Domain-Independent Exception Handling Services That Increase Robustness in Open Multi-Agent Systems

Mark Klein and Chrysanthos Dellarocas

CCS WP #211 SWP # 4115

May 2000

Domain-Independent Exception Handling Services That Increase Robustness in Open Multi-Agent Systems[1]

MARK KLEIN, PHD
Center for Coordination Science
Massachusetts Institute of Technology
m_klein@mit.edu
http://ccs.mit.edu/klein/

CHRYSANTHOS DELLAROCAS, PHD
Sloan School of Management
Massachusetts Institute of Technology
dell@mit.edu
http://mit.ccs.edu/dell/

Abstract. A critical challenge to creating effective multi-agent systems is allowing them to operate effectively in environments where failures ('exceptions') can occur. This paper describes the motivation, progress and plans for work being pursued in this area by the MIT Adaptive Systems and Evolutionary Software research group (http://ccs.mit.edu/ases/).

1. The Challenge: Enabling Robust Open Multi-Agent Systems

"open systems ... represent arguably the most important application for multi-agent systems" (Wooldridge, Jennings et al. 1999)

This paper addresses one simple question: how can we develop effective multi-agent systems out of the diverse and unreliable (buggy, malicious, or simply "dumb") agents and infrastructures we can expect to encounter in open system contexts? This is becoming an increasingly critical question because of emerging changes in the way human organizations work. Globalization, enabled by ubiquitous telecommunications, has increasingly required that organizations be assembled and re-configured within small time frames, often bringing together partners that have never worked together before. Examples of this include international coalition military forces, disaster recovery operations, open electronic marketplaces and virtual supply chains (1999) (Fischer, Muller et al. 1996) (Tsvetovatyy, Gini et al. 1997).

[1] Working Paper ASES-WP-2000-02, Center for Coordination Science, Massachusetts Institute of Technology, Cambridge MA USA. February 2000 (http://ccs.mit.edu/ases).

Multi-agent systems (MAS) represent one of the most promising approaches for creating the agile information systems needed to support these kinds of applications, because of their ability to use multi-agent coordination protocols to dynamically self-organize themselves as their problems and constituent agents change (Jennings, Sycara et al. 1998) (Wooldridge, Jennings et al. 1999). A critical open challenge remains, however. The vast majority of MAS work to date has considered well-behaved agents running on reliable infrastructures in relatively simple domains (Hägg 1996). These have been almost exclusively *closed* systems, i.e. where the agents and their infrastructure are developed and enacted under centralized control. It is clear however that these assumptions do not hold for the *open* contexts described above, where agents can come from multiple sources and must operate on the infrastructures at hand (Hewitt and Jong 1982). For these contexts we can expect, in contrast, to find:

- *Unreliable Infrastructures*. In large distributed systems like the Internet, unpredictable host and communication problems can cause agents to slow down or die unexpectedly, messages to be delayed, garbled or lost, etc. These problems become worse as the applications increase in size, due to the increase in potential points of failure.

- *Non-compliant agents*. In open systems, agents are developed independently, come and go freely, and can not always be trusted to follow the rules properly due to bugs, programmer malice and so on. This can be expected to be especially prevalent and important in contexts such as electronic commerce or military operations where there may be significant incentives for fraud or malice.

- *Emergent dysfunctions*. Emerging multi-agent system applications are likely to involve complex and dynamic interactions that can lead to emergent dysfunctions, such as chaotic behavior, with the multi-agent coordination mechanisms that have proved most popular to date (Youssefmir and Huberman 1995) (Sterman 1994). This is especially true since agent societies operate in a realm where relative communication and computational costs and capabilities can be radically different from those in human society, leading to behaviors with which we have little previous experience. It has been argued, for example, that the 1987 stock crash was due in part to the action of computer-based "program traders" that were able to execute trade decisions at unprecedented speed and volume, leading to unprecedented stock market volatility (Waldrop 1987).

All of these departures from "ideal" multi-agent system behavior can be called *exceptions*, and the results of inadequate exception handling include the potential for poor performance, system shutdowns, and security vulnerabilities.

2. Our Approach: Knowledge-Based Exception Handling Services

Our group is addressing this problem directly by developing technologies to help increase the robustness of open multi-agent systems, based on two key and novel ideas:

- It is possible to identify highly reusable *domain-independent exception handling expertise* that describes the characteristic failure modes for different MAS coordination protocols, as well as how they can be handled.

- This expertise can be instantiated in the form of *exception handling services* that outsource exception handling from the agents in a multi-agent system, requiring only that they adhere to some relatively simple guidelines.

The remainder of this section will examine these two ideas in more detail, describing the key concepts, the progress we have made to date, the open research questions that remain, and how we are addressing them. In a nutshell, our work to date has shown that exception handling based on these ideas is workable and has substantial advantages over previously explored approaches. Critical areas for further research include (1) methodologies and tools to support the rapid analysis of how coordination protocols can fail and be "fixed", and (2) more sophisticated run-time services that can adaptively diagnose and resolve exceptions in complex MAS environments where many, and even simultaneous, exceptions can occur.

In order to make the ideas more concrete, we will present them in the context of the well-known MAS protocol known as the Contract Net (Smith and Davis 1978). The Contract Net (henceforth called CNET) is probably the most widely-used MAS protocol, and has been applied to many domains including manufacturing control (Baker 1988) (Sousa and Ramos 1999), tactical simulations (Boettcher, Perschbacher et al. 1987), transportation scheduling (Bouzid and Mouaddib 1998), and distributed sensing (Smith and Davis 1978). CNET operates as follows (Figure 1):

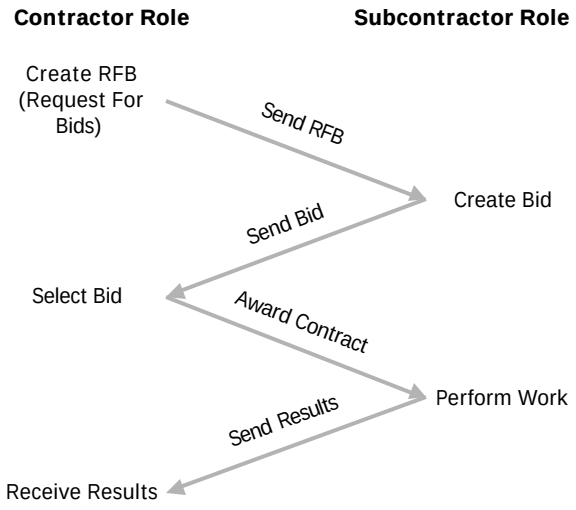

Figure 1. A simple version of the Contract Net protocol.

An agent (hereafter called the "contractor") identifies a task that it cannot or chooses not to do locally and attempts to find another agent (hereafter called the "subcontractor") to perform the task. It begins by creating a Request For Bids (RFB) which describes the desired work, and then sending it to potential subcontractors (typically identified using a matchmaker that indexes agents by the skills they claim to have). Interested subcontractors respond with bids (specifying such issues as the time needed to perform the task) from which the contractor selects a winner. The winning agent, once notified of the award, performs the work (potentially subcontracting out its own subtasks as needed) and submits the results to the contractor.

2.1. The Power of Domain-Independent Exception Handling

The first key idea underlying our work is that the characteristic exceptions and applicable exception handling techniques for a multi-agent system can be usefully treated as dependent on the coordination protocol used but *independent of the domain the agents work in*. This is a key insight because it means that we can build exception handling knowledge bases (and associated run-time services) that are generic and thereby highly reusable.

Evidence for the existence of domain-independent exception handling expertise: There is substantial evidence for the validity of this claim. Early work on expert systems development revealed that it is useful to separate domain-specific problem solving and generic control knowledge (Gruber 1989) (Barnett 1984) (Malone and Crowston 1994). Our own work has confirmed the applicability of this insight to exception handling in domains including collaborative design (Klein 1989; Klein 1991), requirements capture (Klein 1997), workflow management (Klein and Dellarocas 2000) and multi-agent systems (Klein and Dellarocas 1999) (Dellarocas and Klein 2000).

Analysis of the CNET protocol, for example, reveals that it is prone to domain-independent exceptions from all three of the categories (unreliable infrastructures, non-compliant agents, emergent dysfunctions) described above. Some examples include:

- *Agent death:* If a CNET agent dies there are several immediate consequences. If the agent is acting as a subcontractor, its customer clearly will not receive the results it is expecting. In addition, if the agent has subcontracted out one or more subtasks, these subtasks and all the sub-sub-… tasks created to achieve them become "orphaned", in the sense that there is no longer any real purpose for them and they are uselessly tying up potentially scarce subcontractor resources. Finally, if the system uses a matchmaker, it will continue to offer the now dead subcontractor as a candidate (a "false positive"), resulting in wasted message traffic.

- *Fraudulent [sub]contractor:* A buggy or intentionally malicious CNET agent can wreak havoc through fraudulent advertising, bidding or subcontracting. Imagine, for example, an agent that falsely informs the matchmaker that it has a comprehensive set of skills, sends in highly attractive but fraudulent bids (e.g. specifying it can do any task almost instantaneously) for all the RFBs it receives, and once it wins the awards returns either no results, or simply incorrect ones. The result would be that many if not all of the system's tasks would be awarded to a non-performing agent.

- *Resource poaching:* It is typical for CNET systems to annotate tasks with priorities, so that when a subcontractor is considering several RFBs, it will bid (first) for the RFB with the greatest priority. One emergent dysfunction that can occur in such contexts is "resource poaching", wherein a slew of low-priority but long-duration tasks tie up the subcontractors, thereby freezing out resources needed for the higher-priority tasks that arrive later (Chia, Neiman et al. 1998). This does not represent an error per se, but rather an unexpected consequence of the protocol when applied in a complex environment.

A wide range of domain-independent techniques ("handlers") can be used to deal with such exceptions. Some examples include:

- To detect agent death, periodically poll active subcontractors. If an agent dies, clear the agent record from the matchmaker(s), and instruct the contractors for that agent to re-run the bidding process for the failed tasks. One can cancel the orphaned sub-sub-tasks if any. Keep track of agent MTBF (mean time between failure) statistics to help avoid relying on unreliable agents for critical tasks in the future.

- To detect fraudulent agents, keep track of their performance over time and note which agents consistently fail to produce results with the contracted cost, quality and duration. To foil them, "kill" or "exile" or simply "ignore" them when they appear.

- To detect resource poaching, compare the average priority of pending tasks to the average priority of in-work tasks. To resolve it, instruct subcontractors to preempt current lower priority tasks and bid for the pending higher priority tasks. To avoid it, impose a longer waiting period before agents select an RFB, increasing the chance that higher-priority tasks will not be frozen out.

We have identified, to date, over 300 exception types and associated handlers for representative examples of the most widely-used and studied classes of MAS coordination protocols including

CNET (Smith and Davis 1978) (Aryananda 1999), multi-level coordination (Durfee and Montgomery 1990), and team-based approaches (Tambe 1997) (Xu 1999).

A systematic methodology for identifying exceptions: Given that such domain-independent exception handling expertise exists, the key question becomes: How can we quickly (and ideally automatically) identify the characteristic exceptions, as well as the appropriate handlers, for the coordination protocol(s) being used in a particular MAS? We have developed the beginnings of a systematic methodology for identifying characteristic exceptions, as well as a repository-based approach for organizing knowledge about exceptions and their handlers so as to facilitate its rapid retrieval.

The exceptions that characterize a given MAS protocol can be uncovered using an emerging technique we call Role Commitment Violation (RCV) analysis. RCV analysis is based on the insight that coordination fundamentally involves the process of agents making commitments to each other (Singh 1999) (Jennings 1996) (Gasser 1992). Exceptions can thus be viewed as the ways in which the components of a MAS can fail to achieve their commitments. RCV works, more specifically, as follows:

- Identify all the *roles* involved in the multi-agent system. This generally includes a wider range of roles than typically considered when describing a MAS protocol, including for example the computational and communications infrastructure.

- For each role, identify the *commitments* that each role *ideally* (and implicitly or explicitly) requires of the other roles in the system in order to meet its own commitments. Identifying 'idealized' commitments allows us to better identify exceptions that result because one role makes assumptions that another role is not guaranteed to satisfy.

- For each commitment, identify the ways the commitment may be *violated* (i.e. identify the exceptions). Since each agent in an open system is in principle a 'black box', this analysis must be done based on general principles rather than on an understanding of precisely how each role is implemented by the agent.

Let us make this more concrete but looking at several simple examples of the RCV analysis for CNET. There are three classes of commitment types, corresponding to the three classes of exceptions identified above. One type is infrastructural commitments, i.e. the implicit commitments made by the infrastructure to provide reliable agent hosts and communication links. Every agent, for example, relies upon the messaging infrastructure to send the right messages to the right place at the right time. We can thereby infer that message sending is prone to three generic exceptions: wrong thing (the message is garbled), wrong place (the message is dropped or misrouted), and wrong time (the message is excessively delayed, or perhaps even comes unexpectedly fast). A second class of commitment is that made between agents as part of their coordination protocol. Our analysis has identified 17 such commitments in CNET, including "correctly inform matchmaker about skill changes in a timely way" (a commitment to the matchmaker), and "send only legitimate RFBs" (a commitment to a subcontractor). These commitments have associated exceptions such as "agent fails to inform matchmaker about capability loss" (e.g. if the agent dies), and "contractor sends invalid RFB" (e.g. if the customer for the contractor cancelled its request and thereby made the RFB obsolete). Finally we have the commitments made by the

system manager to ensure overall effective system operation e.g. by maintaining the appropriate mix of subcontractor skills.

RCV can be viewed, we believe, as a useful complement to traditional failure identification techniques (known collectively as failure mode effects analysis **(Hunt, Pugh et al. 1995)**) and appears to be particularly well-suited to open systems exception analysis, because it treats components as black boxes and focuses on the failure modes for their inter-relationships rather than the failure modes for the components themselves. Using RCV, we believe that we have exhaustively (or nearly so) identified all the exception types for the CNET variant described above. In any case, our analysis has uncovered all the exceptions revealed by an exhaustive review of the CNET literature, as well as many others.

Once we have identified the exceptions that characterize a given MAS protocol we need to uncover the handlers that are appropriate for dealing with them. Unlike exceptions, the range of possible handlers is not a closed easily enumerable set but seems to be limited only by human ingenuity. Exception handling techniques have been developed in many disciplines including multi-agent systems (Hägg 1996) (Kaminka and Tambe 1998) (Tambe 1997) (Horling, Lesser et al. 1999) (Venkatraman and Singh 1999) (Bansal, Ramohanarao et al. 1997), distributed and real-time systems (Burns and Wellings 1996) (Mullender 1993), planning and robotics (Traverso, Spalazzi et al. 1996) (Howe 1995) (Birnbaum, Collins et al. 1990) (Broverman and Croft 1987) (Firby 1987) (Hindriks, de Boer et al. 1998), computer-supported cooperative work (Mi and Scacchi 1993) (Chiu, Karlapalem et al. 1997) (Klein 1998) (Auramaki and Leppanen 1989) (Finkelstein, Gabbay et al. 1994), operations research (Fletcher and Misbah 1999) (Adamides and Bonvin 1993) (Katz 1993), and management science (`Milgrom and Roberts` 1992) (Loomis 1979) to mention just a few. Our task, therefore, has been to mine these literatures, and abstract out the key ideas as handlers appropriate for the MAS context. We have, to date, reviewed over 60 research publications for this purpose.

A knowledge base for representing exception handling expertise: We have developed a simple but effective schema for capturing this knowledge in a way that facilitates quickly finding the appropriate handlers for given protocols, based on an extension of concepts developed by the MIT Process Handbook project (Klein 2000) (Malone, Crowston et al. 1998). The scheme is based on three interlinked taxonomies representing coordination protocols (AKA mechanisms), their characteristic exceptions, and the handlers appropriate for dealing with them:

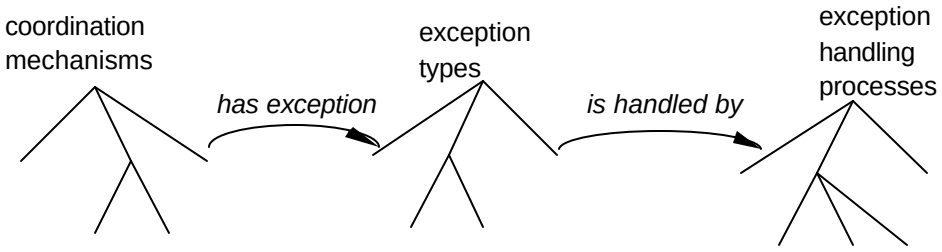

Figure 2. Overview of the schema for exception handling expertise.

The first taxonomy captures MAS coordination protocols, arranged in an abstraction hierarchy with abstract protocol classes on the left and more specific ones on the right:

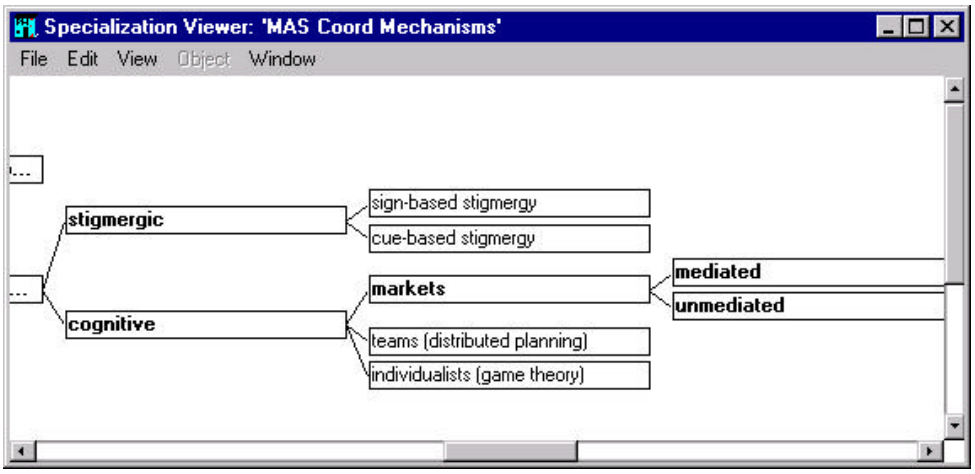

Figure 3. A part of the coordination mechanism taxonomy.

Each protocol has pointers to the exceptions that characterize it, themselves stored in an exception taxonomy:

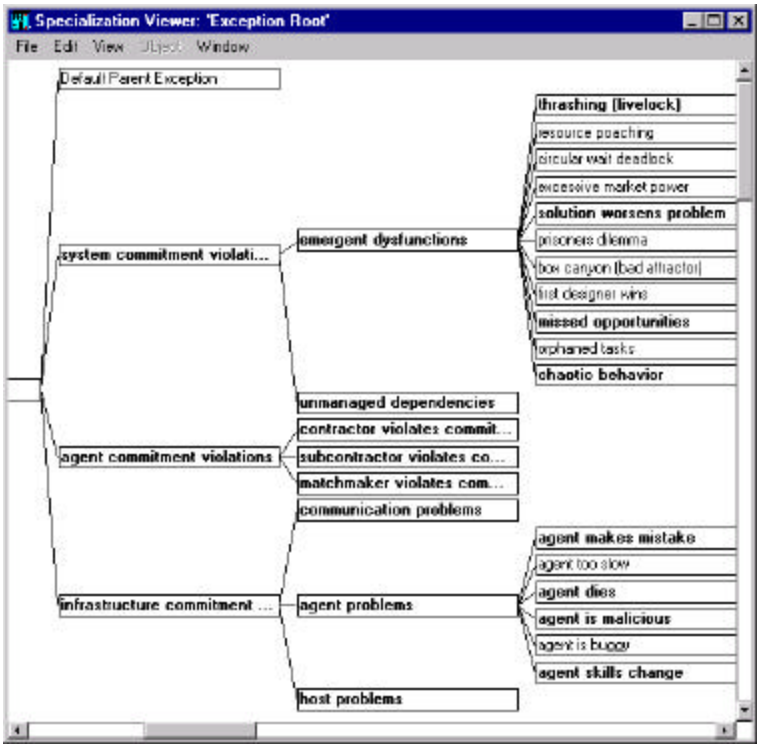

Figure 4. A part of the exception type taxonomy.

Exceptions are themselves linked, finally, to the potential applicable handlers in an exception handler taxonomy. We have found that there are four main types of handlers; those suitable for

anticipating and *avoiding* exceptions (before they occur), or *detecting* and *resolving* them (after they occur):

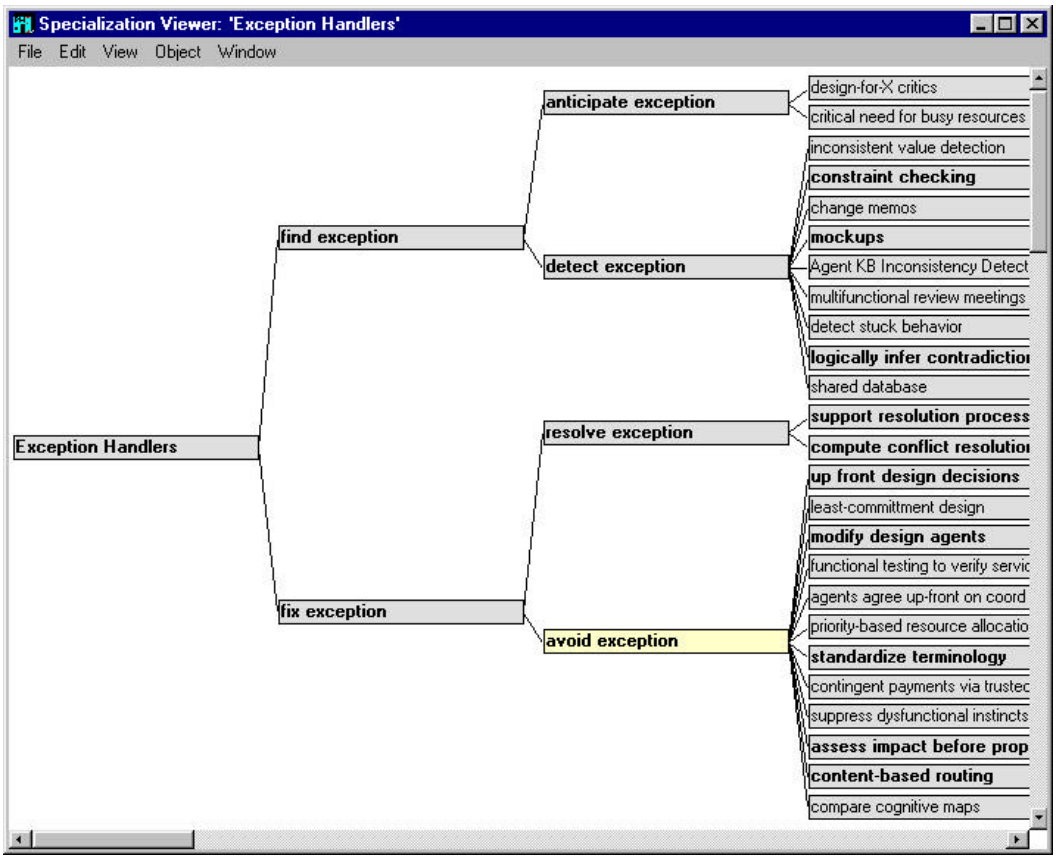

Figure 5. A part of the exception handler taxonomy.

The power of this approach is that we can use inheritance to simplify identifying the exceptions and handlers for a given MAS protocol. Some exceptions are characteristic of a whole class of protocols, and therefore are potentially inherited by all of its subclasses. Any protocol, for example, that implements 'pull-based resource sharing' (where resources are allocated by selecting among requests) potentially faces the problem of resource poaching. Similarly, a handler suitable for an abstract exception type is likely to be suitable for subclasses of that type. Preemptive re-allocation of resources, for example, is a reasonable potential candidate for any subclass of the resource poaching exception. Finally, if a handler is suitable for an exception, it is likely that subclasses of that handler will be useful for that exception as well.

Open issues: While we have made substantial progress in terms of being able to rapidly identify the exceptions and handlers for some widely-used MAS protocols, important issues remain to be addressed:

- How can we simplify exception analysis? The RCV technique described above currently requires careful human judgement in order to successfully identify all exceptions for a protocol. Is there a way to make this procedure easier and guarantee completeness of exception identification? One option is to extend the taxonomic schema describe above to

exploit inheritance to identify role commitment violations. Another is to develop techniques to automatically identify exceptions based on a formal commitment-based protocol representation such as that proposed by Singh et al (Venkatraman and Singh 1999).

- What is the space of critical exceptions and handlers for the most important MAS protocols? We have focused so far on infrastructural exceptions in CNET and a small handful of other protocols. It would be useful to get a more comprehensive view of what exception types are of greatest practical relevance, scope out the major classes of useful handlers, understand what their tradeoffs are, and identify gaps that should be addressed by future research.

2.2. An Architecture for Exception Handling Services

Weaknesses with current 'survivalist' approaches: One can imagine at least two possible approaches to developing more robust multi-agent systems. One, which we can call the "survivalist" approach, involves using our knowledge of exceptions and their handlers to design more sophisticated agents exploiting more complex coordination protocols to handle exceptions. Most MAS research has in fact taken this direction. Even the original CNET protocol (Smith and Davis 1978) included such augmentations as an "immediate response bid', which allowed a contractor to determine whether the lack of bids was due to all eligible subcontractors being busy (in which case a retry is appropriate) or due to the outright lack of appropriate subcontractors. This "survivalist" approach to multi-agent exception handling faces, however, a number of serious shortcomings.

First of all, it places a heavy burden on agent developers, who must implement potentially complicated exception handling behaviors capable of dealing with all the exceptions the agent might possibly encounter; this is difficult at best. Agents become harder to maintain, understand and reuse because a potentially large body of exception handling code obscures the relatively simple normative behavior of an agent.

This approach can, in addition, result in poor exception handling performance. Finding the appropriate responses to some kinds of exceptions (notably emergent exceptions such as resource poaching) requires for example that the agents achieve a more or less global view of the multi-agent system state, which is notoriously difficult to establish among multiple agents without heavy bandwidth requirements.

Most seriously, however, this approach is unrealistic for open agent contexts. Some kinds of exception handling require carefully coordinated behaviors across multiple agents. Agents may not comply properly with the more sophisticated protocols, or violate some of their underlying assumptions. Some exception avoidance approaches (e.g. (Sandholm, Sikka et al. 1999)) assume, for example, that all agents are rational utility maximizers, which obviously may not always be the case. Some kinds of interventions (such as "killing" a malicious agent) may be difficult to realize because the agents do not have the legitimacy needed to apply such interventions to their peers.

Our 'citizen' approach: For these reasons, our approach has been to develop instead a set of services that *offload* the exception handling burden from agents. We call this the "citizen" approach by analogy to the way exceptions are handled in human society. In such contexts, agents

adopt relatively simple and optimistic rules of behavior and rely on a whole host of institutions (the police, the Better Business Bureau, Security and Exchange Commission, and so on) to handle most exceptions. This is generally a good tradeoff because such institutions are able, by virtue of specialized expertise, widely accepted legitimacy and economies of scale, to deal with exceptions more effectively and efficiently than individual citizens, while making relatively few demands of most agents (e.g. pay your taxes, obey police officers, report crimes) and being able to deal with the subset of agents that do not cooperate. This same approach, we believe, has directly analogous benefits in the MAS context.

Our approach instantiates these ideas in an open MAS setting using the following functional architecture:

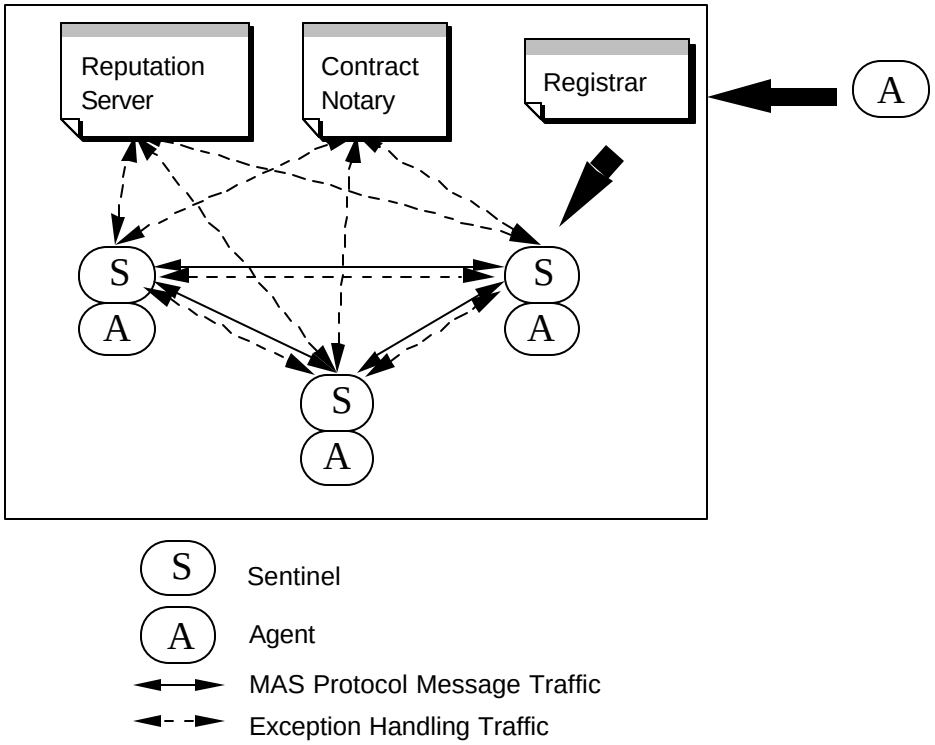

Figure 6. Functional architecture for open MAS with exception handling services.

When an agent joins an open MAS served by the exception handling (EH) services, it must register with a *registrar* responsible for assigning it a *sentinel* that will mediate all of the agents' further interactions with other agents in the system. The agents so 'wrapped' can include problem solving agents as well as components such as matchmakers that support the protocols they enact.

Sentinels are the central element in this approach. They can be viewed as "commitment monitors" whose role is to observe and influence agent behavior as necessary to ensure the robust functioning of the system as a whole. Each sentinel includes a repository of domain-independent EH expertise that describes the characteristic exceptions and associated handlers for the protocol(s) enacted by the agents in that MAS. Sentinels monitor message traffic to develop a model of the commitments their agent(s) are involved in, use the appropriate anticipation and/or detection handlers to uncover when these commitments are violated, diagnose the underlying

causes to identify the appropriate avoidance and/or resolution handlers, and enact these handlers to help re-establish the violated commitments, or at least minimize the impact of them having been violated. Ancillary services such as the contract notary and reputation server keep track of global state information such as commitment structures and reliability statistics. Agents, for their part, must be able to respond appropriately to a relatively small set of EH directives to support the action of the sentinels.

Let us consider the example of handling the 'agent death' exception in CNET. Sentinels can use the handlers described above to detect and resolve this exception as follows:

- Whenever a contractor sends an award message to an subcontractor, the subcontractor's sentinel informs the contract notary about the commitment and ensures (by periodic polling) that the subcontractor is still functioning.

- If the subcontractor dies, its sentinel then instructs the matchmaker to remove the dead subcontractor from its database, and directs the contractor to re-start the bidding process for the task previously allocated to the deceased agent. The sentinel also queries the contract notary to see if the dead subcontractor had awarded any subtasks to other CNET agents; if so, these agents are instructed to cancel these 'orphaned' tasks. Finally, the sentinel informs the reputation server of the subcontractors' death in order to update the reliability statistics for that agent in case it should re-join the MAS at some later time.

Sentinels can implement the agent death anticipation and avoidance handlers as follows:

- When a contractor agent sends out an RFB, its sentinel makes a note of the deadline for bids for that RFB.

- When a subcontractor agent sends a bid for the RFB, its sentinel automatically annotates this outgoing message with the agents' reliability (received from the reputation server when the sentinel was assigned to that agent).

- When the contractor sentinel receives the bids, it caches them until just before the RFB deadline, and then selectively filters out the bids from unreliable agents before passing them on. A more sophisticated version of this handler might take into account the scarcity of subcontractors with this skill (also maintained by the reputation server) in deciding whether to filter out a bid or not.

While we can not, in this limited space, exhaustively enumerate all the ways sentinels can handle different exceptions, we hope that we have sufficiently illustrated the point that distinct EH services configured according to the principles described above can help ensure reliable MAS operation in the face of at least some important exceptions.

We have made substantial progress in our work on the EH services. We have implemented and empirically evaluated a simplified subset of these services applied to the agent death exception for CNET. Our results have shown that the EH services substantially out-performed the widely-used 'survivalist' exception handling technique (timeout and retry), with comparable agent complexity. In one experimental condition, for example, the EH services reduced average task completion

times, when agent death occurs, by a factor of nearly four (Dellarocas and Klein 2000) (Figure 7):

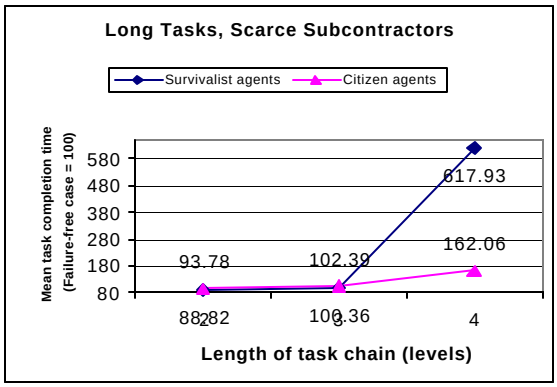

Figure 7: An Example of Mean Task Completion Times for Citizen vs Survivalist Agents

We have also made substantial progress in defining an EH services architecture that is both *scalable* (with respect to number of agents) and *generic* (can be applied to a wide range of MAS contexts). The architecture is fundamentally distributed: it is based on a distributed sentinel population plus EH services that are essentially database applications and thus can be replicated using well-known techniques. We have found that, with careful design, it has been possible to minimize EH related message traffic among these components. The architecture makes few assumptions about the MAS system the EH services operate in. Agents need implement only a relatively small number of simple EH directives (e.g. "are you alive?" and "cancel task X") in order to participate fully. Significant benefit (such as selective filtering of bids from unreliable agents) is possible even if the agents implement no EH directives whatsoever. Finally, if sentinels are integrated into the messaging infrastructure, the EH services can be totally transparent to the agents. If this can not be done, all that is required is that agents be able to treat sentinels as proxies for their in and out-going message traffic.

Open Issues: Important issues remain. Some of the most pressing include:

- As we extend our approach to handle a wider range of exceptions, we will increasingly encounter situations where a given exception (e.g. a task is late) can have multiple possible causes (e.g. the subcontractor died, the results message was lost, we can not find subcontractors with the necessary skills, etc). How can the EH services most effectively diagnose (and thereby select interventions for) exceptions in that context? There are two main contenders: a *fault-tree* approach (based on exhaustive pre-enumeration of the causal relationships among failure modes) requires extensive up-front knowledge acquisition, but *model-based* approaches, requiring as they do a complete behavioral model, may not be applicable to an open system made up of agents whose behavior is, in principle, a priori unknowable (Hamscher, Console et al. 1992).

- There appears to be a direct relationship between the number of EH directives an agent can handle, and the level of support that the EH services can provide; the more directives agents

implement, the more the EH services can do for them. These tradeoffs need to be better understood to provide guidelines for agent developers.

- How do we deal with agents that include some native ('survivalist') exception handling capabilities, and may therefore prefer to use their own techniques rather than outsource handling of some exceptions to the EH services? One of the team members (Prof. Dellarocas) is exploring the notion of using explicit contracts defining the normative behavior of open MAS agents to facilitate agent interoperability. One possibility is to extend this notion to capture an explicit EH 'contract' wherein agents specify, for each of the exceptions characteristic of their protocol, which exceptions they wish to outsource and which they prefer to handle themselves.

- What about exceptions in the exception handling process itself, e.g. agents that do not respond correctly to EH directives, sentinels that die, or EH messages that get lost? Our preliminary analysis suggests that the principles we described above can apply. Every sentinel, for example, can have another sentinel assigned to monitor it, and all sentinels can register their monitoring commitments with the contract notary. If a sentinel dies, therefore, we will be able to detect it, and shift its duties to a replacement sentinel.

3. Our Research Activities

Our group' efforts in this area include the following key objectives:

Objective 1: Develop Exception Analysis Methodologies: We are continuing to develop the RCV exception analysis methodology by applying it to other MAS protocols, and are exploring such techniques as inheritance and the use of formal commitment models (Venkatraman and Singh 1999) to substantially simplify the task of analyzing a new protocols.

Objective 2: Acquire Exception Handling Expertise: There are relatively few classes of MAS coordination protocols: the main types include market mechanisms (such as CNET, auctions and General Equilibrium Markets) (Radhakrishnan, Sandholm et al. 1997), distributed planning (Durfee 1996) (Lesser 1990) , game theory (Rosenschein and Zlotkin 1994), team-based approaches (Tambe 1997) (Jennings and Campos 1997) and stigmergic coordination (Ferber 1996) (Ferber and Jacopin 1991) (Beckers, Holland et al. 1994). The long-term goal of this line of research is to systematically perform an EH analysis for all the most important protocols. For now, however, we are focusing on market mechanisms. Such mechanisms are unique in that they are scalable to large numbers of agents, are backed up with substantive theoretical and empirical results from economics, computer science and other disciplines, and are immediately applicable to many of today's most compelling practical challenges such as e-commerce. We believe that the most interesting results will come from study of exceptions due to non-compliant agents (e.g. agents lying or under-performing), emergent problems (e.g. thrashing), and failures in the exception handling process itself.

Objective 3: Develop Exception Handling Services: We are defining the agent development guidelines and developing the EH service components needed to exploit the EH expertise acquired above in order to improve the robustness of open multi-agent systems, in a way that is both

scalable and generic. We are following a three phase approach that tackles increasingly sophisticated capabilities over time. In phase I, we focus on implementing the handlers and associated EH services needed to diagnose and resolve market exceptions with 'pure' citizen agents (where all exception handling is outsourced). In phase II we address the issue of dealing with agents that have some degree of 'survivalist' exception handling capabilities: in an open system such agents are likely to exist, and it is important to avoid the possibility of the agents and EH services working at cross-purposes. Finally, in phase III we explore making the EH services more adaptive in terms of what exceptions they address when. All exception handling involves some kind of cost, and it would be useful to design the services so they can allocate EH effort intelligently, for example by increasing the vigilance for non-compliant agents when new unfamiliar agents arrive.

Objective 4: Evaluate Services in Simulated and Real MAS Environments: We are evaluating the EH services we develop by simulating a range of realistic MAS scenarios, based on the analysis of the abstract properties of representative problem domains such as information retrieval, e-commerce and the like. Our work to date has applied this strategy successfully in our work on agent death in CNET, where we used the Swarm agent-based simulation tool (Minar, Burkhart et al. 1996) to evaluate the impact of the EH services for a variety of experimental conditions that varied task decomposition topologies (e.g. tall skinny task decompositions vs shallow and broad task decompositions), task lengths, and exception frequencies (Dellarocas and Klein 2000). The net result of this work is empirically-based, generalizable results concerning the relative efficacy for different exception handler approaches for different exceptions, which can captured as tradeoff information in the EH knowledge base.

The evaluation component of our work also includes studying how our EH services can be integrated with third-party market-based MAS. This allows us to thoroughly understand how developers can "plug" our technology into their open systems. We are currently involved in performing an initial evaluation of this sort in the context of the DARPA CoABS (control of agent-based systems) program, where we are creating a simplified version of the EH services to be incorporated into the MAS infrastructure being developed in this program. Further work will look at other systems, e.g. the University of Michigan AuctionBot system (Wurman, Wellman et al. 1998)).

Objective 5: Disseminate Results: The results of our work are being disseminated via publications as well as planned workshops, tutorials, and special issues in this area. Our growing exception handling knowledge base is also publicly available via the Web version of the MIT Process Handbook (see http://process.mit.edu/handbook.html): we anticipate that this medium will facilitate ready access by researchers, practitioners, and students interested in this topic (Klein and Dellarocas 2000).

To summarize, the direct deliverables of our research include::

- Improved techniques for analyzing the exception modes in open multi-agent systems, which will be valuable for distributed system development irregardless of whether the developers take advantage of the EH services approach we are developing or not.

- A knowledge base of exceptions and associated handlers for several important classes of market-based MAS coordination protocols, made freely available in highly organized form over the Web. While we are orienting our work towards the challenge of open agent systems, we believe that this knowledge base will also be useful for closed systems since the best exception handling techniques, in our experience, are often the same.

- Agent development guidelines, as well as implementations of key EH service components, as needed to enhance the robustness of open multi-agent systems.

- All of the above verified by substantial empirical evaluations in simulated MAS settings as well as the exercise of introducing this technology into a working MAS.

A key long-term aspiration of this work to develop a general coordination-theory-centric perspective on failures in complex distributed systems. Coordination theory has been successful in using powerful abstractions such as dependencies and coordination mechanisms to better understand normative behavior in complex distributed systems, irregardless of the nature of the agents and the domain (Malone and Crowston 1994). It is our hope that this work will lead to the development of similarly powerful abstractions concerning exception handling behavior which can be applied to domains ranging from computational systems to markets and human organizations.

4. Related Work

As we have already seen, the problem of robustness in multi-agent systems has been largely ignored by the MAS community, and the little previous work in this area has adopted 'survivalist' approaches that are unsuitable for open agent systems. Several lines of research have begun to explore concepts similar to those presented here, but none as far as we know have explored the combination of domain-independent exception handling implemented as distinct services. Hägg (Hägg 1996) presents the concept of sentinel agents; these are distinct services which monitor the agent system and intervene when necessary by selecting alternative problem solving methods, excluding faulty agents, or reporting to human operators. This approach is not domain-independent, however: sentinels must be customized for each new application. Kaminka et al (Kaminka and Tambe 1998) created Social Attentive Monitoring (SAM), wherein agents detect exceptions via uncovering violations of normative relationships with their teammates, and exploit a teamwork model to diagnose and fix these problems. This approach does have generic elements, but it is limited to teamwork protocols like TEAMCORE (Tambe 1997) and requires domain-dependent exception detection procedures. Horling et al. (Horling, Lesser et al. 1999) have explored the use of domain-independent tools to detect and resolve the exception wherein the agents have a harmfully inaccurate picture of the inter-agent dependencies in their current context. This approach is limited to a single exception type, however, and like SAM applies to just one class of coordination protocol. Venkatraman et al (Venkatraman and Singh 1999) describe a generic approach to uncovering agents that do not comply with coordination protocols. This approach only addresses one subclass of exception types, however, and does not include a resolution component. Decker et al. (Decker, Williamson et al. 1996) analyze the relative failure-proneness of matchmaker vs brokered middle-agents, but offer no prescriptions about how to deal with failures when they do occur. Huberman et al. (Youssefmir and Huberman 1995) and Sterman et al. (Sterman 1994) have investigated emergent dysfunctions (in particular, chaotic dynamics) can

appear in resource acquisition contexts, but their prescriptions make strong assumptions concerning the agent implementations (e.g. that agents can switch among multiple resource acquisition strategies) and thus are less well suited for open MAS contexts. Finally, Nodine et al (Nodine and Unruh 1999) and Kuwabara (Kuwabara 1996) discuss generic mechanisms for extending coordination protocols to handle errors, but include at best rudimentary prescriptions concerning what extensions are appropriate when.

Related work can also be found if we go farther afield into such disciplines as planning, distributed systems, manufacturing process control, and the like. Distributed database and real-time systems research has produced useful techniques such as error-correcting protocols, checkpointing and rollbacks (Holzmann 1991) (Burns and Wellings 1996) (Mullender 1993), but these techniques are aimed at closed systems and achieve generality at the cost of the efficiencies that can result from coordination-mechanism specific exception handling. Forrest et al. (Somayaji, Hofmeyr et al. 1998) (Dasgupta and Forrest 1999) describe an intriguing set of ideas, inspired by the immune system, for dealing with behavioral anomalies; they have identified little generic EH expertise to date, however, mainly concerning the detection of machine tool breakage and some kinds of security intrusions. There has also been substantial work in the planning and robotics communities on dealing with unexpected world states (Traverso, Spalazzi et al. 1996) (Howe 1995) (Birnbaum, Collins et al. 1990) (Broverman and Croft 1987) (Firby 1987) (Hindriks, de Boer et al. 1998). This work focuses almost exclusively on exceptions (e.g. failed operations, unexpected events) in the world manipulated by the agents, and not on exceptions concerning the agent world itself. Finally, there has been substantial work on detecting and resolving exceptions in computer-supported cooperative work (Mi and Scacchi 1993) (Chiu, Karlapalem et al. 1997) (Klein 1998) (Auramaki and Leppanen 1989) (Finkelstein, Gabbay et al. 1994) and manufacturing control (Fletcher and Misbah 1999) (Adamides and Bonvin 1993) (Katz 1993) but this has been applied to a very limited range of domains (e.g. just software engineering or flexible manufacturing cell control) and exception types (e.g. just inappropriate task assignments).

In an important sense we can say that the approach presented in this paper attempts to subsume much of the previous work in this area, in that our goal is to provide a unifying framework to exploit exception handling techniques derived from multiple disparate disciplines, for the benefit of improved robustness in open multi-agent systems.

5. References

(1999). Proceedings of the International Workshop on Knowledge-Based Planning for Coalition Forces. Edinburgh, Scotland.

Adamides, E. and D. Bonvin (1993). "Failure recovery of flexible production systems through cooperation of distributed agents." Ifip Transactions B: Computer Applications in Technology **11**: 227-38.

Aryananda, L. (1999). An Exception Handling Service for the Contract Net Protocol. Department of Electrical Engineering and Computer Science. Cambridge MA, MIT.

Auramaki, E. and M. Leppanen (1989). Exceptions and office information systems. Proceedings of the IFIP WG 8.4 Working Conference on Office Information Systems: The Design Process., Linz, Austria.

Baker, A. (1988). Complete manufacturing control using a contract net: a simulation study. Proceedings of the International Conference on Computer Integrated Manufacturing, Troy New York USA, IEEE Computer Society Press.

Bansal, A. K., K. Ramohanarao, et al. (1997). Distributed Storage of Replicated Beliefs to Facilitate Recovery of Distributed Intelligent Agents. Intelligent Agents IV; Proceedings of ATAL-97. M. P. Singh, A. Rao and M. J. Wooldridge**:** 77-91.

Barnett, J. A. (1984). "How Much Is Control Knowledge Worth? A Primitive Example." Artificial Intelligence **22**(1): 77-89.

Beckers, R., O. E. Holland, et al. (1994). From local actions to global tasks: stigmergy and collective robotics. Proceedings of the Fourth International Workshop on the Syntheses and Simulation of Living Systems, Cambridge, MA, USA, MIT Press; Cambridge, MA, USA.

Birnbaum, L., G. Collins, et al. (1990). Model-Based Diagnosis of Planning Failures. Proceedings of the National Conference on Artificial Intelligence (AAAI-90).

Boettcher, K., D. Perschbacher, et al. (1987). "Coordination of distributed agents in tactical situations." Ieee(87CH2450): 1421-6.

Bouzid, M. and A.-I. Mouaddib (1998). "Cooperative uncertain temporal reasoning for distributed transportation scheduling." Proceedings International Conference on Multi Agent Systems.

Broverman, C. A. and W. B. Croft (1987). Reasoning About Exceptions During Plan Execution Monitoring. Proceedings of the National Conference on Artificial Intelligence (AAAI-87).

Burns, A. and A. Wellings (1996). Real-Time Systems and Their Programming Languages, Addison-Wesley.

Chia, M. H., D. E. Neiman, et al. (1998). Poaching and distraction in asynchronous agent activities. Proceedings of the Third International Conference on Multi-Agent Systems, Paris, France.

Chiu, D. K. W., K. Karlapalem, et al. (1997). Exception Handling in ADOME Workflow System. Hong Kong, Hong Kong University of Science and Technology.

Dasgupta, D. and S. Forrest (1999). Artificial immune systems in industrial applications. Proceedings of the Second International Conference on Intelligent Processing and Manufacturing of Materials. IPMM'99.

Decker, K., M. Williamson, et al. (1996). Matchmaking and brokering. Proceedings of ICMAS '96: 2nd International Conference on Multiagent Systems, Kyoto, Japan, AAAI Press.

Dellarocas, C. and M. Klein (2000). An Experimental Evaluation of Domain-Independent Fault Handling Services in Open Multi-Agent Systems. Proceedings of The International Conference on Multi-Agent Systems (ICMAS-2000), Boston, MA.

Durfee, E. H. (1996). Planning in Distributed Artificial Intelligence. Foundations of Distributed Artificial Intelligence. G. M. P. O'Hare and N. R. Jennings, John Wiley & Sons**:** 231-245.

Durfee, E. H. and T. A. Montgomery (1990). A Hierarchical Protocol for Coordinating Multiagent Behaviors.

Ferber, J. (1996). Reactive Distributed Artificial Intelligence: Principles and Applications. Foundations of Distributed Artificial Intelligence. G. M. P. O'Hare and N. R. Jennings, John Wiley and Sons**:** 287.

Ferber, J. and E. Jacopin (1991). "The framework of eco-problem solving." Decentralized A.I. **2**: 181-93.

Finkelstein, A., D. Gabbay, et al. (1994). "Inconsistency Handling in Multi-perspective Systems." IEEE Transactions on Software Engineering **20**(8): 569-578.

Firby, R. J. (1987). An Investigation into Reactive Planning in Complex Domains. Proceedings of AAAI-87.

Fischer, K., J. P. Muller, et al. (1996). Intelligent agents in virtual enterprises. Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM'96), Blackpool, UK.

Fletcher, M. and D. S. Misbah (1999). "Task rescheduling in multi-agent manufacturing." Proceedings. Tenth International Workshop on Database and Expert Systems Applications. DEXA **99**: 689-94.

Gasser, L. (1992). DAI Approaches to Coordination. Distributed Artificial Intelligence: Theory and Praxis. N. M. Avouris and L. Gasser, Kluwer Academic Publishers**:** 31-51.

Gruber, T. R. (1989). "A Method For Acquiring Strategic Knowledge." Knowledge Acquisition **1**(3): 255-277.

Hägg, S. (1996). A Sentinel Approach to Fault Handling in Multi-Agent Systems. Proceedings of the Second Australian Workshop on Distributed AI, in conjunction with Fourth Pacific Rim International Conference on Artificial Intelligence (PRICAI'96), Cairns, Australia.

Hamscher, W., L. Console, et al. (1992). Readings in model-based diagnosis. San Mateo, CA, Morgan Kaufmann Publishers.

Hewitt, C. and P. D. Jong (1982). Open Systems, Massachusetts Institute of Technology.

Hindriks, K., F. de Boer, et al. (1998). "Failure, monitoring and recovery in the agent language 3APL." Cognitive Robotics. Papers from the.

Holzmann, G. J. (1991). Design and validation of computer protocols. Englewood Cliffs, N.J., Prentice Hall.

Horling, B., V. Lesser, et al. (1999). Diagnosis as an Integral Part of Multi-Agent Adaptability. Amherst, Massachussets, University of Massachussets at Amherst Department of Computer Science.

Howe, A. E. (1995). "Improving the reliability of artificial intelligence planning systems by analyzing their failure recovery." IEEE Transactions on Knowledge and Data Engineering **7**(1): 14-25.

Hunt, J., D. R. Pugh, et al. (1995). "Failure mode effects analysis: a practical application of functional modeling." Applied Artificial Intelligence **9**(1): 33-44.

Jennings, N. and J. Campos (1997). "Toward a Social Level Characterization of Socially Responsible Agents." IEE Proceedings on Software Engineering **144**(1).

Jennings, N. R. (1996). Coordination Techniques for Distributed Artificial Intelligence. Foundations of Distributed Artificial Intelligence. G. M. P. O'Hare and N. R. Jennings, John Wiley & Sons**:** 187-210.

Jennings, N. R., K. Sycara, et al. (1998). "A Roadmap of Agent Research and Development." Autonomus Agents and Multi-Agent Systems **1**: 275-306.

Kaminka, G. A. and M. Tambe (1998). What is Wrong With Us? Improving Robustness Through Social Diagnosis. Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98).

Katz, D. M., S. (1993). Exception management on a shop floor using online simulation. Proceedings of 1993 Winter Simulation Conference - (WSC '93), Los Angeles, CA, USA, IEEE; New York, NY, USA.

Klein, M. (1989). Conflict Resolution in Cooperative Design. PhD thesis. Computer Science. Urbana-Champaign, IL., University of Illinois.

Klein, M. (1991). "Supporting Conflict Resolution in Cooperative Design Systems." IEEE Systems Man and Cybernetics **21**(6): 1379-1390.

Klein, M. (1997). "An exception handling approach to enhancing consistency, completeness, and correctness in collaborative requirements capture." Concurrent Engineering Research & Applications **5**(1): 37-46.

Klein, M. (1998). A Knowledge-Based Approach to Handling Exceptions in Workflow Systems. Cambridge MA USA, MIT Center for Coordination Science.

Klein, M. (2000). Towards a Systematic Repository of Knowledge About Managing Collaborative Design Conflicts. Proceedings of the International Conference on AI in Design (AID-2000), Boston MA.

Klein, M. and C. Dellarocas (1999). Exception Handling in Agent Systems. Proceedings of the Third International Conference on AUTONOMOUS AGENTS (Agents '99), Seattle, Washington.

Klein, M. and C. Dellarocas (2000). "A Knowledge-Based Approach to Handling Exceptions in Workflow Systems." Journal of Computer-Supported Collaborative Work. Special Issue on Adaptive Workflow Systems. **9**(3/4).

Klein, M. and C. Dellarocas (2000). Towards a Systematic Repository of Knowledge about Managing Multi-Agent System Exceptions. Cambridge MA USA, Massachusetts Institute of Technology.

Kuwabara, K. (1996). Meta-level control of coordination protocols. Procedings of The International Conference on Multi-Agent Systems (ICMAS-96), Kyoto, Japan.

Lesser, V. R. (1990). "An overview of DAI: viewing distributed AI as distributed search." Journal of Japanese Society for Artificial Intelligence **5**(4): 392-400.

Loomis, J. C. (1979). "Management by Exception." Journal of Property Management **44**(3): 171-173.

Malone, T. W. and K. Crowston (1994). "The interdisciplinary study of coordination." ACM Computing Surveys **26**(1): 87-119.

Malone, T. W., K. Crowston, et al. (1998). "Tools for inventing organizations: Toward a handbook of organizational processes." Management Science **45**(3): 425-443.

Mi, P. and W. Scacchi (1993). Articulation: An Integrated Approach to the Diagnosis, Replanning and Rescheduling of Software Process Failures. Proceedings of 8th Knowledge-Based Software Engineering Conference, Chicago, IL, USA, IEEE Comput. Soc. Press; Los Alamitos, CA, USA.

Milgrom, P. and J. Roberts (1992). Economics, Organization and Management, Prentice Hall.

Minar, N., R. Burkhart, et al. (1996). The Swarm Simulation System: A Toolkit for Building Multi-Agent Systems. Santa Fe, New Mexico, USA, Santa Fe Institute.

Mullender, S. J. (1993). Distributed systems. New York

Wokingham, England ; Reading, Mass., ACM Press ;

Addison-Wesley Pub. Co.

Nodine, M. H. and A. Unruh (1999). Constructing Robust Conversation Policies in Dynamic Agent Communities. Proceedings of the Workshop on Specifying and Implementing Conversation Policies, part of The Third International Conference on Autonomous Agents (AA-99), Seattle, Washington USA.

Radhakrishnan, T., T. Sandholm, et al. (1997). Market-Oriented Approaches to Multi-Agent Systems.

Rosenschein, J. S. and G. Zlotkin (1994). Rules of encounter : designing conventions for automated negotiation among computers. Cambridge, Mass., MIT Press.

Sandholm, T., S. Sikka, et al. (1999). Algorithms for Optimizing Leveled Commitment Contracts. Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-99), Stockholm, Sweden.

Singh, M. P. (1999). "An Ontology for Commitments in Multiagent Systems: Toward a Unification of Normative Concepts." Artificial Intelligence and Law.

Smith, R. G. and R. Davis (1978). "Applications Of The Contract Net Framework: Distributed Sensing." Distributed Sensor Nets: Proceedings of a Workshop.

Smith, R. G. and R. Davis (1978). "Distributed Problem Solving: The Contract Net Approach." Proceedings of the 2nd National Conference of the Canadian Society for Computational Studies of Intelligence.

Somayaji, A., S. Hofmeyr, et al. (1998). Principles of a computer immune system. New Security Paradigms Workshop. Proceedings., Langdale, Cumbria UK, ACM.

Sousa, P. and C. Ramos (1999). "A distributed architecture and negotiation protocol for scheduling in manufacturing systems." Computers in Industry **38**(2): 103-13.

Sterman, J. D. (1994). Learning in and about complex systems. Cambridge, Mass., Alfred P. Sloan School of Management, Massachusetts Institute of Technology.

Tambe, M. (1997). "Towards flexible teamwork." Journal of Artificial Intelligence Research **7**: 83-124.

Traverso, P., L. Spalazzi, et al. (1996). "Reasoning about acting, sensing and failure handling: a logic for agents embedded in the real world." Intelligent Agents II. Agent Theories, Architectures, and Languages. IJCAI`95 Workshop.

Tsvetovatyy, M. B., M. Gini, et al. (1997). "MAGMA: An agent-based virtual marketplace for electronic commerce." Applied Artificial Intelligence **11**(6): 501-524.

Venkatraman, M. and M. P. Singh (1999). "Verifying Compliance with Commitment Protocols: Enabling Open Web-Based Multiagent Systems." Autonomous Agents and Multi-Agent Systems **3**(3).

Waldrop, M. (1987). "Computers amplify Black Monday." Science **238**: 602-604.

Wooldridge, M., N. R. Jennings, et al. (1999). A Methodology for Agent-Oriented Analysis and Design. Proceedings of the Third Annual Conference on Autonomous Agents (AA-99), Seattle WA USA, ACM Press.

Wurman, P. R., M. P. Wellman, et al. (1998). "The Michigan Internet AuctionBot: a configurable auction server for human and software agents." Proceedings of the Second International Conference on Autonomous Agents. ACM.

Xu, W. (1999). Generic Exception Analysis in a Dynamic Multi-Agent Environment. Department of Electrical Engineering and Computer Science. Cambridge MA, MIT.

Youssefmir, M. and B. Huberman (1995). Resource contention in multi-agent systems. First International Conference on Multi-Agent Systems (ICMAS-95), San Francisco, CA, USA, AAAI Press.