SYSTEM FAULT-TOLERANCE USING MULTI-AGENT SYSTEM

P. Lokesh Kumar Reddy*

B. Rama Bhupal Reddy**

S. Rama Krishna***

Abstract: The presence of errors and faults in systems is still a problem that needs to be handled. This paper tries to examine such a problem in the context of multi-agent systems. For solving it we analyze the use of two mechanisms: self-healing and self-organization, in order to manage fault at agent level as well as at the overall network infrastructure level. In order to realize these we proposed the use of adaptive agents which change state change their status according to the current situation and transmit their decisions to other agents by the means of a gossip communication protocol.

Keywords: Multi-agent system, self-healing, self-organization, gossip communication

* Assistant Professor, Department of Computer Science, Rama Raja Institute of Technology and Science, Tirupati, A.P., India

** Associate Professor, Dept. of Mathematics, K.S.R.M. College of Engineering, Kadapa, A.P., India

*** Professor, Dept. of Computer Science, S.V. University, Tirupati, A.P., India

1. INTRODUCTION

As the field and utilization of multi-agent systems is growing also the necessity for systems to be up and running 24 h of 24 h became mandatory. If a minor glitch or failure occurs it is mandatory to be treated in time so the damage to be minimal.

Systems in large companies today can't afford to have failures that can affect the entire system. Unfortunately bad things happen and as things evolve it is not possible to have foolproof systems. Cases of failure reported by big companies like Skype in 2007, shown that systems are not perfect, although we want to believe they are. What we can do in case of a failure is to reduce it to an acceptable level or even to recover from it, depending on the situation. So to do this what we need to do is to make a system fault tolerance.

1.1 Fault-tolerance overview

Fault-tolerance is the property that allows a system to continue operating properly in the event of the failure of some of its components. This concept do not refer only to individual agents, it can refer also to the interaction of agents from a multi-agent system.

When working on building such system it is fair to assume that we know what should be treated as being a correct behavior and what seems to be a deviation from it or how a fault can be recognized. This problem has been analyzed in many papers. One of the most used classifications of faults is the one described in [7], according to which faults can be classified according to seven attributes: phase of creation or occurrence, system boundaries, dimension, phenomenological cause, objective, capability, persistence.

1.2 Multi-Agent System overview

A multi-agent system (MAS) is a loosely coupled network of software agents that interact to solve problems that are beyond the individual capacities or knowledge of each problem solver" [16].

Analyzing the field of multi-agent system development tools we can discover that there are a lot of such tools ones better than other depending on the problem you want to solve and the developer knowledge. Some of these toolkits are: Retsine [10], Agent Builder, Jake [17], and Jade [2].

An extended comparison of these platforms can be found in paper [13]. Our main criteria, base on which we selected Jade as an environment for developing a multi-agent system,

were: familiarity with the specific environment, license type, system adaptability, system capabilities to support advanced development interaction, tools debugging facilities etc.

2. SELF-HEALING AND SELF-ORGANIZATION STUDY

2.1 Overview

When talking about fault-tolerance systems in a multi-agent environment we are talking about a reliable system capable of treating and recovering from faults. Such a system in the current literature is defined in term of a self managed system.

The spread of capabilities and actions that a system can encounter and manage has led to a classification of self-* systems according to the features that each class is mapped on. So we can have: Self-healing, Self-organization, Self-optimizing, Self-protection etc.

For the moment we are going to focus on self-healing at agent level and self-organization features of the whole architecture of a system.

2.2 Self-healing

Self-healing is a modern approach to the problem of detecting failure, diagnose and recovering from it in order to make the system more reliable. Our analysis is focused on system repair and recovery.

First when we think of a self-healing mechanism what we need to define are the states of the system. Normally as shown in the paper [1] there are three main states in which a system can be: normal state, degraded state, broken state. Usually it is tricky to define each of this states the line between them is sometimes gray. For this purpose is very important to define for every system in part as many characteristics as needed for the proposed mechanism of classification, making it clear for different situation encountered in which state the system is.

There are many different approaches on how a self-healing system should look [6], [14] but all of this approaches spin around the same main mechanisms: detect, diagnose, analyze, plan, knowledge, recover.

Fault detection in systems which have a self-healing mechanism usually is obtained by having mechanism able to recognize degradation. For this to be realized different papers show two main approaches for detecting and reporting suspicious behavior: searching for inconsistencies in the sensed data or triggered by inappropriate behavior[4].

Very important for a self-healing system are its policies. In [4] are defined three groups of

policies: Action Policies, Goal Policies, and Utility Function Policies.

And last but not the least are the recovery techniques that show what behavior should be applied: replacement, balancing, isolation, persistence, redirection, relocation, diversity.

2.3 Self-Organization

There are many definitions of self-organization in almost every technical and social field. From the point of view of computer science the trend is usually focused on trying to understand how nature do thing because it usually knows to do it best. According to "Self-organization is defined as the mechanism or the process enabling a system to change its organization without explicit external command during its execution time." In this pa- per the author also has classified self-organization as being weak or strong according to the presence or not of a central control and planning agent. A typical self-organizational system needs to be capable of adaptation, evolution and emergence.

When creating a self-organizing mechanisms there are also some specific that need to be taken into account and determined for the system[5]: self-determined boundaries, operational closure and energetic openness, independence of identity and structure, maintenance(self-healing), feedback and hierarchy, criticality, emergence, self-determined reaction to perturbations, reduction of complexity.

According to [3] the approaches that can be explored when designing a self-organizational system can be divided in five classes depending on the mechanisms they are based on: direct interactions between agents using basic principles such as broadcast and localization, indirect interactions between agents, reinforcement of agent behaviors, cooperation behavior of individual agents, choice of a generic architecture.

One of the main problems in a multi-agent system is the consensus between agents. For solving this problem a step forward is the use of a good communication protocol. Such mechanism is gossip-based protocols that have been proved to be able to maintain connectivity in large-scale dynamic systems in the presence of high churn and other extreme failure scenarios [11].

Examples of gossip based protocols are newscast [8], T-MAN [12], Ranking protocol, Sliver protocol [15] Etc.

3. PROJECT DESCRIPTION

In the current paper we are presenting the general architecture and implementation of an agent-based system, built using Jade [2].

3.1 Systemarchitectue

Our model considers a network of agents that are assigned unique identifiers and that communicate using message exchange. The system is going to be able to communicate also, with other agents that are not in our proposed environment. There are multiple available approaches to building and designing a fault-tolerance system in the context of multi-agent systems. We propose a framework based approach that imply for every agent to have on top of the Multi-agent development tool a fault-tolerance layer.

3.2 Agents architecture

Our proposed agents are going to be autonomous agents capable of taking their own decision based on specific situations. They will adapt to system conditions and take decisions based on internal state and environment information. These decisions are going to be the best solutions for the welfare of the overall system. Agents are going to be cooperative agents with the following general characteristics: Skills (what is able to do), Social aptitude(cooperation), Knowledge (about itself, other agents, environment) etc.

Proposed architecture at agent level:

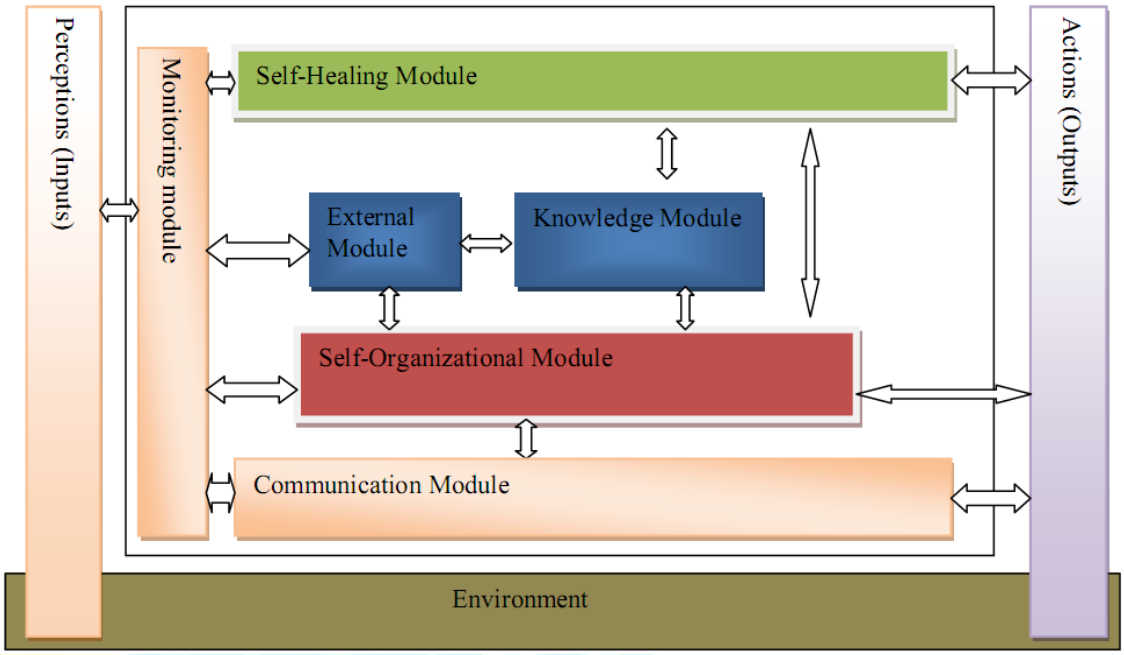

Figure 1: Agent Fault-tolerance layer architecture

3.3 Agent modules

As seen in the above figure our solution is based on six modules for the moment: monitoring, communication, knowledge, external, organizational and healing module. The number of modules although can rise in time, if we map to our framework others self methods

3.3.1 The Monitoring module

This module is intended to be a mechanism of healing modules interfacing with the environment and organization. Together with the communication module is intended to be a complete solution for all reception and delivery of information from and in the environment.

3.3.2 Communication Module

This module is responsible with the communication between agents. This is a very important module because based on it the self-organization module can do its job. Implementation is made according to considerations. Our protocol extends the following generic protocol:

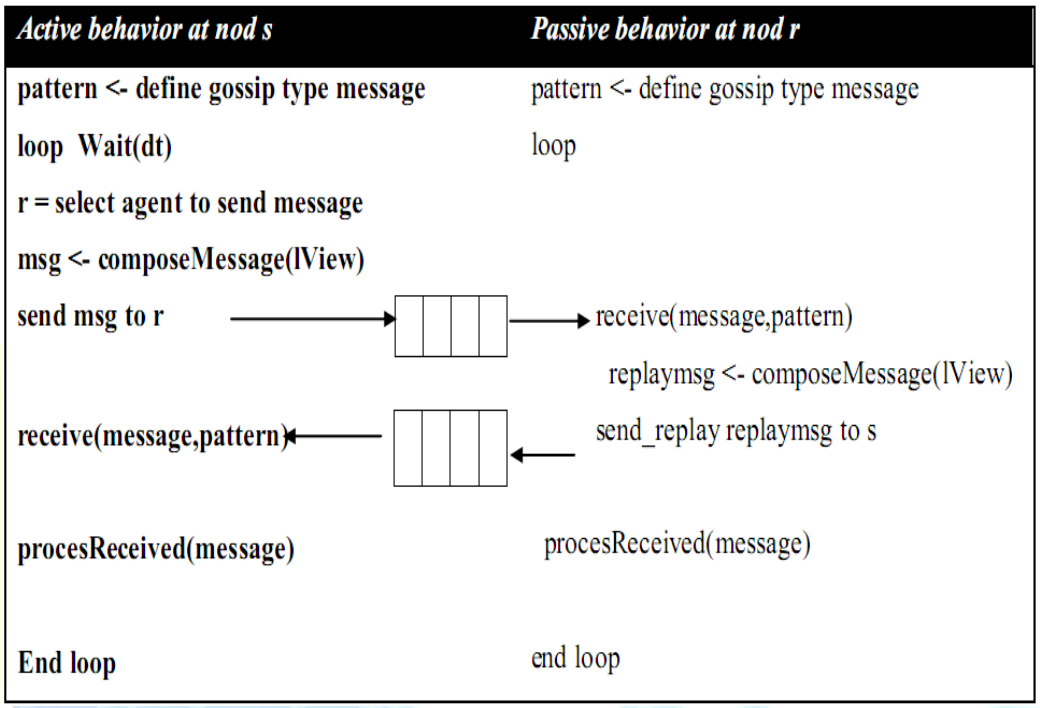

Figure 2: Communication protocol

3.3.3 Knowledge module

We are going to use this module for storing all the information needed by the others modules. It will interact with all modules and offers to other modules the possibility of storing relevant information during processing and after it.

Every module attached to the architecture will have an interface which facilitates communication with the knowledge module. In it will store patterns for abnormal and normal behavior, also actions that need to be taken in case a specific situation was detected together with other relevant information that will help achieve a fault-tolerant system.

3.3.4 External module

External module is used to collect information and maintain a database of known failures and solutions and also for interaction with the external environment.

3.3.5 Healing module

This module is responsible with detection and recovering from system failures. Module structure:

- Monitor - offers the interface between this module and the monitor one. It makes the necessary filtering and changes to the monitored data so that in can be easier processed by de

diagnosing engine;

- Communication - in order to send information about its state in the system
- Knowledge - this is an interface with the general Knowledge module;
- Diagnosing - analyze(responsible for analyzing the situation), plan(based on situation proper actions are generated and analyzed), execute(actions proposed by the plan stage are being executed).

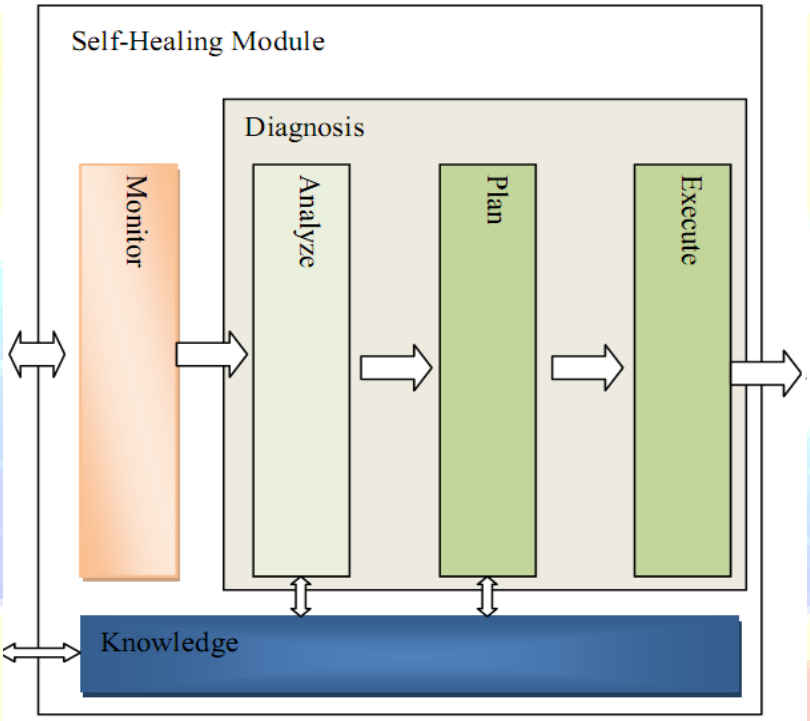

Figure 3: Self-healing module

3.3.6 Organizational Module

It is a module that implements the self-organization algorithm for the proposed architecture. It is responsible with selecting the agents to which the current agent need to communicate in order to achieve desired overlay an to maintains it. This module interacts with other modules using: communication interface (provides the interface between current module and the communication one), knowledge interface (this is an interface with the general Knowledge module).

Apart from these interfaces we will also have a situation recognition module and a situation management module that decides what information need to be stored and sent.

4. CONCLUSION AND FUTURE WORK

Software reliability in complex systems is an important issue for big companies with certain prestige and not only. To prevent losses of customers and money is important to have a fault-tolerant system capable of encapsulating many capabilities, which ultimately serve to prevent the entire system failure.

Current paper was divided into two parts. In the first part we have described the general mechanisms used in handling faults, that for the moment implies the use of self-healing and self-organization approaches in order to achieve our goal. The second part of this document has been focused on designing a fault-tolerance multi-agent system that implemented and analyzed the features described in the first part of the document.

The next step of this project is going to be the implementation of these concepts and proposed architecture along with a more detailed description of it in Jade environment. Another possible extension is going to be the prevention of failures in self mechanism by using mechanisms like the use of computational verification or replication of key healing components.

REFERENCES

[1] Upadhyaya Debanjan Ghosh H. Raghav Rao Shambhu and Raj Sharman. Self-healing systems - survey and synthesis. Decision Support Systems, 42(4):2164–2185, January 2007.

[2] G. Caire. D. Greenwood F. Bellifemine. Developing Multi-Agent Systems with Jade. Wiley Series in Agent Technology, 2007.

[3] M.-P. Gleizes G. D. M. Serugendo and A. Karageorgos. Self-organization in multi-agent systems, volume 20(2):165-189. The Knowledge Engineering Review, 2005.

[4] Schahram Dustdar Harald Psaier. A survey on self-healing systems: approaches and systems. Computing, pages 91(1):43–73. August 2010.

[5] Christian Koppen Hermann De Meer. Characterization of self-organization. In peer- to-peer systems and applications. 2005.

[6] McCann JA Huebscher MC. A survey of autonomic computing - degrees, models, and applications. 40(3):1-28, August 2008.

[7] Yves Crouzet Yves Deswarte Jean-Charles Fabre Jean-Claude Laprie Jean Arlat and David Powell. Tolerance aux fautes. Pages 241–270. In Encyclopedia del' Informatique et des Systems d'Information, 2006.

[8] M. Kowalczyk W. Jelasity and M. van Steen. Newscast computing. Technical report, Vrije Universities Amsterdam, Department of Computer Science, Amsterdam, 2003.

[9] Patrick Taillibert Katia Potiron and Amal El Fallah Seghrouchni. A step towards fault tolerance for multi-agent systems., volume 5118/2008 of In Languages, Methodologies and Development Tools for Multi-Agent Systems, pages 156–172. Springer Berlin / Heidelberg, France, 2008.

[10] Massimo Paolucci. Martin Van Velsenv-Joseph Giampapa Katia Sycara. The retsinamas infrastructure. In autonomous agents and multi-agent systems. Volume 7, pages 29–48. Springer Netherlands, Pittsburgh, 2004.

[11] Rachid Guerraoui Anne-Marie Kermarrec, Mark Jelasity and Maarten van Steen. The peer sampling service: Experimental evaluation of unstructured gossip-based implementations. Volume 3231 of In Lecture Notes in Computer Science. Springer-Verlag, 2004.

[12] Ozalp Babaoglu Mark Jelasity. T-man: Gossip-based overlay topology management. Volume 3910 of In Engineering Self-Organizing Systems, pages 1–15. Springer Berlin / Heidelberg, 2006.

[13] Cynthia Nikolai and Gregory Madey. Tools of the trade: A survey of various agent based modeling platforms. Artificial Societies and Social Simulations. 2009.