

# A MULTI-ROBOT ARCHITECTURE FOR PLANETARY ROVERS

R. S. Aylett<sup>†</sup> and D. P. Barnes<sup>‡</sup>

<sup>†</sup>Centre for Virtual Environments

<sup>‡</sup>Department Electronic & Electrical Engineering  
University of Salford, M5 4WT, UK

## ABSTRACT

Salford University has for some time been working in the area of multiple co-operant autonomous mobile robots. While there are a number of applications for this technology in remote hazardous environments, such as rover based planetary exploration, the required science and engineering is only just beginning to be realised. The key questions are firstly, how to control the interaction of these robots with themselves and their environment? and secondly, how to interact with this group of robots from the point of view of a remote operator? What has emerged is a novel hybrid architecture that contains a reflective planning agent which is capable of translating high level operator goals into low level behaviour missions that can be executed by multiple autonomous mobile robots. Two real robots have been used as part of our studies and this paper details our hybrid approach and the results obtained so far.

**Keywords:** co-operating mobile robots, hybrid architecture, behavioural agents, reflective agent.

## 1. INTRODUCTION

The University of Salford, in collaboration with UK Robotics Ltd., is currently investigating the issues associated with the application of multiple mobile robots to remote hazardous environment operations. We have recently completed a project titled: Multiple Automata for Complex Task Achievement (MACTA<sup>1</sup>), which focused upon a laboratory based robot task involving two of our research robots, Fred and Ginger. This task was based upon an object acquisition and relocation exercise that required both robots. Firstly, Fred had to acquire an object whilst being monitored closely by Ginger. Secondly, both robots then had to co-operatively transfer the object to a given location. Thirdly, Fred had to release the object to Ginger who then had to deposit the object at the desired location. This laboratory task contained many salient co-operant robot problems, and a major result of this work was the design of a *hybrid* architecture for supervising multiple co-operant autonomous mobile robots.

Figure 1 shows an overview of the resultant MACTA architecture. A single *reflective agent* directs *behavioural*

<sup>1</sup>The MACTA research was funded by the UK EPSRC, Control, Design and Production Group, GR/J49785.

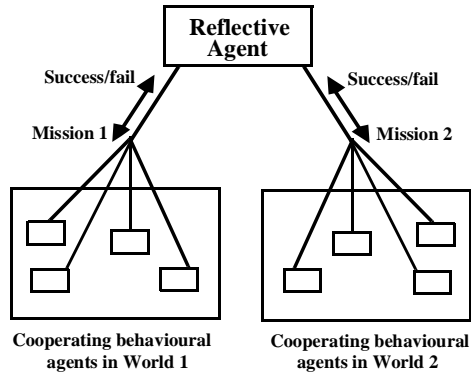


Figure 1. MACTA high level architecture.

*agents* (robots) organised in world-based clusters. The reflective agent accepts goals from a human operator, each of which is translated into a mission for a group of behavioural co-operating agents. These carry out the mission autonomously, without reference to the reflective agent until either the mission is complete, or until any agent is involved in a non-recoverable failure. The success of an abstract plan, Gat [6], generated by our reflective agent depends upon the ability of the behavioural agents to respond to local conditions in the environment and to cope with the detail of execution. Indeed, many human organisations work in precisely this way: a supervisor produces a plan and operatives are responsible for determining how these instructions are carried out at the detailed level. The following sections of this paper discuss the components of our hybrid MACTA architecture.

## 2. RESEARCH BACKGROUND

During the initial design of our MACTA architecture, it was realised that such a system might find application in a wide variety of remote hazardous environments, including planetary exploration using space rover technology. Consequently, throughout our work we have been mindful of relevant space research, most notably in the areas of AI task planning and planetary rover control design.

**AI task planning.** This has been widely used by NASA, with the Planning and Scheduling AI Group at JPL fielding a number of applications. Two of the best known are the Multimission VICAR Planner (MVP) and DPLAN.

MVP uses AI planning techniques to automatically synthesise executable image processing procedures to satisfy science requests, so that users can produce tailored images without having to master the complexity of the underlying software suites. DPLAN is a tool for automatically generating antenna operations procedures for facilities of the Deep Space Network used to communicate with NASA's probes and exploration craft. It is a good example of the use of AI planning to provide a simple, clear, interface in command and control.

Having proved the worth of AI task planning in ground-based systems, NASA are now pioneering the use of AI planning in the Deep Space 1 probe. Here it is specifically intended to provide a level of autonomy never before tested in a space exploration craft, with a clear human interface for ground-based interaction. Their planning work has resulted in a general planning and scheduling package called ASPEN (Automated Scheduling and Planning Environment). ASPEN is a modular, reconfigurable application framework which is capable of supporting a wide variety of planning and scheduling applications: its primary application area is in the spacecraft/rover operations domain. Its purpose is to reduce operations cost and increase the autonomy of rover operations by automating the command sequence generation process, including the encapsulation of operation specific knowledge. This allows a small operations team without subsystem experts to carry out rover command. It is being integrated with NASA's Web Interface for Telescience (WITS).

However, a key question with the use of AI planning has been whether the command sequences generated map directly onto the actual operations on the rover. If a planner is used in this deterministic fashion, then there is limited scope for the rover to use its own sensory information (which is likely to be more accurate and up-to-date than the world model used for planning), due to the planner sequences having complete control. Thus a rigid use of AI planning can actually reduce the amount of rover autonomy. What is required is a more equal relationship in which command sequences can be interpreted by the rover according to local conditions.

**Planetary rover control.** As is well known, in July 1997, the NASA probe Pathfinder reached Mars and entered its atmosphere. Among the payload of the probe was the rover Sojourner, the first semi-autonomous robotic vehicle ever launched on Mars. Sojourner possessed a large number of sensors which provided internal state and external environment information. Despite limited low level motor speed control, it was possible to control the vehicle either directly via a remote human operator or via two higher level autonomous commands. The first and most important was the *GoTo-XY* command which made the robot go to a specified  $x, y$  point. If any insurmountable obstacles were encountered whilst travelling to this point, then an obstacle avoidance algorithm would take control whilst still ensuring that the desired end point was reached. A second high-level command was the *seek-nearest-stone* which provided the position of a large stone to which Sojourner was to travel.

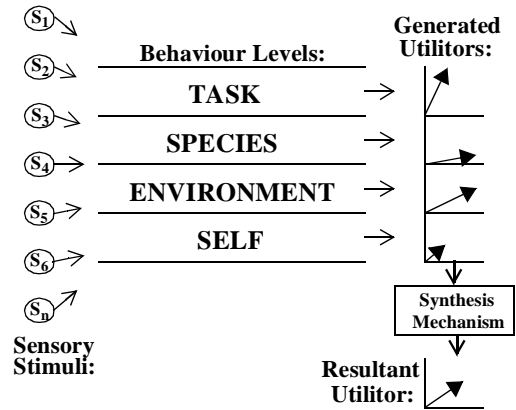


Figure 2. *The behaviour synthesis architecture (BSA).*

The control system worked in three stages: first the vehicle's position was determined from appropriate sensors and data; second, its sensors were used to locate any obstacles; third, it moved on one step, which approximated to 10cm.; fourth, the routine was repeated. One can regard this *sense then act* cycle as a primitive form of sensor-driven or behavioural control and the obstacle avoidance and 'beacon' navigation algorithms constituted important autonomous behaviours when not under manual tele-operator control. If NASA is to achieve their aim of transporting samples back from Mars to Earth, then it is very probable that a greater repertoire of autonomous behaviours will be needed. A sample retrieval robot will be needed to collect the Mars samples obtained by other rovers. The samples will then need to be taken and stored on an Earth-return ascent vehicle prior to their transportation. This retrieval and storage task will be considerably more complex than those activities undertaken by Sojourner, and task planning and robot behavioural control, and their integration, will undoubtedly play a major rôle.

Having identified the salient aspects associated with AI task planning for space applications, and planetary rover control, we were able to proceed with the specification of our MACTA architecture. First this involved the design of a behavioural multi-agent control method and secondly a reflective agent architecture.

### 3. MULTI-AGENT CONTROL

The interaction of multiple behavioural agents can be regarded as a continuum between two diverse types of behaviour. At one extreme, the behaviour can be regarded as being egotistic, where an agent is concerned purely with self directed behaviour, e.g. energy conservation. At the other extreme their behaviour can be regarded as being altruistic, e.g. when a group of agents need to work together to perform some common task. Given the different behaviours that can be found in single agent and multi-agent scenarios, the research focused upon the design of a control architecture that could accommodate

diverse and usually conflicting behaviour types. What emerged was the BSA, see figure 2, which complements the mobile robot control architectures of Arkin [1] and Brooks [3]. For purely conceptual convenience, four different behaviour levels in the architecture were identified: A **self** level contains those behaviours concerned with the maximisation, and if possible, replenishment of internal resources, (e.g. modifying a rover’s locomotion based upon available on-board power). An **environment** level contains those motion behaviours associated with activities involving other objects within the robot’s environment, (e.g. avoiding collision with rocks, other rovers and lander vehicle). A **species** level contains those behaviours associated with co-operant activities, (e.g. maintaining a correct position and orientation with respect to a rock sample while co-operatively relocating this sample with the aid of another rover). A **task** level contains those behaviours specific to a particular task, (e.g. navigating to the initial location of an rock sample to be relocated, then subsequent navigation to the desired lander location).

Sensory stimuli, from our developed robot sensor systems, provide the internal and external state information needed for the various levels and from each level, appropriate motion responses are generated that relate to the desired actuation. Any level can contain a number of *behaviour patterns*, **bp**’s, where

$$\mathbf{bp} = \begin{bmatrix} r \\ u \end{bmatrix} \quad \text{and} \quad \begin{cases} r = f_r(s) \\ u = f_u(s) \end{cases} \quad (1)$$

$r$  is the desired motion response and is a function,  $f_r$ , of a given sensory stimulus,  $s$ . Associated to every response is a measure of its utility or importance,  $u$ . This quantity is a function,  $f_u$ , of the same sensory stimulus. Hence a **bp** defines not only what a robot’s motion response should be for a given sensor input, but it also provides a measure as to how the relative importance of this response varies with respect to the same input. The values of  $r$  and  $u$  constitute a vector known as a *utilitor*. At any point in time,  $t$ , multiple conflicting motion responses are typically generated. For example, a robot may be navigating towards a lander location while co-operatively relocating a rock sample with the aid of another robot, when an obstacle unexpectedly appears in its path, and at the same time it senses that it must move out of a shadow to maximise its solar energy resource. In such a situation, what should it do? In the BSA, conflicting motion responses are resolved by a behaviour synthesis mechanism to produce a resultant motion response. Competing utilitors are resolved by a process of linear superposition which generates a resultant utilitor,  $UX_t$  where:

$$UX_t = \sum_{n=1}^m u_{(t,n)} \cdot e^{j \cdot r_{(t,n)}} \quad (2)$$

and  $m$  equals the total number of related utilitors generated from the different behaviour levels, e.g. all those concerned with translation motion or those concerned with rotation motion. Given a resultant utilitor, a resultant utility,  $uX_t$ , and a resultant motion response,  $rX_t$  are simply obtained from

$$uX_t = \frac{|UX_t|}{m} \quad \text{and} \quad rX_t = \arg(UX_t) \quad (3)$$

$X$  identifies the relevant degree of freedom, e.g. translate or rotate, and the resultant motion response,  $rX_t$ , is then executed by the robot.

Despite the success of the BSA, it did initially suffer from a problem common to all behavioural architectures. Effectively, **bp**’s may interact in ways which are not useful to the robot. While utility functions are ideal in the context of generating a resultant robot motion, they are sensor dependent not sub-task dependent. Hence situations can arise when the associated utility for a particular **bp** needs to be forced to zero, irrespective of its input sensor value. This effectively produces a **bp** which does not contribute to the resultant motion response. We argue that the root of the problem is in allowing all **bp**’s to be active at all times rather than restricting active behaviours to those most useful for the achievement of a particular sub-task. What was required was a means of allowing the task structure to create a context in which only appropriate **bp**’s would be activated. In the BSA, a structure known as a *behaviour script* was designed for this purpose. A behaviour script consists of *behaviour packets*, each of which contain a triplet: [sensor pre-condition(s), **bp**’s, sensor post-condition(s)]. Sensor pre- and post-conditions are a combination of a particular sensor and either an initiating or terminating condition. These are similar to the continuous action model implemented by Gat [5] in which activities are initiated and terminated by conditions, while Zelinsky’s ‘graphical sketches’ [9] represent a more specialised form of this approach to navigation only. As each behaviour packet within the behaviour script is carried out, the pre-condition for the next is encountered so that finally, the whole script is executed. Hence this process constitutes an ideal mechanism for sequencing behaviours, see figure 3. Further BSA details can be found in Barnes [2].

#### 4. OUR REFLECTIVE AGENT

The rôle of the reflective agent is to accept an overall goal (or mission) from a human operator, which is then accomplished by a particular world-cluster of robots. Our design includes a publicly available non-linear planner, UCPOP, which can be used to decompose an operator goal into planning primitives. As part of this process, behavioural agents with the appropriate behavioural repertoire can be recruited for the mission: this is an example of intertwined planning and scheduling where the agents selected for the mission impact on the sub-goals to be met and the final content of the abstract plan, Muscettola [7]. The planner produces a directed graph of planner operators, (*plan-net actions*), which can then be converted into behaviour scripts for the recruited behavioural agents and communicated to them. Upon receipt of a behaviour script, the behavioural agents can proceed to carry out the mission, and when it is complete, report back success to the reflective agent. Note that in our design there is no assumption of continuous communication between the reflective and behavioural agents.

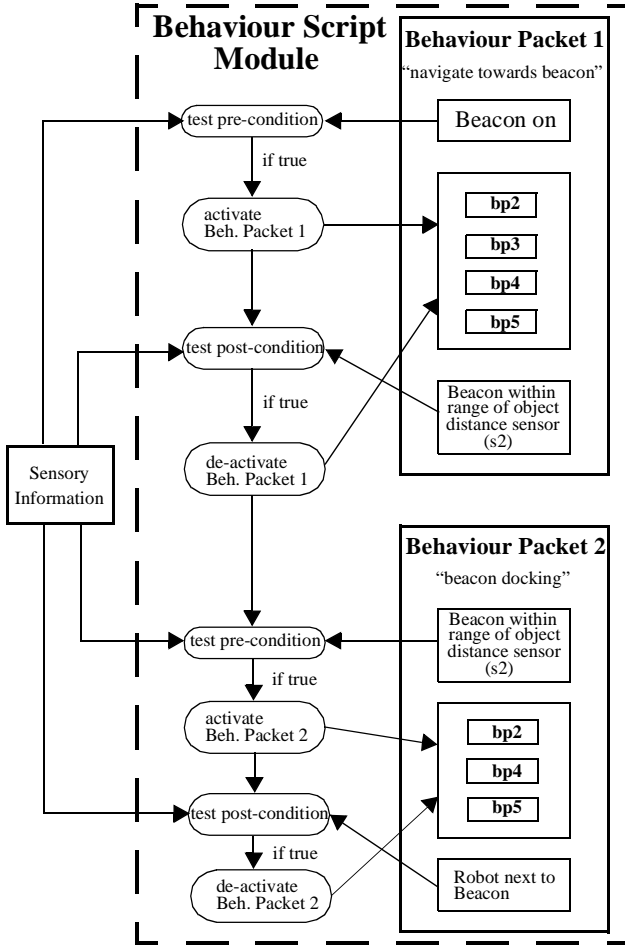


Figure 3. *Behaviour script example. bp2 - bp5 represent appropriate behaviour patterns while s2 is a robot to beacon distance measuring sensor.*

The function of each component in our design may be clarified by describing how information flows through the system in a sample interaction between a user. The object relocation task already discussed will be used as an example. The user sets a goal for the robots to achieve via the *HCI* (Human/Computer Interface) component in figure 4. An example in the object relocation scenario might be “*move rock\_sample to lander*”. This goal is passed to the *planner* which, using the *world model* describing the initial state of the world and a *knowledge base* containing abstract representations of the actions capable of being executed by the behavioural agents, produces a non-linear plan and returns it to the user via the *HCI* for acceptance or rejection. This may involve communication with other agents since as already stated, the reflective agent attempts to allocate behavioural agents to its plan as part of the planning process. If accepted, the plan is then passed to the *mission organiser*, which interprets the plan and converts it into behaviour scripts. This conversion process is carried out by matching each of the linking goals in the plan-net of primitive actions, generated by the planning process, against the index-

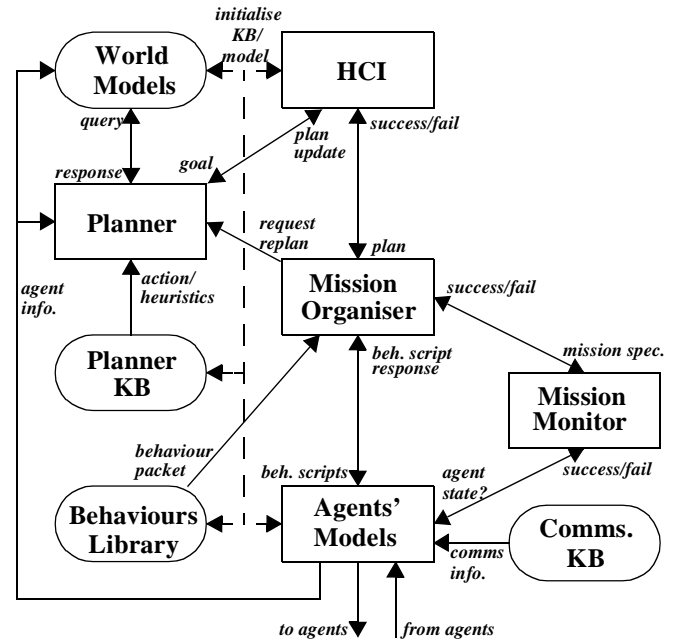


Figure 4. *Reflective agent architecture (round cornered boxes represent knowledge bases while sharp cornered boxes contain active processes).*

ing conditions in the *behaviours library* component in figure 4. The behaviours library records the available capabilities of agents in a form that can be utilised by the reflective agent’s planning system. This is achieved by grouping together behaviour names in packets, each of which is labelled with the symbolic preconditions it requires to be incorporated in a plan, and by the symbolic goals it achieves. The content of a behaviour is not accessible to the reflective agent since it resides in the individual behavioural agents and even if it were, the planner could make no use of the non-symbolic sensor information within it. The behaviours library defines the level of primitive actions for the reflective agent’s planning system and thus the base abstraction level of the planning process. Note that many of the details involved in achieving a goal are abstracted away from planning owing to the behavioural agents’ capabilities. For example, the behavioural agents may navigate towards a beacon whilst avoiding obstacles, so that obstacle avoidance ceases to be a planning problem.

The mission organiser then passes the behaviour scripts to the appropriate robots by sending them to the corresponding *agent model*. This incorporates the relevant *communications* routine for the robot in question and delivers the behaviour script to the robot (via RF) and passes its acknowledgement back to the mission organiser. The mission organiser then spawns a *mission monitor* to which it passes details of the robots involved and an estimated time for the mission. If the robots have not replied with a success or fail message by the end of this time, the mission monitor will try to establish the status of the mission by contacting the robots. Initially we have assumed successful missions with each robot eventually returning a success message which is ultimately deliv-



Figure 5. *Fred and Ginger with capture-heads, manipulators, ultra-sonic obstacle and infra red beacon sensors and RF communications.*

ered back to the HCI. However the problems of failing missions are under active investigation and mechanisms for coping with failure will need to be incorporated into reflective and behavioural agents in due course.

## 5. RESULTS OBTAINED

**Our behavioural co-operating robots** - To perform the laboratory based co-operant object relocation task, Fred and Ginger were originally equipped with ultra-sonic sensors for obstacle detection, and an instrumented self-centering  $x-y$  table (historically known as the capture-head) upon which the object to be relocated could be placed. The capture-heads serve the purpose of providing each robot with information concerning the Cartesian position of the object relative to each robot's central axis. When an object is in contact with each capture-head, the two robots are effectively mechanically coupled (just as two people are when jointly carrying an object) and this means that the relative motion of one robot can be transmitted to the other robot and vice versa. Distance data from an array of ultra-sonic sensors and the  $x-y$  data from a capture-head formed the sensory input to each robot's BSA. Work on the project to date has resulted in additional beacon sensors, manipulators and RF communications being incorporated into both Fred and Ginger, see figure 5. Our infra-red beacon design has been extended to function as an inter-robot tracking system as well as being used for fixed beacon location. When the two robots are coupled, when co-operatively transporting an object, they act as an articulated unit. This is due to the 3 DOF passive compliant capture-heads and an induced rotation motion about the gripper vertical axes due to gripper slip on the object. The RF facility allows the reflective agent to communicate with a particular robot e.g. for delivering a behaviour script or obtaining status information. So far, we have found no need for the robots to directly communicate with each other via this facility.

**Behaviour patterns, packets and scripts** - To move towards being able to achieve our laboratory based robot

SLOTS	CONTENTS
name	A
tag	TRANSPORT
Active Behaviour Patterns	1. <i>translate</i> 2.& 4. <i>obstacle avoid</i> 3.& 5. <i>centre capture head</i>
Sensory Postcondition	robot detects beacon

Table 1. *TRANSPORT behaviour packet.*

Description	Translate bp's
1. <i>translate...</i> Generates a constant response with constant utility.	$r1 = 0.5$ $u1 = 0.5$
2. <i>obstacle avoid...</i> $s1$ - ultra sonic array. Decelerates robot when object detected. Negates any +ve bp's with max. initial utility.	$r2 = f_r(s1) = -s1$ $u2 = f_u(s1) = s1$
3. <i>centre capture head...</i> $s2$ - optical encoder. 'Over-damped' response to sensor input with 'under-damped' utility.	$r3 = f_r(s2) = \sqrt{s2} + 1$ $u3 = f_u(s2) = \sqrt{s2}$
Rotate bp's	
4. <i>obstacle avoid...</i> $s1$ - ultra sonic array. Rotate away from object with max. initial utility.	$r4 = f_r(s1) = s1 + 1$ $u4 = f_u(s1) = s1$
5. <i>centre capture head...</i> $s2$ - optical encoder. 'Under-damped' response to sensor input with 'under-damped' utility.	$r5 = f_r(s2) = \sqrt{s2}$ $u5 = f_u(s2) = \sqrt{s2}$

Table 2. *Active bp's for the TRANSPORT behaviour packet. All  $r$ ,  $s$  and  $u$  are normalised.*

task, we have had to develop new bp's, packets and scripts. Appropriate  $f_r(s)$  and  $f_u(s)$  functions were initially hand-crafted via experimentation. When satisfied with the performance of a set of bp's, they are grouped into a behaviour packet for inclusion into a script. Generally, behaviour packet generation is an iterative process requiring a good deal of bp refinement before they can be put into a given task script. So far we have created behaviour packets to co-operatively TRANSPORT an object, NAVIGATE to a beacon, DOCK with a beacon, RELEASE an object and TRACK another robot. Table 1 shows details of our TRANSPORT behaviour packet. Identified are those bp's that are active during this sub-task and table 2 shows their associated functions. As can be seen, simple functions have been used for each required  $f_r(s)$  and  $f_u(s)$ . As the robots are capable of translate (forward/backwards) and rotate (clockwise/anti-clockwise) motion, then the bp's have been designed accordingly.

**Plan-net actions** - An analysis of the various sub-tasks performed by our robots has led to the creation of a set of actions which form part of the planner KB component within our reflective agent architecture. Actions are required by the planner in order to generate plans and correspond to the set of behaviour packages which form the behaviours library component. A key part of the

SLOTS		CONTENTS
Identifier		1
Name		<i>TRANSPORT</i> (object, [robots], rel1, ref1, rel2, ref2, time1, time2)
Agents		[robots]
Pre-cond.		robot-position([robots], rel1, ref1, time1) robot-attribute([robots], [grippers], full, time1) object-position(object, in, [grippers], time1)
Post-cond.	Delete	robot-position ([robots], rel1, ref1, time1)
	Add	Intend. Effects transported (object, [robots], rel1, ref1, rel2, ref2, time1, time2)
	Other Effects	robot-position ([robots], rel2, ref2, time2)
Duration		time2-time1
Description		"[robots]" transport "object" from "rel1, ref1" at "time1" to "rel2, ref2" at "time2"

Table 3. *TRANSPORT* plan-net action (*pa*).

design of an action hierarchy for a planner is the specification of plan-net actions (*pa*'s). So far we have created the following: *TRANSPORT*, *NAVIGATE*, *DOCK*, *RELEASE1*, *RELEASE2* and *TRACK*. These have been implemented as CLOS objects and details of our *TRANSPORT pa* are shown in table 3. *Rel1*, *ref1* etc. refer to relationship-reference pairs such as *near*, *location1*, *at*, *location2* and *time1*, *time2* are expected start and finish times if appropriate. The *pa* ID, Name, Agents and Preconditions slots are self explanatory, but the Postconditions need explanation: *Delete* is a conjunction of predicates which are no longer true once the *pa* has been executed. *Add/Intended Effects* is a conjunction of predicates which become true once the *pa* has been executed. These represent the intended outcome. A planner can use these effects to achieve goals. *Add/Other Effects* is a conjunction of predicates which become true once the *pa* has been executed. The planner does not use these effects to achieve goals and these effects are not necessarily brought about only by executing this particular *pa*. *Duration* is the time required to execute the *pa* and the *Description* slot is there to provide an operator with details of those activities the *pa* symbolically represents.

Our created *pa*'s have been designed to enable a planner to generate a plan for our laboratory based robot task. These are mapped into behaviour packets, which when down-loaded (via our RF link) onto our robots, allow the robots to execute the desired task. Details of all the major planner design issues can be found in Coddington [4].

**MACTA simulator** - In addition to the co-operant object relocation task, as performed by our two real robots, a sample store scenario was also chosen both for its relevance to real-world space problems, and as a way of investigating the effect of scaling-up a planning problem. To study the application of our hybrid reflective/reactive agent approach to this scenario, a simulator was implemented in C using PHIGS (Programmers Hierarchical Interactive Graphics System), a graphics library publicly available from the MIT X Consortium. The sample

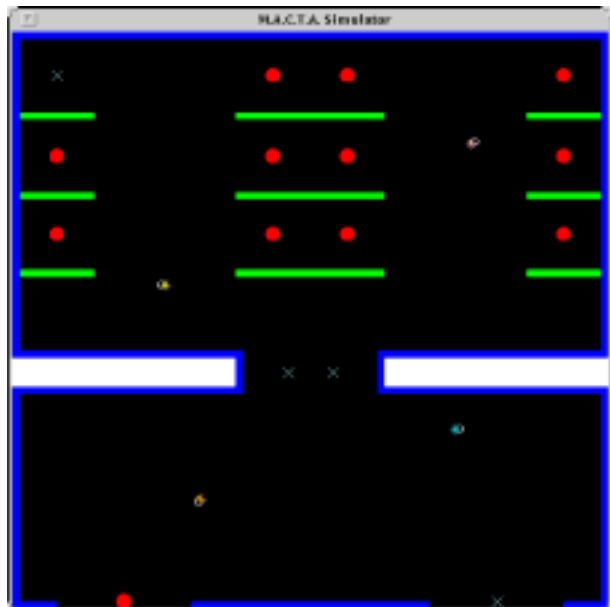


Figure 6. The MACTA simulator showing four robots capable of transporting and storing objects.

store scenario was implemented in our MACTA simulator and is shown in figure 6. This consists of two connected compartments. The upper area in the diagram is a sample store consisting of 12 storage locations divided by two aisles, while the lower area is entered by two doors. There are two interchange points between the two compartments (marked by two crosses). There are four robots (shown as unfilled circles with attached 'whiskers'), two in each compartment: each robot must remain within its allocated area. The two robots within the sample store are responsible for storing and retrieving samples (shown as filled circles) from their storage locations. Each of these robots is allocated an aisle. An example task involving the storage of samples might consist of the following manoeuvres: a sample deposited at one of the doorways in the lower compartment is acquired by one of the robots within that area and taken to one of the interchange beacons situated between the two compartments. One of the robots in the sample store goes to the appropriate interchange beacon and acquires the sample prior to its storage at the desired location.

The plan-net actions and behaviour packets for this scenario were the same as those for the object relocation task described previously. However, this scenario was considerably more complex from the point of view of the planner - there were four robots, each capable of sensing five beacon locations. Behavioural control of this scenario was a success, but it did expose the deficiencies of the UCPOP (University of Washington) planning system. UCPOP is a non-hierarchical planner and required excessive computation resources for this scenario. Nevertheless, the experiments undertaken using the simulator did demonstrate the ability of our MACTA architecture to deal with more complex multi-robot tasks than was previously achieved with just two real mobile robots.

## 6. CONCLUSIONS

Our early work into co-operant object relocation and the resultant design of the behaviour synthesis architecture incorporating behaviour patterns, packets and scripts, laid the foundations for this MACTA project. We have significantly increased our desired robot task achieving capability by focusing on a complex object acquisition and co-operative relocation problem. To investigate this application domain our studies have involved the use of real mobile robots, Fred and Ginger, and these have been considerably enhanced in terms of both hardware and software. They now support new beacon detection sensors, RF communications and manipulators for object acquisition and release. Additional behaviour patterns, packets and scripts have been created to utilise this new hardware. We have formulated plan-net actions for use by the planner component of our hybrid architecture. These actions constituted a significant milestone in our research as they were the key to successful plan to behaviour script translation. Using our architecture we have successfully demonstrated our two robots performing key sub-tasks of the laboratory based scenario, and thus have shown the viability of our hybrid approach in supervising real multiple co-operant autonomous mobile robots.

## 7. THE WAY FORWARD

Planetary rovers as shown by Sojourner, benefit from local autonomy since successful interaction with the planetary environment requires a real-time response to local sensor data. This is all the more essential when missions of increasing complexity are considered, such as a lunar rover exploring the bottoms of craters at the lunar pole, or Mars rovers such as Beagle 2, in which rock samples must be carried back to a lander for laboratory processing. Behavioural systems are strong candidates for an on-robot architecture since they are reactive and robust. Moreover they cope well with inter-robot cooperation, and thus support multi-robot activity which may be required for example, if micro-rovers or even mini-rovers are deployed, [8].

However, behavioural architectures have provided an opaque user interface and task inflexibility with new tasks requiring a new behavioural design. The Earth bound operator must be able to set global objectives for planetary rovers, and to sequence the overall components of the task without having to deal with low-level detail. It is here that AI task planning has a major rôle to play.

AI task planning is already being used by ESA for ground based or non critical tasks. For example, ARIANE IV payload management is carried out by the OPTIMUM AIV system, while the Integrated Learning System uses AI planning to generate just-in-time schedules for on-board astronaut training. In mission planning, complex missions such as ERS-1 have revealed the need for generic planning tools, and work has been carried out on

a Generic Mission Planning Toolset.

We have argued in this paper that one of the key issues in using AI task planning for systems such as planetary rovers, is creating a flexible relationship between top-down direction, and bottom-up reactivity. It is this type of user interface that our MACTA research has generated, and we believe it would allow cooperating planetary rovers to be supervised at a high-level remotely, whilst still retaining their local autonomy for real-time interaction with their planetary environment.

Our recommendation is that given the likelihood of an increase in the number of planetary rover projects in the future, it is worth considering a detailed feasibility study into the hybrid approach presented in this paper. This could involve an existing ESA planetary rover simulation, or a simulation of a forthcoming project such as Beagle. The study could investigate what additions should be made to the Generic Mission Planning Toolset to support operator interaction with behaviourally driven robots, both for solitary activities and those involving multi-robot cooperation.

## 8. REFERENCES

- [1] Arkin, R. C. 1989. Motor schema-based mobile robot navigation. *Int. Journal of Robotics Research* 81(4): 92-112.
- [2] Barnes, D. P., Ghanea-Hercock, R. A., Aylett, R. S., and Coddington, A. M. 1997. Many hands make light work? An investigation into behaviourally controlled co-operant autonomous mobile robots. In *Proc. 1st. Int. Conf. on Autonomous Agents*. Marina del Rey, CA, 413-420.
- [3] Brooks, R. A. 1986. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation* RA-2: 14-23.
- [4] Coddington, A. M., and Aylett, R. S. 1996. Plan generation for multiple autonomous agents: an evaluation. In *Proc. 15th Workshop of the UK Planning and Scheduling SIG*. Liverpool, Vol 1, 126-137.
- [5] Gat, E. 1992. Integrating planning and reaction in a heterogeneous asynchronous architecture for controlling real-world mobile robots. In *Proc. 10th National Conf. on Artificial Intelligence*. AAAI-92, 809-815.
- [6] Gat, E. 1993. On the role of stored internal state in the control of autonomous mobile robots. *AI Magazine*, AAAI, 64-73.
- [7] Muscettola, N. 1994. HSTS: Integrating planning and scheduling. *Intelligent Scheduling*, Zweben, M., and Fox, M. S. eds., Morgan Kaufmann, 169-212.
- [8] Putz, P. 1998. Space robotics in Europe: A survey. *Journal of Robotics and Autonomous Systems* 23, 3-16.
- [9] Zelinsky, A., Kuniyoshi, Y., and Tsukune, H. 1994. Monitoring and coordinating behaviours for purposive robot navigation. In *Proc. Int. Conf. on Intelligent Robots and Systems, Munich*. Vol 2, 894-901.