A MOBILE AGENTS APPROACH TO VIRTUAL LABORATORIES: ENABLING REMOTE OPERATION OVER THE INTERNET

L. M. Camarinha-Matos*, Walter Vieira*, and Octavio Castolo***

*New University of Lisbon - Faculty of Sciences and Technology
Quinta da Torre, 2825-114 Monte Caparica, Portugal
Tel.: +351-1-2954464 / ext. 3728 Fax: +351-1-2957786
**Instituto Superior de Engenharia de Lisboa, DEEC, Portugal
{cam, wv, lgc}@uninova.pt

Abstract: This paper presents the use of adaptive mobile agents for remote operation, enabling real-time response in spite of the characteristics of the communication channels, such as time-delays, availability, and reliability. Autonomy of the mobile agents is achieved through high levels of intelligence including execution monitoring and error recovery. Potential applications range from traditional telerobotics to virtual laboratories where mobile agents act as representatives of users in scientific experiments. Practical results are presented in a scenario where a SCARA-type robot is remotely commanded through the Internet.

Keywords: Agents, Teleoperation, Monitoring

1. INTRODUCTION

Virtual laboratories, and remote operation and supervision, in general, are attracting growing interest, due to the many potential applications, ranging from sharing expensive equipment, machines operating in hazardous or inaccessible environments, to spatial vehicles operating with large autonomy. In special, the number of applications using remote operation over the Internet has been growing at significant rates in various domains, such as remotely operated robots and telescopes, manufacturing systems, remote elderly care, remote surveillance systems, etc.

In this paper the specific case of the Internet, whose low costs and widespread availability make it very appealing as a basis for remote operation, is considered. The proposed approach addresses some difficulties, such as: i) when reasonable practical application domains are considered, high levels of heterogeneity are expected in the sensorial and equipment richness of the remote places, which can degrade the flexibility and scalability of the system; ii) Internet is characterized by long and variable time-delays and, very often, suffers from low levels of availability, raising new challenges in what concerns the reliability of the implemented system and its dependence on the characteristics of the network; and iii) the execution environments are potentially unstructured and uncertain which means that it is not adequate to resort to deterministically programmed systems. Therefore, special attention is given to the aspects related to how to deal with the large time-delays and the poor availability of the network connections, although related aspects, such as heterogeneity and uncertainty of the execution environments, are also addressed, as they have strong impact on the found solutions.

Another important aspect is concerned with execution monitoring. It is fundamental that

monitoring be a process running completely in parallel to the plan execution, which turns embedded exception handling mechanisms implemented in languages such as C++ and JAVA inappropriate.

In this paper a solution is discussed for enabling remote operation over communication channels with large time-delays and/or poor availability, and results of its application to remotely control a robot are presented. The approach is based on adaptive mobile agents (Fuggeta, *et al.*, 1998; Kotz and Gray, 1999; White, 1997) that carry hierarchical high level abstract plans which they adapt to the actual capabilities of each visited remote place (Camarinha-Matos and Vieira, 1999a). The hierarchical plans are annotated with information intended to guide the plan adaptation, execution monitoring, and error recovery processes. The mobile agents have a multi-thread architecture, which is a fundamental characteristic for the implementation of the necessary concurrency between plan execution and monitoring.

The proposed (multi) mobile-agent approach represents a sound basis for a flexible implementation of remote manipulation or virtual laboratory systems. Potential applications are in providing remote access to experiments in collaborative research, specially, in the case of expensive setups only available in a few places. Another application area is remote learning, especially for life-long training programs. Remote access to virtual teaching labs gives the student the opportunity to get almost "hands on" experience.

In addition to the mentioned generic advantages of the mobile agents approach, the virtual lab application can also benefit from the autonomy of the agents in order to not require a synchronous availability of the participants in a given experiment. These participants can delegate on their agent representatives the actual realization of some task, which will be done when the necessary conditions are satisfied. This approach allows high levels of decoupling between the participants in the experiment, both in spatial and temporal terms, since they don't need to be physically present in the place where the experiment takes place and neither do they need to participate on-line in the experiment.

2. THE INTERNET

Like in many other communication infrastructures, the Internet puts severe constraints in what concerns real-time operation, mainly:

Large time-delays. This has been addressed by various authors, and solutions have been proposed for very specific domains falling in two main classes. In one class the time-delay is estimated and somehow presented to the operator (for example, a cursor may be put over the image of the scene describing the current remote state pointing to the new position of a robot arm after the time-delay). In the second class, the intelligence and autonomy of the remote place is increased in order to enable supervisory control strategies in the sense that the operator only gives high level commands to the remote place when the remote place cannot proceed by its own means. The first case is only applicable to very specific domains whereas the second reduces the dependency on the characteristics of the network at the expenses of the flexibility, since any change of functionality or the alteration of the remote environments require the update of all the remote places. The solution described in this paper is based on mobile agents that are sent to the remote place to implement some desired functionality. Since the agents run on the remote place, high levels of independence on the characteristics of the network are achieved, yet preserving high levels of flexibility (new mobile agents can be built whenever needed to accomplish the desired functionality).

Poor availability of the network. This demands high levels of autonomy in the remote place that must have skills to allow the execution supervision and to adopt intelligent recovery strategies when errors occur. Some authors have tried to attack this problem with full deliberative agents (Wooldridge, 1999) equipped with planning capabilities that allow them to re-plan the new sequence of actions when errors occur. This is impracticable for many real domains due to performance insufficiency or to the complexity of the considered domain, which makes difficult the implementation of full autonomous systems. In our proposal the mobile agents carry with them hierarchical abstract plans annotated with monitoring and error recovery information. These annotations and the hierarchical abstract plan are used as guidelines by the execution supervision component of the agents. Since the abstract plans may be generated off-line by mixed initiative planners, it is possible to embed very sophisticated execution and error recovery strategies. On the other hand, the hierarchical structure of the plans allows an easier control of the complexity of plan repair during error recovery.

Heterogeneity of the execution environments. This is a crucial aspect when considering remote operation in large scale. It is not practical to have a different agent for each differently equipped remote place, even in those cases where the desired functionality at some reasonable abstraction level is the same. In this work the abstract plans carried by the agents are adapted (i.e. refined) according to the actual capabilities the agents find in the remote places they visit. As in the previous point, the hierarchical nature of the abstract plans is the key to control the complexity of the plan adaptation. This refinement capability is, of course, limited to a target domain for which the remote agent was designed.

3. AN ARCHITECTURE FOR ADAPTIVE MOBILE AGENTS

In order to cope with these characteristics of the communication channels, an architecture for adaptive mobile agents was previously defined. It is called IMAJ (Intelligent Mobile Agents in JAVA) and comprises three main components:

1) The mobility component that implements the process of agent migration (see (Camarinha-Matos and Vieira, 1998) for more details on this component).
2) The coordination component that implements the coordination infrastructure allowing the agents to coordinate their activities in order to achieve a well behaved system. Basically, each agent communicates with the other agents currently executing in the same server through a local tuple-space. Each agent also sees the tuple-space of its home place (the server where it was launched for the first time) through which it can cooperate with agents originated in the same server (Cabri and Zambonelli, 1998).
3) The execution supervision component that implements the mechanisms that allow one agent to adapt its high level abstract plan to the exact environment it finds in each visited place, and to execute the adapted plan, including execution monitoring and error recovery (Vieira and Camarinha-Matos, 1999).

Execution supervision involves plan adaptation, execution monitoring and error recovery. Plan adaptation allows an agent to adapt its high level plan for execution in each place, according to the capabilities found there; execution monitoring and error recovery are crucial since the agents may be operating autonomously for long periods of time.

In contrast with most previous works on execution supervision that assumed very restricted application domains, this work is concerned with more general solutions that address a wide range of application domains implying that the actual composition of each execution environment cannot be known in advance. Furthermore, when dealing with remote operation, the agents must run with a high degree of autonomy in uncertain environments. To achieve this goal, an approach based on general monitoring and recovery methods and on plans with annotations intended to help the execution monitoring and error recovery was adopted. A hierarchical plan structure was considered, since, besides other advantages of hierarchical planning, it allows the specification of monitors at various levels of detail, which is very appropriate for complex domains. Furthermore, the hierarchical approach is a powerful mean to structure interesting monitoring strategies that range over a set of low level actions.

Another important aspect when considering real-time response is that the execution of one task must not cause significant degradation of the execution of other tasks, which turns the sequential execution approach found in many systems inappropriate. This is the reason why the architecture of the execution supervision component (Fig. 1) extensively uses multi-thread programming. This means that provided enough resources are available, parallel branches in a plan are executed in parallel, because, besides other concurrent activities, each primitive action runs in its own execution thread. Furthermore, multi-threading simplifies the implementation of reactive behaviors according to the events that occur during the execution of an action.

In this architecture, executable actions (EA) are performed as concurrent threads and interface the local environment via a set of effectors. They call local procedures determined by the capabilities they are associated with. It is admitted that the local execution environment is equipped with adequate monitoring and error recovery mechanisms at the execution level, in such a way that if execution of an EA fails, sufficient information for characterizing the failure is obtained from the environment. This information is used for high-level error recovery.

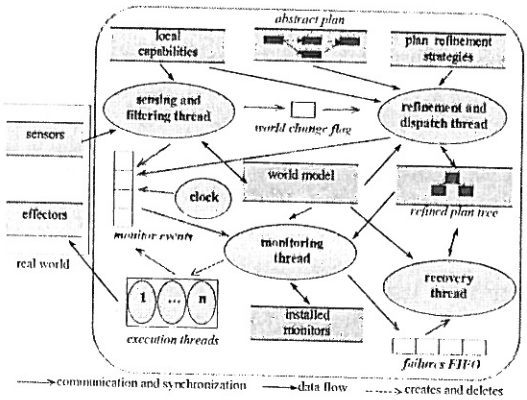

Fig. 1. Architecture of the execution supervision component of a mobile agent

The sensing and filtering thread is a high priority task that runs periodically and is responsible for maintaining an updated persistent world model, at a rate determined by the dynamics of the domain. A notification event is sent to the monitoring thread (via the monitor events FIFO) for each perceived change of the world state. The refinement and dispatch thread is also notified of these changes (via the world change flag - a binary semaphore).

The refinement and dispatch thread runs in two phases: i) in the first phase it adapts the carried abstract plan; ii) in the second phase it acts as a dispatcher, scheduling next EAs for execution. When all pending EAs are waiting for the completion of precedent actions, this thread blocks in the world change flag to wait for a world change.

The monitoring thread handles all the events in the agent (change of the world state, clock ticks, start and

termination of EAs). It installs and removes monitors (when EAs start and terminate), and checks the installed monitors when clock events or world change events are received. When a monitor condition is violated, the recovery thread is notified by the failures FIFO.

The recovery thread executes the recovery procedures associated to the notified error, trying to obtain a suitable plan repair that hopefully will allow the normal continuation of the agent's execution.

The specification of the abstract plans carried by the agent is done through the language MAAPL (Mobile Agents Abstract Plan Language) (Camarinha-Matos and Vieira, 1999b). This language allows the definition of hierarchical abstract plans with the specification of several monitors (Reece and Tate, 1994) and several recovery strategies if a monitor fires. Currently the following types of monitors may be specified: **pre-condition monitors** that fire if the specified condition on the preconditions of the action fails; **effect monitors** that fire when the specified condition on the effects of the action fail; **maintenance monitors** that are used for continuous observation of some condition; **failure monitors** that enable the definition of recovery strategies at the parent level when some child action fails; and **time-out monitors** that fire if the action doesn't terminate within the specified time limit. Each monitor has associated one of the following recovery strategies: **repair** is the default for pre-condition monitors and tries to find a patch plan that repairs the failed conditions; **retry** implies the repetition of the action with its current refinement; **redo** is used to obtain an alternative plan refinement for the action; **ignore** is the default for effect monitors and ignores the error; **fail** is used to propagate the failure to the upper levels in the plan hierarchy.

Hierarchical refinement in MAAPL can be one of the following: **seq** that specifies a sequence of sub-actions; **choice** that specifies a set of actions one of which must be chosen; **par** that specifies a set of actions whose refinements will be executed concurrently; and **adapt** and **goal** that specify that a plan must be obtained locally taking in account the capabilities of the environment. The difference between **adapt** and **goal** is that **adapt** takes the preconditions of the action as the initial state of the planner whereas **goal** takes the current real state. The former is used for obtaining plans that are executed as protocols and so require that execution of the refined plan starts at the preconditions of the upper level action; the later is used for less restrictive situations where all that is required is the achievement of the effects of the action.

4. AN EVALUATION SCENARIO

For evaluation of the proposed solution an assembly scenario where a simplified version of the Cranfield benchmark is performed was considered. The experiments consist of assembling three parts, as in Fig. 2, using a task-level control strategy.

The original environment is composed of a SCARA-type robot (SONY SRX-4CH), a pallet for assembly, and two dispensers, one for parts of type **A** and the other for parts of type **B**. This environment was enriched with a set of infrared sensors for detecting the presence of parts in several points, as shown in Fig. 2. An inexpensive video camera was also included, which allows visual feedback to be provided at the client side.

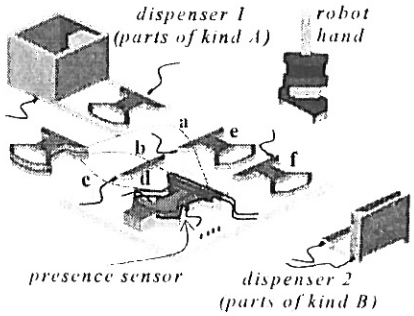

Fig. 2. The remote environment layout.

An IMAJ server (remote server) was integrated in this environment. A second IMAJ server (client server) was installed in another machine that communicates with the first server through the Internet. In this server a client environment was set up to receive sensorial feedback from the remote server and to allow the user to send mobile agents to there. This was accomplished by using two agents: the **Feedback Collector** that is launched in the client server and immediately migrates to the remote server where it continually senses the environment and puts the sensed information in its home tuple-space, and the **Visualization and Simulation** agent that runs in the client server and displays the information sent by the **Feedback Collector**, as illustrated in Fig. 3.

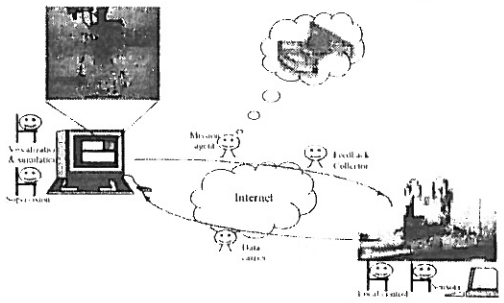

Fig. 3. Feedback collector and Visualization agents

Two types of experiments were conducted:

Implications of time delays

Since all the experiments were conducted using a task-level control strategy, the main implication of the time-delays is on the time taken to complete one

assembly task. In the first experiment the assembly was done using the traditional remote procedure calling mechanism under a (simulated) 10 second round-trip time-delay and using low level commands such as **move**, **open grip**, **get grip**, **close grip**, etc. The task was terminated after about 18 minutes. A second experiment was done using a mobile agent that was launched in the remote server where it did the same assembly task spending about 3 minutes. These results confirm that remote operation using low-level commands is impracticable when the communication channels have large time-delays. One alternative would be to increase the level of the commands implemented in the remote server, but this solution sacrifices the flexibility of the system, as it was discussed previously. The use of mobile agents is a good solution, but high levels of autonomy are required.

Autonomy of the mobile agents:

The following set of experiments was conducted in order to evaluate the level of autonomy of the mobile agents allowed by the execution monitoring and error recovery capabilities.

Experiment 1: In this case an agent was sent to the remote server with the abstract plan illustrated in Fig. 4. The refined executable plan is shown in Fig. 5.

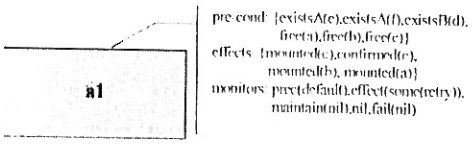

Fig. 4. Abstract plan for experiment 1

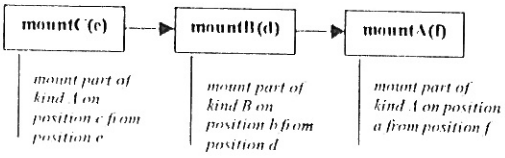

Fig. 5. Refinement of the plan of Fig. 4

With this simple plan several errors were intentionally introduced. The agent recovered from the errors in all situations where a recovery plan was theoretically possible. For example, when after action **mountA(f)** an error was introduced telling the agent that the first part was not mounted, the agent fired the effect monitor of action **a1** which indicates recovery strategy **retry**. As a consequence, the agent tried to re-execute the same executable plan, but then the pre-conditions of **a1** where not available, which caused firing of its pre-condition monitor. Since the default strategy for pre-condition monitors is **repair**, the agent generated and executed a repair plan that un-mounted all parts before resuming the initial plan. Several other errors were provoked with this abstract plan and other similar plans with variations in the monitors. The agents could recover from all recoverable errors.

Experiment 2: Since this is a simple domain, the agents from the previous experiments were able to recover from all recoverable errors, although sometimes with recovery plans involving too many actions. In order to show the advantages of the use of hierarchical plans an agent with the abstract plan depicted in Fig. 6 was sent to the remote server. In this plan, action **a1** is refined in a lower level composed of the sequence **a2**, **a3** and **a4**. Action **a2** is in turn refined in a choice between actions **a5** and **a6**, and **a4** is refined in a choice between **a7** and **a8**.

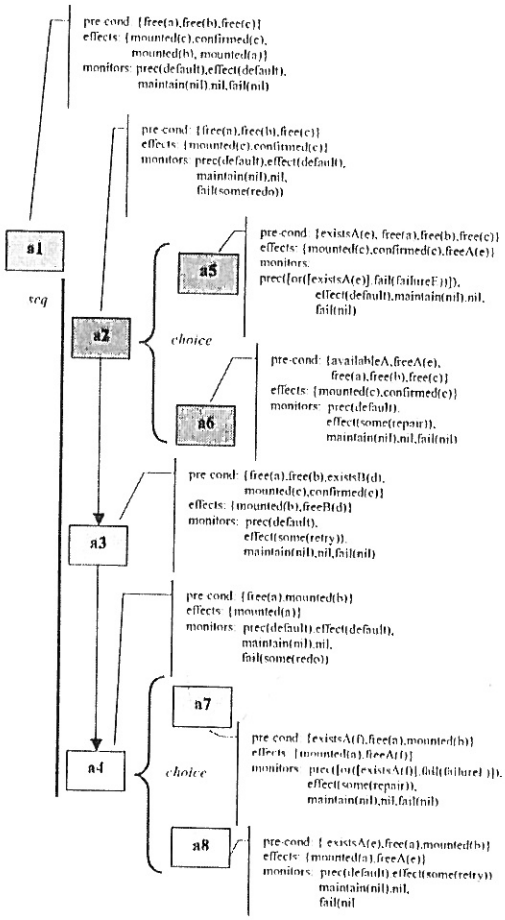

Fig. 6. Abstract plan for experiment 2

The refined executable plan is the same as in the previous experiment (Fig. 5). When the plan started an error was introduced which caused pre-condition **existsA(e)** to fail and firing of the pre-condition monitor of **a5**, which in turn caused propagation of failure to the upper level and, consequently, fired the failure monitor of **a2** whose recovery strategy is **redo**. As a consequence, the second branch of the choice in **a2** was selected and the new plan composed of the sequence "**moveA(e)**, **mountC(e)**" substituted the sub-set of the original executable plan composed only of action **mountC(e)** (Fig. 7).

However, since there was no part **A** available on the output of the dispenser, the pre-condition **availableA** of **a6** failed and fired its monitor whose recovery strategy repaired the error with one single action

placeA that directs the dispenser to move a new part to its output (see Fig. 8).

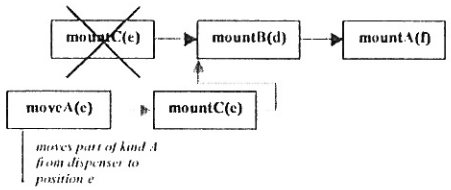

Fig. 7. New executable plan resulting from failure of action **a5** in the plan of Fig. 6

Later on in the same experiment, another error occurred because a part of type **B** was not available in position **d**, causing the pre-condition monitor of **a3** to fire and repairing the plan with the sequence "**placeB, moveB(d)**" shown in Fig. 8. The agent took about 5 minutes to complete the assembly, against the 3 minutes that it would take if no errors had occurred. Again, the agent was capable of recovering from errors whenever a recovery plan existed theoretically. On the other hand, the recovery plans tended to be smaller and more opportune than when no hierarchical decomposition is used.

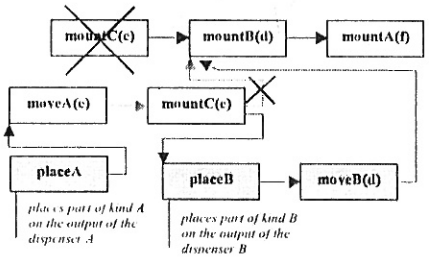

Fig. 8. Repairing pre-condition **existsB(d)** of **a3**.

5. CONCLUSIONS

In this paper a solution for remote operation based on adaptive mobile agents was presented. This solution allows a conciliation between the flexibility achieved with remote operation based on low level commands implemented in the remote site and the independence on the characteristics of the communication channels achieved by remote operation based on high level commands implemented in the remote site. In order to cope with the high levels of heterogeneity expected in the remote places the agents carry hierarchical abstract plans not completely specified. The agents obtain an executable plan by refining these abstract plans according to the capabilities they find in each site. Experiments conducted with a real robot showed that the proposed solution is very independent of the characteristics of the communication channels such as time-delays and availability. Concerning autonomy, these experiments showed that the agents could recover from errors whenever it was theoretically possible. Although the considered domain is very simple, the number of possible situations is considerable and turns difficult the adoption of deterministic solutions, especially if we consider that the same agent can visit several places with different execution environments. In fact, recovering from many of the errors introduced in the experiments would require human intervention if a traditional approach would be adopted. Therefore it is the opinion of the authors that the proposed solution is very effective in supporting remote operation over the Internet and long distance scenarios such as the spatial exploration domain.

ACKNOWLEDGEMENTS

This work was funded in part by the Portuguese Ministry of Science and Technology through the PRAXIS-XXI project Telecare

6. REFERENCES

Cabri, G. Leonardi and F. L. Zambonelli (1998). How to coordinate internet applications based on mobile agents. In: *Proc. of WETICE'98 - workshop on coordination architectures for distributed web applications.*

Camarinha-Matos, L. M. and W. Vieira (1998). Adaptive mobile agents for teleroboties and telesupervision. In: *Proc. of INES'98*. pp. 79-84.

Camarinha-Matos, L. M. and W. Vieira (1999a). Intelligent mobile agents in elderly care. *Robotics and Autonomous Systems* **27**(1-2), 59-75.

Camarinha-Matos, L. M. and W. Vieira (1999b). MAAPL: A language for adaptive mobile agents with execution monitoring. In: *Proc. of INES'99*. pp. 171-176.

Fuggeta, A., G. P. Pico and G. Vigna (1998). Understanding code mobility, *IEEE Transactions on Software Engineering* **24**(5), 346-361.

Kotz, D. and R. S. Gray (1999). Mobile agents and the future of the internet. *ACM Operating Systems Review* **33**(3), 7-13.

Reece, G. and A. Tate (1994). Synthesizing protection monitors from causal structure. In: *Proc. of the 2nd Int. Conf. on Planning Systems*. AAAI Press.

Vieira, W. and L. M. Camarinha-Matos (1999). Execution monitoring in adaptive mobile agents. In: *Cooperative Information Agents III, LNAI* (M. Klusch, O. Shehory and G. Weiss, Eds.). Vol. 1652. pp. 220-231. Springer Verlag. Berlin et al.

White, J. E. (1997). Mobile agents. In: *Software Agents* (J. M. Bradshaw, Ed.). pp. 437-472. MIT Press.

Wooldridge, M. (1999). Intelligent agents. In: *Multiagent systems* (G. Weiss, Ed.). The MIT Press.