Handling Exceptions in Multi Agent Systems using Learning Agents

Mounira Bouzahzah[1]and Ramdane Maamri[2]
[1]*LIRE Laboratory, Science and Technologie Institute, University Center of Mila, Mila, Algeria*
[2]*LIRE Laboratory, Computer Science departement, Mentouri University, Constantine, Algeria*

Keywords: Exception Handling, Learning Agents, Q Learning Algorithm, Reinforcement Learning, Hierarchical Plans.

Abstract: In this paper we address the problem of exception handling in multi agent systems and we propose an approach based learning agents to provide fault tolerance in multi agent systems. In reality, agents systems are used as a perfect solution to design recent applications that are characterized by being decentralized and dispersed. These systems are subject of errors that occur during execution and that may cause system's failure; Exceptions are the main cause of the system's errors. Researchers in fault tolerance field use handling exceptions technique to provide error-prone systems. Through this work, we propose an approach for handling exception using learning agents; this approach assures the most efficient handler for each exception mainly in case of the existence of many handlers and allows the adaptation of decision according to the environment changes. The learning agent is given the capacity to learn about new exceptions from the extern.

1 INTRODUCTION

A critical challenge for fault tolerance researcher is to provide robust multi agent systems in terms of effectiveness and errors-prone.

Several approaches are proposed to deal with exceptions in multi agent systems, some solutions are based on the use of additional agents such as (Hagg, 1996), (Anand and Robert, 2000) and (Bouzahzah and Maamri, 2012).

(Platon, 2007) proposes the idea to give the system's agents the ability to handle the exception by themselves. Other solutions use frameworks to handle exceptions in multi agent systems such as (Souchon and Christophe, 2002) and (Klein and Dellarocas, 1999).

Our Approach assures exceptions handling in multi agent systems using a learning agent that applies a reinforcement learning algorithm (Rejeb, 2005) called the Q learning algorithm (Hoet and Sabouret, 2009). this algorithm joins each undesirable state of the agent (error detected) with its efficient handler that is decided using agent previous experiences.

This paper is organized as follows: the second section reviews the limitation of some existing approaches proposed to deal with the problem of exceptions in multi agent systems. The next section gives a detailed idea concerning agent's representation, error detection and the communication protocol. The forth section defines the learning's concepts and describes how learning agent is used to handle exceptions. The last section concludes this paper and gives indications concerning our future works.

2 EXISTING APPROACHES AND THEIR LIMITATIONS

Several approaches are proposed to solve the problem of errors in multi agent systems. Some approaches are based on decentralized solutions, which use of controlling agents to monitor the system's agents and to handle exceptions such as:

(Hagg, 1996) That describes an approach based sentinels. Sentinels are guardian agents introduced in the multi agent system application in order to provide a fault tolerance layer for the system. These guardian agents protect the system from failing in undesirable states. They have the authority to react to faults. (Anand and Robert, 2000) Introduce the concept of the supervisor which has the role of a handler for a group of system's agents. He defines two types of exceptions: the internal exceptions that

Bouzahzah M. and Maamri R..
Handling Exceptions in Multi Agent Systems using Learning Agents.
DOI: 10.5220/0004263304230426
In *Proceedings of the 5th International Conference on Agents and Artificial Intelligence* (ICAART-2013), pages 423-426
ISBN: 978-989-8565-38-9

are treated by the agent itself and the external ones handled by the supervisor which has a global access to the agents that it supervises.

These approaches are costly in terms of computation and communication and they cause point of failure since controlling agents, also, are subjects of fault. There are other approaches that treat the exception using a centralized solution. In this case, a framework is created to handle the different exceptions that occur in the system:

The SaGE system described in (Souchon and Christophe, 2002) and (Kchir, 2010) gives more important to agents' communications in terms in request/ response. An exception is treated by handlers that try to solve the problem locally, but if it is not the case the request will be returned toward the agent asking for it. Complex services that invocate other agents to solve a request use a concertation function. The approach proposed in (Klein and Dellarocas, 1999) uses a separate system called the exception handling service; this service is viewed as a coordination doctor that knows about the different cases to handle exception; exceptions, In this case, are considered as illness.

These two approaches are effective in term of decreasing the charges; but they are not able to recognize if an exception has being handle in previous times or not which cause a great waste of time, and they cannot deal with new exceptions that are not existing in their knowledge bases.

The next sections describe our new approach that tries to use learning agents to have the opportunity to handle exception according to thier experiences.

3 AGENT EXECUTION MODEL AND ERROR DETECTION

This section allows the definition of the agent's model used by the approach proposed.

3.1 Agent's Model and Plans

Representation. We use hierarchical plans (Bradley and Edmund, 1999) to model the agent's activities. The hierarchical plans representation allows specifying different combinations of alternatives to accomplish a given goal with a particular context. A hierarchical plan identifies promising classes of long-term activities, and incrementally refines these classes to eventually converge in specific actions.

The aim of our work is to develop a simple and a formal model that represents the agent's activities and allows error detection in order to solve the problem of failure in multi-agent systems.

Definition:
A hierarchical plan P is defined in our model by a tuple (Pre (P), In (P), Post (p), type (P), sub plans (P), Order (P)):

- Pre (p): set of conditions and data needed for the execution.
- In (p): set of conditions that must be verified during the execution.
- Post (P): denote the set of conditions verified and the data provided at the end of the execution.
- The type of plan P type (P) can be: a simple plan (action), OR plan which is a hierarchical plan that is accomplished by carrying out one of its sub plans, AND plan is also a hierarchical plan that is accomplished by carrying out its entire sub plans.
- Sub plans: is a set of plans. In case of simple plans the set of sub plans is empty.
- Order (P) represents the order of execution of plan P.

3.2 The Error Detection

To determine that an error occurs at one point of time, we have to define to a new concept that is the execution.

Definition:
An execution of a plan P is an instance of decomposition and ordering of its sub plans' executions. The set E (P) represents the possible executions of a plan P.

An execution e of P is recursively defined as a tuple <D (e), ts, tf, V (Pre (e), In (e), Post (e))>:

- ts, tf are positive, non zero real numbers representing the start and finish times of execution e, and $t_s < t_f$.
- D (e) is the set of sub plans executions representing the decomposition of plan P under this execution e.
- V (Pre (e), In (e), Post (e)): a Boolean function that verify the previous conditions. It returns true if the execution succeeds and returns false if an error occurs.

The agent has to compare the expected conditions that appear in his hierarchical plan with the real conditions that occur at the moment of execution; if all the conditions are verified then the action is executed with success otherwise an exception is detected.

3.3 The Communication Protocol

The agent has to communicate the learning agent and the other system's agents using messages. The general model of messages in our approach has the following structure:

M=<Send, Recept, C>

- M represents the message or the communication protocol.
- Send represents the code that identifies the agent who sends the message and who is searching for a handler for the detected exception.
- Recept identifies the learning agent.
- C represents the content of the message that may design an exception.

The content of the message, in our work, represents the current state of the agent that may be the expected value of conditions and the resulting conditions.

C=< (Pre (P), In (P), Post (p)), (Pre (e), In (e), Post (e))>

4 LEARNING AGENT AND EXCEPTION HANDLING

In reality, two types of learning can be used in multi-agent systems:

- Individual learning occurs independently of other agents. It is taken into consideration only when all the elements of the learning process are executed by the same agent and requires no interaction with other agents.
- Multi-agent learning requires the presence of other agents and their interaction (Watkins,1989).

We consider the first type of learning in our model since we choose to have a centralized solution in order to decrease the charges. So, the learning agent is the one concerned by solving the exception signaled.

4.1 The Learning Models

The learning models are classified into two main categories:

- Unconscious learning or reinforcement learning that occurs when the agent does not know what he is learning. It is a mechanism that operates automatically and continuously.
- The conscious learning takes place when the agents are able to reflect on their actions and their consequences (Brenner, 2005).

We are interesting to have a learning agent that can learn automatically and continuously, thus, we choose reinforcement learning model and we apply the Q learning algorithm known by being simple and gives clear results.

4.2 Q Learning Algorithm and Exception

Handling. Q-learning is a reinforcement learning technique that works by learning an action-value function that gives the expected utility of taking a given action in a given state and following a fixed policy thereafter. One of the strengths of Q-learning is that it is able to compare the expected utility of the available actions without requiring a model of the environment (Qlearning, Wikipedia).

Concerning our approach the learning agent uses the Q learning algorithm in order to join the couple (exception, handler) and a value Q (exception, handler) that represents the reward or the effectiveness of the handler to solve this exception. So, this algorithm is based on the use of a function whose value is:

$$Q: S \times A \longrightarrow R$$

- S is the set of the agent s states in other words it represents the set of errors or exception detected. When the learning agent receives the message concerning the exception it chooses the handlers that can be associated within this exception.
- A represents the set of actions or handlers proposed to solve errors.
- Q(s, a) this value represents the reward when using the handler a to solve the exception s.

The agent can move from exception state to a normal one. Each state provides the agent a reward (a real or natural number). The goal of the agent is to maximize the reward. It does this by learning which action is optimal for each state. So, the best handler for an exception is the one whose reward is the maximum.

Before learning has started, the learning agent is initialized by some handlers and the function Q returns a fixed value, chosen by the designer. Then, each time the agent is given a reward (the error is solved) new values are calculated for each combination of a state s from S, and action a from A.

The core of the algorithm is a simple value iteration update. It assumes the old value and makes a correction based on the new information. The update of the reward value is done according to the following formula:

$$Q(s,a) \leftarrow (1- \propto t(s,a))Q(s,a) + \propto t(s,a)[r + \gamma \max Q(s',a')]$$

The goal of the Q learning algorithm is to build the reward function Q for each couple (s,a) according to the result given after the use of the handler a. αt has a value in [0,1] it refers to the learning rate. γ is the actuel lisation factor, it allows the learning agent to determinr the best reward. If the agent needs immediate rewards so the actual lisation factor must be near 0.

4.3 The Learning Agent's Memory

The learning agent has to remember the historical value of rewards for each handler in order to choose the best handler in case where an exception appears for the second time.

As we show in the previous paragraphs the Q learning algorithm builds its decisions depending on its perception of the system. But in our approach we need a decision that depends also on the agent's historical experiences. As a solution we propose to use the learning agent memory that includes the past decisions about handlers. According to this idea the agent will be able to recognize an action that has appears in the past and choose its best handler.
In case where the exception has reward =0 using all the available handlers. We give the learning agent the authority to ask for new solutions from the extern designer. The knowledge base will be extended and the new added solution will be treated as the initial ones.

5 CONCLUSIONS AND FUTURE WORK

Throughout this paper, we have proposed an effective approach for fault tolerance in multi-agent systems based on learning agent. We use a formal model for agent activities representation called hierarchical plans. It allows the detection of errors in a simple and automatic way. We choose Q learning algorithm to handle exceptions. Learning agent has the opportunity to handle exceptions according to his experiences; to choose the most effective handler in case where may handlers exist, and we give the agent the ability to learn from the extern in case of new exceptions.

Finally, we are interested in validating this work through a simulation that can provide real results on the effectiveness of this approach and compare it with other approaches.

REFERENCES

Anand T. and Robert M., 2000. *Exception Handling in Agent Oriented Systems*, Springer-Verlag,

Bouzahzah M. And Maamri R., 2012. A Proposed Architecture for a Fault Tolerant Multi Agents System Using Extern Agents, *6th International Conference, K.E.S.-A.M.S.T.A, proceedings Springer LNAI* 7327pp 282-289,

Bradley J. C., Edmund H. D., 1999. Identifying and Resolving Conflicts among Agents with Hierarchical Plans, *American Association for Artificial Intelligence*.

Brenner T., 2005. Handbook of Computational Economics Vol: 2. *Agent-Based Computational Economics*, (Handbooks in Economics Series),

Hagg S., 1996. A Sentinel Approach to Fault Handling in Multi-Agent Systems. In: *Proceedings of the Second Australian Workshop on Distributed AI*, Cairns, Australia.

Hoet S., and Sabouret N., 2009. Apprentissage par Reforcement d'acte de Communication Dans un Contexte Multi agents", *RJCIA*.

Kchir S., 2010. Gestion des Exceptions dans un Système Multi Agents avec Replication, Master2*, Laboratoire d'Informatique, de Robotique et de Micro-electronique*, Montpellier.

Klein M., Dellarocas C., 1999. Exception Handling in Agent Systems, *Antonymous agents*, USA.

REJEB L., 2005. *Simulation Multi agents des Modèles Economiques Vers des Systèmes Multi Agent Adaptatif*'', Reims Champagne-Ardennes University.

Platon E., 2007. *Modeling Exception Management in Multi Agent Systems*, Doctorate thesis, department of informatics, the graduate university for advanced studies.

Souchon F., Christophe D., Christelle U., Sylvain V.,and Jacques F., 2002. *SaGE: une proposition pour la gestion des exceptions dans les system multi agents*, Internal repport-LIRMM-02205,

Q-learning –Wikipedia, the free encyclopaedia: http://en.wikipedia.org/wiki/Q-learning

Watkins C. J. C. H., 1989. *Learning from delayed rewards*, PhD thesis, Cambridge University, Cambridge, United Kingdom.