Design A Hardware Support For An Architecture Based On Multi-Agent Systems For M2M Services

Abdellah KRIOUILE, Hicham MEDROMI

Abstract— The exchange of computer data, visual, graphic or audio for remote treatment becomes very complex when it comes to large structures. The Machine to Machine - M2M - allows exchanges between equipment (servers, cameras ...) and information systems automatically through networks. There are innumerable application areas (security and monitoring, traffic management ...). The idea of our work is to design and implement an architecture based on multi-agent systems in compliance with the criteria cited for M2M services. This architecture should be scalable and applicable to all existing structures M2M. The purpose will be to develop a solution based on multi-agent systems to apply to all M2M services respecting three parts, information systems, networks, and the remote equipment. We will also apply optimization methods based on operational research in this architecture. The proposed architecture will be developed in Open source and ensure, the reliability as the core software used is reliable (Eclipse, Apache ...), viability as it is standards-based computing (J2EE ...); optimization because it uses technical based on operational research; interoperability because of the compatibility with different devices and finally portability to operate on all operating systems.

II. MULTI-AGENT APPROACH

A. Agent

An agent is an autonomous and intelligent, real or abstract entity, who is capable of acting on itself and on her environment in a multi-universe agent also, she can communicate with the other agents and whose behavior is the consequence of her observations, her knowledge and the interactions with the other agents [2].

B. Multi-Agent System

A multi-system agent is a compound distributed system of a set of agents, who feign to a certain extent the capacities of the human reasoning, the SMA is ideally conceived and implemented as a set of agents interacting between each other (Fig. 1 [3]), mostly, according to modes of cooperation, competition or coexistence.

A system multi-agent can be:
- Opened: the agents enter it and go out of it freely. (An example of that is in e-commerce).
- Closed: group agents stay the same. (A football match is an example).
- Homogeneous: all the agents are built on the same model. (A working meeting, a colony of ants).
- Heterogeneous: agents of different models, granularities different. (An ecosystem).
- Mixed (or not): the agents “human beings” are integral part of the system. It is opened and heterogeneous. (A Medie workgroup by agents assistants) [4].

Fig. 1: Principle of a Multi agents System
III. CHARACTERISTICS OF M2M SERVICES

The main elements of a M2M system include:

- A device or group of devices capable of responding to requests for data contained in these devices or capable of transmitting data contained in these autonomous devices.
- A communication link to connect the device, or a group of devices to a computer server or to other device.
- Software, process, or interface through which can analyzed, reported and/or implemented by data.

IV. PRESENTATION OF CASE STUDY

To achieve this solution, we started to work on a specific case. Others from our laboratory had worked on robots dedicated to research. The idea is to conceive design a solution that will bring all the robots together (in the same application). The adaptation to other equipment will be very practicable.

A. Expression of needs

The robots (NXT, Khepera III Hemisson) are used by a telecom company, the experts of the company cannot access to some places which are difficult to achieve for maintenance or control. In these cases robots make the work. The expert has to communicate with the robot. It can even control multiple remote locations simultaneously. His supervisor may intervene or control the actions of maintenance. To better illustrate these requirements, the diagram of use case shows the different actors of this system and also shows the great deeds that players can realize.

B. Diagram of use case

The diagram below outlines the requirements of our application:

C. The actors on the system

The actors who use the system are:

- Manager (supervisor): Responsible of the team must have access to databases, monitor connections, plan the tasks complicated...
- Expert: An expert will connect to power or control of robots or view data.
- Robot: (NXT, Khepera III Hemisson) the robots must perform tasks or detect events.

V. GENERAL PROPOSED ARCHITECTURE

The architecture specifies the structure of a system. We speak of functional architecture to define system services, technical architecture for the technical components used and application architecture to describe the division into sub-systems [7]. In our application, the application must be usable by the team supervisor, the expert and the robot too. The action of these actors run from graphics interfaces that access a server which handles incoming data and then stores them in a database. The application will be divided into several layers.

A. The 3-tier architecture

The 3-tier architecture or architecture in three layers generalized the multilayer architectures or multi-levels. We can therefore say that our application is in three layers:
Note:
- The use of layers costs a lot (communication inter-layer = 3 to 6 times more than a 'normal' method call)
- But it greatly facilitates the scalability and maintenance.

Presentation Layer
For our work, the expert, going on the internet to use the application found interface components which assist in data entry, are not changing and adjust the data for proper display...

Business logic layer
This layer serves to manage the process involving several steps and long transactions (Optimization). It is also served to use services, access layer 'data' and handle exceptions and error.

Layer Data Access
The access layer data attend to create, read, update and delete data. These data are related with technicians (history, password ...) and robots (history, location, integrated features ...).

V. Proposed Application Architecture Based on SMA
To solve any major problem, it is better to divide it into sub-problems. But this approach may not always work because not all problems are not always separable. Alternatively, components of the main problem must be able to argue the knowledge and abilities of the other aims of effective cooperation [5]. For this, these components must be able to perception and action on their environment and should have an autonomous behavior so we talk about agents and therefore about multi-agent systems [10].

A. Concept of Agent
Agents are entities created for specific environments. The characteristics of these environments influence the design of these agents. If the environment is not stable, the agents must react quickly to events, and then a reactive architecture is appropriated. However, if the environment requires that the agent reasons for reaching his goal, a deliberative architecture is more appropriated. Some environments require that in some situations agents have reactive behavior and deliberative behavior in other circumstances. Hybrid architecture is appropriate for this kind of environment it is a mixture of the first two architectures. The agents apply predefined rules to choose their actions according to the applied architecture used.

Definition:
An agent [11] is an entity, actual or virtual, acting in an environment, able to perceive, act and interact with the surrounding components. An entity is an agent if it is able to exercise local control over its process of perception, communication, learning, acquirement, and reasoning, make decision and implementation. The agents have two trends: Social trend: We speak about knowledge of an agent in relation to other elements in society. This last is a representation of the mechanisms needed to manage interactions and the organization or structuring agents in the system.

Trend individual:
We speak about mechanisms of knowledge containing internal rules to the agent.

We can add the group trend when we need an aggregation of agents into subsets.

All this is to organize the problem by reducing its complexity and clarifying the division of tasks between agents in establishing the relationship between them.

B. Architecture
In our architecture, each agent has a role, every two agents can be connected, and each agent belongs to a group or several.

Role:
The role is often defined as a service class that offers an agent in the system [12]. In our case we have the expert roles, accountable, NXT robot, robot Khepera III robot Hemisson...

Link:
The links are structured exchanges between agents using the specified relationships in the roles of these agents. In our work we need to determine some types of links. Communication links for exchanging data between expert and supervisor for example. Authority links to express, for example, the subordination of the supervisor.

Group:
The group is a subset of agents. In our work we can say that each group depends on the expert's mission and roles involved to accomplish.

Fig. 5: Structure of the Architecture
The structure of our architecture is like this:
VI. PRINCIPLE OF OPERATION

We propose a new architecture model for M2M services, based on a distributed approach and the multi-agent systems. For this use case, an expert connects via the internet with a login and password, to order or combine tasks to robots. When the expert try to connect to a robot, an interface agent, with the role of expert, sends a request to the interface agent, with a supervisor role. If the connection is made, the expert responsible interface transmits information to the agent interface robot. The agent robot integrates the expert in a group and contacts the software agent responsible for the database. The agent of the robot executes the robot mission. But if the mission requires cooperation from another robot, the agent's first robot communicates to the second necessary tasks. Except when the distribution of tasks is complex (multiple robot and many tasks), the solution is to hand the agent responsible for organizing and planning. Because the agent supervisor integrates operations research methods to optimize time and manage resources.

VII. THE HARDWARE ARCHITECTURE

The Hardware support that will integrate our architecture is as follows:

Servers trades:
- Contains application servers (eg JBOSS J2EE services). A trade server contains in this case a container of agents and shared services.

JDBC:
- It is an interface or an API that allows access to relational databases.

RMI:
- It is an interface that allows to agents or remote object to communicate as even if they aren’t in the same places.

Presentation Servers:
- Contains a web container or container Servelet. This is software that performs Servelet (eg Tomcat).

Web Servers:
- This is to accommodate the Internet and runs the HTTP server (Apache for example) and server database (MySQL for example).

The expert connects through a computer or a mobile to internet to use distant robots. These robots provide data to the system.

VIII. USE OF THE OPERATIONAL RESEARCH

Robots have for objective, sometimes, to calculate the shortest ways to affect purposes. For it they can use algorithms and methods of operational research. But it does not make an objective among the objectives of this work. But the point of the organization of the depending tasks some of the others is very essential for this work, especially when it is about tasks distributed on several robots. The idea is to integrate the method of the potential for this problem of organization. The problems of organization are a direct application of the methods of research for the optimal ways in a graph [13]. The robot can have a mission where, with the aim of the realization of any objective, it is necessary to achieve a set of tasks or operations; them were subjected in a set of constraints and sometimes distributed on the other robots also. When the robot is in this situation, it sends a request with the list of the tasks and the constraints so that the responsible agent plans and organizes spots in an optimal way. We distinguish: The constraints of the potential type, which are the constraints of temporal localization (the task i does not have to begin before such date, or, on the contrary, must be finished in such date), or the constraints of succession (the task j cannot begin before the end of the task i, or before the task i was half finished); The constraints of the disjunctif, impressive type disjunction of two relative intervals of time, for example, in the execution of two tasks i and j, which cannot be simultaneously realized; A mission is decomposable into tasks, these tasks being subjected to diverse constraints, it is important to determine a schedule of execution of the tasks, compatible with the constraints. To find such a schedule, it is to obtain a solution of the problem of organization. However, among the diverse solutions, it is the best and less good with regard to a given criterion there. In most of the cases, we try to reach the objective as soon as
possible. The method of the potential (MPM) bases on a graph the plan of which is immediate; summits represent the operations and the arcs of the constraints; these arcs are values by a number indicating the minimal duration that must pass by between the beginning of the task initial extremity and that of the final task-extremity. In practice, often this graph is not drawn but not represented by successors’ list. The list while expressing the order of the spots which imports us.

Example:

Let us admit that a robot received the order to execute the following spots:

<table>
<thead>
<tr>
<th>Teaches</th>
<th>Durée de la tâche (minutes)</th>
<th>Tâches antérieures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task a</td>
<td>2 min</td>
<td>Nothingness (first task)</td>
</tr>
<tr>
<td>Task b</td>
<td>1 min</td>
<td>Task a (the same execute)</td>
</tr>
<tr>
<td>Task c</td>
<td>3 min</td>
<td>Task b</td>
</tr>
<tr>
<td>Task d</td>
<td>2 min</td>
<td>Task b and Task c</td>
</tr>
<tr>
<td>Task e</td>
<td>5 min</td>
<td>Task a</td>
</tr>
<tr>
<td>Task f</td>
<td>1 min</td>
<td>Task e et Task d</td>
</tr>
</tbody>
</table>

Table I: Executed spots

If robots execute these tasks one after the other as indicated in the list, the mission will be achieved in 14 minutes. Let us apply now the (implemented) method MPM one find this graph:

![Graph](image)

Fig. 8: Orderly Spots

By applying the method we re-order the tasks;

Table II: Orderly List

<table>
<thead>
<tr>
<th>Orderly list</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task a</td>
</tr>
<tr>
<td>Task b et Task e</td>
</tr>
<tr>
<td>Task c</td>
</tr>
<tr>
<td>Task d</td>
</tr>
<tr>
<td>Task f</td>
</tr>
</tbody>
</table>

By respecting this order the mission comes true in 9 minutes instead of 14 minutes. We thus see that this method is going to add some value to this architecture.

IX. CONCLUSION

In this article we have presented an architecture based on multi-agent systems for M2M services. We have detailed this proposal by applying it to a use case. We have developed a structure based on three areas: the role or function of each element of an M2M system, the links that might be between these elements and finally the group or subset according to function. We find these three lines in M2M services. Basing ourselves on it, we have designed a multi-agent architecture. We have also shown how operational research could be useful for significant optimizations. A software solution will be developed and deployed on UNIX and Windows. The solution will require significant resources. It contains three main layers. We will need to develop different servers. But once developed we will need just to reconfigure the interfaces will for new use cases. We will use Java language to develop the software. Given the complexity of the architecture we should rework this concept during development. Human discovered M2M to help mankind. Our research falls within this context.

REFERENCES

AUTHOR'S PROFILE

Abdellah KRIOUILE Engineering degree of French State IT option industrial engineers' national School of Brest ENIB on 2009 In 2010s he joined the System Architecture Team of the ENSEM School, Casablanca, Morocco. His current main research interests embedded systems Based on Multi-agents Systems.

Hicham Medromi received his PhD in engineering science from the Sophia Antipolis University in 1996, Nice, France. He is responsible of the system architecture team of the ENSEM Hassan II University, Casablanca, Morocco. His current main research interests concern Control Architecture of Mobile Systems Based on Multi Agents Systems. Since 2003 he is a full professor for automatic productive and computer sciences at the ENSEM School, Hassan II University, Casablanca.