Dogo Rangsang Research Journal
ISSN : 2347-7180 Vol-08 Issue-01 2018

The Initial Global Conference on Perceptual Computing in Data Sciences Moving Forward with Including Fault Tolerance in Agent-based Systems

Sachin Kumar Patra, Pinaki Bhusan Nayak
Department of Computer Science and Engineering
Gandhi Academy of Technology and Engineering,Berhampur

Abstract

Because of the intricacy of each entity's structure and the many ways in which the multiple agents interact with one another, distributed systems based on agent entities are prone to failure. Even though some approaches deal with fault failure, models that objectively address the field of fault tolerance are still needed. This is especially true when it comes to existing approaches that cover fault tolerance using conventional techniques that rely on communication and the underlying operating system. In this work, we propose a sentinel-based paradigm for handling agent conflict that incorporates fault tolerance directly into the language that describes the agent's behavior.

1. Introduction

The research community has been interested in fault-tolerance of multi-agent systems for the last ten years. Numerous academics have focused on the necessity of managing system complexity by taking system faults into account in the various agent software engineering processes as a result of the advancement of consensus theory in MAS [1-2]. In fact, MAS is thought to be more susceptible to interface, physical, and development errors since it is built on dispersed entities. The majority of research on managing faults in multi-agent systems focuses on fault detection and recovery, which are derived from conventional methods for recovering from other

distributed systems failure [3]. However, MAS has its own specificities and characteristics, and applying traditional tolerance techniques can be suitable just for specific situation and require special infrastructural support [4]. In this paper, we are interested to integrate and implement fault tolerance in the agent's components in order to build agents system by taking into account fault tolerance in the design phase of the agent software engineering.

Multiple agents work together to address challenges that are outside each agent's scope of expertise in agent-based systems [5]. We provide a model to define the behavior of the agents in the three operating modes—normal, temporal tolerance, and degraded—in order to include fault tolerance into the MAS system. Modeling the agent's behavior on the presumption that no errors occur is the focus of the normal mode. The temporal tolerance mode simulates a system where agents can function for a finite amount of time even when faults are present. When errors arise, the degraded mode takes into account simulating how the agent's functionality deteriorates.

This is how the remainder of the paper is structured. A similar work is presented in

 Copyright @ 2018 Authors

Section .

2. Connected work

There has been some work done on MAS in the area of fault tolerance to enable designers to create agents that can handle errors. Fault tolerance is one area of study that is interested in incorporating into the agent behavior model. The author of [6] suggests using a meta-script to control the agent's behavior. The latter uses fault tolerance by handling messages that reflect the exception. suggests the fault tolerant agent communication language (FT-ACL), an extension of the FIPA Agent Communication Language (ACL), to handle agent communication crash failures.

Using sentinels to address MAS flaws is the focus of another area of study [8]. For agents who are not agents, the sentinels are able to identify and manage exceptions. By cloning the essential agents system, other researchers are now embracing the replication notion [9, 10]. A suggested exception handling service in [11]'s work enables processing of agent-to-agent communication in pre-defined languages for the purpose of learning about exceptions and specifying actions taken in response to them.

2. The internal agent behavior incorporates fault tolerance.

In order to properly define the behavior of the system by accounting for fault tolerance, the study suggests a model called Extended Multi Decisional Reactive (E-MDRA), which is based on three modes. Four basic functions (action, decision, decision acknowledgement, and signal) are used in the first mode to characterize the typical system behavior. Fault-tolerance in the second and third modes is recognized and articulated to define temporal tolerance and degraded modes.

The following describe the typical mode: Action A: an operation that might be carried out on the agent; Decision D: an answer that the agent offers in response to the action. It is connected to a decision horizon Hd, which shows how long the decision will remain in effect.

• External State E': indicates the agent's state following the resolution of the action;

• External Objective O': indicates the behavior the agent adopts after receiving an action;

• Signal S: indicates an answer or an acknowledgment that the agent expects from an external entity;

• Internal State E: indicates the agent's current state at a given moment, and which is validated by the comprehension and understanding of a received signal.

Conversely, the following components form the basis of the deteriorated tolerance and temporal modes:

 Copyright @ 2018 Authors

• Degraded external objective: If the acknowledgment S is not received, the agent will cause event F. Degraded internal state N: represents the condition indicating the anomaly subsequent to the receipt of the event F. G: reflects the conduct that the agent adopted in response to the system dysfunction

• Tolerant internal state T: represents the condition in which the agent prolongs the waiting period of the signal S.

• Degraded External State L: reflects the remedy offered by the agent to address the anomaly of the system.

1.1. Standard mode

Assuming that there are no defects, we define the system's behavior in the normal mode. This leads to the definition of
four functions: the Action function (Act), the Decision function (Dec), the Decision Acknowledgment function (DecAck), and the Signal function (Sig).

1.1. Normal mode

In the normal mode, we specify the behavior of the system under the assumption that no faults are occurred. Four functions are defined for this purpose: Action function (Act), Decision function (Dec), Decision Acknowledgement function (DecAck) and finally the Signal function (Sig).

• The action feature

These stimuli may be viewed as actions applied to the reactive agent in order to carry out tasks or provide services, as the reactive agent is driven by stimulus-response behavior. Any agent that receives an action responds by modifying its internal behavior in accordance with its predetermined logic. Therefore, we define the Act function, which links an action A to a single External objective O', to reflect the dynamics of the agent. The latter depicts how the agent behaved after getting the action.

.

Act : A □□ O’
a □□ o’
□*a* □ *A,* □*! o’*□ *O’ / Act(a) = o’*

- **Decision function**

Reactive agents interpret stimuli from their surroundings as immediate reactions that they deliberate to define the External Objective. These responses may be viewed as Decisions, which stand for a particular problem's solution. Furthermore, we may use the Dec function to describe the immediate reaction capability (as a decision) because the behavior of the agent depends on its internal state

 Copyright @ 2018 Authors

Dec: O’ □
E □□
D □ O
(o’, e)
□□ (d,
o)

□ *(o’, e)* □ *O’* □ *E,* □*! (d, o)*□ *D* □ *O / Dec(o’, e) = (d, o)*

• The function that acknowledges decisions

Decisions that provide the system actions answers determine the dynamic behavior of the system. But the agent has a limited amount of time to carry out its function in order to promptly fix an issue. Every decision is therefore identified by its decision horizon Dur Dec, which denotes the amount of time that the decision is still in effect. The Ack Dec function is a representation of this process, linking every decision to a single signal and a decision horizon. This is officially translated as follows: AckDec:

D □□ S □ IN
d □□ (s, Hd)

[□*d*□*D,* □*! (s, Hd)* □ *S* □ *IN / AckDec(d) = (s, Hd)]* □*[d* □ □*≤Hd s]*

Using: - IN stands for all potential durations; - 𝓐 for the Real-Time Temporal Logic (RTTL) future operator

Quantitative temporal features are expressed using Real-Time Temporal Logic (RTTL) [Bellini et al., 2000].

⇒ For instance: ***A*** □ □***≤t B*** : means that when A occurs, B must occur in t units of time;

⇒ ***A*** □ □***[t, t] B*** : means that when A occurs, B must occur after t units of time

- **Signal function**

When an agent makes a decision on a perceived action, both the environment and the agent's behavior are impacted. The Sig function outcome of the Dec and AckDec functions, which links an external objective and signal to an internal and external state, is thus presented in order to provide the agent the power to modify its state. Formally speaking, the function is translated by:by :

Sig: O’ □ O □
S □□ E
□ E’(o’,
o, s)

□□ (e,
e’)

As [□ (o’, e1) □ O’□E , □! (d, o)□ D □ O
/ Dec(o’, e1) = (d, o)]and [□! (s, Hd) □ S
□ IN / AckDec(d) = (s, Hd)]

□ *[*□ *(e2, e’)* □ *E* □ *E’* □ *(e1 ≠ e2) / Sig(o’, o, s) = (e2, e’)]*

 Copyright @ 2018 Authors

This indicates that an external state e' is instantly transmitted and the new agent internal state is changed to e when the agent gets a signal s, contingent upon the current external objective o' and the predicted internal objective o.

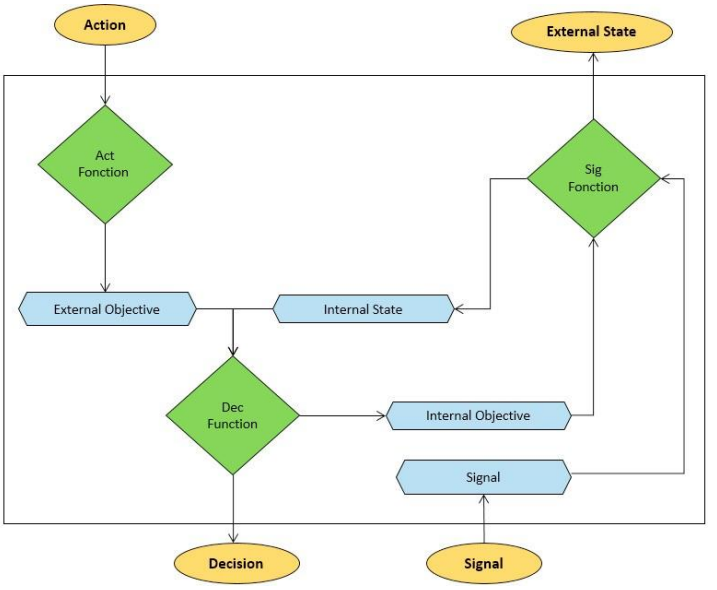


Fig. 1. Normal mode

- 1.2. Mode of Temporal Tolerance

For a brief period of time, the system may work and act as usual even when there are mistakes thanks to the Temporal Tolerance mode. The ability to respond locally against any additional delay that results from the choice not being recognized represents this mode at the agent level. Two functions—the internal objective tolerance function (AddTol) and the internal state tolerance function (TolSnt)—represent the implementation of this system.

• Function of internal state tolerance

In typical behavior, the agent's state transforms to an internal objective—a representation of the anticipated state following the execution of a choice—when the decision is carried out. The latter is limited by an acknowledgment signal's related time. The agent modifies its internal state representing the tolerance mode and produces an acknowledgment overflow event in the case that the signal is not acknowledged. The TolSnt function, which is a formal translation of this process, is represented by:

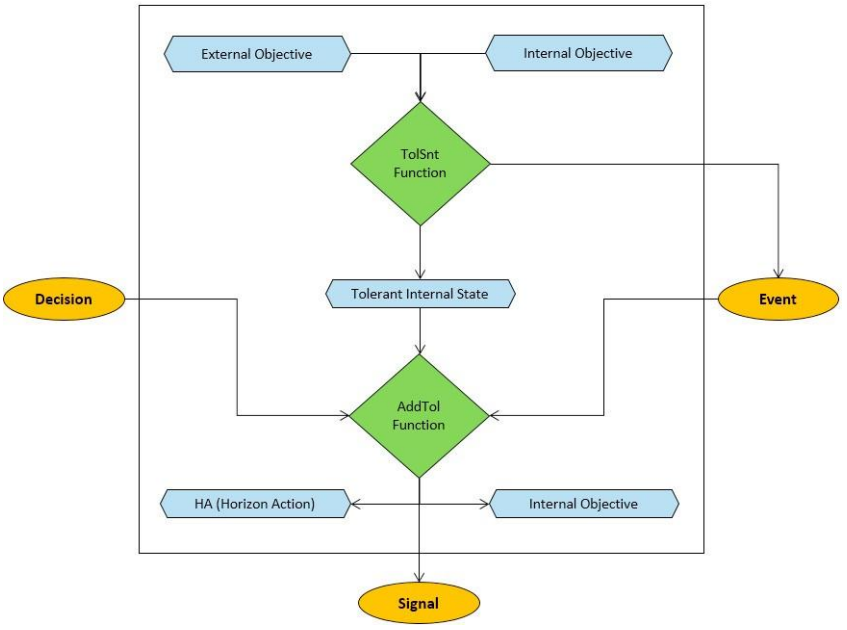
 Copyright @ 2018 Authors

TolSnt: O’ □ O □□ F □ T
(o’, o) □□ (f, t)
As [□ (o’, e) □ O’□E , □! (d, o)□ D □ O /
Dec(o’, e) = (d, o)] and [□! (s, Hd) □ S □
IN / AckDec(d) = (s, Hd)] □ [d □ □≤Hd s]
□ *[□! (f, t)* □ *F* □ *T* ⍰ *(o* □ □*[Hd, Hd] f) / TolSnt(o’, o) = (f, t)]* (o □ □[Hd, Hd] f) indicates that the occurrence f happens beyond the decision horizon Hd and after achieving the internal goal o.

Internal objective tolerance function:
Internal objective tolerance function: To allow the agent to prolong the time it takes to acknowledge a decision, we define the AddTol function. This function associates, for each decision d, an event f and the tolerance state t that comes from the TolSnt function. It also associates an additional time and a new internal objective that indicates the agent's expected state. The formal translation of this function is

AddTol: D □ F □ T □□ O □ S □ IN
(d, f, t) □□ (o, s, AH)
As [□ (o’, e) □ O’□E , □! (d, o1)□ D □ O
/ Dec(o’, e) = (d, o1)]and [TolSnt(o’, o1) =
(f, t) with (f, t) □ F □ T]

 Copyright @ 2018 Authors



□ *[□! (o2, s, AH)* □ *(O* □ *S* □ *IN)* □ *(o1 ≠o2)* □ *[d* □ *□≤(Hd + AH) s] / AddTol(d, f, t) = (o2, s, AH)*With: IN stands for every potential length of time.The mechanism of the temporal tolerance mode is shown in Figure 2.

Fig. 2. Temporal Tolerance mode

1.2. Lesser mode

During the repair process, the agent consents to a reduction in functionality in order to maintain operation despite the mistakes that have occurred. The primary concept involves seeing the deteriorated state as an external goal for the agent to accomplish, and regarding any corrective action as a new external goal. The non-compliance with time limitations or the changing environmental variables are what cause this deteriorated mode to occur.

to indicate the agent's switch to the weakened mode. Four functions are recommended for use: degraded acknowledgment function (Ack Deg), degraded exterior objective (Deg Oxt), degraded internal state function (Deg Snt), and degraded signal function (Sig Deg).

• A decline in the internal state function

When the agent behaves differently from how it did at first, as seen by its disregard for time limits, the transition to the degraded mode is initiated. When we have more time to move to the temporal tolerance mode or when the decision horizon is surpassed, these limitations arise. The mechanism is characterized by the degraded internal state function (Deg Snt), which is linked to both internal and external objectives, an event f that arises from exceeding temporal constraints, and a degraded internal state that

 Copyright @ 2018 Authors

signifies the shift in the agent behave. The function is formally translated by:

Deg Snt: O’ □ O □□ F □ G
(o’, o) □□ (f, g)
As [□ (o’, e) □ O’□ E , □! (d, o1)□ D □ O /
Dec(o’, e) = (d, o1)] And [□! (s, Hd) □ S □
IN / Ack Dec(d) = (s, Hd)] □ [d □ □≤Hd s]
□ [□! (f, g) □ F □ G A (o1 □ □[Hd, Hd] f) / Deg Snt(o’, o1) = (f, g)]
If [□ (f, t) □ F □ T / Add Tol(f, t, d) = (o2, s, AH) with (o2, s, AH) □ (O □ IN □ S) A (o1 ≠o2)]
□ *[□! (f, g) □ F □ G □ (o2 □ □[Hd + AH, Hd + AH] f) / Deg Snt(o’, o2) = (f, g)]*

• A reduced ability to achieve external goals

Degrading an agent's regular behavior and assigning a new external goal to fulfill is the aim of the degraded external objective function, or Deg Oxt. The agent's deteriorated condition, which symbolizes this process, signifies the oddity it encounters. Beginning in this condition, a new external aim is officially translated by the deteriorated external state that results from it.

Deg Oxt: G □ N □□ L □ O'
(g, n) □□ (l , o')
As [□ (o’1, o) □ O’□ O / Deg Oxt(o’1, o) = (f, g) with (f, g) □ *F* □ *G*]
□ *[□! (l, o’2) □ L □ O' □ (o'1 ≠o'2) / Deg Oxt(g, n) = (l , o'2)]*

• A diminished ability to acknowledge

By accepting a reduction in its functions, the agent can use the degraded mode to address abnormalities that arise. Because the degradations are carried out by outside goals. The latter then indicate to the agent that their relevant duty has been completed. The mechanism in question is characterized by the degraded acknowledgment function (Ack Deg), which is designed to verify the accomplishment of external goals through the generation of a report for the agent. The formal translation of this function is:

Ack Deg: G □ L □ O' □□ S
(g, l, o') □□ s
[□ (g , n) □ G □ N / DegSnt(g, n) = (l , o') with (l, o') □ L □ O']
□ *[□ s □ S / AckDeg(g, l, o') = s]*

• Reduced signal performance

Allowing the agent to revert to its regular state upon reception of the signal is the aim of the degraded signal function (SigDeg). The latter is contingent upon the acknowledgement transmitted by the external goals as a consequence of the mode's degradation. which, when translated officially, is:

Sig Deg: S □□ O' □ O

 Copyright @ 2018 Authors

s □□ (o' , o)
As [□ (o'1, o1) □ O'□ O / DegOxt(o'1, o1) = (f, g) with (f, g) □ F □ G] ,
[□ n □ N/ DegSnt(g, n) = (l , o'2) with (l, o'2) □ L □ O' A (o'1 ≠o'2)]and [AckDeg(g, l, o'2) = s avec *s* □ *S*]
□ *[SigDeg(s) = (o'1, o1)]*

The Figure 3 illustrates the mechanism of the degraded mode.

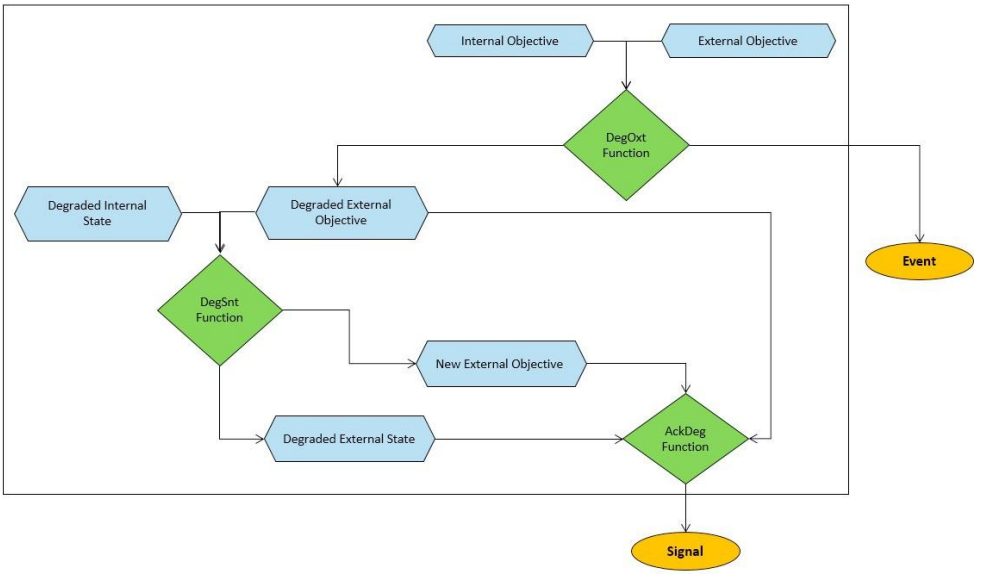


Fig. 3. Degraded mode

2. Integrating fault tolerance into the actions of external agents

The Extended Multi Decisional Reactive (E-MDRA) model is composed of a number of agents that are linked to one another by communication interfaces. This creates a two-level tree-based hierarchical structure that consists of a DRA Supervisor (E-DRAS) and two or more sub-agent components (E-MDRASi). Two communication interfaces, the Decisional Interface (DI) and the Signal Interface (SI), are used to establish a link between the supervisor and its sub-agents. An environment's actions and external states' emissions to the environment are how such a system interacts with its surroundings**.**

Many Agents An E-DRAS Supervisory Agent is in charge of several E-DRAC

 Copyright @ 2018 Authors

component agents in Extended Decision Reactive (E-MDRA).

The component of E-DRAC can be:

And an agent Supervisor: Engages with the environment directly or manages other agents

• An agent element is someone who works directly with the environment and is not in charge of any other agents.

Figure 4 depicts the Agent's Hierarchical Structure (AHS) hierarchical structure. AHS describes subagents and supervisor agents..

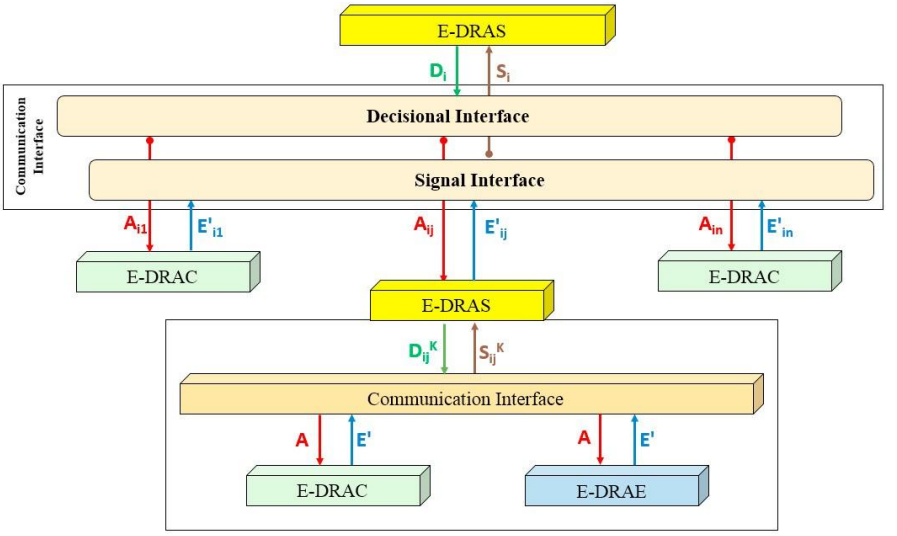


Fig. 4. Multi-Agent structure

- A quadruplet AHS = <E-DRAS, n, E-DRAC, AgentSup> defines the hierarchical structure of Agent's Hierarchical Structure (AHS), where:
- • E-DRAS: supervisory set of agents
- • n = dim(AHS), which is the dimension of AHS (number of subagents under the agent supervisor's direction).
- • E-DRAC: collection of agent constituents (n-tuple of agents)
- • Agent Sup: Function that links the components of an agent's subagent to the agent supervisors

Agent Sup : E-DRAS □□ E-DRAC^n

Ags □□ (Agc1, Agc2, …, Agcn)

[□Ags □ E-DRAS(AHS), □(Agc1, Agc2, ..., Agcn) □ E-DRAC(AHS) / Agent Sup(Ags) = (Agc1, Agc2, ..., Agcn)]

1.2. Two information flows form the basis of the communication interface: • A

 Copyright @ 2018 Authors

descending flow that is represented by the Decisional Interface, which moves choices and actions from E-DRAS to E-DRACs.

1.3. • An amount flow that is represented by the Signal Interface, which communicates with the E-DRAS the signals and external states of the E-DRACs.

1.4. In the sections that follow, the two decision and signal interfaces are formalized.

1.5. *Formalization of the Decision Interface*

- Transforming the decisions made by the agent supervisor into a set of instructions for the components of the lower-level sub-agents is the aim of the Decision Interface. It also describes the common acts that the actors take.
- Dec Int = <Dec Input, n, Dec Output, Tra Dec> is a quadruplet that formalizes the Decisional Interface (Dec Int).
- • Dec Input = (E-DRAS ₴ D) is a series of choices related to the agent supervisor.
- n is the dimension of Dec Int (the collection of Agc subagents related to (Ags, d)); n = dim(Dec Int) <= dim(AHS)
- • Dec Output = (E-DRAC ₴ A)n, which is the collection of actions related to the component agents.
- Tra Dec is a function that converts a decision into many concurrent actions directed at agents at a lower level.

Tra Dec : Dec Input □□ Dec Output

(Ags, d) □□ ((Agc1, a1), (Agc2, a2), …, (Agcn, an))

□ (Ags , d) □ Dec Input(Dec Int), □ (Agc, a) □ Dec Output(Dec Int), Agc □ Agent Sup(Ags) /Tra Dec(Ags ,d) = ((Agc1, a1), (Agc2, a2), …, (Agcn, an))

- *1.6. Signal interface formalization*
-
- *Through the decisional interface, their actions are received by the lower level agents. These agents then communicate an external state to the agent supervisor to verify that their operations were carried out as intended. These external states are successfully received by the Signal interface.*
- *A quintuple defines the signal interface (Sig Int). Sig Int = <Sig Error, Tra Sig, Sig Output, n, Sig Input>*
- *Dim (Sig Int) = dim (Dec Int) < dim (AHS) is the dimension of Sig Int, which represents the number of sub-agents Agc connected with (Ags, s).*
- *• Sig Input = (E-DRAC □ E')n, the collection of external states produced by the agents component*
- *• Sig Output: signals that the agent supervisor received; Sig Output = E-DRAS ₴ S*
- *• Tra Sig: A function that unifies the several signals the agents' component generates into a single signal connected to the agent supervisor.*

Tra Sig: Sig Input □□ Sig Output

 Copyright @ 2018 Authors

((Agc1, e'1), (Agc2, e'2), …, (Agcn, e'n)) □□ (Ags, s)

□ *(Agc, e')* □ *Sig Input,* □ *(Ags , s)* □ *Sig Output / Tra Sig((Agc1, e'1), (Agc2, e'2), …, (Agcn, e'n)) = (Ags, s)*

Sig Error: Sig Output □□ Sig Input (Ags , s) □□ ((Agc1, e'1), (Agc2, e'2), …, (Agcn, e'n))

□ *(Ags , s)* □ *Sig Output,* □ *(Agc, e')* □ *Sig Input, / Sig Error(Ags, s) = ((Agc1, e'1), (Agc2, e'2), …, (Agcn, e'n))*

2. Examine cases

We suggest using the detecting intrusion system case study to demonstrate our methodology. To detect, diagnose, and transmit information about any hostile incursion in the protected region, a number of distributed agents must collaborate in this kind of system. Figure 5 illustrates how to simulate an agent's unique behavior while accounting for fault tolerance. The UML State chart diagram represents the internal behavior throughout. After receiving the order to "detect intrusion," the agent decides to "get video surveillance." If he obtains the footage from the scene, he examines it, processes the data, and notifies the user of the out come **I.e.** He extends the time he waits for video surveillance if retrieving the videos is not feasible. If, after that period, he still does not get any information, he modifies his behavior by choosing to detect movement and temperature. A report is delivered to the user once all scene information has been gathered.

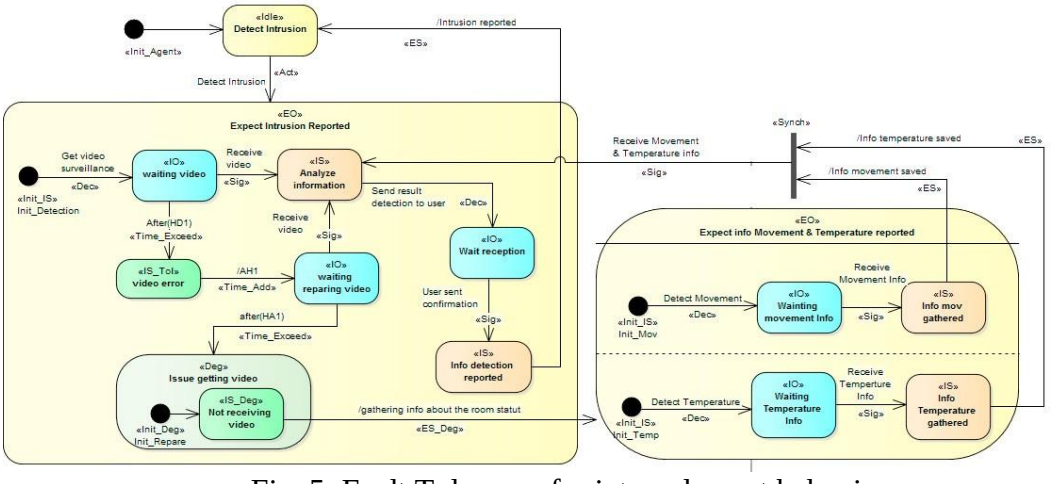


Fig. 5. Fault Tolerance for internal agent behavior

 Copyright @ 2018 Authors

Four options may be identified based on the internal behavior of the "Detect intrusion" action: "Get video surveillance," "send info to the user," "detect movement," and "detect temperature." As a result, we can describe the agent's exterior behavior (Fig.6) all the way through the UML activity diagram. In its initial choice, the coordinator agent selects the action "collect image from scene," which is connected to the agent in charge of the surveillance camera. If the latter transmits the video, the coordinator agent will review its content; if not, he will make a call to the Movement sensor agent to obtain movement data and the Temperature sensor agent to obtain temperature

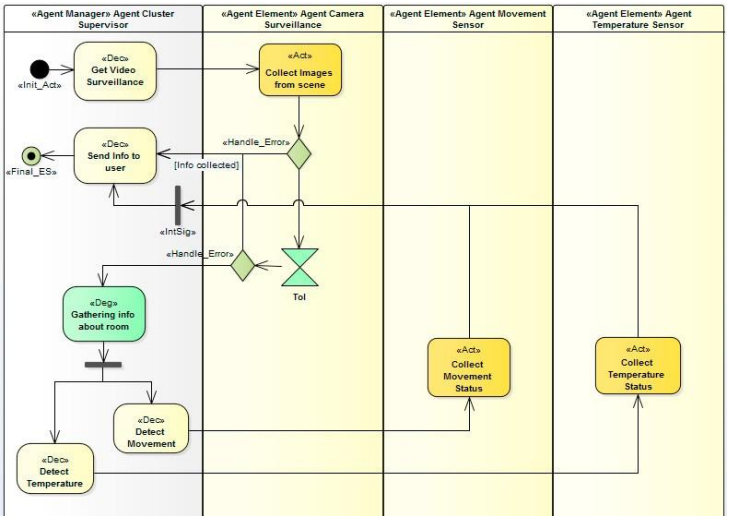


Fig. 6. Fault Tolerance for external agent behavior

Remarks

We have provided a model in this research to include fault tolerance into the MAS behavior definition. This paradigm allows designers to construct agents with degraded fault tolerance and temporal tolerance. The suggested paradigm, which is based on the sentinel method to handling agent conflict, incorporates fault tolerance directly into the vocabulary that describes the behavior of the agent. We are presently working on the implementation phase as future work to assess the execution process' fault tolerance.

References

[1] Su, Lili, and Nitin H. Vaidya. “Fault-tolerant multi-agent optimization: optimal iterative distributed algorithms.” *In Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing.* 2016, p. 425-434

[2] Zhang, Gaosheng, et al. (2018) “Fault-tolerant coordination control for second-order multi-agent systems with partial actuator effectiveness.” *Information*

 Copyright @ 2018 Authors

Sciences 423: 115-127

[3] Hua, Yongzhao, et al. “Distributed fault-tolerant time-varying formation control for second-order multi-agent systems with actuator failures and directed topologies.” *IEEE Transactions on Circuits and Systems II: Express Briefs*. 2017

[4] Wangapisit, Ornkamon, et al. (2014) “Multi-agent systems modelling for evaluating joint delivery systems.” *Procedia-Social and Behavioral Sciences* 125: 472-483

[5] Shakshuki, Elhadi, and Malcolm Reid. (2015) “Multi-agent system applications in healthcare: current technology and future roadmap.” *Procedia Computer Science* 52: 252-261

[6] Potiron, Katia, Amal El Fallah Seghrouchni, and Patrick Taillibert. “Fault Tolerance for MAS Specific Faults.” *From Fault Classification to Fault Tolerance for Multi-Agent Systems*. Springer London, 2013, p. 37-57

[7] Dragoni, Nicola, Mauro Gaspari, and Davide Guidi. (2007) "An ACL for specifying fault-tolerant protocols." *Applied Artificial Intelligence* 21 (4-5): 361-381

[8] Basilico, Nicola, Timothy H. Chung, and Stefano Carpin. “Distributed online patrolling with multi-agent teams of sentinels and searchers.” *Distributed Autonomous Robotic Systems*. Springer, Tokyo, 2016, p. 3-16

[9] Platania, Marco, et al. “Towards a practical survivable intrusion tolerant replication system.” *In IEEE 33rd International Symposium on Reliable Distributed Systems (SRDS)*. 2014, p. 242-252

[10] Bhanot, Rajdeep, and Rahul Hans. (2015) “A Secure and Fault Tolerant Platform for Mobile Agent Systems.” *International Journal of Security and Its Applications* 9 (5): 85-94

[11] Fang, A., et al. “Fault Tolerance Assistant (FTA): An Exception Handling Programming Model for MPI Applications”. *No. LLNL--TR- 692704. Lawrence Livermore National Lab, Livermore, CA (United States)*. 2016

 Copyright @ 2018 Authors