Modeling of Errors Realized by a Human Learner in Virtual Environment for Training

Thanh Hai Trinh[1, 2], Cédric Buche[1], Jacques Tisseau[1]

Université Européenne de Bretagne – ENIB – LISyc – CERV
Technopôle Brest-Iroise, 29238 Brest Cedex 3, FRANCE
(2) Institut de la Francophonie pour l'Informatique
42 Ta Quang Buu, Ha Noi, VIETNAM
E-mail: tthai@ifi.edu.vn; buche@enib.fr; tisseau@enib.fr

Abstract

This study focuses on the notion of erroneous actions realized by human learners in Virtual Environments for Training. Our principal objective is to develop an Intelligent Tutoring System (ITS) suggesting pedagogical assistances to the teacher. For that, the ITS must obviously detect and classify erroneous actions produced by learners during their realization of procedural and collaborative work. Further, in order to better support human teacher and facilitate his comprehension, it is necessary to show the teacher why learner made an error. Addressing this issue, we firstly model the Cognitive Reliability and Error Analysis Method (CREAM). Then, we integrate the retrospective analysis mechanism of CREAM into our existing ITS, thus enable the system to indicate the path of probable cause-effect explaining reasons why errors have occurred.

***Keywords**:* Intelligent tutoring system, Erroneous actions, Retrospective analysis.

1. Introduction

In order to simulate procedural and collaborative work, we previously developed the model MASCARET (Multi-Agent System for Collaborative Adaptive and Realistic Environment for Training) where human learners and agents collaborate to realize a mission (Querrec *et al.*, 2004). Learners are gathered in team consisting of several predefined roles, every role contains a number of tasks to be realized by learners with accurate resources. During realisation of the tasks, it is essential to take into account that human learners could make erroneous actions in comparing to their predefined correct procedure.

In (Buche and Querrec, 2005), we have proposed a model of Intelligent Tutoring System (ITS) whose principal objective is to suggest pedagogical assistances to the teacher adapted to the simulation context and to the learner's behaviours (including erroneous actions). However, this works exclusively concerns errors detection and tagging. Once erroneous actions are detected in our existing ITS, it were be classified in different types (see Figure 1a) whose explications are based on a knowledge base on

classical errors. In order to better support the teacher and facilitate his comprehension, it lacks a model that could explain reasons why the learner made an error.

Our approach bases on the Cognitive Reliability and Error Analysis Method (CREAM) in Human Reliability Analysis field (Hollnagel, 1998). This approach proposed a *classification scheme* which makes a distinction between observations of errors (*phenotypes*, see Figure 1b) and its causes (*genotypes*) classified in three categories: M(an), T(echnology) and O(rganization). The causal links between phenotype-genotype are represented using a number of *consequent-antecedent* links. Finally, the scheme could be associated with both a method of retrospective analysis (the search for causes) and a performance prediction method. However, in our goal of erroneous actions detection and then searching for the causes, we interested in human learner's performance analyses, in other words, in retrospective analyses.

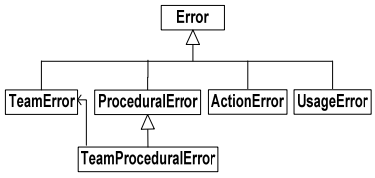

Figure 4a. Errors types in ITS (Buche and Querrec, 2005)

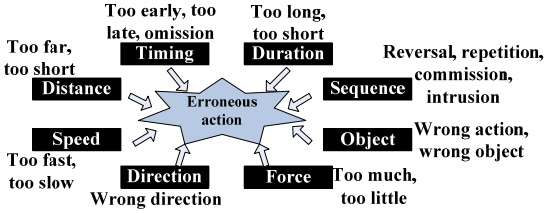

Figure 1b. Dimensions of error modes (Hollnagel, 1998)

Implementation of CREAM was object in the work of (El-Kechaï, 2006, 2007) which firstly proposed a task model named METISSE in order to recognize learner's plans in Virtual Environments for Training (VET), then this model could be used to detect for erroneous actions according to classification of Hollnagel. Nevertheless, implementation of METISSE was not complete, and integration of CREAM into a really ITS was not performed.

In this paper, we will firstly propose an approach to model CREAM (section 2). Next, in section 3, we will present the integration of retrospective analysis mechanism of CREAM into our existing ITS as well as our evaluation. Finally, section 4 summarizes the present work.

2. Implementation of CREAM

2.1. Classification Scheme Representation

There are several graphic tools that permit to keep track of analyses processes such as *CREAM Navigator* developed by (Serwy and Rantanen, 2007). However, this navigator is completely closed in the sense that it does not maintain an explicit

representation of possible errors modes and probable causes. For that, (El-Kechaï and Després, 2007) proposed using a rules base for represent consequent-antecedent links, hence the search for the causes was executed by backward inferences. Limitation of this method obviously lies on the performance of inference mechanism, other problem maybe occurs in adding, removing another potential errors that will demand a considerable modification on the rules base. For our development, as suggested in (Hollnagel, 1998), we intent to separate the analysis method (cf. section 2.3 and 2.4) and the representation of errors modes using a group of four data files in format XML detailed below:

– *Questionnaire.xml*: proposing to represent a list of questions from which we could evaluate the Common Performance Conditions (see section 2.2 in following)

– *Phenotype.xml*: proposing to maintain the phenotypes and its antecedents

```
<Phenotypes>
 <Phenotype name="Time/During" description="...”>
        <GeneralAntecedents>
              <item>Inadequate plan</item>
              <item>Inattention</item>
        </GeneralAntecedents>
        <SpecificAntecedents>
               <item>Earlier omission</item>
        </SpecificAntecedents>
  </Phenotype>...
</Phenotypes>
```

– *Genotype.xml*: containing all possible causes classified in three groups (M,T,O), each group is then detailed into several categories. The important point is that this data file also represents relations between each consequent and its antecedents

```
<Genotypes>
        <Group name="Man">
              <Category name="planning">
                    <GeneralConsequent name=" Inadequate plan" description="...”>
                          <GeneralAntecedents>
                                <item>Distraction</item>
                                <item>Excessive demand</item>
                          </GeneralAntecedents>
                          <SpecificAntecedents>
                                <item>Error in goal</item>
                                <item>Inadequate training</item>
                          </SpecificAntecedents>
                    </GeneralConsequent>
              </ Category >
        </ Group >...
</Genotypes>
```

– *Repartition.xml*: proposing to determine repartition of specific antecedents in three factors (M,T,O) which serves to initialize the mass of each specific antecedent as a probable cause (cf. section 2.4)

```
< Repartition >
        <item name=" Earlier omission" group="Man" description="..."/>
        <item name=" Message misunderstood" group="Technology" description="..."/>...
</ Repartition >
```

Finally, in considering that CREAM is naturally a flexible method and adaptable to different analysis contexts, this strategy of classification scheme representation permits customize the scheme without any modification on analysis method.

2.2. Define the Common Performance Conditions (CPC's)

In CREAM, Hollnagel highlighted that the context strongly influence human actions. It is therefore essential to take into account the description of virtual environment in which the human learner is immersed. The objective is to determine how each factor (M,T,O) influences the training context. Here, we are inspired from the proposition presented in (El-Kechaï and Després, 2007) using a predefined questionnaire which will be answered by the teacher before training session:

```
<Questionnaire>
 <Question name="The visual quality of the interface is it bad?" group="Technology" answer="No"/>
 <Question name="Does the learner have concentration trouble?" group="Man" answer="No"/>...
</Questionnaire>
```

Next, each factor will be assigned one coefficient calculated using formula below:

$$[1]\quad Coefficient(group\ \mathrm{i}) = \frac{\text{Number of } Yes \text{ answers associated to group } i}{\text{Total number of } Yes \text{ answers}}$$

where *group i* is respectively in (Man, Technology, Organization). These values permit define the most probable factor leading to erroneous actions.

2.3. Modelling of Consequent-Antecedent Relations

One advantage of CREAM lies on its recursive analysis approach, rather than strictly sequential in compare with other traditional analysis methods. So that, it also conducts to a non-hierarchical data structure to connect the direct as well as indirect links:

(i) between a phenotype and its antecedent; and (ii) between a consequent and its antecedents. Figure 2 shows our model to represent the connection between consequent – antecedent.

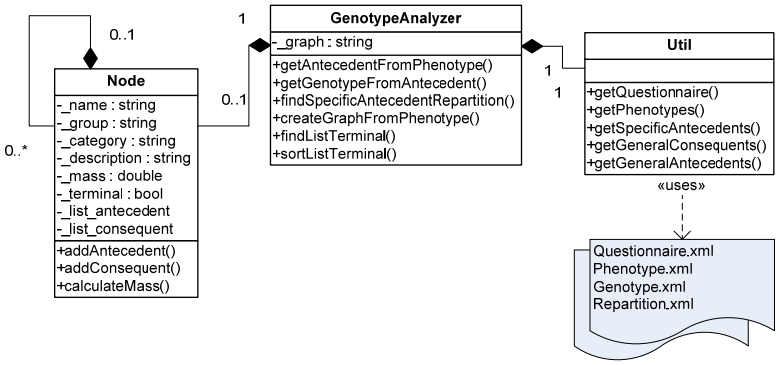

Figure 2. Our UML diagram for modeling consequent-antecedent links

Here, we are going to construct a causal graph where we use the term "*node*" to point to either a consequent or an antecedent. Each node is described by its *name*; the *group* of errors modes that it is associated and its *category* in group; the *description* in text helps better explain the error's semantics in particular context. The boolean attribute *terminal* permit to identify if that is a terminal-cause or not. The most important is that, each node contains two lists: one includes its *antecedents*, other points to its *consequents,* in others words, they represent edges in/out one node in the causal graph. At last, each node must also include a value of *mass* which represent the certitude of choosing this node as a probable cause. The two methods *addAntecedent()* and *addConsequent()* serve for maintaining the two lists of antecedents and consequents of one node. Note that once a node calls the method *addAntecedent()* serving for adding a "*parent*" node like one of its antecedents, this node will also add itself to the consequents list of the "parent" node (using the method *addConsequent()* of the parent node) , the value of the attribute *terminal* then will be set to *false*.

2.4. Search for the Causes

Finally, the retrospective analysis is executed by a *GenotypeAnalyzer* containing *graph* attribute which is initialized by pointing to the initiating phenotype ("*root*" node), then the analyzer calls accurate methods to find the "*root*" causes (the nodes with the attribute terminal having value false). This mechanism is presented below:

Input: *Phenotype of erroneous action*
Initialization: *Construct the "root" node pointing to phenotype input*

Step 1: *Read from file Phenotype.xml, find all general antecedents of phenotype input*
For each antecedent Do
Add it into antecedents list of "root" node

Step 2: *For each unvisited node in the graph Do*
- *Find its antecedents from file Genotype.xml & add them to list*
- *Return step 2. This recursive search terminates when the nodes selected is a specific antecedent node or a general consequent node without antecedents*.

With this algorithm, we finally attain a causal network where each node is associated with its antecedents and consequents. The "*leaves*" are terminal nodes (or "*root*" causes) whose antecedents list is empty. In order to calculate the certitude of choosing each node as a probable cause, we inherit the proposition presented in (El-Kechaï and Després, 2007) using Dempster-Shafer's evidence theory:

$$[2]\ mass(a) = coefficient(g(a)) \times \sum_{\forall c \in Cons(a)} \left(\frac{mass(c)}{\sum_{\forall i \in \{M,T,O\}} (coefficient(i) \times n_{ic})} \right)$$

where:

- *mass(a)*: mass of antecedent *a*
- *g(a)*: group of *a*
- *Cons(a)*: consequents list of *a*
- *coefficient (i)*: coefficient of group *i* calculated in formula *[1]*
- n_{ic}: number of antecedents of *c* classified in group *i*

3.Integration of Retrospective Analysis into our existing ITS

3.1. Learner's Plans Recognition

In order to detect the erroneous actions realized by a human learner, it is indispensable to know: (1) the learner's activities in the past; (2) his current action (in the meaning that the action has just been done); and (3) the actions that the human learner intents to do in according to a predefined correct procedure. Our existing ITS as proposed in (Buche and Querrec, 2005) bases on the model MASCARET (Querrec *et al.*, 2004) where we used an approach using multi-agent system to simulate collaboration between human learners and agents during their realization of missions. Learners are gathered in team consisting of several predefined roles, every role contains a number of tasks associated eventually with accurate resources, every leaner also owns an epistemic memory containing all actions realized in the past, etc. Finally, we could retrieve from MASCARET following informations relating to learner's plan in VET:

– *action(s) before*: learner's action(s) in the past (note that, in MASCARET, every action is eventually associated with its accurate resource(s))

– *current action*: action has just been done by learner

– *action(s) correct (according to role)*: action(s) must be done by learner in his role(s)

– *action(s) correct (according to plan)*: action(s) may be done by learners in the context. Here, it is essential to make distinction betweens "*action(s) correct according to role*" and "*action(s) correct according to plan*". In the first case, because the learner could play several roles, it represents all correct actions that the system expects from the learners. The second one concerns the cases where there are more than one learner in VET to realize together a mission. Therefore, in this case, it is possible that a leaner performs a correct action according to the plan but it is not correct in compare to his role.

– *next correct action(s) in the role*: next action(s) must be done by learner in his role(s)

– *full correct plan*: description of all accurate actions (associated with resources) in predetermined procedure that the learner must respect.

In next section, we present our mechanism for mapping erroneous actions detected by our existing ITS with Hollnagel's classification scheme of errors modes.

3.2. Classification of Erroneous Actions according to the Scheme of CREAM

Erroneous Actions in Phenotype "Sequence"

According to Hollnagel, performing an action at the wrong place in a sequence or procedure is a common erroneous action, and it is more realistic in our context of simulation of procedural and collaborative work. The "*Sequence*" problem consists of several specific effects: *Omission* (an action was not carried out); *Jump forward/ Jump backwards* (actions in a sequence were skipped/carried out again); *Repetition* (the previous action is repeated); *Reversal* (the order of two neighbouring action is reversed); *Wrong action* (an extraneous or irrelevant action is carried out). We present in following our mechanism to detect erroneous actions in phenotype "*Sequence*":

– If *current action* exists in *action(s) correct according to role*:
this is a correct action (phenotype Sequence does not occur).
Else: + If *current action* does not exist in *action(s) correct (according to plan)*:
specific effect = "*Wrong action*"
Else: * If *current action* exist in *last action before*: specific effect = "*Repetition*"
* Compare the relative order of *current action* to the order of *next correct action(s) in the role* using the *full correct plan*:
– If *id_current_action* < *id_correct_action_in_rol*e:
specific effect = "*Jump backwards and/or Omission* "
Else: specific effect = "*Jump forward and/or Omission*"
– If *id_current_action* = *id_correct_action_in_role* +1:
specific effect = "*Reversal* "

Erroneous Actions in Phenotype "Wrong object"

In (Hollnagel, 1998), the author clarified that "*action at wrong object*" is one of the more frequent error modes, such as pressing the wrong button, looking at the wrong indicator, etc. In our context, during realisation of collaborative work, it is possible that learner performs a correct action but on a wrong object. Therefore, the detection of erroneous actions in phenotype "*Wrong object*" must be implemented independently with the detection of phenotype "*Sequence*". This phenotype is detailed into following specific effects: *Neighbour/Similar object* (an object that is proximity/similar to the object that should have been used); *Unrelated object* (an object that was used by mistake).

In order to detect erroneous actions in phenotype "*Wrong object*", we use the same principle presented in the case of phenotype "*Sequence*" by using following informations retrieved from model MASCARET:

– *current resource*: resource associated with *current action*

– *resource(s) correct (according to role)*: resource(s) must be used by learner in his role(s)

– *resource(s) correct (according to plan)*: list of resource(s) associated with all action(s) in *action(s) correct according to plan*.

Our algorithm is detailed in following:

– If *current resource* exists in *resource(s) correct according to role*:
this is a correct resource (phenotype Wrong object does not occur).
Else: + If *current resource* does not exist in *resource(s) correct (according to plan)*:
specific effect = "*Unrelated object*"
Else specific effect = "*Neighbour and/or Similar object* "

Erroneous Actions in Phenotype "Time/During"

The phenotype "*Time/During*" is divided in several specific effects: *Too early/ Too late* (an action started too early/too late); *Omission* (an action that was not done at all); *Too long/Too short* (an action that continued/was stopped beyond the point when it should have been). Hollnagel noted that the error modes of timing and duration refer to a single action, rather than to the temporal relation between two or more actions. In our context, the realization of tasks in model MASCARET is sequential, therefore, an action is considered to be too early when it was realized before several actions in plan; also, action(s) are considered to be omitted when they were not carried out.

Finally, in order to detect erroneous actions in phenotype "*Time/Durring*", we propose that:

– action having specific effect ""*Jump forward*" also has specific effect "*Too early*"

– action described by specific effect "*Omission*"(in error mode "Sequence") will be considered as an action having specific effect "*Omission*" (in error mode "Time/During")

3.3. Experiment & Results

In order to evaluate our integration of retrospective analysis into ITS, we take place in GASPAR application (Marion et al., 2007) whose objective aims at simulate aviation activities by virtual reality. We use the classification scheme of error modes proposed in (El-Kechaï and Després, 2007) which were particularly adapted to VET. Table 1 illustrates results of retrospective analysis for the phenotype Sequence.

Table 1

Causal links of phenotype "Sequence"

Coefficient (M,T,O)	Causal links
0.333 - 0.333 - 0.333	*1, Design failure (0.125) -> Inadequate scenario (0.125) -> Sequence*
	2, Adverse ambient condition (0.125) -> Inattention (0.125) -> Sequence
	3, Long time since learning (0.042) -> Memory failure (0.125) -> Sequence
1 - 0 - 0	*1, Other priority (0.2) -> Memory failure (0.2) -> Sequence*
	2, Error in mental model (0.067) -> Faulty diagnosis (0.2) -> Sequence
	3, Erroneous analogy (0.067) -> Faulty diagnosis (0.2) -> Sequence
0 - 1 - 0	*1, Equipment failure (0.1) -> Access problems (0.5) -> Sequence*
	2, Distance (0.1) -> Access problems (0.5) -> Sequence
	3, Localisation problem (0.1) -> Access problems (0.5) -> Sequence
0 - 0 - 1	*1, Noise (1) -> Communication failure (1) -> Sequence*

We change coefficients of three factors (M,T,O) for evaluating how CPC's influence the analysis result. For each phase in analysis process, we select and display the most probable cause by ordering mass values.

4. Conclusion and Future Work

In this paper, we proposed an approach to modelling the Cognitive Reliability and Error Analysis Method (CREAM). We separated the representation of classification scheme of erroneous actions and the analysis method; therefore, our description of errors modes is adaptable to different training context without any modification on analysis method. We started by defining the Common Performance Conditions, then the direct and indirect relations between consequent-antecedent are modelled using a non-hierarchical data structure. Finally, the most probable cause-effect links could be found using Dempster-Shafer's theory presented in (El-Kechaï and Després, 2007).

In order to integrate the retrospective analysis described above into our existing ITS, we based on the model MASCARET to retrieve information concerning learner's plans and then detect erroneous actions. Finally, we presented our proposition to mapping erroneous actions with Hollnagel's classification. The experimental results in GASPAR

project are also presented. So that, in addition to the detection and tagging of erroneous actions, the ITS could furthermore indicate the path of probable cause-effect explaining reasons that the errors occur.

In the future work, we will concentrate our attention on evaluation of MASCARET so that this model could permit to describe more complex tasks in taking into account other factors such as force, distance, speed, direction, etc. Hence, other different types of errors modes could be detected and then explained using the retrospective analysis.

REFERENCES

QUERREC R., BUCHE C., MAFFRE E., and CHEVAILLIER P. (2004), Multiagents systems for virtual environment for training: application to fire-fighting. *International Journal of Computers and Applications (IJCA)*, June 2004, 25-34.

BUCHE C. and QUERREC R. (2005), Intelligent tutoring system for MASCARET, in *Simon Richir and Bernard Taravel, editors, 7th Virtual Reality International Conference (VRIC'05)*, April 2005, Laval, France, 105-108.

HOLLNAGEL, E. (1998), *Cognitive Reliability and Error Analysis Method*, Oxford: Elsevier Science Ltd.

EL-KECHAÏ N. and DESPRÉS C. (2007), Proposing the underlying causes that lead to the trainee's erroneous actions to the trainer, in *EC-TEL: European Conference on Technology Enhanced Learning*, September 2007, Crète (Grèce), 41-55.

EL-KECHAÏ N. and DESPRÉS C. (2006), A Plan Recognition Process, Based on a Task Model, for Detecting Learner's Erroneous Actions, in *Intelligent Tutoring Systems ITS 2006*, June 2006, Jhongli (Taïwan), 329-338.

MARION N., SEPTSEAULT C., BOUDINOT A. and QUERREC R. (2007), GASPAR: Aviation management on an aircraft carrier using virtual reality, in *Cyberworlds 2007*.

SERWY R. D. and RANTANEN E. M. (2007), *CREAM Navigator* http://www.ews.uiuc.edu/~serwy/cream/v0.6beta/, [version 0.6, September, 2007]