Ensuring Correctness of Agent Interactions in Collaborative Environments

Elena Troubitsyna
Åbo Akademi University, Faculty of Science and Engineering
Domkyrkotorget 3, 20500,
Turku, Finland
e-mail: Elena.Troubitsyna@abo.fi

***Abstract*— Ensuring correctness of agent interactions in complex systems constitutes a significant research challenge. The highly dynamic nature of the system makes it hard to predict all possible collaborations that the agents form during the system functioning. Therefore, it is desirable to create a generic abstract model that can facilitate reasoning about correctness of agent interactions in the complex dynamic collaborative environments. In this paper, we adopt a goal-oriented approach to reasoning about agent collaboration and define the basic abstractions underlying the behaviour of complex collaborative systems. Each agent has individual capabilities that are complemented and enhanced via cooperation to allow the system to achieve the desired goals. We define an abstract model of a system whose behaviour can be structured as a set of dynamic coalitions. We propose a structured approach to analysing possible deviations in the component interactions based on Hazard and Operability Study (HAZOP) and formally define the impact of deviations in agent interactions on achieving the required goals.**

Keywords -dynamic coalitions; interactions; goals; deviation analysis; formal modelling.

I. INTRODUCTION

Over the recent years, creating new services and applications via collaboration has gained a significant popularity. Dynamic collaborations and compositions of agent components allow the designers to achieve agility and high productivity in the development of new features and functions. Dynamic collaboration is in the heart of such major trends as the Internet of Things [10], industrial internet and Internet of Everything. These concepts are built on the pervasive connectivity and openness towards sensors, machines and devices. The opportunities offered by dynamically composed collaborative environments offer rich technical and business opportunities that can be efficiently utilised to dynamically create novel flexible architectures.

The highly dynamic nature of collaborations requires novel approaches that allow the designers to systematically analyse the dynamics of collaborative environments and in particular, predict how deviations in the agent behaviour and interactions impact the functions that a collaborative environment should implement.

Currently, it has been commonly accepted that the notion of goals provides us with a suitable basis for formalising the objectives that a system should achieve [4]. The agents form coalitions to interactively work on achieving certain goals. The agents forming collaboration provide certain individual functionality that contributes to achieving a common goal. When an agent or a communication infrastructure fails, a coalition might fail to achieve the desired goal. Therefore, we should systematically explore the possible deviations in the agent and communication infrastructure behaviour and study the impact of these deviations on the possibility of achieving the required goals.

In this paper, we demonstrate how to use the HAZOP method [1][2] to systematically study possible deviations in the agent interactions. We propose a classification of the types of deviations in the agent interactions and define their impact on achieving system goals.

We define a generic model that formalises the relationships between the system goals and possible deviations in agent behaviour and interactions. Since the proposed model explicitly links the system goals with the behaviour of the individual agents in a coalition, it can facilitate design of complex collaborative systems.

The paper is structured as follows. In Section II, we define collaborative environments in terms of the goals that should be achieved by agent coalitions. In Section III, we describe generic scenarios of agent interactions. Section IV shows how to systematically analyse deviations in the component interactions using HAZOP. Finally, in Section V, we discuss the proposed approach and overview the related work.

II. TOWARDS FORMAL MODELLING OF COLLABORATIVE ENVIRONMENTS

In this paper, we define a *collaborative environment* as a set of coalitions, i.e.,

$$ColENV = \{C_1, C_2, ..., C_N\}$$

where *Ci* is an id of a coalition.

A coalition is a dynamic composition of the agents. The agents join and leave a coalition depending on their states and the current goal. As soon as an agent joins a coalition, it can communicate and collaborate with all the agents involved into it. In general, any coalition is formed to fulfil a certain goal [4]. The set of goals, which an entire collaborative environment can achieve, is denoted as *GOALS:*

$$GOALS = \{G_1, G_2, ..., G_M\}$$

 ISBN: 978-1-61208-575-3

The set consists of the constants defining the names of the goals. We assume that each particular coalition is formed to perform a certain goal from the set *GOALS*.

A *goal* is an objective that a coalition should achieve. A goal can be decomposed into a set of subgoals, and furthermore, into a set of sub-subgoals that each component in the collaboration should perform. Each agent carries a special attribute describing the functionality that it implements. Often these attributes are called roles. Usually, an agent implements a set of roles chosen according to the tasks that it should perform in each particular coalition.

For each coalition, we can define a configuration function that indicates how many agents in certain roles a coalition should have for a goal to be achievable. Hence, a configuration can be defined as follows:

$Config \in CONFIG$, where $CONFIG : \mathscr{P}(ROLES \rightarrow N)$

where *ROLES* is a set of roles, $\mathscr{P}$ denotes a powerset and *N* is a set of natural numbers.

Often a configuration of a coalition is defined not only by a goal but also non-functional parameters, e.g., performance. We assume that goals are distinct if their non-functional parameters are different. Therefore, we can unambiguously map a set of goals onto the set of configurations.

For each goal *Gi*, $Gi \in GOALS$, we can define the minimal sufficient configuration as a function

$$MINCONF : GOALS \rightarrow CONFIG$$

The function defines how many agents in each role a coalition should have to be able to achieve a certain goal. The function *MINCONF* defines the minimal necessary conditions. Obviously, a coalition can have more agents than required by *MINCONF*. The additional agents can remain inactive while achieving a certain goal and become activated to replace failed agents in case some of initially involved agents fail.

In practice, at each particular moment of time, a collaborative environment *ColENV* does not try to achieve all the goals defined by the set *GOALS* at once. Therefore, we can distinguish between a set of the active (triggered) goals *Act_G*, i.e., the goals that a collaborative environment tries to achieve at a certain moment of time and the goals that are not triggered, i.e., passive goals *Pas_G*,. This defines a partitioning of the set of goals into two non-intersecting subsets:

$$GOALS = Act\_G \cup Pas\_G,$$
$$\text{where } Act\_G \cap Pas\_G = \emptyset$$

In our modelling, we assume that the agents are not dormant and hence, they are getting engaged in the different coalitions (as soon as their roles match the roles required in the coalition. Therefore, when a goal is activated, it might take some time to fulfil the conditions defined by *MINCONF* because some of the required agents are still engaged in another coalitions. If the required configuration is established, then, the coalition executes the required actions to achieve the goal. We introduce a set

$$C\_STATE : \{Active, Activated, Dormant\}$$

to designate the status of the coalition. The constant *Active* means that the coalition has the required configuration and is assigned a goal to achieve. The constant *Activated* means that the coalition is assigned a goal but it has not established the required configuration. Correspondingly, the constant *Dormant* means that the configuration is currently not involved into an execution of any goal. We introduce the function *C_STATUS* that maps the id of the collaboration to its status:

$$C\_STATUS : CNAME \rightarrow C\_STATE$$

The function *CUR_CONFIG* is defined as follows:

$$CUR\_CONFIG : CNAME \rightarrow CONFIG$$

It designates the current configuration of the coalition.

Next, we formally define the relationships between the status of a coalition, goals and configurations.

The coalition *Ci* is active, i.e.,

$$C\_STATUS(Ci) = Active$$

if

$$Gj \in GOALS \wedge$$
$$Gj \in Act\_G \wedge$$
$$MINCONF(Gi) \leq CUR\_CONFIG(Ci)$$

where the ordering relation $\leq$ is defined over the configurations as follows:

For $Conf_k$ and $Conf_l$, such that $Conf_k$, $Conf_l \in CONFIG$, $Conf_k \leq Conf_l$ if

$$\forall r_n . r_n \in dom(Conf_k) \Leftrightarrow r_n \in dom(Conf_l)$$
$$\forall r_n . r_n \in dom(Conf_k) \Leftrightarrow Conf_k(r_n) \leq Conf_l(r_n)$$

where *dom* denotes the domain of function or relation.

When a coalition *Ci* is set to achieve a certain goal but has not established the required configuration or an execution of a scenario required to achieve a goal is suspended due to failures, its status is *Activated, i.e.,*

$$C\_STATUS(Ci) = Activated$$

if

$$Gj \in GOALS \wedge$$
$$Gj \in Act\_G \wedge$$
$$\neg (MINCONF(Gi) \leq CUR\_CONFIG(Ci))$$

Finally, a coalition can be inactive, i.e.,

$$C\_STATUS(Ci) = Activated$$

if

$$Gj \in GOALS \wedge$$
$$Gj \in Pas\_G$$

We assume that agents are involved in the coalition with the status *ACTIVE* communicate with each other by exchanging messages. To achieve a certain goal, a coalition should perform a predefined scenario. In the next section, we define generic scenarios performed by the components in a coalition.

III. MODELLING SYSTEM ARCHITECTURE AND AGENT INTERACTIONS

A goal defines a set of states that a collaborative environment should achieve. While working of achieving a certain goal, a coalition executes a certain scenario. An execution of a scenario is triggered by a coordinator of the coalition. It is an agent with the specific rights to initiate and finalise the scenario execution. We can describe a scenario by a UML [5] use case model and supplement it by a description of the flow of events associated with it. The actors of the use case model are the agent roles and the use cases are the coalitions achieving the corresponding goals. Due to the generic nature of our model, we omit its graphical representation. A description of typical and abnormal flows of events in a generic use case associated with our system can be defined as shown below:

Description of use case
Coalition Ci achieves goal Gj

Precondition *Goal is eligible for execution and triggered*

$$Gj \in GOALS \wedge$$
$$Gj \in Act\_G \wedge$$

Postcondition *Collaboration achieves goal or Collaboration reports failure*

Includes: *Recover_Scenario_Ci_Gj*

Normal sequence of events:

1. The coordinator of *Ci* receives a notification that a goal is activated and changes the status of the coalition, i.e.,

$$C\_STATUS\ (Ci) := Activated$$

2. The coordinator broadcasts an invitation to join a coalition to the agents of *ColENV* and monitors that the required configuration is established
3. When a configuration is established, i.e.,

$$MINCONF(Gi) \leq CUR\_CONFIG(Ci)$$

it broadcasts the message engaged to the involved components and changes the status of the collaboration, i.e.,

$$C\_STATUS\ (Ci) := Active$$

4. Agents collaborate and communicate with each other to perform the tasks required to achieve the required goal and the coordinator monitors the status of the agents in the duration of the scenario execution. If it discovers an agent failure then go to step 8.
5. When goal is achieved the agents report to the coordinator about completion of scenario.
6. Coordinator hands over the control to the collaborative environment manager and changes the status of the collaboration, i.e.,

$$C\_STATUS\ (Ci) := Dormant$$

7. The coordinator broadcasts disengage message to all agents.
8. The collaboration coordinator re-evaluates the status of the coalition. If the condition of the sufficient configuration is not satisfied then it changes the status of the collaboration to *Activated* and activates timer.
9. If the agents recover within the timeout then the status is changed to *Active* and the normal execution is resumed.
10. If the agents fail to recover within timeout then switch to executing failure recovery scenario *Recover_Scenario_Ci_Gj*.

Description of use case *Recover_Scenario_Ci_Gj*

Precondition
Normal execution of scenario to achieve goal Gj by coalition Ci failed.
Status of Ci is Activated

Postcondition Reconfiguration and resuming normal execution or permanent failure

Extends: *Coalition Ci achieves goal Gj*

Sequence of events:

1. The coordination of *Ci* broadcasts a new invitation to all agents of the collaborative environment to join a coalition and activates a timer

2. If within the timeout the coordinator receives a respond from the agents whose roles match the roles of failed agents then continue. Otherwise the scenario terminates, i.e., go to 4.

3. The coordinator sends engagement message to the newly joining agents and changes the status of the coalition to *Active*. After this, the normal execution

 ISBN: 978-1-61208-575-3

resumes, i.e., the use case *Collaboration Ci achieves goal Gj* resumes.

4. The coalition sends the failure message to the collaborative environment manager and changes the status of the collaboration *Ci* to *Dormant.*

5. The coordinator broadcasts disengage message to all agents.

Let us now depict the proposed system structure. We distinguish between three layers: the collaborative manager layer, coalition coordinators and, finally, agents. The structure is presented in Figure 1.

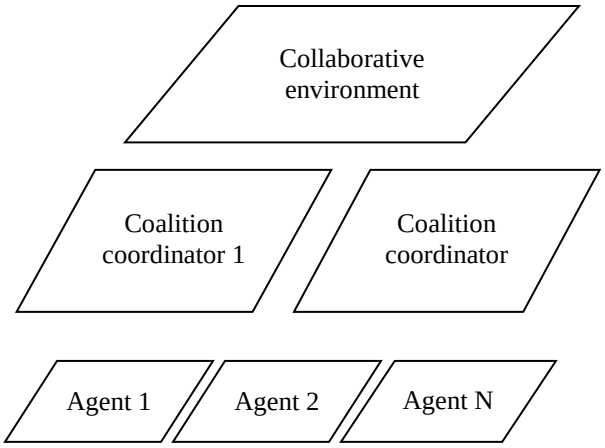

Figure 1. System architecture

The collaborative environment manager is responsible for triggering the system goals and broadcasting the corresponding messages to the coalition coordinators. The coalition coordinators (a special kind of agents) check whether they are eligible to initiate a coalition and if this is the case, broadcast the invitations to all agents of the collaborative environment. The coalition coordinator monitors the coalition forming and as soon as the configuration conditions are fulfilled, monitors an execution of the scenario associated with the given goal. Upon completion of the scenario, it acknowledges it to the collaborative environment manager and disengage the agents. If the execution of the scenario fails and cannot be recovered then the coalition coordinator reports the failure the coalition manager.

To join a coalition, each agent a check that it has the eligible role and becomes engaged in the coalition. If the resources permit then an agent can join several coalitions at the same time but typically in different roles.

As it is easy to note, the main complexity of ensuring correctness of agent collaboration is associated with handling agent failures and recovery. Indeed, it is easy to run into a deadlock situation, i.e., reach the state that no progress can be achieved because the agents are engaged in different coalitions and the system lack the resources to recover and resume its execution. Next, we discuss how to systematically analyse agent failure and ensure correctness of agent collaboration even in presence of failures using our analysis method, HAZOP, adapted for the analysis of the dynamic behaviour.

IV. SYSTEMATIC ANALYSIS OF CORRECTNESS OF AGENT COLLABORATION

HAZOP – Hazard and Operability Study – is a well-established technique in safety analysis [1][2]. It was originally developed in chemical industry. HAZOP provides a group of safety experts with a structured basis for brainstorming possible deviations in the behaviour of the system and analysing their impact on safety. As a result of performing HAZOP, the safety experts typically identify hazards associated with the system and propose the means to mitigate them.

HAZOP defines a list of guideword that can be systematically applied to certain system parameters to identify whether the deviations in these parameters can cause safety hazards. The list of the guidewords is presented in Table I.

TABLE I. LIST OF GENERIC HAZOP GUIDE WORDS

Guideword	Interpretation
No/None	Complete negation of the design intention. No part of the intention is achieved and nothing else happens
More	Quantitative increase
Less	Qualitative increase
As Well As	All the design intentions is achieved together with additions
Part of	Only some of the design intention is achieved
Reverse	The logical opposite to design intention is achieved but something quite different happens
Early	Something happens earlier than expected relative to clock time
Late	Something happens later than expected relative to clock time
Before	Something happens before it is expected, relating to order of sequence
After	Something happens after it is expected, relating to order or sequence

Table I presents the generic guideword list from the Defence Standard 00-58 [1] and IEC-61882 [2]. Since the HAZOP method has been used in different domains, it has received several interpretations that allow the engineers to focus on a wide spectrum of aspects – from human errors to software.

To analyse the dynamic aspect of the system behaviour, we can interpret the guidewords in a variety of ways. While choosing the interpretation, we aim at understanding how the deviations in the agent behaviour and interactions in a coalition affect the likelihood of achieving the desired goals. In this paper, we adopt the reinterpretation of the HAZOP

guidewords proposed in [3]. The adopted interpretation of the HAZOP guidewords [3] focuses on the message exchange between the agents, as shown Table II.

Let us now explain how to apply the guidewords to the basic scenario of the agent interactions. We present the examples illustrating the situation in which the deviations in the agent behaviour result in a failure or a delay in achieving the desired goals.

Messages outgoing from the coordinator:

Invite message:
No: Execution of scenario is not triggered
Before: Message sent when the goal is not triggered
Earlier: Message sent before the goal is triggered
Later: Message sent with the delay

Messages from the agents:
Confirm participation
No: Message might block execution of the goal if no other agent confirm
After: Message delays execution of scenario

Inter-agent Communication Message:
No: No message is sent after completing execution: Deadlocks goal execution
More than: several messages sent after completing execution: scenario is executed in wrong order
Before /Early: message is sent before task completes and triggers earlier than required execution of tasks in another agents
Later: execution of the goal is delayed.

TABLE II. INTERPRETATION OF HAZOP GUIDE WORDS

Attribute	Guideword	Interpretation
Predecessor/ successors during interactions	No	Message is not sent
	Other than	Unexpected message sent
	As well as	Message is sent as well as another message
	More than	Message sent more often than intended
	Less than	Message sent is often as intended
	Before	Message sent before intended
	After	Message sent after intended
	Part of	Only a part of a set of messages is sent
	Reverse	Reverse order of expected messages
Message timing	As well as	Message sent at correct time and also incorrect time
	Early	Message sent earlier than intended time
	Later	Message sent later than intended time
Sender/ receiver objects	No	Message sent but never received by intended recipient
	Other than	Message sent to wrong recipient
	As well as	Message sent to correct recipient and also an incorrect recipient
	Reverse	Source and destination are reversed
	More	Message sent to more recipients than intended
	Less	Message sent to fewer recipients than intended
Message guard conditions	No/none	The conditions is not evaluated and can have any value (omission)
	Other than	The condition is evaluated true whereas it is false, or vice versa (commission)
	As well as	The condition is well evaluated but other unexpected conditions are true
	Part of	Only a part of conditions is correctly evaluated
Message guard conditions (cont.)	Late	The conditions is evaluated later than required (other dependant conditions have been tested before)
		The conditions is evaluated later than correct synchronisation with environment
Message parameters/ return parameters	No/None	Expected parameters are never set/returned
	More	Parameters values are higher than intended
	Less	Parameter values are lower than intended
	As Well As	Parameters are also transmitted with unexpected ones

 ISBN: 978-1-61208-575-3

	Part of	Only some parameters are transmitted Some parameters are missing
	Other than	Parameter type/number are different from those expected by receiver

Our analysis allows us to derive recommendation how to mitigate the impact of deviations. For instance, it clearly demonstrates that a message omission leads to the system deadlock. Therefore, a timeout mechanism should be implemented to ensure that the goal execution progresses despite possible message omissions.

If an agent sends a confirmation of a task completion then the consequent task might start in an incorrect state. To mitigate this hazard, a coordinator might additionally check to ensure that the required task was indeed completed.

Our analysis of the deviations in the agent behaviour allows us to derive the following recommendations to ensure correctness of agent interactions in the collaborative environments:

- Implement acknowledgement and timeout mechanisms on the communication between the collaborative environment manager and the coalition coordinators during the goal triggering
- Implement acknowledgment, timeout and resend mechanism between the collaborative environment manager and the coalition coordinators for the task completion communication
- Ensure that a reliable level of connectivity is maintained in the collaborative environment to support inter-agent communication.

V. CONCLUSION AND RELATED WORK

In this paper, we have proposed a general model facilitating reasoning about correctness of agent interactions in the collaborative environments. Our analysis is based on formal definition of relations between the goals that collaboration should achieve and states of the agent. A formalization of a goal-oriented development was proposed in [6]. In this paper, the focus was not only on formal representation of relationships between the agents and goals but also on the systematic analysis of deviations.

An approach to integration with other techniques for safety analysis was proposed in [8]. This work is relevant to a high-level analysis of collaboration. An approach to analysis of collaborative behaviour in the context of mode-rich systems was proposed in [9]. The focus of this work was on reasoning about modes of collaborating components.

A formalization of agent collaboration has been performed in [7]. The focus of this work was on tolerating temporal agent failures, while in our work we focused on systematic analysis of deviations in component interactions.

HAZOP analysis has been adapted to analyse human computer-interactions, as well as process deviations. Our use of HAZOP is similar to the former and allows us to reason about interactions of components participating in collaboration.

In this paper, we proposed a systematic approach to analyse agent interactions in collaborative environments. We formally defined relationships between the state of agents and ability of coalition to achieve the required goals. We have demonstrated that the HAZOP method allows us systematically study deviations in the agent interactions and establish a link between errors in interactions and goal achieving.

As a future work, it would be interesting to apply the proposed approach to complex collaborative environment from the Internet of Things domain.

REFERENCES

[1] DefStan00-58, HAZOP studies on systems containing programmable electronics. Defence Standard, Ministry of Defence, UK, 2007.

[2] IEC61882, Hazard and operability studies (HAZOP studies). Application guide. International Electrotechnical Comission, 2001.

[3] J. Guiochet, D. Martin-Guillerez, and D. Powell, “Experience with Model-Based User-Centered Risk Assessment for Service Robots”, HASE 2010: pp. 104-113.

[4] A. van Lamsweerde, “Goal-Oriented Requirements Engineering: A Roundtrip from Research to Practice”, Proceedings of RE’04, 12th IEEE Joint International Requirements Engineering Conference, Kyoto, Sept. 2004, pp. 4-8.

[5] OMG-UML2.OMG unified modeling language (UML). Superstructure, v2.1.2, Onject Management Group.

[6] I. Pereverzeva, E. Troubitsyna, and L. Laibinis, “Formal Goal-Oriented Development of Resilient MAS in Event-B”, in Proc. of Ada-Europe 2012 --17th International Conference on Reliable Software Technologies. Lecture Notes in Computer Science 7308, pp. 147–161, Springer, June 2012.

[7] L. Laibinis, E. Troubitsyna, A. Iliasov, and A. Romanovsky, “Rigorous Development of Fault-Tolerant Agent Systems”, in M. Butler, C. Jones, A. Romanovsky and E. Troubitsyna (Eds.), Rigorous Development of Complex Fault-Tolerant Systems. Lecture Notes in Computer Science, vol. 4157, pp. 241-260, Springer, Berlin, November 2006.

[8] K. Sere and E. Troubitsyna, “Safety Analysis in Formal Specification”. In J.M. Wing, J. Woodcock, J. Davies (Eds.) Proc. of FM'99 - Formal Methods: World Congress on Formal Methods in the Development of Computing Systems, Lecture Notes in Computer Science 1709, pp. 1564 – 1583, Springer, France, September 1999.

[9] A. Iliasov, E. Troubitsyna, L. Laibinis, A. Romanovsky, K. Varpaaniemi, D. Ilic et al, “Developing Mode-Rich Satellite Software by Refinement in Event-B”, Science of Computer Programming, 78(7), pp. 884-905, 2013.

[10] Internet of Things, [Online]. Available from: www.internet-of-things.eu/ March, 2017.

 ISBN: 978-1-61208-575-3