16

ADAPTIVE MOBILE AGENTS: ENHANCED FLEXIBILITY IN INTERNET-BASED REMOTE OPERATION

Walter Vieira[1][2] and L. M. Camarinha-Matos[1]
[1]*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
[2]*Instituto Superior de Engenharia de Lisboa, DEEC, Portugal*
{wv, cam}@uninova.pt

The number of applications involving some kind of remote operation over the Internet has been growing in the last years. Most of these applications show high levels of heterogeneity of the equipment to be operated which, in conjunction with the low availability and long time-delays that characterize the Internet, pose serious difficulties to the traditional approaches based on remote procedure calling mechanisms, either in terms of reliability and flexibility. Solutions have been tried to circumvent the dependency on the characteristics of the network, but with the cost of a further degradation in flexibility. In this paper an alternative solution based on adaptive mobile agents is described which overtakes most of these difficulties.

1. INTRODUCTION

There is a growing interest in issues related to remote operation and remote supervision, due to the many potential applications, ranging from machines operating in hazardous or inaccessible environments to spatial vehicles operating with large autonomy as a consequence of the recent developments in computer networks and ubiquitous computing. In special, the number of applications using remote operation over the Internet have been growing at significant rates and spreading over various other domains, such as remotely operated robots and telescopes, manufacturing systems, virtual laboratories, remote elderly care, etc.

The low costs and widespread availability of the Internet make it very appealing as a basis for remote operation, but also raise 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 demands appropriate solutions to guarantee the appropriate levels of flexibility and scalability; ii) the 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; iii) the execution environments are potentially

unstructured and uncertain which means that it is difficult to cope with these environments by resorting to deterministically programmed systems.

To adequately face these difficulties more sophisticated and more reliable solutions have to be found allowing the increase of the autonomy of the equipment being operated, yet preserving a high degree of flexibility.

The mobile agents paradigm (Fuggeta et al, 1998) offers interesting characteristics when these issues are considered (Camarinha-Matos et al, 1998-b), because: i) moving the code to the places where the devices are located enables real-time response with reduced dependency on network availability and delays; ii) since new mobile agents can be built and sent for remote execution whenever needed, higher levels of flexibility and scalability are achieved.

Flexibility may be further improved if we take into account that in many situations the same abstract execution plan can be executed using different resources (machines), which is a very important aspect when we consider remote operation of many heterogeneously equipped execution environments. In order to take advantage of this possibility, the authors have proposed a solution based on mobile agents that carry general execution plans that must be refined/adapted when the agents reach their target places. An architecture to support remote operation over the Internet, based on mobile has been developed, which includes mobility, execution control (plan adaptation and execution monitoring), and coordination components. The issues related with plan adaptation were discussed in (Camarinha-Matos et al, 1999-a); execution monitoring and error recovery were discussed in (Vieira et al, 1999); and the language for MAAPL (Mobile Agents Abstract Plan Language) for description of hierarchical abstract plans with execution monitoring was described in (Camarinha-Matos et al, 1999-b). In this paper the global architecture of the mobile agents is presented which integrates new aspects related to coordination of the agents.

The rest of the paper is structured as follows: in section 2 some issues related to the mobile agents paradigm are presented and the advantages of this paradigm in the context of remote operation are discussed; the architecture of adaptive mobile agents for remote operation, including plan specification and adaptation, execution monitoring and coordination issues, is presented in section 3; in section 4 an application scenario which has been used for validation of the proposed architecture is described; and in section 5 some conclusions are presented and areas where further work is needed are pointed out.

2. MOBILE AGENTS

Mobile agents (White, 1997) (Kotz et al, 1999) are software agents (Wooldridge, 1999) that have the ability to move from one machine to another. Two main approaches have been considered for mobility (Fuggeta et al, 1998). In *weak mobility*, mobility is achieved by transference of code to another network node, possibly accompanied by initialization data, but without any transfer of the agent's dynamic execution state. Most of the JAVA mobile agents, (such as IBM Aglets, MOLE, Concordia, etc.), are implemented this way. In *strong mobility*, when the agent is transferred, its execution is suspended, its code and execution state (both

static data and dynamic state) are moved to the destination node where the agent's execution is resumed. This is the kind of mobility exhibited by such systems as Telescript and Agent Tcl.

Several advantages of mobile agents have been pointed out: i) they may reduce bandwidth requirements (Kotz et al, 1999), since in moving the code to the machine where the data is, they can perform all the computation there and transmit just the result to the original machine; ii) they can act autonomously, even in the case of temporary absence of a network connection to the machine from where they were launched; iii) they provide high scalability to the systems since functionality can be added whenever needed; and iv) they are specially well suited for remote operation, as they conciliate reliability and flexibility (Camarinha-Matos et al, 1998-b) (Camarinha-Matos et al, 1999-a), since, on one hand, in moving to the place where the machines or devices to be operated are, they provide execution supervision locally, and therefore they do not rely on the availability and time-delays of the network, and, on the other hand, they allow an easy way of extending functionality.

Why mobile agents for remote operation?

One popular solution for remote operation assumes low level services available in the remote place, being high level tasks composed of a sequence of calls to these services by means of the traditional remote procedure calling (RPC) mechanism, or even the middleware layers such as CORBA, RMI and JINI, as is illustrated in Figure 1. This solution is very flexible in what concerns functionality, since clients can easily achieve new functionality by composition of the low level commands the remote place recognizes. Unfortunately, it is extremely dependent on the characteristics of the communication links, such as availability and time-delays. That is, for example, the case when some error occurs in the remote place and the communication link crashes being very difficult to define a recovery strategy, since, in general, intelligent recovery strategies depend not only on the individual low level commands, but also on the overall plan to be executed whose information is in the client side.

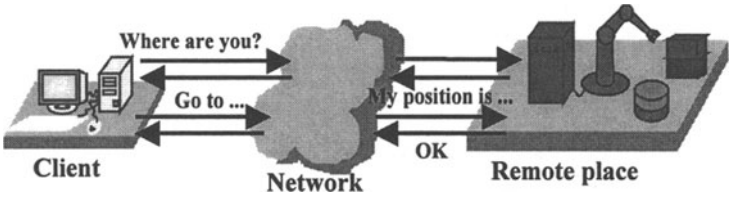

Figure 1 - Remote operation based on low-level commands

The network dependency may be reduced increasing the intelligence of the remote places, by implementing there high level commands which, when activated from the client side, execute almost entirely in the remote place. In this way it is possible to embed effective recovery strategies since the high level execution plan exists in the remote place.

However, this solution is quite limited in what concerns flexibility, since it implies that any change in functionality requires the corresponding update of all remote places where that changed functionality has to be recognized.

Mobile agents allow a solution where flexibility and network independence are conciliated. In this solution, remote places still implement a set of low level services but are augmented with the capability to run mobile agents. On the client side, mobile agents are implemented according to the desired functionality and sent to the remote places where they run independently of the characteristics of the network. Intelligent recovery strategies may be implemented in the agents, since they carry the overall execution plan.

Very often the same desired functionality can be achieved using quite differently equipped execution environments. This means that if an execution plan is described at a sufficiently high level of abstraction it can be adapted for execution in execution environments using quite different resources, as is illustrated in Figure 2. This strategy results in systems with enhanced flexibility, when compared with the traditional RPC solutions and the most primitive use of non-adaptive mobile agents.

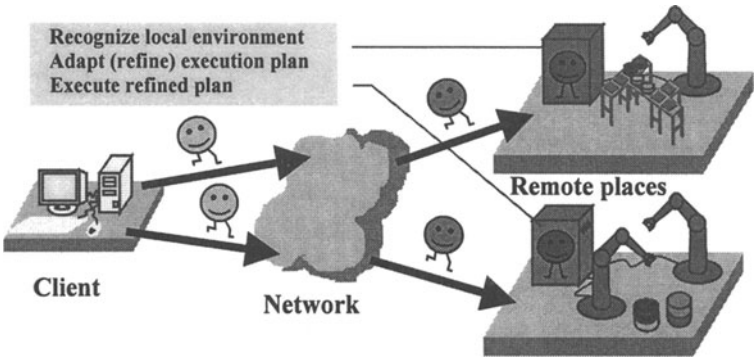

Figure 2 - Remote operation based on adaptive mobile agents

In principle, one could think of complex autonomous mobile agents to manage the adaptation mechanism. These agents would have planning capabilities which allow them to achieve the high level goals they carry with them. However, in many situations, the planning process may be so complex that full automated planning systems would be impracticable due to either performance insufficiency or the associated complexity of the considered domain. For example, in many situations a plan may be better achieved by mixed initiative planners where several special purpose planners are used according to guidelines defined by the users in an interactive way. Moreover, in complex domains, rich execution monitoring strategies may better be achieved with powerful mixed initiative off-line planners than with local planners embedded in the agents.

In the approach taken in this work the mobile agents carry with them high level abstract plans obtained a priori (using a mixed initiative planner, for example). These plans are refined according to the capabilities found in each place visited by the agents, and the resulting refined plan is executed.

The abstract plans may have special annotations (generated automatically or with the user support) intended to guide the local plan refinement process, the local execution monitoring process, or the local error recovery process.

Another significant advantage of the mobile agents paradigm is that the agents control when and where they execute their plans, including the ability to suspend

their execution and move to another place where execution is resumed. For example, if an unrecoverable error occurs, a mobile agent can move to another place where it proceeds the execution of its plan, or it can choose the most appropriate places for execution of the various parts of its plan.

3. THE ARCHITECTURE

In this multi-agent architecture there are several places where mobile agents can reside in one precise moment of their lives (see Figure 3).

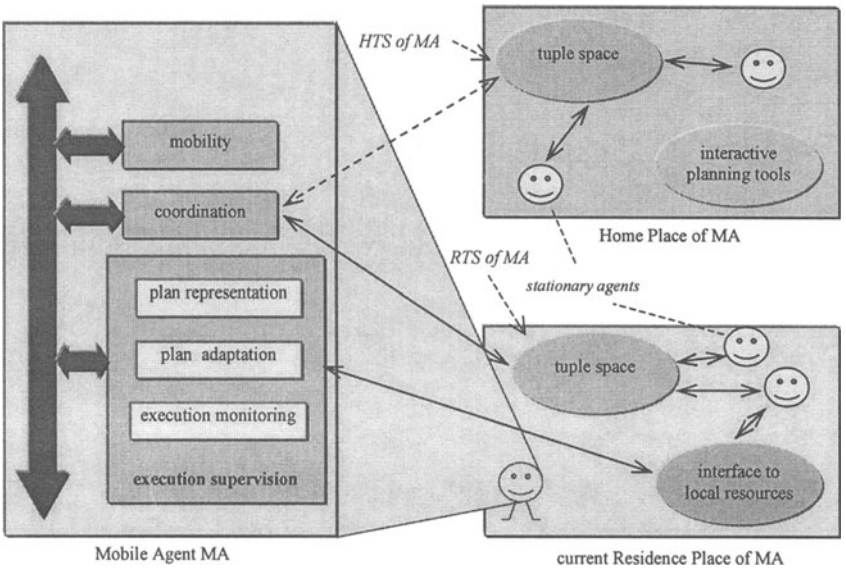

Figure 3 - Main components of a mobile agent

These places may include several stationary agents which implement some specific functionality within that place. Stationary agents are involved only in cooperative work with the agents executing in the same place, or mobile agents having this place as home, while inter-place cooperation is achieved using mobile agents. There are several factors motivating this approach as will be seen later when coordination is discussed.

It is assumed that all the agents have the same structure, being the only difference between mobile and stationary agents the fact that the latter do not use the mobility capabilities. In fact, besides the homogeneity associated with this solution, it would also allow high levels of flexibility (for example, during the installation of the software component in a new place, or when updating some stationary agent, mobile agents could be launched to the place, adapt their behavior to the specific resources they find and, persistently, spend the rest of their lives there, as if they always were stationary agents).

In the proposed architecture, the main components of a mobile agent are: i) the mobility component that implements the process of agent migration; ii) the coordination component that implements the coordination infrastructure that allows the agents to coordinate their activities in order to achieve a well behaved system; and iii) the execution control component that implements the mechanisms that allow one agent to adapt its high level abstract plan to the exact environment it finds in each place it visits, and to execute the adapted plan, including execution monitoring and error recovery.

Mobility

A platform for execution of mobile agents, called IMAJ (originally standing from "Inceptive Mobile Agents in Java", but now meaning "Intelligent Mobile Agents in Java"), was developed in JAVA (Camarinha-Matos et al, 1998-a). Agents use *weak mobility* by JAVA serialization.

JAVA was chosen as the basic development language due to its multi-platform nature, widespread utilization and richness of built-in services it offers, but lacks symbolic processing capabilities that would reduce programming effort when inherently symbolic processing is necessary (e.g. in planning).

In the last few years some developments have been made on light versions of symbolic processing languages/shells to be used in the Internet and which allow some sort of integration with the JAVA language, as are the cases of JESS (Java Expert System Shell), a clone of the popular CLIPS expert system shell, and JINNI (Java INference engine and Networked Interactor) a Prolog-like language.

Both JESS and JINNI were integrated in IMAJ in a way that allows the programmer of the agents to use the most suitable language for each type of processing.

Coordination

In (Cabri, 1998) coordination models for mobile agents are discussed according to the spatial and temporal coupling they imply. *Spatial coupling* implies that the agents involved in a mutual interaction must share a common name space, whereas *temporal coupling* means that communication between the agents is synchronous in the sense that both the sender and the receiver must exist at the time of the communication.

In the direct coordination model (e.g. the client-server model), one agent sends a message to a specified destination agent that must be present at the time the message is sent, thus implying both *spatial* and *temporal coupling*.

In the meeting-oriented model two agents can communicate only if they reside on the same place. One of the agents acts as a server and the other as a client. Although the agents don't have to know each other's names, they both must know the name of the meeting point, and, thus, only partial *spatial de-coupling* is achieved. On the other hand, this scheme imposes strict synchronization between the agents, thus suffering from *temporal coupling*.

One alternative to these coordination models is the blackboard model with associative access, or *tuple space* model (commonly known as the Linda model). In this model, agents communicate through a persistent information space using commonly agreed patterns of information. A tuple of information is put in the

information space regardless of which agent will get it. When an agent wants to retrieve information from the information space, it gives the desired pattern which is used to filter the information currently available in such a way that only tuples that match the given pattern are received. This solution allows both *spatial* and *temporal de-coupling* because, on one hand, no destination agent is named, and, on the other hand, there is no need for the partner involved in an interaction to be present during all its duration (persistence of the exchanged information is provided by the tuple space).

The first two coordination models require some explicit naming of the partners or meeting points (*spatial coupling*) and imply synchronization of the partners (*temporal coupling*), what turns them unsuitable for many mobile agents applications where the adoption of global naming schemes and guaranteeing synchronization between agents may be difficult.

Specifically in the context of adaptive mobile agents used for remote operation as described in section 2, one mobile agent sent to a remote place is involved in interaction with the resident agents it finds in the remote place and with the agents residing in its home place. If the agent would have to know the names of its partners, flexibility would be sacrificed. Alternatively, the agent can ask for services other agents can provide without the need to know their names. Also, the agent may announce its results without the necessity to have the destination agents directly connected to it.

To overtake these difficulties, in this work, a coordination model based on Linda-like tuple spaces is adopted. As can be seen in Figure 3, a mobile agent has access to two tuple spaces. One of the tuple spaces is merged with the tuple space of the remote place where the mobile agent is executing (RTS - residence tuple space), thus allowing the agent to interact with all the agents that co-reside there (both stationary agents and other mobile agents currently executing in the same place). The other tuple space is merged with the tuple space of the agent's home place (HTS - home tuple space) and allows the agent to interact with the agents located at its home. The part of this tuple space in the agent's side is cached in the current place where the agent is executing and automatically synchronized with the home place whenever a reliable network connection is established. If the agent decides to move to another place and the cache has unsynchronized tuples, they are saved in the agent's state, transferred along with the agent and used to restore the HTS cache when the agent reaches the destination place.

As it was previously mentioned, stationary agents in the proposed architecture are only directly involved in intra-place interactions, being inter-place interactions done by means of mobile agents. This means that if interaction is necessary between two places, the place where the interaction is initiated sends a mobile agent to the other place. Since this mobile agent has access to its HTS, cooperation is possible with the two places using the mobile agent as an intermediary. This scheme results in very homogeneous systems where high levels of spatial and temporal de-coupling are achieved.

However, two factors may be raised against this solution: i) this scheme may require a large number of (intermediary) mobile agents, particularly if interaction is needed among mobile agents coming from different home places and executing in different places; ii) sending mobile agents to some place means that some degree of spatial coupling exists, as the address of the place must be known.

Regarding the first factor, it can be said that the proposed solution privileges cooperation among agents coming from the same home (regardless of where they are executing) or coming from different homes but executing in the same place, which are the most common situations in many applications, being this disadvantage highly overtaken by the benefits resulting from homogeneity and high levels of spatial and temporal de-coupling. If remote operation is needed, then the advantages of sending mobile agents are evident as explained in previous sections. Regarding the second factor, one must agree that the problem of some degree of spatial coupling exists. However, it appears only during the agents' launching, being the work, and hence the interactions, of the agent in the destination place completely de-coupled. This low level of spatial coupling is negligible in many applications. Nevertheless it is difficult to imagine a completely spatial and temporal de-coupled application.

Execution supervision

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

Most works on execution supervision have assumed very restricted application domains, such as assembly in flexible manufacturing cells (Meijer, 1991), because this allows the enrichment of the knowledge about the domain and, therefore, eases the definition of execution monitoring and error recovery strategies. In this work more general solutions are searched for a very wide range of applications, implying that the actual composition of the execution environments is not known in advance. Furthermore, as we are dealing with remote operation, the agents must run with a high degree of autonomy in uncertain environments.

Consequently, 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 (Vieira et al, 1999). A hierarchical plan structure was considered, since it allows the specification of monitors (Reece et al, 1994) 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. Also, hierarchical plans contribute to the reduction of the complexity of the plan adaptation mechanisms.

As it will be seen below, monitors of various types may be specified on a per-action basis.

The architecture of the execution control component extensively uses multi-thread programming in order to avoid the problems associated to the sequential execution found in many intelligent agent architectures. 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.

Plan representation

Hierarchical partial order plans (HPO) are considered. In the HPO abstract plan carried by the agent, two types of abstract actions (AA) exist: i) high level abstract actions (HLAA) whose refinements are specified in terms of other abstract actions, and ii) low level abstract actions (LLAA) that must be refined locally in each site visited by the agent. A third type of action is generated by the refinement process which corresponds to executable procedures using local capabilities and are called executable actions (EA). A simplified version of the object model of the actions considered in an agent's plan is depicted in Figure 4 using the Unified Modeling Language (UML).

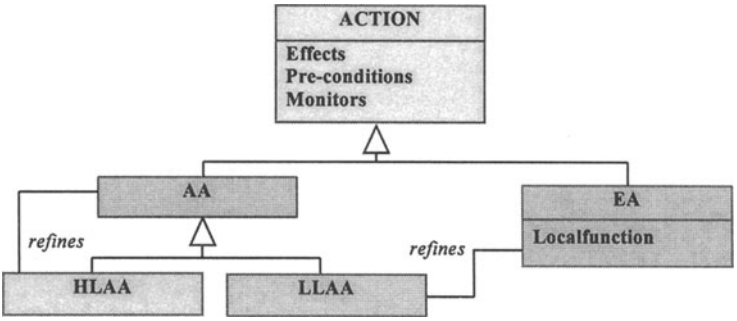

Figure 4- Hierarchy of actions in a plan

An abstract plan specification always begins with one abstract action, whose refinement methods determine the agent's behavior.

For the specification of the abstract hierarchical plans, a plan specification language, called mobile agents abstract plan language (MAAPL) (Camarinha-Matos et al, 1999-b), was defined. For each action a set of annotated information can be added, such as monitoring and recovery information. Figure 5 shows an example of a simple plan specification.

Plan adaptation

Previous work related to plan adaptation in mobile agents was described elsewhere (Camarinha-Matos, 1998-a). Basically, a least commitment strategy of the kind found in partial order causal link planners is followed. Each abstract action is refined according to the capabilities found in the local environment and the obtained refined sub-plan is merged with the current refined plan. The result of the refinement process is a refined plan tree where the full hierarchical plan is represented along with the corresponding annotated information. The leaves of this tree are the primitive executable actions (EAs). Each node maintains information about its monitors, its pre-conditions and its expected effects. The nodes corresponding to the abstract actions that have various choices for refinement also maintain information about the current refinement and the refinements that were not tried yet in order they can be tried during some error recovery procedure.

```
action(a1,                                          action name
       NIL,                                         parameters
       0,                                           priority
       {c1,c2},                                     pre-conditions
       {c4},                                        effects
       CHOICE {({a2,a3},{a2 < a3}), ({a2,a4},{a2 < a4})},   refinement (set of partial orders)
       DEFAULT,                                     effect monitors
       MAINTAIN {(c2,REDO),(c2,REDO)},              maintenance monitors
       TOUT (100, FAIL),                            time-out monitor
       FAIL {(a2, RETRY),(a4, FAIL)}                fail monitor
    )
...
action(a4,                                          action name
       {(first_param, "first parameter"), (second_param,1200)},   parameters (two parameters)
       0,                                           priority
       {c2,c3},                                     pre-conditions
       {c4,c5},                                     effects
       ADAPT (LIBRARY, ACT_1),                      refinement (adapt action type
                                                    ACT_1 using library of plans)
       DEFAULT,                                     effect monitors
       NIL,                                         maintenance monitors
       TOUT (20, RETRY)                             time-out monitor
)
```

Figure 5 - Example of a plan specification in MAAPL

Execution monitoring

Contrary to the works described in (Veloso et al, 1998) and (Pollack et al, 1999) where monitors are used in the context of dynamic planning, in this paper monitors are used mainly for execution control.

As it was said before, each action specified in the abstract plan may, optionally, include a set of monitors of the following types:

- *Effect monitors* specify monitors that verify the achievement of the effects of actions upon their completion. If *default* is indicated, the default recovery methods are used. Otherwise, a set of monitors may be specified, being each one a pair composed of a condition on the effects of the action and a recovery method which indicates one of the available recovery strategies: i) RETRY for retrying the action with the current refined plan, ii) REDO for trying an alternative refinement, and iii) FAIL for gracefully failing the action, and propagate error recovery to the upper level.
- *Maintenance monitors* which are pairs composed of a world state condition that must be maintained during the execution of the action, and a recovery method as specified for the effect monitors. The maintenance condition must be one of the pre-conditions of the action where the monitor is defined.
- A *time-out monitor* may be specified which indicates the maximum allowable duration for the action's execution, and a recovery method as specified in the effect and maintenance monitors.
- A *fail monitor* specifies which recovery method must be used when the named child action fails after trying all its applicable recovery methods.

Maintenance monitors are propagated from the actions at higher levels to their descendents at the lower levels. The monitors of one action at a higher level are

installed when its first primitive descendent starts execution, and uninstalled when its last primitive descendent terminates execution, as can be observed in Figure 6.

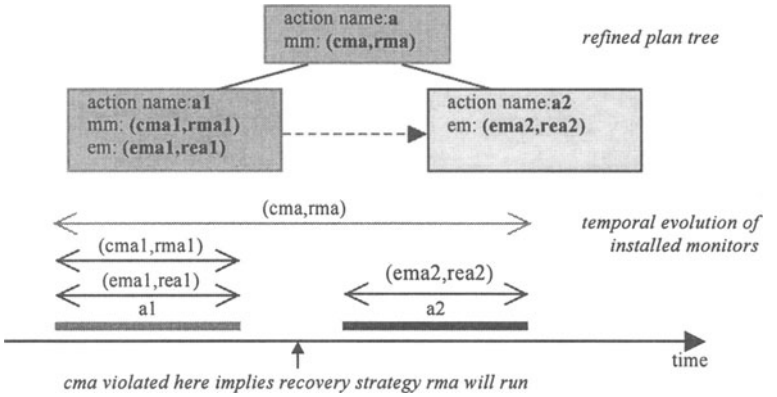

Figure 6 - Example of monitors' installation

When errors are detected, the recovery procedure is activated with the information related to the action involved, the classification of the occurred error and the identification of the related monitor.

One important aspect is that there are implicit monitors associated with the pre-conditions of the actions. When some actions that have all its precedent actions already executed have pre-conditions not satisfied by the current world state, a recovery strategy is fired which will try to find a patch plan for achieving these not satisfied pre-conditions from the current world state. This recovery strategy must preserve all causal relations associated with the part of the plan not yet executed and the pre-conditions of all the actions currently being executed.

For more details on execution supervision, please refer to (Vieira et al, 1999).

4. AN APPLICATION SCENARIO

The concept of adaptive mobile agents applied to remote operation as described in the previous sections has been tested in a prototype system to support old adults living alone at their homes. This is a multi-agent system structured around a set of places interconnected through the Internet.

At the heart of the system there are the home places (HP) which are located at the homes where elderly people (the users) live. HPs include a set of sensors (temperature, humidity, etc.), measurement devices (blood pressure, body temperature, etc.), and a set of robotized home appliances. This equipment may be operated both locally, by local monitoring agents, and remotely, from social care centers (care center places - CCP), health care centers (health center places - HCP), or from relative's jobs (relative's places - RP). The system may also include leisure places (LP) which the user may visit to participate in some social activity, and commerce places which give support to some electronic supported trade (CP).

Figure 7 depicts the simplified structure of an HCP, a CCP and a HP. As can be observed, in a HP there are several stationary agents: the monitor agent for monitoring the evolution of a set of sensors and to observe the behavior of the user; the agenda agent for guaranteeing the fulfillment of some pre-defined agenda (e. g. reminding the user that it is time to take some medicine), etc.

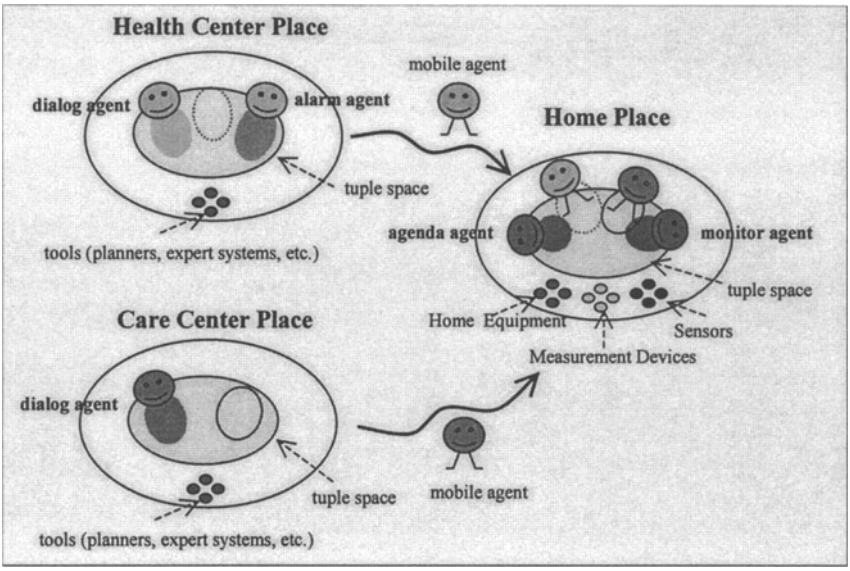

Figure 7 - Example of the use of mobile agents in elderly care

Mobile agents can be used for several purposes in this application scenario. For example, mobile agents can be sent from HCPs, CCPs and RPs to a HP to give some help/advice to the user or to observe his/her health state and behavior, or mobile agents can be sent from a HP to a CP to buy some goods. As an example, a mobile agent could be sent from a CCP to a HP to help the user in preparing some meal. This agent would carry an abstract plan considering abstract actions such as set the table, "warm-up the meal", "serve the meal", etc. When in the destination HP, the agent would adapt its abstract plan to the local capabilities of the HP. For example, would the HP have a robotized microwave and the agent could perform the action "warm-up the meal" almost entirely automatically. Otherwise, if only a conventional gas oven were present the agent would have to resort to a set of help messages sent to the user using some local capability (voice synthesizer, color lights, bell, TV screen, etc.).

Dialog agents are created in HCPs, CCPs and RPs for each mobile agent sent to a HP whose work requires a close interaction with the sender (e. g. a dialog agent that interacts with a relative and uses a graphical interface to show images of the old adult).

It should be mentioned that the emergence of the so-called ubiquitous computing is allowing the introduction of computational power and embedded intelligence in most of the environment of our lives. That is, for instance, the case of our homes for which several manufactures start to offer appliances (washing machines, microwave

ovens, refrigerators, etc.), TV sets, climate-control systems, surveillance cameras and alarm systems, lighting, etc, with embedded microcomputers, sensors, actuators, and the capability to be interlinked in a local network or even connected to the Internet. On the other hand, mobile devices such as mobile phones or PDAs, and the automobiles are offering increasing intelligent functionalities and connection capabilities. The integration of sensors and local processing capabilities in clothes, namely for sports, is also a trend that might be generalized to the garment industry. New protocols and support technologies such as the WAP, JINI, Windows CE and CE bus contribute to this trend (Dutta-Roy, 1999).

5. CONCLUSIONS AND FURTHER WORK

Conciliating flexibility and network independence is a very important factor in remote operation and supervision. Some disadvantages of the traditional RPC or CORBA based approaches were discussed. The concept of adaptive mobile agents was presented as a mean to cope with these disadvantages. Contrary to several other systems described in the literature (Bonasso,1997), (Miller et al, 1996), (Pell et al, 1999), (Haigh et al, 1998), these agents don't have an a-priori deep knowledge of their execution environments, which turn the adoption of very specific execution monitoring strategies very difficult. A solution based on mobile agents that carry off-line generated hierarchical abstract plans annotated with execution monitoring and error recovery information at the various levels of the hierarchy was presented. These carried plans are adapted by the agents to the actual execution environments they find in each place they visit. Since the high level plans may be generated off-line by mixed initiative planners, very effective monitoring strategies may be incorporated. This approach differs from the common use of mobile agents. In (Ohsuga et al, 1997) planning is used in mobile agents, but not as a mechanism to adapt one abstract execution plan to differently equipped places as used in this work.

Coordination mechanisms where mobile agents are involved are also a very important aspect to consider. It was claimed that the spatial and temporal de-coupling provided by Linda-like coordination models makes them good candidates for the considered target application.

Following these principles, an architecture for supporting mobile agents based remote operation was presented, including the structure of the mobile agents, comprising components to support mobility, coordination, and execution supervision.

Some open issues require further research, such as the use of local diagnosis capabilities in order to improve the error recovery strategies implemented in the mobile agents, and extending MAAPL in order to include more standard services, to allow temporal plans, and to allow the inclusion of more monitoring strategies. The integration of emerging protocols is also an important element in terms of interoperability.

Acknowledgements

This work was funded in part by the Portuguese Ministry of Science and Technology through the PRAXIS XXI project Telecare (PRAXIS/C/EEI/12089/98).

6. REFERENCES

1. Bonasso, R., Firby, R. Gat, E. Kortenkamp, D., Miller, Slack, M: Experiences with an Architecture for Intelligent Reactive Agents, Journal of Experimental and Theoretical Artificial Intelligence, Vol. 9, No. 1, 1997.
2. Cabri, G. Leonardi, L. Zambonelli, F.: How to Coordinate Internet Applications based on Mobile Agents, in: Proc. of the 7th IEEE workshops on enabling technologies: infrastructures for collaborative enterprises (WETICE 98) - workshop on coordination architectures for distributed web applications.
3. Camarinha-Matos, L. M. Vieira, W.: Using multiagent systems and the Internet in care services for the ageing society, in: L. M. Camarinha-Matos, H. Afsarmanesh and V. Marik (Ed.), Intelligent Systems for Manufacturing: multi-agent systems and virtual enterprises, Kluwer Academic Publishers, 1998: 33-47.
4. Camarinha-Matos, L. M. Vieira, W.: Adaptive Mobile Agents for Telerobotics and Telesupervision, in: Proc. of the IEEE International Conference on Intelligent Engineering Systems, INES'98, 1998: 79-84.
5. Camarinha-Matos, L. M. Vieira, W.: Intelligent mobile agents in elderly care, Robotics and Autonomous Systems, 1999, Vol. 27 No 1-2: 59-75.
6. Camarinha-Matos, L. M. Vieira, W.: MAAPL: A language for adaptive mobile agents with execution monitoring, Proceedings of INES'99, 3rd IEEE International Conference on Intelligent Engineering Systems (1999) 171-176, ISBN 80-88964-25-3, Poprad, High Tatras, Stará Lesná, Slovakia, Nov. 1-3, 1999.
7. Dutta-Roy, A.: Networks for Homes, IEEE Spectrum, Vol. 35, No. 12, December 1999
8. Fuggeta, A., Picco, G. P., Vigna, G.; Understanding Code Mobility, IEEE Transactions on Software Engineering, 1998, Vol. 24, No. 5: 346-361.
9. Haigh, K., Veloso, M.: Interleaving Planning and Robot Execution for Asynchronous User Requests, Autonomous Robots, 1998, Vol. 5, No.1: 79-95.
10. Kotz, D. Gray, R. S.; Mobile Agents and the Future of the Internet, ACM Operating Systems Review, 1999, Vol. 33, No. 3: 7-13.
11. Müller, J. P.: The design of Intelligent Agents: a layered approach, Springer-Verlag, 1996.
12. Ohsuga, A. Nagai, Y. Irie, Y. Hattori, M. Honiden, S.: Plangent: An Approach to Making Mobile Agents Intelligent, IEEE Internet Computing, 1997, Vol. 1, No 4: 50-57
13. Pell, B., Dorais, G. Plaunt, C. Washington, R: The Remote Agent Executive: Capabilities to Support Integrated Robotic Agents, submitted to Autonomous Robotics Journal , 1999.
14. Pollack, M., McCarthy, C.: Towards Focused Plan Monitoring: A Technique and an Application to Mobile Robots, IEEE International Symposium on Computational Intelligence in Robotics and Automation, 1999.
15. Reece, G., Tate, A.: Synthesizing Protection Monitors from Causal Structure, Proc. of the 2nd International Conference on Planning Systems, AAAI Press, 1994.
16. Vieira, W. Camarinha-Matos, L. M.: Execution Monitoring in Adaptive Mobile Agents, in: M. Klusch, O. Shehory, G. Weiss (Eds.), Cooperative Information Agents III, LNAI, Springer-Verlag, Berlin et al, 1999, , Vol. 1652: 220-231.
17. Veloso, M., Pollack, M., Cox, M.: Rationale-Based Monitoring for Planning in Dynamic Environments, 4th International Conference on AI , 1998.
18. White, J. E.: Mobile Agents, in: Bradshaw, J. M. (eds): Software Agents, MIT Press (1997) 437-472
19. Wooldridge, M.: Intelligent Agents, in: Weiss, G. (Ed): Multiagent Systems, The MIT Press, 1999.