Eric Platon

PhD Candidate

Final Presentation

Exception Management in Multi-Agent Systems

Dr. Nicolas Sabouret

Pr. Shinichi Honiden

June 25th, 2007
Multi-Agent Systems
Distributed Systems
“Fallacy”
Robustness is inherent with MAS

[Haegg, 1996; Brewer, 2000; Klein et al., 2003; Hohpe, 2007]
Robustness is inherent with MAS

Robustness is not free

[Haegg, 1996; Brewer, 2000; Klein et al., 2003; Hohpe, 2007]
How to build more robust distributed systems with *multi-agent technologies*?
Robustness in Software

Exception Handling

Simple, Powerful, Elegant

Limited in Distributed Systems

[Goodenough, 1975; Xu et al., 1998; Issarny, 2000/2001; Romanovsky, 2001; Klein et al., 2003]
Exception Management in Multi-Agent Systems

Thesis, Part 1
Exception in MAS are not only programming-like exceptions

Thesis, Part 2
Exception in MAS should be supported at three levels:
• Code level
• Agent level
• System level
Exception in MAS are not only programming-like exceptions

What are MAS?

Engineering model for distributed, open, heterogeneous systems

What are MAS?

Interactive software process

Autonomous: Capable to decide

Resources

- DB
- DB
- WS

Deployment Environment

Agents

- Knowledge
- Processing
- Perception
- Action

Hardware
What are MAS?

Interactive software process
Autonomous: Capable to decide

Resources
DB DB WS

Deployment Environment

 Hardware

Agents
Knowledge
Processing
Perception Action
Exceptional Conditions

Mission: Register to CS-2

Related to Interaction Protocols
Exceptional Conditions

Related to Interaction Protocols
Exceptional Conditions

Related to the Environment
Expected Reactions to Exceptional Conditions

Self-Adaptive Behaviors

Based on Interaction & Autonomy
Target Notion of Exception
What is an exception usually?

Class MailBox, Method sendMessage

```java
try {
    Message m = new Message(id, from, to, content);
    send(m);
} catch (SendException exc) {
    System.err.print("Exception!");
}
```

[Goodenough, 1975; Parnas 1976; Randell, 1976]
Class MailBox, Method sendMessage

```java
try {
    Message m = new Message(id, from, to, content);
    send(m);
} catch (SendException exc) {
    System.err.print("Exception!"瘙);
}
```
Class MailBox, Method sendMessage

```java
try {
    Message m = new Message(id, from, to, content);
    send(m);
} catch (SendException exc) {
    System.err.print("Exception!");
}
```
Class *MailBox*, Method *sendMessage*

```java
try {
    Message m = new Message(id, from, to, content);
    send(m);
} catch (SendException exc) {
    System.err.print("Exception!");
}
```
Class MailBox, Method sendMessage

```java
try {
    Message m = new Message(id, from, to, content);
    send(m);
} catch (SendException exc) {
    System.err.print("Exception!");
}
```
An **agent exception** is the evaluation by the agent of a perceived event as unexpected.
Two categories

Agent Level  System Level
Exceptions in Multi-Agent Systems
Exceptions in Multi-Agent Systems

Code

```java
try {
    ...  
} catch (IOException e) {
    ...
}
```
Exceptions in Multi-Agent Systems
Exceptions in Multi-Agent Systems
Exceptions in Multi-Agent Systems

Agent

Exception

Programming Exception

System

Agent

Code
Exception Management in Multi-Agent Systems

Thesis, Part 1
Exception in MAS are not only programming-like exceptions

Thesis, Part 2
Exception in MAS should be supported at three levels:
• Code level
• Agent level
• System level
Exception in MAS should be supported at three levels, CAS

System Designer

Toward Engineering Practice
Separation of Concerns
System

Agent

Code
Our Proposal To Agent-Level Exception Management
Exception-Ready Agents
Exception-Ready Agents

Structure of Knowledge

Structure of Execution

Environment

Perception

Action
Exception-Ready Agents

Goals \rightarrow Plans \rightarrow Operators

Perception \rightarrow Environment \rightarrow Action

Operator Expectations \rightarrow Protocol Expectations \rightarrow Protocol Enactments

Action Revision \rightarrow ASM
Exception-Ready Agents

Goals | Plans | Operators | Protocols | Protocol Enactments

Environment

Perception

Operator Expectations

Protocol Expectations

Exception Detection

Violation

ASM

Action
Exception Detection

\[ action ::= \text{op}(arg[n]) \mid \text{protocol}(arg[n]) \quad \text{Apply an operator} \]
\[ \text{reflect}(action, expression, effect) \quad \text{Execute a protocol} \]
reffect(doAction, expression, effect)
reffect(doAction, expression, effect)
Perception Processing Perception Processing Perception Action

Protocol Enactments

Operator

Goals

Plans

Envelop

Protocol Enactments

ASM

Exception Detection

Environment

Goals

Plans

Operators

Protocols

Protocol Enactments

Operator Expectations

Protocol Expectations

Violations

Perception Action

Environment

Environment

Environment

Environment
Action Selection

\[\text{action ::= } \text{op}(\text{arg}[n]) \mid \text{protocol(arg}[n])\]

Apply an operator

Execute a protocol

\[\text{context(action, expression)}\]

\[\text{reffect(action, expression, effect)}\]
Expectations

- Update the agent state
- Select next action
  - Operator
  - Protocol
  - Goal & Plan Revision

\[
\text{context}(\text{action}, \text{expression}) \\
\text{reffect}(\text{action}, \text{expression}, \text{effect})
\]

- Commit action

Related to ANA, [Maes, 1991]
Example

cnet(x, [])

Initiator  Participant

cfp

accept

go(x)
\text{reffect}(\text{cnet}(x, []), \text{T}, \text{have}(x) \land \text{money}(y=y-\text{price}(x)))

\text{reffect}(\text{go}(x), \text{T}, \text{position}(x))
context( go(x), \neg position(x) )
context( \( \text{cnet}(x, [\text{shop}_1, \ldots, \text{shop}_n]) \), \
need(x) \land \text{available}(x, [\text{shop}_1, \ldots, \text{shop}_n]) \) )
Validation

Energy Consortium Scenario

Distributed
Open
Heterogeneous

Energy producers

Machinery producers

Machine parts producers
Validation

Energy Consortium Scenario

Test Exception: Delay Requests

Distributed
Open
Heterogeneous

Energy producers

Machinery producers

Machine parts producers
Implementation Styles

No Exception Management

Plain Management

EMS Approach
<table>
<thead>
<tr>
<th>(Intrinsic) Robustness</th>
<th>Plain</th>
<th>EMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tasks for Agent Exception Code</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Ad hoc</td>
<td>Actions</td>
<td></td>
</tr>
</tbody>
</table>
Execution Cycles per 100 ms

- **Baseline**
- **Plain**
- **EMS**

<table>
<thead>
<tr>
<th>Time (s)</th>
<th>Min</th>
<th>Max</th>
<th>Plateau</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.25</td>
<td>3.75</td>
<td>1.25</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>3</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>0.75</td>
<td>2.75</td>
<td>1.75</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>2.25</td>
</tr>
<tr>
<td></td>
<td>1.25</td>
<td>3.25</td>
<td>2.5</td>
</tr>
<tr>
<td></td>
<td>1.5</td>
<td>3.5</td>
<td>2.75</td>
</tr>
<tr>
<td></td>
<td>1.75</td>
<td>3.75</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>3.25</td>
<td>3.25</td>
</tr>
<tr>
<td></td>
<td>2.25</td>
<td>3.5</td>
<td>3.5</td>
</tr>
<tr>
<td></td>
<td>2.5</td>
<td>3.75</td>
<td>3.75</td>
</tr>
<tr>
<td></td>
<td>2.75</td>
<td>3.75</td>
<td>3.75</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>3.5</td>
<td>3.5</td>
</tr>
<tr>
<td></td>
<td>3.25</td>
<td>3.75</td>
<td>3.75</td>
</tr>
<tr>
<td></td>
<td>3.5</td>
<td>3.75</td>
<td>3.75</td>
</tr>
<tr>
<td></td>
<td>3.75</td>
<td>3.75</td>
<td>3.75</td>
</tr>
</tbody>
</table>
Average Execution Time

Trade-off
Robustness vs.
Execution Cost
EMS

Agent-Level Exception Management

Separation of Concerns
Task reduced to Action Design
Execution Overhead
Our Proposal
To
System-Level Exception Management
Application layer
Application layer

Sentinels

System-Level Exception Management

[Haegg, 1996; Klein et al., 2003; Shah et al., 2005]
Overridden Autonomy

Agent

- Knowledge
- Processing
- Perception
- Action

Sentinel

- Knowledge
- Processing
- Perception
- Action

Overridden Autonomy
Overridden Autonomy

Agent

Knowledge

Process

Perception

Sentinel

Knowledge

Learning

Perception

Action
System-Wide Support

Agent

Local View

Why Waiting?
System-Wide Support

Sentinel

Enlarged View
Multi-Agent System

App Agent

Sentinel

Environment
Multi-Agent System

Architecture with Softbody

App Agent

Sentinel

Environment
Agent

Perception  Action

Public State

Update

Softbody

Sentinel

Knowledge

Processing

Perception  Action
Validation

Energy Consortium Scenario

Sentinels

Machinery producers

Distributed
Open
Heterogeneous

Energy producers

Machine parts producers
Agent

Perception  Action

Public State

Running Order

Expected Completion Date
Contracting
Contracting

Notification
Analytical Evaluation

**Compared to Original Sentinels**

Communication Cost (n agents, worst)\[O(n^2)\]

Heuristics allowing \(O(n)\) and \(O(1)\)

[Haegg, 1996; Dellarocas et al., 2000; Klein et al., 2003; Shah et al., 2005]
Analytical Evaluation

Compared to Original Sentinels

Computational Complexity Unchanged
Architecture Change

“Economical” Change for Autonomy Respect

[Haegg, 1996; Dellarocas et al., 2000; Klein et al., 2003; Shah et al., 2005]
Conclusion
Exception Management in Multi-Agent Systems

Towards more robust distributed systems with multi-agent technologies
Thesis and Contributions

Exception in MAS should be supported at three levels:

- **Code level**
- **Agent level**
- **System level**

```
try {
    ...
} catch (IOException e) {
    ....
}
```
Perspectives on Exception Management in Multi-Agent Systems

Coupling between Exception Levels

Support for Action Design and Consistency

EM as Self-Organizing Mechanism
Acknowledgements
Thank you for your attention

Eric Platon

June 25th, 2007