Handling the Misunderstanding in Interactions: Definition and Solution

Phuong Thao Pham, Mourad Rabah, Pascal Estraillier

Handling the Misunderstanding in Interactions: Definition and Solution

Phuong Thao Pham
L3I Laboratory
University of La Rochelle
La Rochelle, France
phuong_thao.pham@univ-lr.fr

Mourad Rabah
L3I Laboratory
University of La Rochelle
La Rochelle, France
mourad.rabah@univ-lr.fr

Pascal Estraillier
L3I Laboratory
University of La Rochelle
La Rochelle, France
pascal.estraillier@univ-lr.fr

***Abstract*— In this paper we introduce our explorative work on the concepts and taxonomy of the misunderstanding in interaction and on the architectural mechanism solving this problem in interactive systems. System actors interacting during the software execution may face misunderstandings when the internal data of at least one actor evolve differently from other actors' data, after the same sequence of interactions within a common context. We formally define the misunderstandings in interaction and propose an architecture including mechanisms to detect, treat and avoid them. These mechanisms insure the consistency of system's actor data at the end of each interaction sequence in the application scenario.**

Interactive system; adaptativity; misunderstanding; agent-based approach; fault-tolerance

I. INTRODUCTION

Early in the history of computer science, the users looked forward to affect dynamically programs execution according to their actions. This promoted the interactivity and gave rise to the category of interactive systems. Games and simulators are examples of such systems. In these systems, the users and the internal agents can modify systems content and progress in real time through input adjustments. The evolution of interactive system introduced the need to adapt the system execution not only to user's actions, but to user's behaviors. They hence became adaptive to support the execution of adaptive applications. Our work deals with the management of the interactivity in interactive systems with adaptive execution.

The interactive applications adaptativity is based on three logics: i) logic of application's designer called the designer logic – what the system is supposed to do, ii) system's internal logic, called the system logic and describing system behavior according to system parameters – how the system works, and, iii) logic of application's user called the user logic – what the user is trying to do. The interactive systems involve advanced human-machine interactions and storylines based on predefined application scenarios. In order to perform the adaptativity, the system must capture users' behaviors from their interactions. Then, according to system logic and designer logic, the system adjusts its execution to what it perceives of users' logic. In general, the execution process of an interactive system is not constant, even not predictable.

In an interactive system, the human users are the main participants. Besides, there are also other types of actors, such as virtual characters in video games, narrator who manages the application scenario plot domain expert, system components… With different roles in the system, they represent the class of **actors** involved in system's interactions. From this point of view, [1] defines the notion of interaction as "*an action carried out by an actor towards another actor and modifying the state of this last. An interaction uses a shared active resource…*" Obviously, actors' behaviors are the important elements that influence the overall system's progress and content.

Our work focuses on the system logic consistency management according to actors' behavior interpretation. The system may misunderstand actors' intentions. This may lead to an erroneous interpretation of their behavior and an erroneous adaptation of system execution. This misunderstanding may concern user-system interactions, but it can also appear in any kind of interaction between any system's actors. The misunderstanding may results from the incomplete actors' data or the non-determinism of actors' behavior.

In this paper, we define what the **misunderstanding in interaction** is and how it arises, in order to, on one hand, detect and treat it, and on the other hand, prevent it structurally. Our approach for the misunderstanding management is based on: i) system architecture definition, to build the system overall structure, ii) events and interactions observation to detect the misunderstandings, and iii) consistency mechanisms to treat/avoid the misunderstandings. To detect, treat and avoid the misunderstanding in interactions, we suggest adding the appropriate mechanisms in system architecture that will control actors' interactions and insure the coherency between actors' and system's states all along the execution. These mechanisms are inspired by fault-tolerant techniques, since misunderstandings may be seen as threats to system service providing according to dependability point of view. Our aim is to elaborate generic and reusable mechanisms avoiding the user-system misunderstandings and to include them into classical interactive system architecture. We suggest in this papers way to achieve this.

II. RECENT WORK

In the recent research, we can find several works dealing with the user-system dialogue where the communication is done through a real human language [2-5]. According to Rapaport [4], negotiation is the key to understanding. A cognitive agent understands by negotiating with the interlocutor or by hypothesizing the meaning of an unknown word from the context… A cognitive agent can negotiate with

Published in Proceedings of the 2nd International Conference on Software Engineering & Applications (SEA'2011), Singapore, pp. 47-52, december 2011

itself about something external by comparing it perception and internal knowledge in order to change or correct its own misunderstandings. Other works propose to use confidence scores to measure the reliability of each word in a recognized sentence [6]. Besides, Lopez-Cozar proposed to implement a frame correction module, which is independent of speech recognizer [3]. This module corrects misunderstandings in a sentence, caused by the errors in speech recognition, by replacing the incorrect frame with an adequate one.

Karsenty and Botherel applied the adaptable and adaptive transparency strategies to TRAVELS project with the goal of helping the uses to understand and react appropriately to system rejections and misunderstandings [2]. The ability of making the system's interpretations explicit and informing users on how to correct misunderstandings are two ways to help users handle the occurred problems. This strategy is very effective in misunderstanding detection and raises the rate of appropriate user responses after system rejections.

All of these works deal with the problem in speech dialogue where the misunderstandings are the more frequent. But the misunderstanding can be found in other forms of interaction like actions, gesture… Moreover, our question is how we can treat the misunderstandings between the actors themselves, besides the user-system misunderstandings. It is not easy to recognize such class of misunderstandings

III. MISUNDERSTANDING IN INTERACTIONS

A context is "*a set of information that can be used to characterize the situation of an entity*" [7]. An entity in our work is an actor involved in the interaction. A situation is an interaction sequence involving several actors and defining the interaction context. Thus, an interaction during system's execution is carried out between at least two actors, but within a common context.

A. *Actor's Local Vision and the Context*

A lot of work proved that the context is related to the human's activities [8-9]. There is interdependence between the common context and the actors located in this context. An actor performs its activities depending on the current situation and the available contextual information [10]. Each actor has to observe and perceive the world, to interpret with its own logic, to combine with its existing knowledge to construct its own contextual vision and to update the new knowledge. This vision is called the actor's "**local vision**".

The **local vision** is actor's own knowledge about the extern world (environment, resources…), the relations with other actors (other actors' states), and its own profile (internal state). This local vision can be represented by a **state vector**. Thanks to the local vision, actor's actions are not simply feedback. The actors are now able to interact with the others more intelligently with strategy and coordination. However, the local vision is not static. It evolves during interaction sequences. The perceived data are not always identical between different actors due to different capacity of cognition. As consequence, their local visions may start to differ, then become inconsistent. That can lead to different comprehensions of a same fact (a sentence, an action, a state…). If the actors use this inconsistent data in future interactions, that may lead to a misunderstanding.

We give the following definition: *"Two actors are in a misunderstanding state when: they are in interaction with each other and there are incoherent data in their local visions about the same fact. A fact is considered as objective data or absolute reference to system, actors or resources states."* If we consider the interactions between two actors like the acts of language, the misunderstanding can be observed when two actors think that they talk about the same thing whereas they actually talk about different things [4].

B. *Formal Presentation*

Many works, as [11-12] use the finite state system to represent interactive systems. Besides, the linear logic and Petri net has been used by [1] to model the actions and the scenario. We choose the linear logic ([13]) to formalize the misunderstanding because it allows representing states by atoms. Let two actors A and B interacting in the presence of the fact *f* from the extern world. The atom E_f^{real} is the absolute reference to *f*. The knowledge perceived by A and B about the fact *f* is presented by the atoms E_f^A and E_f^B. The perception can be seen as an internal action which cannot be observed by the other actors. The perception of A and B about *f* is shown in the figure 1 and two following logic formulas:

$$A : E_f^{Real} \multimap E_f^A$$
$$B : E_f^{Real} \multimap E_f^B$$

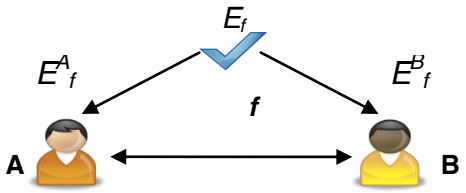

Figure 1 - Actors A and B perceive the fact f in interaction

Misunderstanding in interaction appears when E_f^A is different from E_f^B: $E_f^A \neq E_f^B$

The distance D_f^{AB} measures the difference level between E_f^A and E_f^B: $D_f^{AB} = |E_f^A, E_f^B|$

The ideal situation, e.g. without misunderstanding, is when the perception of A and B on a fact *f* is identical:

$E_f^A = E_f^B$ and the distance $D_f^{AB} = \emptyset$

C. *Classification of Misunderstandings*

We have identified several kinds of misunderstandings.

1) According to dimension: actors' point of views.

- **Symmetric** misunderstanding: none of the actors has correct knowledge on *f*.

  $E_f^A \neq E_f^{real}$, $E_f^B \neq E_f^{real}$ and $E_f^A \neq E_f^B$.
- **Partial** misunderstanding:
  - From A point of view: $E_f^A \neq E_f^{real}$ and $E_f^B = E_f^{real}$
  - From B point of view: $E_f^A = E_f^{real}$ and $E_f^B \neq E_f^{real}$

2) According to severity: misunderstanding consequences on the interactions define two boundary levels.

- **Minor**: non blocking misunderstanding, without significant effect on actors interactions.
- **Catastropic**: misunderstanding causing interaction interruption and/or deadlock.

3) According to frequency: misunderstandings reccurence.

- **Seldom** misunderstanding: rare or single.

- **Frequent** misunderstanding: reccurent.

4) According to revelation: misunderstanding perception.

- **Revealed** misunderstanding: detected by an algorithm or perceived as consequence of the interaction sequence. The revealed misunderstanding is called **active** when it may produce a deadlock in interaction. Otherwise, it is **dormant**.
- **Latent** misunderstanding: not yet detected misunderstanding that may rise, propagate or create new misunderstandings.

D. What Elements Cause Misunderstandings?

Misunderstandings in interaction has various causes

1) Different references: When interacting actors have different contexts. The interactions between actors are carried out under a concrete context that influences their behaviors. The actors locating in different reference worlds will talk about different things. For instance: the word "*bug*", is a kind of insect. But in the computer world, a "*bug*" refers to an error, mistake or fault that produces an incorrect program execution. If the interaction context is not enough clear, the actors' local visions will not be synchronized and misunderstanding conditions may be established.

2) Different logic: The actions of an actor also depend on his own logic which is the deduction rules. For exemple: Two actors interact about the identity of some "old person". For the actor A, an old person means a person over 60 years: old(x) $\Rightarrow$ age(x)>60year). For the actor B, an old person means the oldest known person: old(x) $\Rightarrow \forall$y age(x)$\geq$age(y). If B asks A for an old person, B will expect the oldest person, whereas A will just provide someone old but not especially the oldest. If B asks again, A may provide different answer and A and B will be in a misunderstanding since each actor has his own logic.

3) Semantic ambiguity: The wrong interpretation during the interactions can bring to a different perception. Semantic is "internal" [4], the external world is reflected subjectively in actor's "mind" that creates its own "narrow" knowledge. It is obvious that an actor can interpret as correct or wrong a fact because of the lack of information or the imperfection of observation. For instance: in an e-learning application, a camera has to check student presence. Due to the limitation of the camera scope, a student may be warned because of his absence even he is still there but out of the camera scope.

E. Consequences

Misunderstanding in interaction has various outcomes.

1) Intearction deviation: The interaction chain between two actors diverge from the planned scenario. An actor can estimate incorrectly the state of his interaction partners because of misunderstandings. As result, the actor will make a wrong decision based on the wrong observed state of its partners. Instead of an appropriate action according to the planning, the actor's behavior will diverge from its normal logic and from the logic of other interacting actors.

2) Interaction Deadlock: This problem arises when the misunderstanding is revealed and the actors, of course, get stuck in the middle of the interaction. In this case, an actor receives an answer or a demand that he does not expect, because he is expecting some others reaction. The interaction sequence will be broken. Both or each actor does not know what to do anymore.

3) Propagation of misunderstanding: If the misunderstanding is not detected or revealed, it can be propagated all along the scenario and the execution of the application. Furthermore, the misunderstanding severity may increase.

IV. MEASURES

To estimate misunderstanding importance and assess the efficiency of misunderstanding handling techniques, we propose a set of measures. These measures are inspired by the dependability domain [14].

A. Interaction States

The interaction between the actors can be in one of the three states according to the misunderstandings and their handling. The Markov chain in figure 3 shows the transition between the interactions states. Let $X \in \{-1, 0, 1\}$ the variable associated to the interaction states.

- Stable state ($X = 1$): nominal interactions, without misunderstanding or with potential misunderstanding but non-revealed.
- Adapted state ($X = 0$): adapted (or reinforced) interactions in order to avoid the misunderstandings.
- Misunderstanding state ($X = -1$): the interactions interrupted or deviate totally from the scenario plot with revealed misunderstandings.

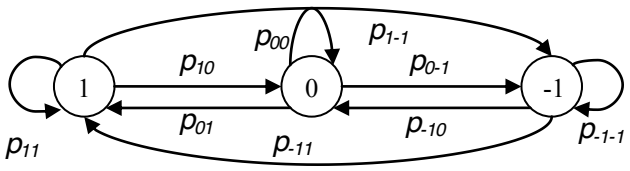

Figure 2 - Transition between the interactions states

p_{10}: adaptation rate in order to avoid misunderstanding occurrences; $p_{1\text{-}1}$: probability that the actors falls in misunderstanding; $p_{\text{-}11}$: rate of successful adaptation and repair. The interactions are restored from misunderstanding state through reparations, and from adapted state through adaptations.

From this Markov chain, we can estimate the efficiency of coherency management mechanisms and the robustness of system architecture from the transition probabilities.

- A reliable system without misunderstanding in interaction should have $p_{\text{-}1\text{-}1} = 0$ and p_{11} high.
- The adaptation is efficient if p_{00} is low and p_{01} higher than $p_{0\text{-}1}$.
- The repair from misunderstanding is efficient if $p_{\text{-}11}$ or $p_{\text{-}10} > 0$, and $p_{\text{-}1\text{-}1}$ is very low.

B. Variables Related to Misunderstandings

To complete the measures by probabilities, we introduce some criteria for the static measures, included variables, rates, and estimators.

- θ_k: Time to the k^{th} misunderstanding after the $(k\text{-}1)^{\text{th}}$ reparation (θ_1: time to the first misunderstanding).
- λ_k: Time to the k^{th} repair after the k^{th} misunderstanding.
- $Y(t), Y$: Cumulative number of all misunderstandings between initial moment and t, between initial and final moment.

C. Static Average Estimators

- N: number of tested interaction sequences, $[i]$: statistic of the i^{th} sequence.
- MTFM (Mean Time to First Misunderstanding):

$$MTFM = \frac{1}{N}\sum_{i=1}^{N} \theta_1[i]$$

- MTBM (Mean Time between Misunderstanding) :

$$MTBM = \frac{1}{N}\sum_{i=1}^{N} MTBM_i = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{Y_i - 1}\sum_{k=2}^{Y_i} \lambda_{k-1}[i] + \theta_k[i]$$

A reliable system should have MTFM and MTBM as high as possible. If MTFM and MTBM are low, that means that the system falls in misunderstanding easily or frequently.

D. Rate of Misunderstanding

- $N_1(t)$: number of sequences with misunderstanding from initial moment to t
- $N(t)$: number of sequences with misunderstanding at the moment t
- Initial misunderstanding rate: $\omega_1(t) = \frac{N_1(t)}{N}$
- Misunderstanding rate at time t: $\omega(t) = \frac{N(t)}{N}$
- Misunderstanding intensity during interaction sequence: $\omega_m = \frac{1}{N}\sum_{i=1}^{N} Y[i]$

A system which is well prevented and repaired from the misunderstanding if $\omega_1(t), \omega(t), \omega_m \approx 0$

V. How to Manage Misunderstandings?

A. Necessary Misunderstanding Occurrence Conditions

An interaction can be in misunderstanding if:

$\mathcal{C}_1$ - *Condition of actors' presence:* At least two actors participate in interaction sequence. The misunderstanding occurs only when actors interact with each other.

$\mathcal{C}_2$ - *Inconsistency of local data:* The knowledge E_f^A *and* E_f^B are totally different or contain a part of different data. There is data inconsistency in the actors' local visions.

$\mathcal{C}_3$ - *Data Sharing:* Different data E_f^A *and* E_f^B is used as shared information or common contents between participant actors during the interactions. If this shared data is neither declared explicitly nor synchronised before the interaction dialogue begins, no one can detect possible misunderstanding.

If these three conditions are met, the misunderstanding will occur. If two actors have inconsistent data, but they never interact with each other, the misunderstanding will never arise. Moreover, they may have different data about a same fact, but if they do not use it as shared data during the interaction, they will not face misunderstanding.

B. Handling Approaches

The aim of misunderstanding management is to avoid misunderstanding occurrence as much as possible and, in the case that the misunderstanding happens anyway, it should be eliminated. In addition, before the misunderstanding is detected, the interactions between two actors may have already deviated from the planned scenario. We must intervene to synchronize their data and their behaviors. We classify the misunderstanding management into three classes:

1) Ignoring: Just ignore the problem if the misunderstanding is minor. It is like the Ostrich Algorithm in deadlock treatment.

2) Prevention: Prevent misunderstanding occurrence by denying one of the three necessary conditions mentioned previously. If one condition is missing, we decrease the possibility of misunderstanding occurence. Hence, for each condition:

$\mathcal{C}_1$: Separate the actors containing the potential misunderstanding in their local visions. If it is possible, it is better not to put them together in interaction.

$\mathcal{C}_2$: Actors' local vision data should not contain inconsistency. Ideally, their knowledge should be identical and coherent all along the interaction. To reach this, the actors' data consistency should be checked after each sequence of interactions and synchronized.

$\mathcal{C}_3$: Declare explicitly and check the consistency of the shared data necessary to a given interaction before the dialogue between the actors begins. Another way is to isolate the different data and avoid its use in during the interactions.

3) Tolerance: Detect the latent misunderstanding during the interaction and resolve it when it became active.

$\mathcal{T}_1$: Regularly check actors' local vision data. The aim of this step is to detect and to eliminate both latent and revealed misunderstandings, if possible, before interaction deadlock.

$\mathcal{T}_2$: The misunderstanding can affect the interaction result by a deviation or a deadlock. Hence, we have to cure this situation by appropriate handling mechanism. Either the system rollbacks to a misunderstanding free state in order to retry the interaction or it continues but with reinforcement actions synchronizing actors' threads, in the most possibly transparent manner for the user, but with respect to the designer's storyline.

C. Misunderstanding, Dependability and Adaptativity

Misunderstandings are similar to the threads (fault, error, failure) affecting system service in the dependability domain. For instance, the byzantine or inconsistent failure happens when some or all the system users perceive differently service correctness [15]. The "automation surprise" and "mode confusion" occur when the system behaves differently than its users expect [12]. These examples show the effects of different users' or actors' perceptions in the system. The principles of misunderstanding tolerance are similar to the fault-tolerance with error detection and system recovery [15]. The implemented mechanisms track down the system service deviation, and put the system into degraded mode or restoration., We suggest adapting fault-tolerant techniques to misunderstanding management in interactive applications.

To prevent the conditions $\mathcal{C}_2$ & $\mathcal{C}_3$, our work will focus on coherency mechanisms and software design structure to improve actors' perception in order to identify more precisely the common context, the system and the others actors' states. Adaptativity is an advanced characteristic of interactive systems that helps the system to reinforce the interaction efficiency [16-19]. It is the aptitude of the system, on one side, to modify or adapt its logic (behaviors, interfaces…) to the users' behaviors, and on the other side, to orient the users' behaviors towards the nominal ones defined in the system logic according to the designer logic. The adaptativity can help the system to solve active misunderstanding case by proposing reinforced interactions in $\mathcal{T}_2$ which are appropriate to insure the final objectives. Furthermore, because the system

has to observe the actors, a good observation will help the system to estimate and perceive more correctly the state of the actors so that the system can detect itself if it misunderstands the actors. The system can also be a middle agent between the actors to detect the inconsistency between them. From the classification of methods in the previous section, we will follow two principal directions: adaptativity and fault tolerance, not in a separated but in combined way.

VI. CONTRIBUTION

A. General Architecture

We propose a robust system architecture with additionnal specific components that ensure misunderstanding in interaction detection and management. We also propose a mechanism of static and structural analysis installed into the proposed architecture to prevent misunderstandings. Finally, in order to handle revealed misunderstandings, we integrate the adaptive treatment mechanismes to the dynamic system's execution. They are inspired and adapted from fault-tolerance techniques cause misunderstanding nature is similar to the error and fault treatment in dependability domain.

Several architecture models have been proposed according to the specific purpose of each work [16-17], [20-22]. We chose the approach in [19] that is a multi-agent-based architecture as a starting point. The advantage of this approach is that each agent can be organized and work autonomously and strategically. We added a special agent called **script agent** besides the adaptation unit to manage the consistency. Figure 3 shows our overall architecture.

1) Observer agent: observes user's behaviors and state, formalize, normalize and transfer them to the scenario agent.

2) Scenario agent: makes decision about scenario orientation according to user's state, planned scenario and permanent objective defined by the designer. This agent tries to find the best way to evolve the application execution. Scenario agent takes charge of a library of "situations" planned by the designer. These "situations" represent scenario components and are all different interaction and activity sequences that can take place in the application, as for instance all the scenes possible in a theater play.

3) Director agent: then receives the decision taken by the scenario agent. In its turn, it will take charge of the production of adaptive scenario such as a modification, an answer, an action adapting to users…

4) Script agent: tracks inconsistency, in 3 steps:

Detection: Detect, confine or partition the inconsistency between the actors in a situation in order to identify the causes of misunderstanding.

Treatment: Apply the mechanism or strategy to remove the inconsistency, to correct the deflected state that brought the situation into the incoherence.

Evaluation: Estimate the efficiency of treatments in order to improve applied mechanism for the next time.

B. Consistence Management

When has the script agent to accomplish its tasks? As said in the previous section, we divide the execution of an interactive application in "situations". A situation is a sequence of interactions between several actors within a certain common context. To enter into a situation, some conditions called "pre-conditions" must be satisfied. And to leave it, it must product some results during the interactions, which fulfill some other conditions called "post-conditions". The confinement of interactions in a situation allows us to set up the treatment mechanisms for ambiguity and misunderstanding inside a "situation". The script agent is placed between the participating actors to synchronize the data and detect the inconsistency. The tasks of consistency manager are: i) to prevent misunderstanding occurrences, ii) to tolerate the misunderstanding in the case of their occurrences, iii) to eliminate both latent and revealed misunderstandings.

If we divide a situation into three phases, here is how our mechanism does work:

1) Prologue phase: To avoid the first cause of misunderstandings – different references, the script agent make the explicite declaration of all the actors' local visions in relation with the interacting content, before the interactions begin. If the knowledge data of all actors are the same since the beginning, they may will talk about the same subject, reducing the possibility of misunderstanding. If the inconsistency exists, a negotiation step will be established between two actors. Then either one or several of them will modify its/their data or the different data will be isolated and not considered in the interactions.

2) Interaction phase: when the interactions are carried out, the actors will update their local data step by step, since they always observe and perceive each other. Despite of agreement in initial local visions, misunderstanding may anyway occur during the interactions. It is why the local data is synchronized all along the interaction to avoid different data in local visions.

3) Epilogue phase: at the end of the second phase, all the interactions have been done. If the post-conditions are

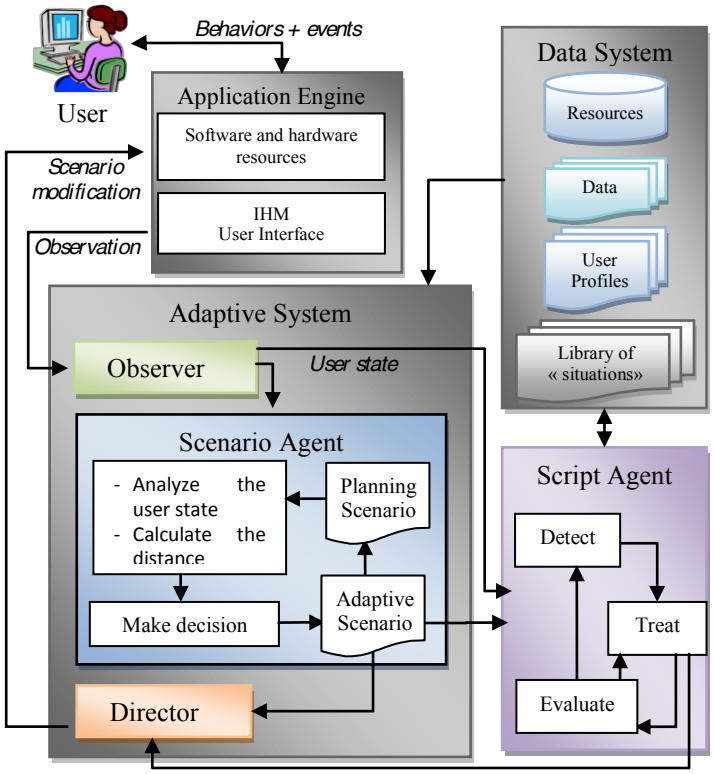

Figure 3. General proposed architecture of interactive system

fulfilled, the situation terminates with the expected results. But, if, for some reason, the post-conditions are not reached, the script agent has to intervene. It has to detect and settle the existing incoherency in order to avoid the propagation of misunderstanding to next situations. The script agent can also propose to the actors to do some supplementary interactions, called the reinforcing interactions, or if necessary, bring the whole situation back to a previous stable state. The final goal of this phase is to quit the situation with the appropriate post-conditions and without latent or active misunderstanding.

VII. CONCLUSION AND PERSPECTIVE

In this paper, we presented our work on actors' interaction misunderstanding in interactive systems. We have defined how the misunderstandings in interaction arise and what consequences that has on left interactions and interacting actors' behaviors. There is misunderstanding when involved actors' local visions about same references start to differ. This may happen, for instance, when actors' data are not on the same granularity level. Among the consequences, we can mention: i) system reactivity slow down, since the system has to compensate the actors local data deviation; ii) players exasperation, since the system does not react as expected; iii) system total or partial deadlock, since the system does not understand what the player is doing.

We showed that the misunderstandings in interactive system research are not explicitly considered in existing work. We proposed to add appropriate mechanisms to interactive system architecture in order to avoid them. These mechanisms are consistency manager components that ensure i) data consistency between interacting actors during the interaction sequences, and ii) data synchronization at the end of the interaction sequences in order to attest a misunderstanding free ending. We have described our architecture including these mechanisms and making the system adaptive even if initially it was not. Indeed, the constancy management can be seen as adaptation mechanism controlling the interactions.

Our approach is based on the notion of "situation" that represents a set of interactions between several actors within a certain context. We use this notion as structuring element of overall system's execution. The aim is to define all system scenarios as situation sequences in a situation graph. The situation structuring and formalization is the other axis of our perspective work. Our work is still in progress. We mentioned that the misunderstanding problem in interactive systems looks like the dependability threats in dependability domain. Thus, we are exploring in our current work to what extent goes this similarity and how we can reuse the fault-tolerance principles in misunderstanding avoidance and treatment. To do this, we are comparing systems proprieties and requirement between interactive and fault-tolerant systems in order to establish their similarities and differences. This will lead us to identify what fault-tolerant mechanisms can be reused and readapted in interactive systems field.

To validate our approach we are applying our architecture in an e-learning project framework. The project is devoted to the development of a virtual classroom and a set of pedagogic e-tools. In this environment, teachers and learners will carry out learning session as in a classical classroom. However, they will face up to misunderstandings due to the system's interfaces and systems mechanisms to catch and manage users' behaviors. In this framework, we aim to apply our architecture and the defined coherency management mechanisms to improve the interactions between teachers and learners and to increase the final system's efficiency.

REFERENCES

[1] A. Prigent, R. Champagnat, and P. Estraillier, "Scenario building based on formal methods and adaptative execution," *ISAGA'2005 (International Simulation and gaming association) Georgia Institute of technology*, pp. 1-19, 2005.

[2] L. Karsenty and V. Botherel, "Transparency strategies to help users handle system errors," *Speech Communication*, vol. 45, no. 3, pp. 305-324, Mar. 2005.

[3] R. López-cózar, Z. Callejas, N. Ábalos, G. Espejo, and D. Griol, "Using Knowledge about Misunderstandings," *Speech Communication*, pp. 523-530, 2010.

[4] W. J. Rapaport, "What Did You Mean by That ? Misunderstanding , Negotiation , and Syntactic Semantics," *Cognition*, pp. 397-427, 2000.

[5] H. Wimelius, "Fundamentals of User Perception and Interaction : Environmental Psychology applied in a study of web pages," vol. 2, no. 3, pp. 282 - 303, 2004.

[6] H.Jiang, "Confidence Measures for Speech Recognition: A survey," *Speech Communication*, vol. 45, no. 5, p. 455_470, 2005.

[7] A. K. Dey, "Understanding and Using Context," *Personal and Ubiquitous Computing*, vol. 5, no. 1, pp. 4-7, Feb. 2001.

[8] P. Dourish, "What we talk about when we talk about context," *Personal and Ubiquitous Computing*, vol. 8, no. 1, pp. 19-30, Feb. 2004.

[9] B. Hommel, "Contextualization in perception and action," *Psychologica Belgica*, vol. 40, no. 227-245, 2000.

[10] F. Picard and P. Estraillier, "Context-dependent Player's Movement Interpretation – Application to Adaptive Game Development," in *3D Image Processing & Applications*, 2010.

[11] D. Javaux, "A method for predicting errors when interacting with finite state systems. How implicit learning shapes the user's knowledge of a system," *Reliability Engineering and System Safety*, vol. 75, no. 2, pp. 147-165, Feb. 2002.

[12] S. Combéfis, P. S. Barbe, and C. Pecheur, "A Bisimulation-Based Approach to the Analysis of Human-Computer Interaction Categories and Subject Descriptors," in *EICS '09 Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems*, 2009, pp. 101-110.

[13] J.-yves Girard, "Linear Logic: Its Syntax And Semantics," in *Advances in Linear Logic*, 1995, pp. 1-42.

[14] M. Giraud, "Sûreté de fonctionnement des systèmes Principes et définitions [Dependability - Principles and Definitions]," *Techniques de l'ingénieur - Electronique,*, vol. 5, pp. 1-19, 2005.

[15] J.-claude Laprie, B. Randell, C. Landwehr, and S. Member, "Basic Concepts and Taxonomy of Dependable and Secure Computing," *IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING*, vol. 1, no. 1, pp. 11-33, 2004.

[16] R. Oppermann, "Adaptively supported adaptability," *Int. J. Hum.-Comput. Stud.*, vol. 40, no. 3, pp. 455-472, 1994.

[17] A. Paramythis, "Adaptive Systems : Development , Evaluation and Evolution," University of Johannes Kepler, 2009.

[18] C. Stephanidis, A. Paramythis, D. Akoumianakis, and M. Sfyrakis, "Self-Adapting Web-based Systems : Towards Universal Accessibility," in *In 4th ERCIM Workshop on "User Interface for All,"* 1998, pp. 1-17.

[19] K. Sahaba, P. Estraillier, and D. Lambert, "Interactive educational games for autistic children with agent-based system," in *4th International Conference on Entertainment Computing (ICEC'05)*, 2005, pp. 422–432.

[20] D. Benyon and D. Murray, "Adaptive systems: from intelligent tutoring to autonomous agents," *Knowledge-Based Systems*, vol. 6, no. 4, pp. 197-219, Dec. 1993.

[21] A. Paramythis, S. Weibelzahl, and J. Masthoff, "Layered evaluation of interactive adaptive systems: framework and formative methods," *User Modeling and User-Adapted Interaction*, vol. 20, no. 5, pp. 383-453, Nov. 2010.

[22] A. Jameson, "Adaptive Interfaces and Agents," *The human-computer interaction handbook: Fundamentals, evolving technologies and emerging applications (2nd ed.)*, pp. 433-458, 2008.