4th Mediterranean Conference on Embedded Computing MECO – 2015 Budva, Montenegro

Works in Progress in Embedded Computing

Wireless Communication Architecture of a Robotic Services System in an Unknown Environment

Eva Cipi,
Department of computer sciences
University of Vlora “Imail Qemali”
Vlore, Albania
eva.cipi@univlora.edu.al

Betim Cico
Department of informatics engineering
Polytechnic University of Tirana
Tirana, Albania
betim.cico@gmail.com

Abstract—**This paper is focused on presenting the architecture and the implementation of a semi autonomous simulation based system which is able to navigate into a partially known environment. Most of robotic agent does not support the mobility according the requirements of application. Our work of implementation provides a semi autonomous system which is guided by an operator performing several services with better quality and low costs. The design is a module based architecture which supports the robot navigation using simulation offline software and on line at the moment when the agency percepts obstacles. The robot changes states of its mobility in real time building a new strategy to achieve the normal path which is received from a simulator that executes and communicates the path calculated by a simple navigational algorithm in the virtual static environment.**

Using a communication protocol between robotic unit and simulator software, it is possible to correct data and to improve the system behavior using a wireless communication. The physical system is tested in a laboratory environment. It is situated in a environment which is the same designed in the simulator interface. The robot updates its coordinates in the virtual environment and the simulation runs exactly according the navigation algorithms until the sensor module transmits to simulation software new data of obstacle presences. We evaluate the performance of the system and the results confirm an improved behavior of robotic agent in extreme situations of dynamic environment.

Keywords—semi-autonomous system, simulation offline, simulation on line, robotic agent, module based architecture, communication protocol

I. Introduction

Autonomous applications are those in which the user is interested in the results of self processing large amounts of data in several distributed locations in order to achieve some goals. These applications are extremely useful in areas such as intrusion detection systems, self service system or unknown environment map analysis.

Robotic agent systems appear to be the most feasible solution for their implementation. In these systems, autonomous software entities (robots) may move across an environment of execution platforms. The application is supported by an architecture which seems to be based on two modalities of agency work.

In this paper aims to study a semi-autonomous system that operates into a partially known environment which is able to bring services of transportation using robotic agents to every point addresses generated from a central management system. The robots move autonomously into a physical environment area controlled by an operator but the robot path has been fixed over a virtual partially known environment. A wireless communication system supports the connection between the central system and robotic entities. There are two channel transmissions in separated frequencies. The system passes from one modality (simulation off-line) in another one (simulation online) when the robots percept the presence of unknown obstacles.

The work aims to contribute on bringing new solutions on solving the problem of the navigation in partially unknown environments in order to avoid collisions with obstacles. Another element is the designing process tents to be easier, programming intelligent behaviors in virtual environments. The robotic entity is equipped by a set of sensors to sense obstacles. We have tested several behaviors of the system in different environments and the performance of the system was very good. In the Fig. 1 you can see a robotic prototype used to be programmed as one of subsystems.

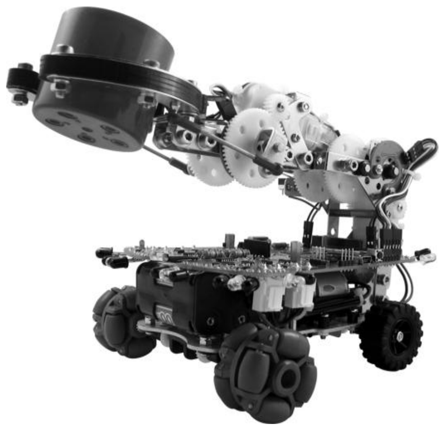
Fig. 1. The prototype used in this work

II. RESEARCH OBJECTIVES

In this research we propose a modulated architecture agent based approach to develop complex applications. The approach aims to increase the performance of systems on managing autonomously the dynamism and changes of the application environment. The general idea of self-management is to endow computing systems with the ability to manage themselves according to high-level objectives specified by humans. Researchers divide self-management into four functional areas [3]:

- Self-configuration: systems automatically configure components to adapt themselves to different environments;
- Self-healing: systems automatically discover, diagnose, and correct faults;
- Self-optimization: system automatically monitor and adapt resources to ensure optimal functioning regarding the defined requirements; and
- Self-protection: they identify and protect against attacks.

The environment will occupy an important role in this project. An agent based system which pretends to be self managed must take the appropriate actions based on a sensed situation of the environment. This requires some functionalities of system such as environment monitoring, decision making, and action execution. This requires the coordination of behavior of agents inside of the same environment. Another positive side is that same approach can be used in different applications. Here we show two modalities of the same system. This helps designers and developers to resolve effectively problems of software engineering processes.

III. MULTIAGENT SYSTEMS AND THE SOFTWARE ARCHITECTURE

A multi agent system provides the software to solve a problem by structuring the system into a number of interacting autonomous entities embedded in an environment in order to achieve the functional and quality requirements of the system. In particular, a multi agent system structures the system as a number of interacting elements in order to achieve the requirements of the system. This is exactly what software architecture is about. [4] defines software architecture as:

Software elements (or in general architectural elements) that provide the functionality of the system, while the required quality attributes (performance, usability, modifiability, etc.) are primarily achieved through the structures of the software architecture.

Typical architectural elements of multi agent system software architecture are agents, environment, resources, services, etc. The relationships between the elements are very diverse. In short, multi agent systems are a rich family of architectural approaches with specific characteristics, useful for a diversity of challenging application domains. [5] There are applications that provide different levels of complexity and various forms of dynamism and change. The architecture for many of them is a set of agents, for reuse, they can serve to develop software architectures transporting them as ready entities with specific attributes such as: each agent has incomplete information or capabilities to solve the problem and, thus, has a limited viewpoint and the computation is an asynchronous process.

A multi agent system consists of a (distributed) environment populated with a set of agents that cooperate to solve a complex problem in a decentralized way. [6] Behavior-based action selection is driven by stimuli perceived in the environment as well as internal stimuli. Situated agents employ internal state for decision making relates to:

- Planning off line (static information of the system).
- Planning on line (dynamic information related to the changes of the environment); or issues internal to the agent.
- Environment encapsulates resources and enables agents to access the resources. [7]

IV. A ROBOTIC SYSTEM: CASE STUDY

The system we propose, is a prototype which has to transports loads from one point to another performing intelligent behaviors being oriented themselves of service points in a partially known market place, completing successfully its tasks, avoiding the collision with obstacles displayed dynamically and creating unexpected situations in its path, which is been pre planed by a simulator software. The tasks are generated by the information agent which is part of a central information system, typically for a business management program discussed in our past works. The task is composed out of some processes like receiving order from service agent acquiring the first service point address, moving to first service point, receiving the second service point address, moving to the second service point. In order to execute this task, the system has to perform:

- Route assignment: paths are generated by the information systems and have to be assigned to robot vehicle that can execute them.
- Routing: the robot must route efficiently through the environment layout of the warehouse when executing transports.
- Gathering traffic information: although the layout of the system is static, the best route for the robot in general is dynamic, and depends on the actual traffic condition. Using visual sensors, the robot routes efficiently without collision with other objects. [8]

This model integrates the environment and agent integrating mechanisms of agent adaption. We have divided the model in two parts: environment and situated agent. The environment model consists of a set of modules with flows between the modules. The modules represent the core functionalities of the environment. The model consists of two main modules:

- the deployment context (referred to the given hardware and software and external resources with which the multi agent system interacts (sensors and actuators, a printer, a network, a database, a web service, etc.).and

- the application environment(refereed to the part of the environment that has to be designed for an application) .

The data flow diagram is showed in the Fig.2.

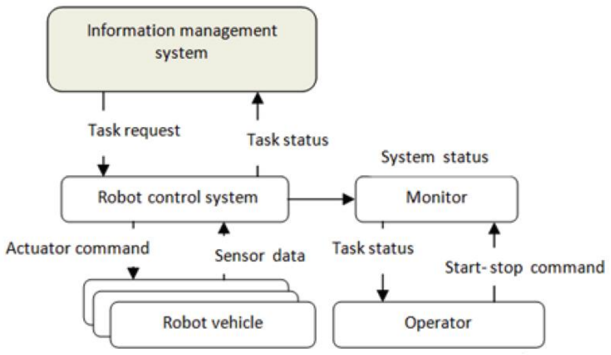

Figure. 2. The diagram of the robotic system for semi-automatic services.

The robotic agent model consists of four modules with flows between the modules. Knowledge module provides the functionality to access and update the agent's current knowledge. Sensing module receives perception requests from the Decision Making and Communication modules to update the agent's knowledge about the environment. Decision making and communication module use the agent's current knowledge to make appropriate decisions. The communication module writes the location in the agent's current knowledge which in turn will be used by the decision making module to move the agent efficiently towards the destination. Decision Making provides the functionality to an agent for selecting and invoking influences in the environment. Decision making consists of two basic functions: Influence selection and actuator execution. To select appropriate influences, an agent uses a behavior based action selection mechanism extended with roles and situated commitments. Execution provides the functionality to invoke selected influences in the environment.

A. The Wireless Communication

The Fig. 3 is a view of environment communication of robotic vehicles with the simulation software. We consider robotic agent as independent entities that communicate with a central management system which generates transportation tasks through a generated path from a simulated known environment but not updated. The control of task distribution is centralized but the vehicle path is not completed autonomous. We consider the presence of collision point but this is provided by robotic agent which changes its status in simulation on line. The central system sends instructions according the simulation software which pretends to know the environment in its first statement. All the exchanged messages are done through a wireless module of communication.

Here is been presented some technical features of the Wireless device which is composed by two modules:

- Receivers RX-B1 (433 MHz and 315 MHz) and
- Transmitters TX-C1 (433 MHz and 315 MHz)

Each message is transmitted as:

- 36 bit training preamble consisting of 0-1 bit pairs
- 12 bit start symbol 0xb38
- 1 byte of message length byte count (4 to 30), count includes byte count and FCS bytes
- 2 bytes FCS, sent low byte-hi byte

Everything after the start symbol is encoded 4 to 6 bits, Therefore a byte in the message is encoded as 2x6 bit symbols, sent hi nibble, low nibble. Each symbol is sent LSB it first. The Arduino clock rate is 16MHz => 62.5ns/cycle. For an RF bit rate of 2000 bps, need 500microsec bit period. The ramp requires 8 samples per bit period, so need 62.5microsec per sample => interrupt tick is 62.5microsec. The maximum message length consists of (6 + 1 + VW_MAX_MESSAGE_LEN) * 6 = 222 bits = 0.11 seconds (at 2000 bps). Throughout the range there are nulls and strong points due to multipath reflection. Similar performance figures were found for DR3100. 9000bps worked. Arduino and TX-C1 transmitter draws 27mA at 9V. Arduino and RX-B1 receiver draws 31mA at 9V.

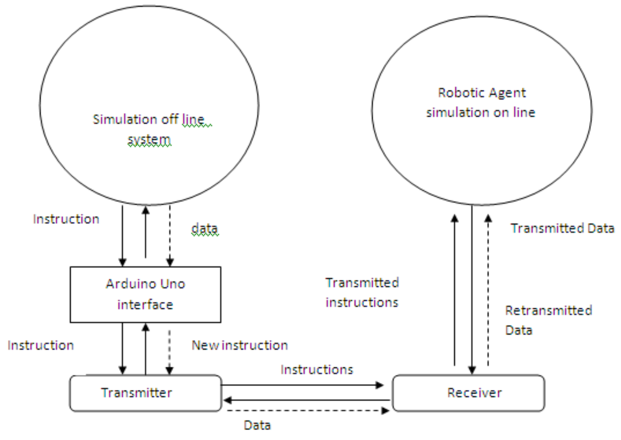

Fig. 3. Receiver and transmitter modules of the wireless communication

Here we present two protocols which allow the system to send commands to robots and to receive data from its. There are two simple protocols which are used to link the simulation software with the physical entity. These messages are transmitted in different frequencies to avoid signal interferences and message errors:

- **Receiving data protocol**

```
int GetMessage()
uint8_t buflen = VW_MAX_MESSAGE_LEN;
   if (vw_get_message(buf, &buflen))
   {
        int i;
     char inStr[25];
     char inChar=-1;
        for (i = 0; i < buflen; i++)
        {
       inChar = buf[i];
       inStr[i] = inChar;
        }
     inStr[i]='\0';
   }
   return  command;
}
```

- **Sending commands protocol**

```
void SendMessage(String msgToSend)
{
   int command=-1;
   uint8_t buf[VW_MAX_MESSAGE_LEN];
{
  char msgtmp[25];
  msgToSend.toCharArray(msgtmp,25);
   vw_send((uint8_t *)msgtmp, strlen(msgtmp));
   vw_wait_tx();
}
```

The operator can use a robot interface to view the messages which are transmitted in both sides of the communication. The fig. 4 presents the interface which is used also to give basic manual command as *connect* or *stop* to the robotic prototype.

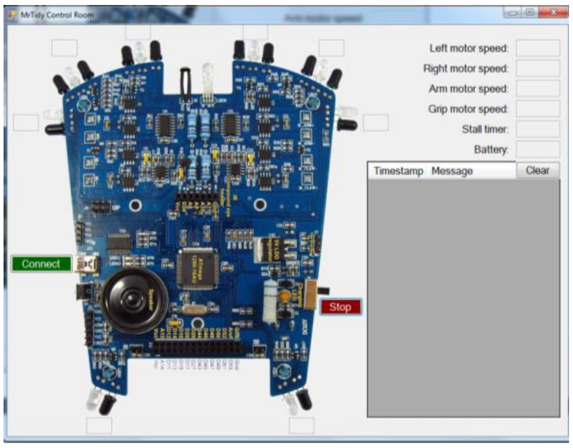

Figure 4. The view of the robot interface

B. *Off line simulation*

Two robotic agents operate in a two dimensional environments, first they act into a space without obstacles; then the obstacles are generated randomly by mouse clicks; i.e. in random sizes and distance between them, and their black shapes are circular. We can add other obstacles during the simulation time. A great number of obstacles increases the complexity of the virtual world but this required more calculation power to manage the dynamic information. At each step, agent should perform two actions:

- interaction with simulation software if agents do not sense an obstacle,
- interaction with environment if agents sense one,
- Orientation to choose the next step.

The final objective is to complete instructions came from the central management system. For this purpose, we combine two types of agent behavior in order to achieve the goal. The system is designed to control constantly the changes of the environment avoiding the failures. Our system is designed to have a set of sensors. The agent can percept obstacles and can overview its distance from them. Hanging around the motion map, the agent can discover invisible areas behind the obstacles to find the target which is not in the visible area. Here he finds the motivation to change position for a new state with purpose to reach his objective. He starts his movement following a predicated plan in base of the visual sensor information and simulation instructions.

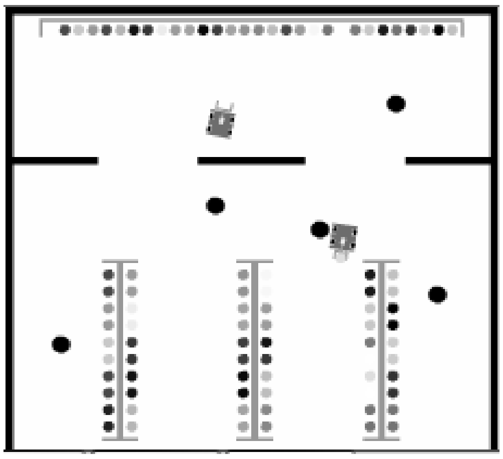

Figure. 5 The virtual environment of the simulation off line with obstacles.

This is an important moment for showing the new state after the action execution. These actions are divided in two categories: normal actions that change the environment sensing actions that accumulate information over the environment to make decisions. In intelligent systems during the execution the agent decides itself about the state of conditions: true or false.

C. *The simulation on line- Explorer modality*

In this modality, the robotic agent does not know the environment. The agent navigates choosing an explorer path until it senses the presence of an obstacle. In this case, the robot transmits data of its location and calculates the distance from the perceived obstacle. Each new state of the robot is calculated and it is showed in the virtual environment reaching the knowledge about it. The robot must perform different ways within the restricted environment by exploring the presence of obstacles and transmitting information about this.

We finally assembled and integrated graphic information of calculated points, showing how the environment becomes more complete and revealing. In the figure 6, we have showed the simulation view. There are two windows, the first is simulation view that would be in a physic environment and the second is the map that the robot makes exploring the world. The yellow obstacles are invisible for a human operator.

After 5 min of simulation, the robotic agents have transmitted data to the environment map window translated in some red points which are points of addresses where the robot senses an obstacle. The third view is the suggested map after discovering process. The same process is for the fourth view after 12 minutes. The simulation integrates more red points considering the distance criteria between them. The green area is considered as obstacle area. The simulation software calculates the rate of error made as the division of green surface with white surface in percentage.

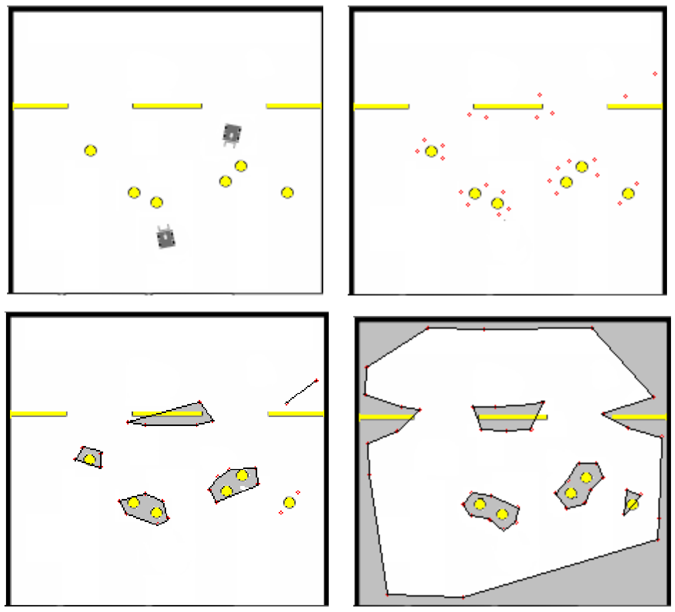

Fig. 6 View of Explorer Simulation modality

$$\text{Error\_rate} = 1 - S_{\text{green}} / S_{\text{white}} \; \%$$

In the first case, the error rate is 94 % considering the walls of environment while, after 12 minute , this error is 66%. The robotic agent needs much more time to discover exactly the map.

V. Conclusions

During the simulation, we can observe the agent environment relations and their dependencies:

- We have designed a software system that is capable to integrate successfully the execution ability of complex tasks using reflexive capacities needed to manage uncertain situations in dynamic environment.
- We observe temptations for more reactive behaviors in expected situations.
- The speed of mission is another of agent exigencies. However the environment changes with certain speed and the agent has not time enough to make a perfect decision and to choose accurately the next action.
- Time in disposition do not compromises the agent performance in the dynamic environment.
- The simulation software that we have designed provides a simple way to study the complex interactions between different types of environment and agents.
- We were interested to study the stability on making decisions of the system. Our attention was focused on how much security offers an agent based system in difficult situations.
- Many different types of robotic agent behaviors have been introduced. We could observe the agent efforts to reach the goal and the results were interesting about agent intelligence.
- Combining simulation off line with simulation on line, the agent can perform better behaviors in a partially known environment giving a new solution in software engineering.
- In further research it would be interesting to introduce new types of environment making them more complex. We aim to extend agent based systems on modeling hierarchical structures of control system and other industrial applications

References

[1] Cipi, E., Cico, B.: Information Agents as a New Paradigm for Developing Software, Applications in Database Systems, Conference Proceeding DSC 2010, Thessaloniki, Greece , Vol. 1, 514–519, (2010).

[2] Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice., Addison Wesley Publishing Comp,95--97, (2003).

[3] Kpheard, J., Kephart., J., Chess, D.:The Vision of Autonomic Computing. IEEE Computer Magazine, No. 36(1), (2004.

[4] Bandini, S., Manzoni, S..and Simone. C.: Dealing with Space in Multiagent Systems: A Model for Situated Multiagent Systems. In 1st International Joint Conference on Autonomous Agents and Multiagent Systems. ACM Press, (2002).

[5] Buchmann, F., Bass, L.: Introduction to the Attribute Driven Design Method. 23rd International Conference on Software Engineering, IEEE Computer Society.Toronto, Ontario, Canada, (2001) .Intelligent Agents Solutions to Improve Strategic Level of Self-Management... 525

[6] Clements, P., Kazman, R., Klein, M.: Evaluating Software Architectures: Methods and Case Studies. Addison Wesley Publishing Comp. (2002).

[7] Steegmans, E., Weyns, D., Holvoet,T., Berbers, Y.: A Design Process for Adaptive Behavior of Situated Agents. In Agent-Oriented Software Engineering V, 5th International Workshop, AOSE, New York, NY, USA, Lecture Notes in Computer Science, Vol. 3382. Springer, (2004).

[8] Weyns, D., Holvoet, T.: Multiagent systems and Software Architecture. In Special Track on Multiagent Systems and Software Architecture, Net.ObjectDays, Erfurt, Germany, (2006).