, expressing the urgency.

Every agent that receives this message responds by sending a priority value that indicates how willing the agent is to help finding the exact location of this particular problem. It calculates the energy it would consume to go to the room, help, and fly to the next battery recharge station:

$$E_{total} = (\text{distance to } Room + \text{distance from } Room \text{ to } RS) \cdot ECR + E_{help},$$

where RS is the next recharge station, ECR is the energy consumption rate, which is assumed to be constant and the same among all PSAs, and E_{help} is the energy consumed during the actual execution of the task.

If E_{total} is bigger than $E_{remaining}$, the remaining energy of the PSA, then the agent cannot help without risking to become unoperational. Therefore, the agent sends a negative priority value, indicating that it does not want to help. If $E_{remaining} \geq E_{total}$, the agent computes a job weight Q that describes the urgency of the problem:

$$Q = \ln(Time + importance(Room)).$$

The function *importance* assigns a certain value to each room type, accounting for relative importance of different room types.

There are three possibilities for what an agent that has received an alert message is currently doing:

1. *Nothing*
   In this case, the priority value returned is:
   $$P = Q \cdot (1 - \frac{\text{distance to target}}{maxDistance}),$$
   where $maxDistance$ is the maximum distance between any two rooms within the space station.

2. *Heading towards a different problem site*
   If a PSA has already been assigned a different task, which it is currently heading towards, it has to deliberate about whether it should change tasks. Thus, the priority value returned in this case is:
   $$P = Q_{new} \cdot (1 - \frac{\text{distance to new target}}{maxDistance}) - Q_{old} \cdot (1 - \frac{\text{distance to old target}}{maxDistance}).$$

3. *Working on a problem*
   If the agent has already arrived at a different problem site and is currently working on a problem, then it reports the priority value:
   $$P = Q_{new} \cdot (1 - \frac{\text{distance to new target}}{maxDistance}) \cdot (1 - completion),$$
   where $completion \in [0, 1]$ indicates the relative amount of work that has already been spent on the current problem. The intention of this rule is to increase system stability and to avoid situations in which certain tasks will never be completely done.

When the sender of the original message (*alert originator*) has received all (or sufficiently many) P values, it selects the two agents with highest P value and asks them to come to the problem site. The other agents get a message informing them that they are not needed any more.

After the communication process has been finished, the alert originator starts to cut down the search space by determining the part of the room where the problem is located. As soon as this process has started, the originator will not leave the problem site until the exact location has been found. A few moments later, the helper PSAs will arrive and help the alert originator in finding the source of the problem.

Although this is not mentioned by Hexmoor and Vaughn, it is not hard to imagine how this architecture could also be used to perform other general tasks on a space station or a permanent Mars station. Consider, for example, the task of growing wheat in a permanent station on Mars, for the purpose of feeding the crew. One PSA could do the job of watering the plants. In order to speed up the task, it could ask another PSA to be the supply unit for this task. The weight of the task would, of course, have to be smaller than that of locating a problem like subnormal temperature or air pressure in one of the rooms.

A second cooperation and task selection strategy, which is similar to the one presented, has been implemented by Hexmoor et al.; the results were comparable. The PSA is not in use on actual space missions, yet. However, prototypes exist, and the system has been tested with PSAs hovering on a table or on the floor. Simulation runs using the Brahms multi-agent simulation environment ([Clan1998], [Sier2001]) have been conducted.

Discussion of the Personal Satellite Assistant

The work on the Personal Satellite Assistant is the first work done in the field of multi-agent environments with *truly autonomous* agents in space missions. Although the control system of the Deep Space One mission could be considered a multi-agent system, too, the individual components of the system (such as the scheduler and the executive) are coupled too tightly to call them truly autonomous. In the PSA architecture, however, an agent can refuse to cooperate with another agent at all. This ability allows us to speak of *true autonomy* here.

Problems with the Existing Architecture

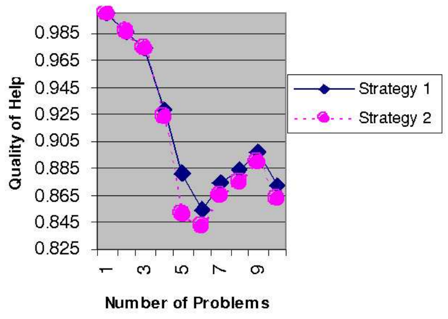

Figure 2. PSA Cooperation Quality. [HV2002]

Simulation runs performed by Hexmoor and Vaughn [HV2002] have shown that the proposed strategies perform quite well, even if multiple problems have been found which the agents have to deal with concurrently. According to the authors, the *Quality of Help* (QoH) varies between 99% and 84%, depending on the number of problems n (n was chosen between 1 and 10 in their experiments). The QoH describes the contentment of the alert originator with the agents that help it in locating the problem. Unfortunately, the authors are very unspecific on how the QoH is actually computed (and it is *not* a terminus technicus, as a Google search for `QOH "quality of help"` has exactly one result). Furthermore, they do not mention how many PSAs are involved in solving the 10 problems. So, it is not easy to determine how well the agents actually performed in the simulation.

The simulations conducted can be considered unrealistic in the sense that all problems simulated were of the same type, i.e. they were not mission-critical. Furthermore, the contentedness of an agent is a bad measure of system efficiency. Human expertise should be consulted when evaluating the performance of the system in a certain scenario – especially if it is as sensitive as a space mission.

In the remainder of this section, a few technical flaws of the PSA cooperation system will be pointed at. After that, the general problems of the chosen approach will be discussed. Suggestions on how to change the system in order to remove these problems will be made.

Technical Problem I – Weighting of time

The weight of a particular problem is calculated as

$$Q = \ln(Time + importance(Room)).$$

The natural logarithm in this equation can have fatal consequences. Consider the scenario in which all PSAs but one are in the laboratory part of the space station, helping with some experiments, and one single PSA is in the storage module. The PSAs in the lab are continuously fixing some minor problems p_{lab}, and the one in the storage module has discovered a problem p_{stor}. Assuming that the storage module's distance to the lab is

$$d(lab, storage) = \frac{3}{4} \cdot maxDistance,$$

the weights of the problems and the corresponding priority values are:

$$Q_{lab} = \ln(1+1), \quad P_{lab} = \ln(2) \cdot 1.0,$$
$$Q_{stor} = \ln(t+1), \quad P_{stor} = \ln(t+1) \cdot 0.25,$$

where t says how long p_{stor} has already been known. It turns out that $P_{lab} = P_{stor}$ for $t = 15$. Unfortunately, 15 minutes can be an illegally long time, especially if it turns out that the problem in the storage module is a major one.

The rationale behind this function is to reduce system dynamics and to avoid agents continuously changing the problems they are working on. However, a slight modification, such as $\ln^{1.5}(x)$ instead of $\ln(x)$, would already reduce t to 5 minutes. It remains to be shown that this function is superior to the one proposed by Hexmoor and Vaughn. But since their sole justification for the function they chose is "because it causes Q to change along a predictable curve" [HV2002], things do not look too bad for the $\ln^{1.5}$.

Technical Problem II – Abandoning tasks

The priority of the current task does not increase monotonically as time goes on. For the sake of system stability, however, this should be the case. Consider the following scenario: An agent heads towards a problem p_1 and receives a message regarding a new problem p_2.

$$Q_1 = \ln(2),\ d(agent, p_1) = 0.5 \cdot maxDistance,\ Q_2 = \ln(2),\ d(agent, p_2) = 0.5 \cdot maxDistance.$$

According to the priority calculation rules, the new problem will get priority

$$P_{new} = Q_2 \cdot d(agent, p_2) - Q_1 \cdot d(agent, p_1) = 0.$$

P_{new} will decrease as the distance between the agent and its original target, $d(agent, p_1)$ is getting smaller. However, as soon as the agent has reached the room that p_1 is located in, the priority calculation rules changes, and the new priority is:

$$P_{new} = Q_2 \cdot d(agent, p_2) \cdot (1 - completion),$$

which suddenly is a positive number! So, it could in fact happen that a PSA makes crazy ping-pong moves between p_1 and p_2.

A better solution would be to calculate the priority according to the rule:

$$P_{new} = Q_2 \cdot d(agent, p_2) \cdot (1 - completion) - Q_1 \cdot d(agent, p_1).$$

This guarantees monotony of the priority value and avoids the ping-pong.

Both technical problems presented can be fixed relatively easily by adjusting the formulas used for calculating the priority values. This does not hold for the following structural problems.

Structural Problem I – Accepting multiple tasks

If a PSA receives a message from an alert originator, it immediately returns a priority value, regardless of how many tasks there are in its current schedule. Assume the following situation: All PSAs are working on a problem. One of them has almost finished the task it is working on, while the others have only just begun their work. Now, 2 alert messages are sent. Chances are that the agent that has almost finished its current task is selected for both new jobs because the priority values calculated only depend on the new task and the task that it is currently working on.

This is fatal because the alert originators of the new problems believe that the agent they selected could be there more quickly than all the other agents. They do not know that its schedule does not allow the helper PSA to help immediately because it has to finish a couple of other tasks first.

The same problem arises if an agent is selected for a different task when it is heading towards the task that was first assigned to it. If a PSA changes its target destination, it does not inform the original alert originator. So, again, the originator could wait for hours, not knowing that the helper agent has lots of other work to do first.

Structural Problem II – Suicidal decisions can be helpful

According to [HV2002], a PSA whose energy level is so low that it could not return to the next recharge station after it has helped locating a problem will not offer its help. This would seem the right decision if the PSA was an expensive device and the problems it deals with are all of innocent nature. However, both assumptions are wrong. A PSA is a relatively (compared with the cost of a space station) cheap device, and it is substitutable, since there are many of them on the space station.

The batteries of a PSA can be recharged even if they have become empty. If the problem that has to be located is a severe one, the agent's decision could, again, be fatal for the success of the entire space mission. Therefore, the decision on whether to risk the temporary loss of working ability should depend on the assumed severity of the problem that is to be located.

A New PSA Architecture

The technical problems described above are not severe. They can be cured by making minor adjustments to the priority calculation rules employed by the agents. The one mistake that is the origin of the structural problems in the approach of Hexmoor and Vaughn is that they do not take into account that their agents have to work in a very special environment, namely on a space station. As stated above, the rule "do not help if this would consume all your energy" is sensible in an everyday environment. On a space station, however, consuming all energy left can sometimes be the best choice.

For general message passing and work coordination purposes, we need a central job/message database in which every entity aboard the space station can put messages and retract messages once they have become obsolete. The sender of a message can be

- a human crew member which needs help in an experiment, needs one of the PSAs for personal purposes, such as voice communication with its family, or needs one of the PSAs to find/fix a problem;
- one of the PSAs that has to perform a job and could use some help;
- a special job scheduling agent that keeps track of all the periodical jobs that have to be done on the space station.

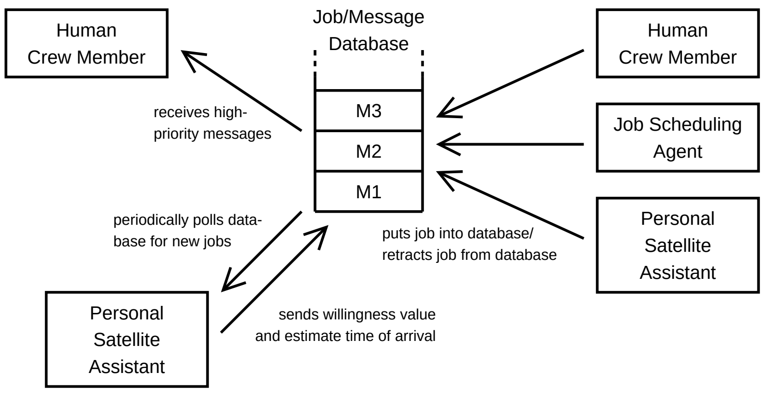
Human Crew Member
Job/Message Database
Human Crew Member
M3
M2
M1
receives high-priority messages
Job Scheduling Agent
periodically polls database for new jobs
puts job into database/ retracts job from database
Personal Satellite Assistant
Personal Satellite Assistant
sends willingness value and estimate time of arrival

Figure 3. PSA/Crew Communication Structure.

Messages put into the job/message database are of the format:

$$M = (\textit{Sender},\ \textit{Time},\ \textit{Priority},\ \textit{Req. Capabilities},\ \#\ \textit{Req. Helpers},\ \textit{Location},\ \textit{Deadline}).$$

- *Sender* is the unique ID of the sender of the message.
- *Time* is the creation time of the message.
- *Priority* is the priority value of the job: *low* (e.g. general, daily monitoring), *medium* (e.g. help in experiments), or *high* (severe problem, must be fixed).

- *Req. Capabilities* is a set of capabilities that an agent needs to have in order to complete the job. This is necessary information if either the PSAs have different capabilities or if a certain component of a PSA has been damaged so that it, for instance, can still visually sense its environment but cannot check the temperature any more.
- *# Req. Helpers* is the number of PSAs required (or desired) to complete the job.
- *Location* is the location of the job, i.e. a room number.
- *Deadline* is the time by when the job has to be finished.

All the PSAs on the space station periodically poll the database for new or changed job messages. In general, an agent is willing to do the job with highest priority value for which it has the required capabilities. If two possible jobs have the same priority, messages whose sender is a human get precedence. If there is still a tie, the job with earlier deadline is chosen. The agent then sends a message to the sender of the job message, containing its ID, its degree of willingness to do the job and the estimated time it takes the PSA to fly to the given location.

The willingness of the PSA to do the job can, for example, be calculated using the priority formulas proposed by Hexmoor and Vaughn. But it is important that the agent also sends its estimated time of arrival because the importance of the job from the point of view of the helping agent is a bad basis for a decision from the point of view of the job originator. The sender of the job message receives a number of responses and calculates subjective preference values

$$pref_A = \frac{arrival_A - minArrival}{maxArrival - minArrival} - \frac{willingness_A - minWillingness}{maxWillingness - minWillingness}$$

for every agent A that it has received a response from, where *minArrival* etc. are the minimum/maximum values received from any agent. It then selects the n PSAs with lowest $pref_A$ value ($n = \#$ *Req. Helpers*). This results in a mediation between both parties' interests.

When the number of PSAs required for a certain job have been found, the job originator retracts the posting from the database. Once an agent has agreed to work on a job, it cannot revoke this decision unless a higher-priority job is posted in the database, in which case the agent is allowed to change its decision and to work on the new job with higher priority. In contrast to the architecture presented by Hexmoor and Vaughn, however, it has to notify the originator of its old job so that the originator can search for a substitute.

One interesting situation that can arise in this system is when the deadline of a given job cannot be met. There are two possible scenarios:

1. The original job had *low* priority, it is retracted from the database and a new job with extended deadline is posted instead. The priority level of this new job is *medium*. The length of the extension is assumed to be predefined by the crew or ground control.
2. The original job had *medium* priority. This either means that the job's original priority level was *low* and it has already missed its deadline once or its original priority level was *medium*. In either case, something went wrong. In order to prevent a damage to the system, two actions are taken: The job's priority is changed to *high*, and the crew member that is responsible for the special kind of problem that the job deals with is informed of the problem.

The third case, in which a job with high priority misses its deadline, should never happen. If it happens, this means that there are some severe problems with the system.

Regardless of whether a job has missed its deadline, a human crew member is informed every time a job with high priority is posted. This is for safety reasons because it is assumed that *high priority* means that something is potentially dangerous. The human can then adjust the message database, give specific commands to the PSAs, or solve a problem herself, without relying on the agents.

Concerning the problem of whether a PSA should risk losing its remaining energy when fulfilling a task, a possible solution is that

- it should always risk reaching a zero-energy state if a high-priority task can be fulfilled by doing so;
- it should risk losing its remaining battery power if a medium-priority task would otherwise miss its deadline and if the job database contains only few tasks with medium priority; if there are many medium-priority jobs in the database, it could happen that all PSAs end up lying on the ground (or rather flying through the space station without control), which could be a more severe problem than having a few jobs miss their deadline.

Shortly before reaching a zero-energy state, the agent should post a message with priority level *high*, informing the crew of its current position and its energy state. This guarantees that the PSA will eventually be found and its batteries will be recharged.

The proposed centralized architecture with one message database and one job scheduler is, of course, not very fail-safe. This problem can be solved if all agents involved in the system are mobile agents, i.e. if they can change their host system in case of a component failure. An introduction to the topic of mobile agents in space missions is given by Papaioannou [Papa1999].

Conclusion

In this project, the general problems that arise in autonomous agents for both manned and unmanned space missions have been analyzed. The dangers as well as the opportunities have been pointed out. Three different agent architectures – aiming at completely different scenarios – have been presented. The Personal Satellite Assistant cooperation structure proposed by Hexmoor and Vaughn has been analyzed; its shortcomings and its inappropriateness for the special scenario of space missions have been pointed out.

A new cooperation structure, involving job deadlines, priority levels and adjustable autonomy of the multi-agent system, has been presented. The agents try to deal with periodical monitoring tasks and problems they discovered independently as long as possible. Humans are only asked for help if the agents cannot deal with the situation or if a problem encountered is too serious to hide it from the crew. This is assumed to lead to increased crew efficiency (because they do not need to know about every detail), while it does not unnecessarily increase risk.

How real-life implementations of the general PSA architecture that will actually be employed on a space station will look like, remains open. We will probably not see them within the next 5 years.

References

[CC2004] B. Cichy, S. Chien et al. *Validating the Autonomous EO-1 Science Agent.* International Workshop on Planning and Scheduling for Space (IWPSS), 2004.

[Clan1998] W. Clancey, P. Sachs, M. Sierhuis, R. van Hoof. *Brahms: simulating practice for work systems design.* International Journal of Human-Computer Studies, 49, 831-865.

[CWRU] Case Western Reserve University. *Journey through the galaxy.* http://home.cwru.edu/ sjr16/20th_far_galileo.html

[Dora1998] G. Dorais, R. Bonasso, D. Kortenkamp, B. Pell, D. Schreckenghost. *Adjustable Autonomy for Human-Centered Autonomous Systems on Mars.* Proceedings of the First International Mars Society Convention, 1998.

[Estl2003] T. Estlin, R. Castano et al. *Learning and Planning for Mars Rover Science.* International Joint Conference on Artificial Intelligence, 2003.

[FN2004] A. Fu, F. Nale. *Communication with the Mars Rovers.* www.personal.psu.edu/users/a/z/azf100/FinalReport.pdf

[Gawd2000] Y. Gawdiak, J. Bradshaw, B. Williams, H. Thomas. *R2D2 in a Softball: The Portable Satellite Assistant.* Proceedings of the 5th international conference on Intelligent user interfaces, 2000.

[HV2002] H. Hexmoor, J. Vaughn. *Computational Adjustable Autonomy for NASA Personal Satellite Assistants.* Proceedings of the ACM symposium on Applied computing, 2002.

[Lewi1998] J. Lewis, N. Packham et al. *The Lunar-Mars Life Support Test Project Phase III 90-day Test: The Crew Perspective.* 28th International Conference on Environmental Systems, 1998.

[Nasa] National Aeronautics and Space Administration. *NASA Homepage.* http://www.nasa.gov/

[Nasa1991] National Aeronautics and Space Administration. *Press Release 1991/1365.* http://www.jpl.nasa.gov/releases/91/release_1991_1365.html

[Nasa1997] National Aeronautics and Space Administration. *Fiscal Year 1997 Budget Summary.* ftp://ftp.hq.nasa.gov/pub/pao/budget/FY97budget/FY97_budget.txt

[Nasa2004a] National Aeronautics and Space Administration. *Mars Fact Sheet.* http://nssdc.gsfc.nasa.gov/planetary/factsheet/marsfact.html

[Papa1999] T. Papaioannou. *Mobile Agents: Are They Useful for Establishing a Virtual Presence in Space?* AAAI Symposium on Agents with Adjustable Autonomy, 1999.

[Pell1997] B. Pell, D. Bernard, S. Chien, et al. *An Autonomous Spacecraft Agent Prototype.* Proceedings of the first international conference on Autonomous agents, 1997.

[PSA] National Aeronautics and Space Administration. *Personal Satellite Assistant Overview.* http://ic.arc.nasa.gov/projects/psa/overview.html

[RESA] Wayne County Regional Education Service Agency. *Galileo Engineering.* http://www.resa.net/nasa/engineer.htm

[Rouf2002] C. Rouff. *Autonomy in Future Space Missions.* AAAI Workshop on Autonomy, Delegation, and Control, 2002.

[Sier2001] M. Sierhuis. *Modeling and Simulating Work Practice; Brahms: A multiagent modeling and simulation language for work system analysis and design.* PhD thesis. University of Amsterdam, 2001.