Fault Tolerant Multi-Agent Systems: its communication and cooperation

Arnulfo Alanis Garza, Juan José Serrano, Rafael Ors Carot, José Mario García Valdez

***Abstract*— Intelligent Agents have originated a lot of discussion about what they are, and how they are different from general programs. We describe in this paper a new paradigm for intelligent agents. This paradigm helped us deal with failures in an independent and efficient way. We proposed three types of agents to treat the system in a hierarchical way. In this I articulate is to the operation of the Agent Node**

A new method to visualize fault tolerant systems (FTS) is proposed, in this paper with the incorporation of intelligent agents, which as they grow and specialized create the Multi-Agent System (MAS). One is the communications diagrams of each of the agents, described in diagrams of transition of states.

***Index Terms*— Intelligent Agents, Fault Tolerance, Distributed System.**

I. Introduction

At the moment the approach using agents for real applications, has worked with movable agents, which work at the level of the client-server architecture. However, in systems where the requirements are higher, as in the field of the architecture of embedded industrial systems, the idea is to innovate in this area by working with the paradigm of intelligent agents. Also, it is a good idea in embedded fault tolerant systems, where it is a new and good strategy for the detection and solution of errors.

A **rational agent** is one that does the right thing. Obviously, this is better than doing the wrong thing, but what does it mean? As a first approximation, we will say that the right action is the one that will cause the agent to be most successful. That leaves us with the problem of deciding *how* and *when* to evaluate the agent's success

By an *agent-based* system, we mean one in which the key abstraction used is that of an agent. In principle, an agent-based system might be conceptualized in terms of agents, but implemented without any software structures corresponding to agents at all. We can again draw a parallel with object-oriented software, where it is entirely possible to design a system in terms of objects, but to implement it without the use of an object-oriented software environment. But this would at best be unusual, and at worst, counterproductive. A similar situation exists with agent technology; we therefore expect an agent-based system to be both designed and implemented in terms of agents. A number of software tools exist that allow a user to implement software systems as agents, and as societies of cooperating agents.

A. Agents

Let's first deal with the notion of intelligent agents. These are generally defined as "software entities", which assist their users and act on their behalf. Agents make your life easier, save you time, and simplify the growing complexity of the world, acting like a personal secretary, assistant, or personal advisor, who learns what you like and can anticipate what you want or need. The principle of such intelligence is practically the same of human intelligence. Through a relation of collaboration-interaction with its user, the agent is able to learn from himself, from the external world and even from other agents, and consequently act autonomously from the user, adapt itself to the multiplicity of experiences and change its behaviour according to them. The possibilities offered for humans, in a world whose complexity is growing exponentially, are enormous [1][4][5][6].

We need to be careful to distinguish between rationality and **omniscience**. An omniscient agent knows the *actual* outcome of its actions, and can act accordingly; but omniscience is impossible in reality. Consider the following example: I am walking along the Champs Elys´ees one day and I see an old friend across the street. There is no traffic nearby and I'm not otherwise engaged, so, being rational, I start to cross the street. Meanwhile, at 33,000 feet, a cargo door falls off a passing airliner, and before I make it to the other side of the street I am

A. Alanis is with the Tijuana institute of technology, Division Graduate and studies Reseach, Calzada Tecnologico, S/N, Tijuana, BC 22379 Mexico: 664-682-72-79; e-mail: alanis@ tectijuana.mx).

J. Jose Serrano is with the Universidad Politécnica de Valencia (España), D. Inf. de Sistemas y Computadoras, Camí de Vera, s/n, 46022 VALÈNCIA, ESPAÑA, ,: 00+34 96387, email:jserrano@disca.upv.es.

R.Ors Carot is with the Universidad Politécnica de Valencia (España), D. Inf. de Sistemas y Computadoras, Camí de Vera, s/n, 46022 VALÈNCIA, ESPAÑA, :00+34 96387, email:rors@disca.upv.es.

J. Mario Garcia is with the Tijuana institute of technology, Division Graduate and studies Reseach, Calzada Tecnologico, S/N, Tijuana, BC 22379 Mexico: 664-682-72-79; e-mail: mario@ tectijuana.mx.

flattened. Was I irrational to cross the street? It is unlikely that my obituary would read “Idiot attempts to cross street.”Rather, this points out that rationality is concerned with *expected* success *given what has been perceived*. Crossing the street was rational because most of the time the crossing would be successful, and there was no way I could have foreseen the falling door. Note that another agent that was equipped with radar for detecting falling doors or a steel cage strong enough to repel them would be more successful, but it would not be any more rational

II. Autonomy

There is one more thing to deal with in the definition of an ideal rational agent: the “built-in knowledge”part. If the agent’s actions are based completely on built-in knowledge, such that it need pay no attention to its percepts, then we say that the agent lacks **autonomy**. For example, if the clock manufacturer was prescient enough to know that the clock’s owner would be going to Australia at some particular date, then a mechanism could be built in to adjust the hands automatically by six hours at just the right time. This would certainly be successful behavior, but the intelligence seems to belong to the clock’s designer rather than to the clock itself. An agent’s behavior can be based on both its own experience and the built-in knowledge used in constructing the agent for the particular environment in which it operates. *A system is autonomous*[4] *to the extent that its behavior is determined by its own experience.* It would be too stringent, though, to require complete autonomy from the word go: when the agent has had little or no experience, it would have to act randomly unless the designer gave some assistance. So, just as evolution provides animals with enough built-in reflexes so that they can survive long enough to learn for themselves, it would be reasonable to provide an artificial intelligent agent with some initial knowledge as well as an ability to learn. Autonomy not only fits in with our intuition, but it is an example of sound engineering practices. An agent that operates on the basis of built-in assumptions will only operate success-fully when those assumptions hold, and thus lack flexibility. Consider, for example, the lowly dung beetle. After digging its nest and laying its eggs, it fetches a ball of dung from a nearby heap to plug the entrance; if the ball of dung is removed from its grasp *en route*, the beetle continues on and pantomimes plugging the nest with the nonexistent dung ball, never noticing that it is missing. Evolution has built an assumption into the beetle’s behavior, and when it is violated, unsuccessful behavior results. A truly autonomous intelligent agent should be able to operate successfully in a wide variety of environments, given sufficient time to adapt.

III FIPA (The Foundation of Intelligence Physical Agents)

FIPA specifications represent a collection of standards, which are intended to promote the interoperation of heterogeneous agents and the services that they can represent

The life cycle [9] of specifications details what stages a specification can attain while it is part of the FIPA standards process. Each specification is assigned a specification identifier [10] as it enters the FIPA specification life cycle. The specifications themselves can be found in the Repository [11]

The Foundation of Intelligent Physical Agents (FIPA) is now an official IEEE Standards Committee.

IV FIPA ACL message

A FIPA ACL message contains a set of one or more message elements. Precisely which elements are needed for effective agent communication will vary according to the situation; the only element that is mandatory in all ACL messages is the performative, although it is expected that most ACL messages will also contain sender, receiver and content elements.

If an agent does not recognize or is unable to process one or more of the elements or element values, it can reply with the appropriate not-understood message.

Specific implementations are free to include user-defined message elements other than the FIPA ACL message elements specified in Table 1. The semantics of these user-defined elements is not defined by FIPA, and FIPA compliance does not require any particular interpretation of these elements.

Some elements of the message might be omitted when their value can be deduced by the context of the conversation. However, FIPA does not specify any mechanism to handle such conditions, therefore those implementations that omit some message elements are not guaranteed to interoperate with each other

The full set of FIPA ACL message elements is shown in Table 1 without regard to their specific encodings in an implementation. FIPA-approved encodings and element orderings for ACL messages are given in other specifications. Each ACL message representation specification contains precise syntax descriptions for ACL message encodings based on XML, text strings and several other schemes.

A FIPA ACL message corresponds to the abstract element message payload identified in the [15]

Table 1: FIPA ACL Message Elements

Element	Category of Elements
`performative`	Type of communicative acts
`sender`	Participant in communication
`receiver`	Participant in communication
`reply-to`	Participant in communication
`content`	Content of message
`language`	Description of Content
`encoding`	Description of Content
`ontology`	Description of Content
`protocol`	Control of conversation
`conversation-id`	Control of conversation
`reply-with`	Control of conversation
`in-reply-to`	Control of conversation
`reply-by`	Control of conversation

.

The following terms are used to define the ontology and the abstract syntax of the FIPA ACL message structure:

Frame. This is the mandatory name of this entity, that must be used to represent each instance of this class.

Ontology. This is the name of the ontology, whose domain of discourse includes their elements described in the table.

Element. This identifies each component within the frame. The type of the element is defined relative to a particular encoding. Encoding specifications for ACL messages are given in their respective specifications.

Description. This is a natural language description of the semantics of each element. Notes are included to clarify typical usage.

Reserved Values. This is a list of FIPA-defined constants associated with each element. This list is typically defined in the specification referenced.

All of the FIPA message elements share the frame and ontology shown in Table 2.

Table 2: FIPA ACL Message Frame and Ontology

Frame	FIPA-ACL-Message
Ontology	FIPA-ACL

V The KQML Language

Communication takes place on several levels. The content of the message is only a part of the communication. Begin able to locate and engage the attention of someone you want to communicate with is apart of the process. Pack-aging your message in a way which makes your purpose in communicating clear is another.

When using KQML, a software agent transmits content messages, composed in a language of its own choice, wrapped inside of a KQML message. The content message can be expressed in any representation language and written in either ASCII strings or one of many binary notations (e.g. network independent XDR representations).All KQML implementations ignore the content portion of the message except to the extent that they need to recognize where it begin sand ends.

The syntax of KQML is based on a balanced parenthesis list. The initial element of the list is the performative and the remaining elements are the performative's arguments as keyword/value pairs. Because the language is relatively simple, the actual syntax is not significant and can be changed if necessary in the future. The syntax reveals the roots of the initial implementations, which were done in Common Lisp, but has turned out to be quite flexible

KQML is expected to be supported by an software substrate which makes it possible for agents to locate one another in a distributed environment. Most current implementations come with custom environments of this type; these are commonly based on helper programs called routers or facilitators. These environments are not a specified part of KQML. They are not standardized and most of the current KQML environments will evolve to use some of the emerging commercial frameworks, such as OMG's CORBA or Microsoft's OLE2, as they become more widely used.

The KQML language supports these implementations by allowing the KQML messages to carry information which is useful to them, such as the names and addresses of the sending and receiving agents, a unique message identifier, and notations by any intervening agents. There are also optional features of the KQML language which contain descriptions of the content: its language, the ontology it assumes, and some type of more general description, such as a descriptor naming a topic within the ontology. These optional features make it possible for the supporting environments to analyze, route and deliver messages based on their content, even though the content itself is inaccessible [17].

VI KQML Software Architectures

KQML was not defined by a single research group for a particular project. It was created by a committee of representatives from different projects, all of which were concerned with managing distributed implementations of systems. One was a distributed collaboration of expert systems in the planning and scheduling domain. Another was concerned with problem decomposition and distribution in the CAD/CAM domain. A common concern was the management of a collection of cooperating processes and the simplification of the programming requirements for implementing a system of this type. However, the groups did not share a common communication architecture. As a result, KQML does not dictate a particular system architecture, and several different systems have evolved [19].

VII Agent Communication Protocols

There are a variety of interprocess information exchange protocols. In the simplest, one agent acts as a client and sends a query to another agent acting as a server and then waits for a reply, as is shown between agents A and B in Figure 1. The server's reply might consist of a single answer or a collection or set of answers. In another common case, shown between agents A and C, the server's reply is not the complete answer but a handle which allows the client to ask for the components of the reply, one at a time. A common example of this exchange occurs when a client queries a relational database or a reasoner which produces a sequence of instantiations in response. Although this exchange requires that the server maintain some internal state, the individual transactions are as before - involving a synchronous communication between the agents. A somewhat different case occurs when the client subscribes to a server's output and an indefinite number of asynchronous replies arrive at irregular intervals, as between agents A and D in Figure 1. The client does not know when each reply message will be arriving and may be busy performing some other task when they do.
There are other variations of these protocols. Messages might not be addressed to specific hosts, but broadcast to a number of them. The replies, arriving synchronously or asynchronously have to be collated and, optionally, associated with the query that they are replying to [18].

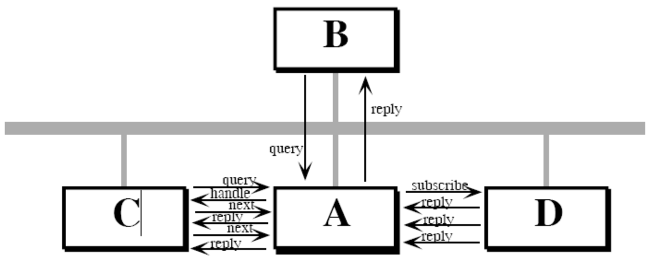

Figure 1: Several basic communication protocols are supported in KQML

VIII Proposed Method

Be a Distributed System (mainly applied to the industrial control), which is made up of a set of Nodes, where each one of them can be constituted by several Devices.

On these Nodes a set of Tasks, ordered is executed all of them to take I finish the functionality of the system. In order to identify this Distributed System the following definitions set out:

Definition 1: = is N {Nor}, the set of the Nodes of the system, being n is the number of units that integrate it.

Definition 2: Be [Di, z], the set of devices that contains Node i. Where z can take value 1, if it is wanted to see the Node like only device, or greater than 1 if it is desired to do visible some of the elements that integrate it.

Definition 3: = is T {T j}, the set of tasks that are executed in the system, being t the number of tasks that integrate the system.

Definition 4: A System Distributed like dupla is defined: SD = (N, T) Once characterized what a Distributed System could be denominated Basic (without no characteristic of Tolerance to Failures), is going away to come to the incorporation on he himself from the paradigm of Intelligent Agents with the purpose of equipping it with a layer with Tolerance to Failures. The Fault tolerant Agents will define themselves now that worked in the SD

Definition 5: An Agent is ANi to whom Agent denominated itself Node, whose mission is the related one to the tolerance to failures at level of the Node Nor. An Agent exists therefore Node, by each Node of the System (biyectiva application).

Definition 6: An Agent is ATj to whom Agent denominated itself Task, whose mission is the related one to the tolerance to failures at level of the Tj Task. An Agent exists therefore Task by each task of the system.

Definition 7: An Agent is AS to whom Agent denominated itself to him System, whose mission is the related one to the tolerance to failures system level.

Definition 8: AN= {ANi} the set of all the Agents Node.

Definition 9: AT= {ATj} the set of all the Agents Task.

Definition 10: A System is SMATF Fault tolerant Multi-Agent, formed by tripla of ANi, ATj, AS.

That is to say, SMATF = <Ni, AT j, AS> with it a Distributed System Fault tolerant SDTF is defined as:

Definition 11: A Distributed System Fault tolerant SDTF like dupla SDTF is defined SDTF =<SD, SMATF > Next goes to describe with greater detail each one of the Agents who compose the SMATF

IX Control of Conversation

In this section we describe the control of conversation between agents. In table 3 we show the protocol. In this table 4 we show the conversation identifier of the node agent. In table 5 we show the reply of an agent.

Table 3 Protocol

Element	Description	Reserved Values
Protocol TCP/IP	Denotes the interaction protocol that the sending agent is employing with this ACL message	See [16]

Table 4 Conversation Identifier of Node Agent ANi

Element	Description	Reserved Values
AN*i*.Phase.Detection y AN*i*.{Input-Error (*i,j*).Error} ANi.Phase.Detection y ASNi. operation .Errordetected ANiS.Stateunderrecovery AN*i*.Phase.Location y AN*i*.Input-Error(*i,j*).Error ANi.Phase.Location y ANi.TestDz.end AN*i*.Phase.Isolation y AN*i*.Device[D*i,z*].Incorrect AN*i*.Phase.Recunfiguration and ANi.Dispositivo[D*i,z*].Incorrect and ANi.Dispositivo[D*i,z*].I criticize. AN*i*.Phase.Recunfiguration and ANi.Dispositivo[D*i,z*].Incorrect and ANi.Dispositivo[D*i,z*] I do not criticize	Introduces an expression (a conversation identifier) which is used to identify the ongoing sequence of communicative acts that together form a conversation	

III. Communications Diagram

Next they are the diagram of transition of states, of the Agent Node

In the figure 2 that is developed in diagram of transition of states, one is, Agent Node (AN), its operation and interchange of messages, as well as the variables that take part in the passage of their internal communication, in addition to concexion with one of the agents of the SMA, the agent system (AS), which as well in its internal states of communication and also, handled its communication with the other agent of the SMATF, the Agent Task.

In this Figure the structure of the Agent can be observed Node, which if it detects an error, activates, the 5 Phases of Tolerance to Failures, and in each one of them, the interchange of messages.

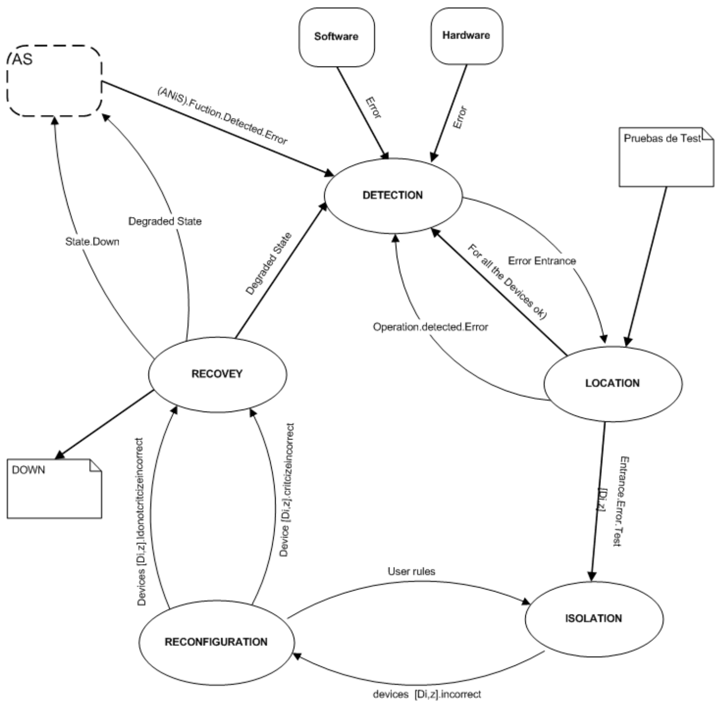

Figure 2: Several basic communication protocols are supported in KQML

X Conclusion

The agent counts on a AID, which is "intelligent Agents as a new paradigm of Distributed Fault tolerant Systems for industrial control" to as Architecture of Reference fipa/Data minimum of an agent is specified in the norms of Fipa (, says: Aid- the agent must have a unique name globally).
The agent contains descriptions of transport in the development of his documentation, which fulfills the specifications of fipa (Architecture of Reference fipa/Data minimum of an agent, says: Localizer one or but descriptions of the transport that as well, contains the type of transport by ej. Protocol), but does not specify the protocol that uses like type of transport, this this in phase of analysis.
It concerns the communication and cooperation between agents, the document "intelligent Agents as New Paradigm of Distributed Fault tolerant Systems for Industrial Control" says to us that the communication between the agents occurs of ascending or descendent form depending on the type of agent. A a little superficial explanation occurs, without specifying for example that type of language of communication between agents uses, or KQML or the Fipa-acl.

XI Considerations

We described in this paper our approach for building multi-agents system for achieving fault tolerant control system in industry. The use of the paradigm of intelligent agents has enabled the profile generation of each of the possible failures in an embedded industrial system. In our approach, each of the intelligent agents is able to deal with a failure and stabilize. It is observed the models and forms to make the communication between the agents' efficient using tools of efficient handling. The system in an independent way, and that the system has a behavior that is transparent for the use application as well as for the user.

References

[1]. Stuart Russell and Peter Norvig, Artificial Intelligence to Modern Aproach, Pretence artificial Hall series in intelligence, Chapter Intelligent Agent, pages. 31-52.
[2]. A.Alanis, Of Architectures for Systems Multi-Agentes, (Master Degree thesis in computer sciences), Tijuana Institute of Technology, November, 1996.
[3]. Michael J. woodridge, Nicholas R. Jennings. (Eds.), Intelligence Agents, Artificial Lecture Notes in 890 Subseries of Lectures Notes in Computer Science, Amsterdam, Ecai-94 Workshop on Agent Theories, Architectures, and languages, The Netherland, Agust 1994 Proceedings, ed. Springer-Verlag, págs. 2-21.
[4]. P.R. Cohen ET al.An Open Agent Architecture, working Notes of the AAAI Spring symp.: Software Agent, AAAI Press, Cambridge, Mass., 1994 págs. 1-8.
[5]. Bratko I. Prolog for Programming Artificial Intelligence, Reding, Ma. Addison-Wesley, 1986.
[6]. Or Etzioni, N. Lesh, and R. Segal Bulding for Softbots UNIX? (preliminary report). Tech. Report 93-09-01. Univ. of Washington, Seattle, 1993.
[7]. Elaine Rich, Kevin Knight, Artificial intelligence, SecondEdition, Ed. Mc Graw-Hill, págs. 476-478.
N. Jennings, M. Wooldridge: Intelligent agents: Theory and practice. The Knowledge Engineering Review 10, 2 (1995), 115– [10] Durfee et al. 89
[8]. E. H. Durfee, V. R. Lesser, D. D. Corkill: Trends in cooperative distributed problem solving. IEEE Transactions on Knowledge and Data Engineering KDE-1, 1(March 1989), 63–83.
[9]. http://www.fipa.org/specifications/lifecycle.html
[10]. http://www.fipa.org/specifications/identifiers.html
[11].http://www.fipa.org/specifications/index.html
[12]. M. Yokoo, T. Ishida, K. Kuwabara: Distributed constraint satis-faction for DAI problems. In Proceedings of the 1990 Distributed AI Workshop (Bandara, TX, Oct. 1990).
[13]. J. Weizenbaum: ELIZA – a computer program for the study of natural language communication between man and machine. Communications of the Association for Computing Machinery 9, 1(Jan. 1965), 36–45.
[14]. T. Winograd: A procedural model of language understanding. In Computer Models of Thought and Language, R.Schank and K. Colby, Eds. W.H.Freeman, New York, 1973, pp. 152–186.
[15] FIPA Abstract Architecture Specification. Foundation for Intelligent Physical Agents, 2000. http://www.fipa.org/specs/fipa00001/
[16] FIPA Interaction Protocol Library Specification. Foundation for Intelligent Physical Agents, 2000. http://www.fipa.org/specs/fipa00025/
[17] External Interfaces Working Group ARPA Knowledge Sharing Initiative. Specification of the KQML agent-communication language Working .
[18] Yannis Labrou and Tim Finin. A semantics approach for KQML
{ a general purpose communication language for software agents. In Third International Conference on Information and Knowledge Management, November 1994.
[19] Tim Finin, Don McKay, Rich Fritzson, and Robin McEntire. KQML: an information and knowledge exchange protocol. In International Conference on Building and Sharing of Very Large-Scale Knowledge Bases, December 1993.(Ed.), "Knowledge Building and Knowledge Sharing", Ohmsha and IOS Press, 1994.

Arnulfo Alanis Garza, Bachelor in Computer Engineering Systems from Technological Institute of San Luis Potosi. Candidate to the Phd Degree in Computer Science from Polytechinical University of Valencia,. Full time Researcher of Tijuana Institute of Technology Since 1995. Current areas of interest include: Intelligent Agents, Expert System, Robotics and Networks, Fault Tolerance System. Published more 20 papers in conference proceedings and journals.

Juan Jose Serrano, , Bachelor in Industrial Engineering from Polytechinical University of Valencia.. Phd Degree in Computer Science form Polytechinical University of Valencia, Full time Researcher of Polytechinical University of Valencia since 1999. Current areas of interest include: Intelligent Fault Tolerance System, Real-time systems, microcontroller.Published more 40 papers in conference proceedings and journals.
.

Rafael Ors Carot, Bachelor in Industrial Engineering from Polytechinical University of Valencia.. Phd Degree in Computer Science form Polytechinical University of Valencia. Full time Researcher of Polytechinical University of Valencia since 1999. Current areas of interest include: Intelligent Faul Tolerance System, Real time systems, micro-controller. Published more 40 papers in conference proceedings and journals.

Jose Mario Garcia Valdez, Bachelor in Computer Engineering System from Technological Institute of Tijuana. Full time Researcher of Tijuana Institute of Technology Since 1994. Current areas of interest include: data base, objects of learning. Published more 10 papers in conference proceedings and journals.