A Flexible Approach to Exception Handling in Open Multi-agent Systems

Nazaraf H Shah, Kuo-Ming Chao, Rachid Anane and Nick Godwin
School of Mathematical and Information Sciences
Coventry University, Coventry UK
{n.shah, k.chao, r.anane, a.n.godwin}@coventry.ac.uk

ABSTRACT

Exception handling in multi-agent systems (MAS) is a complex issue due to distributed and decentralized nature of data and control in such systems. Autonomous agents representing different organizations need to implement a set of behaviors in addition to their problem solving behaviors in order to facilitate the coordinated exception handling processes across organizational boundaries. In this paper we review the limitations of the current domain independent exception handling approaches and propose a flexible approach to exception handling in MAS. The approach works by incorporating the exception handling services provided by the MAS infrastructure owner and local exception handling mechanism of individual agents. It allows the agent to use system provided exception handling service for all of its protocol related (social) exceptions or use its local exception handling mechanism to deal with a subset of these exceptions. It is claimed that this provides flexibility in exception handling. This work provides way of how to increase the reliability and availability of open agent system in exceptional situations.

Keywords

Exception Handling, Robustness, Multi-agent Systems.

1. Introduction

Agents are currently one of the most interesting and popular research topics in distributed AI and other fields of research. They play an important role in the search for solutions to realistic computational problems characterised by incomplete information and autonomy in a dynamic and distributed environment. Open systems represent one of the most important application areas for multi-agent systems [1].

Open MAS evolve dynamically from the interaction of independently developed agents. These independently developed agents cannot always be trusted to be benevolent unlike agents in closed systems. The members of such communities are often unrelated may never met and have no information about each other's reputation [2]. Agents in such environment are prone to different type of failure such as missed deadline, communication channel failure agent deviating from their contracted behaviours intentionally or unintentionally. In order to address these issues there is a need to provide effective runtime exception detection and resolution services. These services monitor the communication between agents and detect exceptions by discovering violations of underlying assumptions of the coordination protocol being used. These services can be implemented as a set of agents called sentinel agents. The purpose of sentinel agent is to maintain the smooth functioning of the system in presence of exceptional situations. The exceptions in a MAS can be broadly divided into two classes known as environmental level and social level exceptions. Social level exceptions result from violations of explicit or implicit assumptions of the coordination protocol during the interaction process. Environmental exceptions are addressed at symbolic level using ordinary software exception handling techniques and individual agents are responsible for dealing with them. The majority of social level exceptions are context dependent. They require the involvement of multiple agents in a coordinated manner during detection and recovery process. The main focus in this research is to provide an architecture/framework to enable flexible exception handling in open multi-agent systems. It relies on the use of a subset of exception classification done by Klein [3] as target exceptions to be addressed.

The proposed approach addresses the handling of agent commitment violation exceptions known as social level exceptions and also allows agents to use their local exception handling techniques at the same time. Agents can employ exception handling service for individual contract not necessarily for the whole duration of their stay in MAS, provided these exceptions do not pose any threat to other agents in the system. This is useful in situations when agents need exception handling service to address a certain type of exception for a given contract at any point in time.

2. Agent and Object

In this section objects and agents are put into perspective. An object is defined as: **"**An object has state, behavior, and identity; the structure and behavior of similar objects are defined in their common class" [4]. There is no precise definition of an agent that is guaranteed to satisfy everyone. However the following definition is much more comprehensive and less controversial. "An agent is an encapsulated computer system that is situated in some environment, and is capable of flexible and autonomous actions in that environment in order to meet its design objectives" [5]. Agent oriented (AO) technology extends object oriented (OO) technology by introducing the notion of flexibility and autonomy in agent behavior. Autonomy means that agents have control over both their internal state and behavior. Flexibility means that agents are both reactive(able to respond in a timely fashion to changes that occur in their environment) and proactive(able to act in anticipation of future goal)[5]. There are number of similarities between agent oriented and object oriented technology e.g. modularity information hiding etc. But there are also essential differences between two approaches.

- Agents are conceived in term of goals and actions i.e. at knowledge level [6]. Agent interactions take place at knowledge level [7]. Whereas objects interactions are primitive and tied to symbolic level. Interactions in object oriented paradigm are simply method invocation with exact format of implementation language. In contrast interactions in AO paradigm take place using agent communication language

(ACL) [8,9] and their message contents are richer than those used in OO paradigm.

- At knowledge level agents are asocial and at social level there is organization or group of agents [10]. This organization also has influence on the behavior of its group members. There no such notion of socialability in object oriented technology. Distributed objects systems can be thought as a counterpart of multi-agent systems, composed of a monolithic entity of tightly coupled objects with hard-coded method calls on known objects. Such systems lack the notion of dynamic evolution of computational organizations, which are at the heart open multi-agent systems.

This qualitative paradigm shift from OO to AO requires exception handling mechanisms for agent systems to accommodate the above mentioned issues.

3. Exception Handling in Agent Systems

Exception handling in OO technology is a well addressed area as compared to agent oriented technology [11, 12, 13]. In OO technology exceptions are handled at symbolic level and they do not consider issues such as autonomy and sociality. In such systems objects are considered as fully trusted and cooperative, which may not be the case in open agent systems. The exception handling techniques provided by OO technology can be used in order to deal with environmental exceptions in an agent's individual plan, but they are unable to deal with exceptions at social level. At social level we expect to deal with systemic dysfunction and agents with limited trust etc. The exception handling mechanisms for agent systems need to address the exceptional situations that arise at social level. Although agents are modular software components, they provide a fundamental requirement for building fault tolerant systems [14]. At the same time they are autonomous and hence they are non-deterministic, i.e. their behaviour is not control by an external entity.

Agents employ the black box control as compared to white box control used in object method invocation. In exceptional situations, it is not possible to know exactly what went wrong inside the agents representing different organization. The necessary information required to handle exceptional situations can only be obtained by requesting the affected agent. So the problem solving agents are required to implement a basic standard interface to provide the necessary information in exceptional situations. In the absence of such interface cooperative recovery in open system is not possible.

We assume that low level exceptions that may arise in an individual plan are dealt with using constructs provided by agent architecture [15, 16]. Theses constructs have similar semantic as used in Java programming language [17]. Their scope is limited to a plan where an exceptional situation arises and they do not take any social context into considerations.

Extant exception handling approaches to MAS can be broadly divided into two classes discussed below, individualistic and social/shared exception handling approaches.

3.1 Individualistic Approach

The earlier multi-agent systems adopted the exception handling approaches employed in traditional software systems. The primary focus of these approaches was individual agent rather than a MAS. This approach focuses on individual agent exception handling mechanism without taking into account the organization of the system. It is extremely difficult to use this approach to address exceptions that require global view of the system. This approach is not suitable for open agent systems where different parties are in control of different subsystems. Open MAS evolve dynamically from interaction of independently developed agents. These independently developed agents cannot be trusted to be benevolent unlike agents in closed systems. Open MAS certainly require exception handling services capable of dealing with exceptions while treating constituent agents as black boxes. Without effective runtime exception handling services in open MAS, it is doubtful that agent oriented system will be able to realize their full potential. The limitations of individualistic approach or survivalist approach are addressed in [18,19].

3.2 Shared Exception Handling Approach

Shared exception handling service consists of a set of specialized agents and is provided by the multi-agent system ensemble for all of its constituent agents. These dedicated exception handling agents are known as sentinel agents. The sentinel agents monitor communication between agents and build a model of the commitments their agents are involved in. They discover exceptions by using information from commitment model and underlying assumptions of coordination protocol or system operation strategies. They take part in exception handling according to given guideline by obtaining state information and issuing directives to affected agents.

3.2.1 Sentinel Approach

Sentinel approach in MAS was first proposed by Stafan Hägg [18]. It uses the sentinel agents to ensure the robust functioning of the system. "A sentinel is an agent, and its mission is to guard specific function or to guard against a specific state in the society of agents" [18]. Sentinel agents build commitment models of their associated agents by monitoring their communication. They only intervene on detection of violations of system operational guideline. This approach is based on graceful degradation of the system and works by excluding faulty components selecting alternatives or by reporting to human operator.

3.2.2 Domain Independent(Citizen) Approach

Domain independent exception handling approach focuses on exceptions that result from violation of underlying assumptions of coordination mechanism [19]. It assumes that lower level exceptions can be dealt by using available exception handling techniques. It addresses the social level exceptions only. It is worth mentioning that any exception that is not dealt at lower level ultimately propagates to social level. This results in agent not being able to honour its commitment. This approach is mainly concerned with the exceptions that result from violations of coordination protocol assumptions rather than domain related exceptions. It also makes use of sentinel agents in order to deal with exceptions. It does not allow agents in a given MAS to use their local exception handling mechanisms in order to deal with any of protocol related exception.

4. Proposed Contract Based Approach

Although Klein et al [19] provide a useful technique for dealing with exceptions in open MAS, but there are still many issues left that need to be addressed. These issues include how to deal with concurrent exceptions, and how to coordinate the exception handling services and survivalist agents using their own exception handling capabilities. In this paper we address the later issue. We propose a flexible contract based approach by extending domain independent exception handling approach. That allows the agent to use its own exception handling capabilities for some exceptions and hire exception handling services from the system to deal with other exceptions which it decides not to deal with itself (or not capable of dealing with them). This approach provides agents with a choice of using their own exception handling capabilities or shared exception handling services provided by the system or both. Whereas domain independent exception handling approaches rely on only sentinel agents for dealing with domain independent or social exceptions. In order to be able to take part in cooperative exception handling process problem solving agent needs to implement a set of plans called interface. Independently developed agent cannot always be expected to provide the full implementation of this interface. It is assumed that all problem solving agents implement at least a basic level of the interface that is crucial to maintain the integrity of the system such as reporting on its current state cancelling the current task etc when asked by its associated sentinel agent.

4.1 Proposed Architecture

The proposed architecture makes use of match maker agent contract notary reputation server and sentinel agents as proposed by Klein et al [19]. The structure of sentinel and registration service is however different from those proposed in [19,20]. These differences provide flexibility in exception handling process in a system where agents with different exception handling capabilities are involved. In current sentinel based domain independent exception handling approaches the sentinel agent takes control of all exception handling apart from low level exceptions. The sentinel agent leaves no choice for the agents to use their own local exception handling mechanism for some exceptions for better efficiency purposes. The proposed architecture facilitates the use of local and system exception handling at same time due to local collaboration between agents and their associated sentinels. The sentinel agents having access to Cooperative Interface, Local Contract DB, and Protocol Exception Handling Capability make informed decision regarding exception handling at runtime. It also allows agents to use their local exception handling mechanism [23] for some exceptions provided all involved agents agree on using local exception handling mechanism for a particular transaction.

The figure 1 shows two agent A and B that have joined a MAS. Two sentinel agents Sentinel A and Sentinel B are assigned to them respectively by the MAS. All communication between the two agents takes place via their associated sentinels. This enables the sentinel agents to intercept protocol violations during the interaction between agents. The proposed architecture is shown is figure 1. The functionality of the each component is briefly described below.

4.1.1 Registration Service Agent

Registration Service Agent (RSA) allows agents to join a MAS after querying them about their coordination protocol and the level of cooperative exception handling interface implemented by them. This service differs from one proposed by Klein et al [19] because it allows RSA to obtain the information about cooperative exception handling behaviour of the agent along with its problem capabilities.

4.1.2 Sentinel Agent

The sentinel agents play an important role in exception handling process in a MAS where agents belonging to different independent

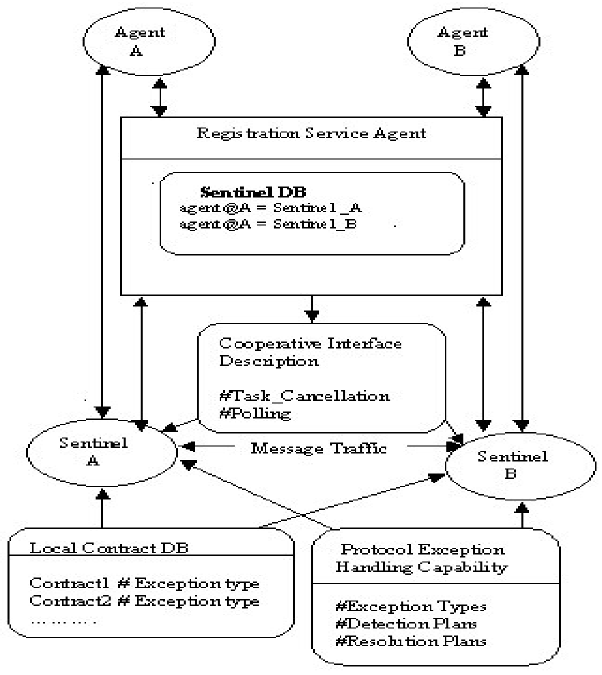

Figure 1: Proposed Architecture

organizations interact with one another in order to achieve their goals. All communication between agents takes place via sentinels. As shown in figure Agent “A” and agent “B” communicate via their associated sentinel agents. Each sentinel agent consists of the following components.

4.1.2.1 Protocol Exception Handling Capability

It contains the characteristic exceptions of the protocol being used by the agents in a MAS and also plans library for detecting and resolving exceptions at runtime. This component is built by identifying the characteristic exceptions of a coordination protocol using Role Commitment Violation Technique [21] and providing implementation of related exceptions detection and recovery plans.

4.1.2.2 Cooperative Interface Description

This database contains the information about a set of exceptions for which the agent is capable of cooperating with sentinel agent during exceptional situations. This information is concerned with the cooperative exception handling interface implemented by the individual agent. RSA populate a private instance of this database for every sentinel by retrieving information from agent when it joins a given MAS. The sentinel agent having access to this database is able to know exactly the level of cooperation that can be obtained from problem solving agent when an exception arises during runtime.

4.1.2.3 Local Contract Database

This database contains the information about specific contract for which agent does not want sentinel agent to intervene on detection of specified exceptions during life of a single contract. The problem solving agent can specify at runtime the particular exceptions for a particular contract be left for problem solving agent to be handled. This can be applied in situations where contractor agents involve in frequent transactions with trusted subcontractor agents. When contract is completed successfully or terminated by the agents abnormally the sentinel agents removes the contracts and its associated exceptions record from database.

5. Conclusion

In a multi-agent system where agents represent different autonomous organizations exception handling process consists of cooperative actions to be taken by responsible agents. This involves the cross organizational cooperation for recovery from an exceptional situation. The decision about the recovery process is context dependent and is taken at runtime. In order to operate flexibly and robustly MAS require runtime monitoring and exception handling mechanisms. In multi-agent system each agent has limited information and limited viewpoint [22]. Using this individualistic and limited information it is not possible for individual agents to handle exceptions, that require information and cooperation from other agents. Runtime exception handling services become necessary to facilitate the exceptions detection and recovery in such situations. These runtime services require problem solving agents to implement a capability to provide the support to exception handling services by providing state information and by acting upon directives received from exception handling service. In open agent system agent with varying degree of cooperative exception handling support capability can join a MAS or some agents may prefer to use local exception handling mechanism for some kind of exceptions. This requires the flexibility in exception handling service in order to address these issues in open multi-agent systems. None of the current exception handling approach allows the agents to use their individual exception handling mechanisms for some social exceptions and employ sentinel agents to deal with the others. In these approaches sentinel agents take the control of dealing with all social exceptions making agent's local social exceptions handling mechanism unusable.

Having pointed the issues involved in exception handling and proposing the architecture for sentinel agents and RSA, our subsequent work aims to address the following issue.

- Classify the characteristic exceptions of a set of MAS coordination protocols by divided them into two groups. Those that can be handled automatically by agents and those that require human intervention.
- Elaborate and implement the sentinel agents and RSA agents according to proposed architecture.

Evaluate the effectiveness of the mechanism by applying it to operational MAS. Compare the results with citizen exception handling approach. Find the trade offs between various exceptions to be handled locally and by the system provided exception handling service.

6. References

[1] Wooldridge, M. and Jennings, N.R. A Methodology for Agent Oriented Analysis and Design: Proc. 3rd Int Conference on Autonomous Agents (Agents-99), 69-76.

[2] Zacharia Giorgos, Moukas Alexanderos, Maes Pattie. Collaborative Reputation Mechanisms in Electronic: Hawaii International conference on System Marketplaces, Proceeding of the 32nd Sciences (1999).

[3] Klein, M. Exception Repository: http://ccs.mit.edu/klein/.

[4] Booch Grady (eds.). Object Oriented Analysis and Design with Applications Second Edition (1994).

[5] Wooldridge M., Jennings, N. R. Intelligent Agent: Theory and Practice: Knowledge Engineering Review,10(2), (1995), 115-152.

[6] Newell, A. The Knowledge Level: Artificial Intelligence, 18(1982), 87-127.

[7] Gaspari ,Mauro. Concurrency and Knowledge Level Communication in Agent Languages: Artificial Intelligence 105, (1998), 1-45.

[8] Labrou Yannis, Finin Tim. A Proposal for a new KQML Specification:. TR CS- 97-03, Computer Science and Electrical Engineering Department, University of Maryland Baltimore County, Baltimore, (1998) 7-38.

[9] Foundation for Intelligent Physical Agents, FIPA Communication Act Library, http://www.fipa.org/specs/fipa00037/ .

[10] Jenninngs N.R. On Agent Oriented Software Engineering: Artificial Intelligence 117(2000), 277-296.

[11] Romanovsky Alexander, Kienzle Jorg. Action-Oriented Exception Handling in Cooperative and Competitive Object Oriented Systems: Advances in Exception Handling Techniques, LNCS-2022, (2001), 147-163.

[12] Xu Jie. Romanovsky, Alexander. Randell Brian. Concurrent Exception Handling and Resolution is Distributed Object Systems: IEEE Transaction on Parallel and Distributed Systems, VOL. 11, NO. 10, (2000), 1019-1032

[13] Cambell H. Roy. Randell, Brian. Error Recovery in Asynchronous Systems: IEEE Transaction on Software Engineering. Vol. Se-12, no. 8 (1986), 811-826.

[14] A.F. Garcia, C. M. F. Rubira, A. Romanovsky, J. Xu. A Comparative Study of Exception Handling Mechanisms for Building Dependable Object Oriented Software: Journal of Systems and Software. 59(2001), 197-222.

[15] JAM Agents in a Nutshell. Intelligent Reasoning Systems. http://www.marcush.net/IRS/index.html

[16] JACK Intelligent Agent: Agent Oriented Software. http://www.agent-software.com/shared/home/index.html

[17] Arnold, K., Gosling, J. (eds.) The Java Programming Language: Second Edition , Addison Wesley 1998.

[18] Hägg, Staffan. A Sentinel Approach to Fault Handling in Multi-Agent Systems: Proc of Second Australian Workshop on Distributed AI Carnis Australia, Verlog-Springer (1997), 181-195.

[19] Klein M., Dellarocas Chrysanthos. Exception Handling in Agent Systems: Proceedings of the Third Annual Conference on Autonomous Agents Seattle, USA (1999), 62-68.

[20] Dellarocas, C. Klein, M. Juan, Anttonio Rodriguez-Anguilar. An Exception-Handling Architecture for Open Electronic Marketplaces of Contract Net Software Agents: Proceedings of the Second ACM Conference on Electronic Commerce Minneapolis Minnesota USA, (2000), 225-232.

[21] Klein, M. Using Role Commitment Violation Analysis to Identify Exceptions in Open Multi-Agent Systems: ASES Working Paper ASES-WP-2000-04. Cambridge MA USA, Massachusetts Institute of Technology (2000).

[22] Jennings, N. R. Sycara, K. and Wooldridge, M. A Roadmap to Agent Research. and Development: Autonomous Agents and Multi-Agent Systems (1998), 7-38.

[23] Sandholm, T. and Lesser, V. Issues in Automated Negotiation and Electronic Commerce: Extending the Contract Net Framework: International Conference on Multi-Agent Systems, San Francisco (1995), 328-335.