Emergent Properties in Natural and Artificial Complex Systems
EPNACS 2010 *September 15-16, 2010*

AN ARCHITECTURE FOR ANALYSING DYNAMIC SITUATIONS WITH A MULTIAGENT SYSTEM

Thierry Galinho, Patrick Person, Hadhoum Boukachour, Florence Lecroq and Jean Grieu *

Abstract. In the educational field the amount of data coming from multiple users, together with the rapidity at which these data occurs is a complex problem. Indeed, on the one hand, data evolution must be accounted for permanently. On the other hand, these data must be incorporated to the representation of the situation dynamically. The data stuctures and algorithms used to face the challenge of representing and interpreting fast-evolving situations are detailed below. Considering dynamic characteristics of the problem, we choose a multiagent system architecture as artificial intelligence component of the intelligent tutoring system. All inputs are transformed into a unique data structure called composite factual semantic feature. Using composite factual semantic features as input, our multiagent system represents the global current situation. From the successive static representations, our multiagent system also address the challenge of deducing possible ulterior evolutions.

Keywords. Factual agent, composite factual semantic feature, intelligent tutoring system, semantic proximity, dynamic representation.

1 Introduction

Quoting Denning: "the fundamental question underlying all of computing is, *what can be (efficiently) automated?*" [1]. That leads to identify three broad categories of problems. Problems that are impossible to solve with a computer, such as predicting the next Lotto numbers which are numbers drawn at random. Problems that are easy to solve with a computer, such as searching if a given integer exists in a billion integers. In that range of problems, there is only one possible answer or one optimal solution. And finally, problems where it is possible to use a computer, from which we are expecting a correct solution, even if not the best one when a best solution exists. This is the situation artificial intelligence has to face when asking computers to bring a solution to complex problems.

In the two feasible categories – simple or complex problems – a correct answer depends on the quality of information used as input: should you omit to provide the information that a number, let's say 12, belongs to the set of numbers that you planned to search for, not surprisingly, the computer returns a negative answer when searching for number 12.

In *The Art of Computer Programming*, Knuth defines a data structure as "A table of data including structural relationships" [2]. This definition does not imply that arrays are the only data structure. The data structure contains the data a user wishes to represent as well as the structural relationships. The design of a data structure not only includes the order of the fields, but also the higher level design goals for the programs which access and manipulate the data structures. For instance, efficiency has long been a desirable aspect of many computer programs. Another important aspect in computing is how to access and manipulate the data structures and their related fields. Knuth defines this as data organization which is "a way to represent information in a data structure, together with algorithms that access and/or modify the structure" [3].

A data structure is an internal representation of a problem which will be processed by a computer. The choice of data structure is closely connected to the choice of the algorithm used to solve the problem. Depending on the choice of data structure, searching for a given integer into a billion integers could be done in thirty comparisons or in a billion comparisons. So the couple made by one algorithm and its associated data structure is suited to a set of problems if it can provide a correct answer in a span of time useful for users. Indeed, an algorithm guessing the weather forecast for tomorow, but needing two days of works from today is useless.

There are several kinds of complex problems depending on their inherent properties: a huge quantity of data to process, a massive amount of computation to deal with, heterogeneous sources to mix and interpret, synchronous co-operation with distributed decisions, merging co-operative works from distributed actors or searching in a large number of distributed sources [4]. The properties of the complex problem that we are working on are detailed below.

In the educational field, one important characteristic of intelligent tutoring systems is the number of dynamic elements to take into account, to interpret and model for providing answers to students [5]. This article presents an architecture dealing with rapid changes implied by all

*LITIS, University of Le Havre, France. E-mails: Thierry.Galinho@univ-lehavre.fr, Patrick.Person@univ-lehavre.fr, Hadhoum.Boukachour@univ-lehavre.fr, Florence.Lecroq@univ-lehavre.fr, Jean.Grieu@univ-lehavre.fr

inputs from students. The system is not determinist: answers to students are not selected a priori neither known in advance. How the answers are choosen or created is presented. Then, we detail the data stuctures and algorithms used to face the challenge of representing and interpreting fast-evolving situations.

2 Two Major Applications

To illustrate the use of our system, we are now going to present two applications: an intelligent tutoring system and the example of the game of Risk.

2.1 An Intelligent Tutoring System

Broadly defined, an intelligent tutoring system (ITS) is educational software containing an artificial intelligence component. The software tracks students' work, tailoring feedback and hints along the way. By collecting information on a particular student's performance, the software can make inferences about strengths and weaknesses, and can suggest additional work [6].

Wenger considers the main goal of ITS to be communicating knowledge efficiently [7]. The model contains four components: the domain model, the student model, the teaching model, and the user interface. Woolf uses a similar model to the one presented by Wenger, with updated computer architectures, one of which being based on multiagent systems [8]. In the challenge for capturing intelligence, Luger also mentions this possible choice [9]. In line with Wooldrigde [10], considering the dynamic characteristics of the problem, we chose a multiagent system architecture for the artificial intelligence component of the ITS designed as in figure 1.

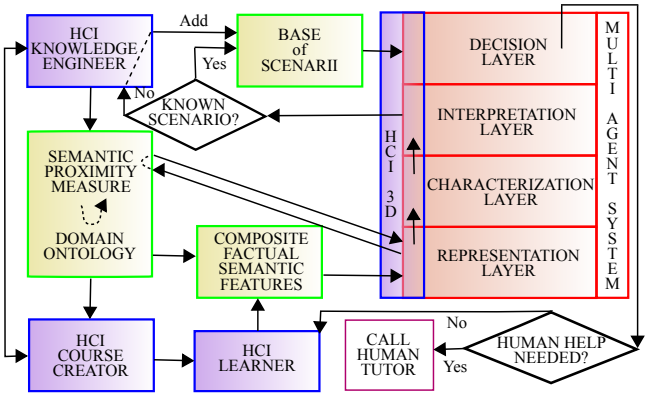

Figure 1: Architecture of the ITS

The three main parts of the ITS in figure 1 are: four Human-Computer Interfaces (HCI) in blue, three knowledge parts in green and a four-layer hierarchical multiagent system (MAS) in red.

Our ITS follows the four components proposed by Wenger which are knowledge of the domain, student model, teaching model and user interface:

1. Knowledge of the domain with domain ontology, semantic proximity measure and composite factual semantic features.

2. Student model using case-based reasoning [11, 12]. During a first step, supervised learning with a group of students generates input from the interpretation layer to the base of scenarii.

3. Teaching models are associated to student models during the supervised learning steps. Decision layer must produce the most appropriate answer for the learner from the data retrieved in the base of scenarii. That layer may also call a human tutor when needed [13].

4. Interfaces for learners and course creator are detailed by Grieu [14]. A 3D visualisation allows developers to see the evolutions and messages exchanged inside MAS.

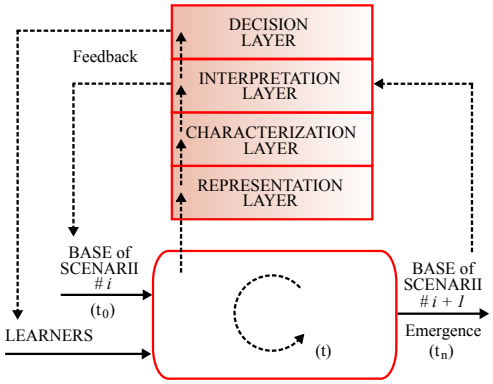

Figure 2: Enriching process of the base of scenarii

One or a few scenarii which are similar to the current situation and associated to the given profile of the student emerge(s) from the system. The system will either adapt the choosen scenario or ask knowledge engineer or/and course creator for possibly creating a new scenario ex nihilo. Figure 2 shows the enriching process of the base of scenarii with students inputs and with new scenario. The aim of this ITS is to stabilize student's behaviour in her/his knowledge acquisition process; this is a negative retroaction loop.

2.2 Game of Risk

Without any loss of generality, another example is chosen to facilitate the presentation of technical details by taking advantage of the assumption of a closed-world. The chosen example is the game of Risk. This is not a toy problem, because, as summarised by Wolf [15] "Risk is far

more complex than any of the traditional boardgames" as chess or go. Risk is a game of strategy for 2 to 6 players. The game board is a map of the world divided into 42 territories. A player wins by conquering all territories. In turn, after an initial placement of armies, each player receives and places new armies and may attack adjacent territories. An attack is one or more battles which are fought with dice. Rules and strategies are detailed in [16].

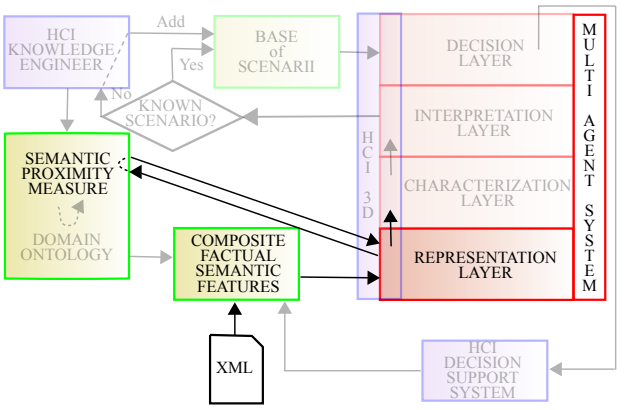

Figure 3: Dynamic representation architecture

Figure 3 outlines the architecture of the decision support system we designed to help Risk players by pointing out previous comparable situations and their consequences [17]. The multiagent component of this system is similar to the one inside the ITS. Instead of using a human-computer interface as input, figure 3 shows a direct input from an XML file. This choice simplifies this presentation because the XML file contains all the successive inputs and thus it is useless to detail the decision support system interface associated to the decision layer. The XML file is chosen from a set of files recorded during test phases of the system.

The next section focuses on how to represent situations the main characteristic of which is to change quickly. This section analyses inputs and the internal organisation of the representation layer in order to illustrate a data structure dealing with numerous data inputs and mixing them into a global evolutive representation.

3 Dynamic Representation: Data Structures and Algorithms

3.1 Composite Factual Semantic Feature

Users inputs from human-computer interface are sent to the system. All inputs are transformed into a unique data structure called *composite factual semantic feature* (CFSF). The successive composite factual semantic features are the inputs of the representation layer of the MAS. In linguistics, a semantic feature is a meaningful component used in text analysis. Here, it is used in another context. A composite factual semantic feature is still a meaningful component. In addition, a CFSF is also an observable fact which is composite because it is composed of a few items. An item is a property-value pair. Three items are compulsory for a CFSF: *Type*, *Name* and *Date*. The generic model of CFSF is:

```
<!ELEMENT CFSF (Type, Name, Date, Item+)>
```

The symbol $^+$ at the end of *Item* means that the number of items could be a given value from 1 to n and that number depends on the *Type* of CFSF. The couple *Type-Name* is the key of the CFSF. Duplicate keys are not allowed. CFSFs are written in XML:

```
<CFSF>
        <Type>    valueType    </Type>
        <Name>    valueName    </Name>
        <Date>    valueDate    </Date>
        <Item1>   value1       </Item1>
        . .       . .          . .
        <ItemN>   valueN       </ItemN>
</CFSF>
```

The following is an example of a CFSF from the game of Risk. Every territory is represented by a CFSF the type of which is territory. On top of the three compulsory ones, this CFSF also has two items which are the player who owns the territory and the number of armies which stand on the territory:

```
<cfsf>
        <type>      territory   </type>
        <name>      Quebec      </name>
        <date>      14          </date>
        <player>    red         </player>
        <nbArmies>  2           </nbArmies>
</cfsf>
```

(1)

Each player is named by a colour. In this example, player red owns the territory called Quebec. The key is the couple (*type : territory - name : Quebec*).

3.2 MultiAgent System

The multiagent system in figure 3 could be seen as dynamic data structures with correlated algorithms. Each layer of the MAS is associated with its own active data structure – agents of the layer – and with its algorithmic processing – interrelation between agents. Functions assumed by each layer are:

Representation: using CFSFs as input, this layer represents the global current situation and its dynamic tendency of ulterior evolutions.

Characterization: this layer classifies subsets of agents of description layer according to levels of internal activity, and then, computes a synthetic measure characteristic of each subset.

Interpretation: this layer associates subsets of characterization layer with sections of scenario.

Decision: this layer finds scenarios closed to the current situation and choses the best one to propose to users.

As an information technology component, an agent can use all abilities of a high level programming language (such as Java). Therefore, an agent can be an advanced dynamic data structure representing a whole situation and its dynamic tendency of evolutions as we will see in following sections.

3.3 Factual Agent

The point of view chosen to explain the data structure is to follow the successive transformations and uses of a given CFSF. The first transformation has already been presented when users inputs are transformed into CFSFs and sent to the representation layer.

The second transformation starts when a given CFSF is encapsulated in a factual agent. Therefore, a presentation of factual agents is needed. A factual agent is made of two parts: a knowledge part – its CFSF – and a behaviour part. Each factual agent contains one CFSF. Each CFSF is associated with only one factual agent. Thus, this is a one-to-one relation between factual agent and its composite factual semantic feature. When a CFSF reaches the representation layer of the MAS, there are two options. If a factual agent contains a CFSF with the same key (couple *type-name*), this factual agent is updated with the incoming CFSF. There is no conservation of the previous one. Else, a new factual agent is created.

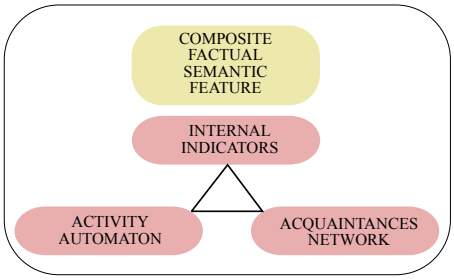

Figure 4: Structure of a factual agent

The challenge is to represent a whole situation and its dynamic tendency of evolutions. The most difficult part is not to represent a static view of the current situation, that is to say a comprehensive representation taken at a given time t. Indeed, the main challenge consists in adding dynamic tendency of evolutions. In other words, the problem is how to make a static data structure such as a CFSF dynamic? From the successive static representations, the main problem is to compute information to describe the tendency of evolutions which would be used by other layers of the MAS. One answer is to use factual agents which compute the behaviour part. Figure 4 shows the structure of a factual agent where the knowledge part is the composite factual semantic feature, and where the behaviour part is made of four internal indicators, an automaton describing the level of internal activity and an acquaintances network. The three components of the behaviour part are strongly bound and work together.

The four internal indicators are *position* p, *celerity* c, *acceleration* a and *satisfaction* s. The computation of *position* p depends on the type of CFSF. This indicator represents the agent in the representation space. Definitions and computations of all the other parts of the behaviour are independent on the type of CFSF. *Celerity* c and *acceleration* a describe the internal dynamics of the agent between two evolutions t and t'.

$$c_{t'} = p_{t'} - p_t$$
$$a_{t'} = c_{t'} - c_t$$

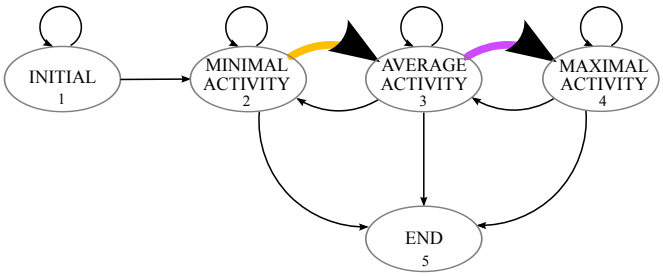

Figure 5: Activity automaton of a factual agent

Satisfaction s is linked to the automaton, and so, will be defined after the presentation of the automaton. Figure 5 shows the activity automaton with its five states. This automaton is set in state *initial* when a factual agent is created. Reaching the state *end* causes the death of its factual agent. The various states are typical of the significance of its associated CFSF in the whole representation, from minimal to maximal activity. There is always only one possible transition from the current state to another or to the same state. As a notation, t_{ij} stands for the transition from state i to state j. Conditions of transitions depend on values of internal indicators. Table 1 contains all the transitions and associated conditions.

The satisfaction indicator measures if there are a lot of transitions from a state to a different state, or if the transition occurs from a state to the same one. This indicator keeps the last ten values in memory and adds them up to give a single value. Thus, satisfaction also indicates the level of activity in [0 .. 20]. In some cases, a negative value can also be set by message.

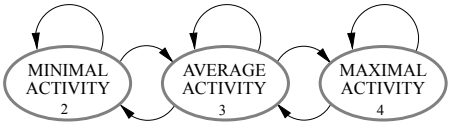

Figure 6: Transitions for computation of satisfaction

Table 1: Conditions of transitions in activity automaton

	p	c		a	s
t_{11}	$p < 1$				
t_{12}	$p \geqslant 1$				
t_{22}		$c \leqslant 0$			$s \geqslant 0$
t_{23}		$c > 0$			$s \geqslant 0$
t_{25}					$s < 0$
t_{32}		$c \leqslant 0$	and	$a \leqslant 0$	$s \geqslant 0$
t_{33}		$c \leqslant 0$	xor	$a \leqslant 0$	$s \geqslant 0$
t_{34}		$c > 0$	and	$a > 0$	$s \geqslant 0$
t_{35}					$s < 0$
t_{43}		$c \leqslant 0$	or	$a \leqslant 0$	$s \geqslant 0$
t_{44}		$c > 0$	and	$a > 0$	$s \geqslant 0$
t_{45}					$s < 0$

Figure 6 shows the transitions which are active for the computation; i.e. when the given transition is activated, the associated value is the new value added to satisfaction (see table 2).

Table 2: Values for computation of satisfaction

t_{22}	0
t_{32}	0
t_{23}	+ 1
t_{33}	+ 1
t_{43}	+ 1
t_{34}	+ 2
t_{44}	+ 2

The third component of behaviour part is an acquaintances network. This is a dynamic memory of factual agents whose semantic proximity measure, between the embedded CFSF and the CFSF of the current agent, is different from 0. As shown in figures 1 and 3, the knowledge part of the whole system contains domain ontology structuring and defining the meaning of the observed facts. The proximity measure uses the ontology to compare CFSFs and returns a value in [-1 .. 1] (see figure 7).

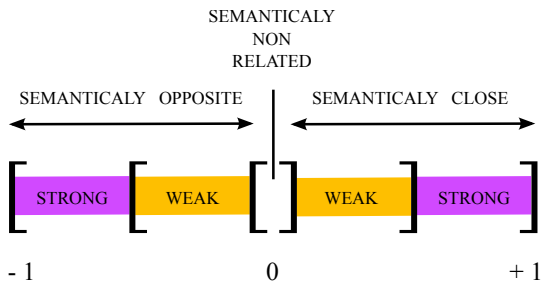

Figure 7: Semantic proximity measure

A value of -1 means a total opposition between the two compared CFSFs. A value of 0 means neutral or non related. A value of 1 means identity between the two CFSFs and any other value in this interval means a semantic connexion in the range from opposite to identical.

Proximity between two CFSFs – opposite or close – is either strong or weak. As described in previous sections, transition occurs when predefined values of internal indicators are reached. Two transitions cause particular actions as displayed in figure 5: transition from state "minimal activity" to state "average activity" appears in orange, and transition from state "average activity" to state "maximal activity" in purple. The first one triggers messages to agents of acquaintances network with weak proximity. The second one triggers messages to agents of acquaintances network with strong proximity.

After this presentation of factual agents, an example of how a given CFSF is encapsulated in a factual agent will illustrate the second transformation of a CFSF. A factual agent was created when the CFSF of example (1) was received by the representation layer. To take into account a change in the game of Risk, the CFSF of example (2) is sent to the representation layer.

```
<cfsf>
        <type>      territory   </type>
        <name>      Quebec      </name>
        <date>      76          </date>
        <player>    red         </player>
        <nbArmies>  7           </nbArmies>
</cfsf>
```
(2)

As both CFSFs have the same key (*type : territory - name : Quebec*), the knowledge part of the associated agent is updated by replacing the previous CFSF with the incoming CFSF displayed in example (2). This event starts computations of the behaviour part. First, position p – the definition of which is specific to the type of CFSF – is revised: when the owner of a territory is the same in both CFSFs, the new position is the new number of armies minus the old one. Next, other indicators are updated accordingly, and a transition from current state in the activity automaton is searched for. If state moves from "minimal" to "average" or from "average" to "maximal", then messages are sent to semantically related factual agents.

3.4 Representation Layer

Previous section details a single factual agent. A multiagent system is made of several factual agents interplaying, i.e. agents exert influence on each other. Systematically, when a factual agent is created or updated, it broadcasts message holding its own CFSF to all the others but itself. Then, each receiving agent uses the semantic proximity measure to compare the CFSF carried by the message with its private CFSF. If, and only if, the result returned

by the semantic proximity is different from 0, receiving agent is activated. In that case, its acquaintances network is kept up to date and its position could be modified. As a consequence, that causes changes in internal indicators and leads to a new transition in the activity automaton. Again, those changes could generate evolution inside the set of semantically connected agents. This third transformation propagates incoming CFSF further than its own factual agent according to the strength of the new CFSF in the global representation. The characterization layer clusters the FAs of the representation layer by using internal indicators of FAs. This partition focuses and identifies the current situation.

4 Discussion

From the game of Risk, ITS prototype re-uses representation and characterization layers specifications. However, for reasons we will not detail here, we changed agent platform for JADE. Therefore rewriting of the prototype was needed.

A 3D Human-Computer Interface for learners is used for the ITS project on Programmable Logic Controllers (PLC's) (see figure 8). The team is presently working on the development of the knowledge part to connect HCI learners to MAS.

Figure 8: A snapshot of 3D HCI learners

The interpretation layer of the ITS is a part of a PhD thesis in progress. The decision layer can modify the learning tools at disposal for of a student: suggesting new series of exercices, proposing complementary reading, taking evaluations again ...

The difficulty of finding the "right values" for the "right parameters" is very often emphasized in literature about multiagent systems. Without any doubt, that was the case for the system detailed above. Definitions and tests of parameters, and finding sets of accurate values for those parameters were time-consuming.

Table 3 summarizes parameters of factual agents sorted into two groups: generic and specific parameters.

Table 3: Characteristics of factual agents

	Factual Agent	Generic	Specific
Knowledge	Composite Factual Semantic Feature		
	. *format*	x	x
	. *content*		x
Behaviour	Internal Indicators		
	. *position*		x
	. *velocity*	x	
	. *acceleration*	x	
	. *satisfaction*	x	
	Activity Automaton		
	. *definition*	x	
	. *use*	x	
	Acquaintances Network		
	. *definition*	x	
	. *use*	x	

Specific parameters needs to be redefined for each new project. The second transformation of CFSF sets the value of position indicator from the CFSF and behaviour computations rely on position values. Therefore, accuracy of calculating position from the content of CFSFs is crucial. By definition, an ontology is a representation of the knowledge of a domain with relationships between concepts of this domain. The ontology supplies information to semantic proximity measure. The latter is involved in the third transformation of CFSFs which affects relations between agents. Thus, the design of both ontology and proximity measure are important. The pattern of the format of composite factual semantic features is generic, but needs to be ajusted to each type of CFSFs.

Generic parameters are defined once for all factual agents. Having done this job makes it easier to create factual agents. Indeed, instead of having eight parameters to consider in the behaviour part every time a new factual agent is needed, knowledge engineers have only one parameter to define. Another important feature of this representation is its ability of dealing with several kinds of complex problems, such as emergency logistics. This system was applied to RoboCupRescue Simulation Project [18] whose main purpose was to provide emergency decision support after an earthquake [19].

5 Conclusion

Human-Computer Interface needs to be designed for helping knowledge engineers to organise the ontology and define composite factual semantic features more easily.

A comprehensive representation of the static situation at any given time is made up of the knowledge part of all agents. The dynamic tendency of evolutions is made up of the behaviour part of all agents. The challenge of both representing fast-evolving situations and computing

information to describe the dynamic tendency of evolutions efficiently was answered by the use of factual agents.

References

[1] P. J. Denning, "Computer Science: The Discipline," in Anthony Ralston & David Hemmindinger (eds.), *Encyclopedia of Computer Science*, 2000.

[2] D. E. Knuth, *The Art of Computer Programming: Fundamental Algorithms*, 3rd ed., Addison-Wesley, 1997.

[3] M. Murr, "How forensic tools recover digital evidence (data structures)," `http://www.forensicblog.org/2007/05/05/`, 2007.

[4] W. Boch, J. Bremer, M. Campolargo, J. Da Silva, T. Van Der Pyl, and J. Bus, "Complex Problem Solving including GRID and Research Networking Infrastructures," *FP6 Internal Reflection Group*, CORDIS, `ftp://ftp.cordis.europa.eu/pub/ist/docs/grid/irg-grid_rni-140502.pdf`, 2002.

[5] M. Clemens, "The Art of Complex Problem Solving," `http://www.idiagram.com/CP/cpprocess.html`, 2005.

[6] K. Hafner, "Software Tutors Offer Help and Customized Hints," AITopics, `http://www.aaai.org/AITopics/pmwiki/pmwiki.php/AITopics/IntelligentTutoringSystems`, 2007.

[7] E. Wenger, *Artificial Intelligence and Tutoring Systems*, Morgan Kaufmann, 1987.

[8] B. P. Woolf, *Building Intelligent Interactive Tutors*, Morgan Kaufmann, 2009.

[9] G. F. Luger, *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*, 6th ed., Addison-Wesley, 2009.

[10] M. Wooldridge, *An Introduction to MultiAgent Systems*, 2nd ed., Wiley, 2009.

[11] H. Boukachour, "Système de veille préventive pour la gestion de situations d'urgence : modélisation par agent," Ph.D. dissertation, Le Havre University, Le Havre, France, 2002.

[12] A. Aamodt, "Case-based reasoning and intelligent tutoring," *AIS-SSLS Proceedings*, Västeras, Sweden, 2005.

[13] J-C. Bertin, and P. Gravé, "How to devise a cognitive agent for distance language learning," *CALICO*, Manoa, United States, 2006.

[14] J. Grieu, F. Lecroq, P. Person, T. Galinho, and H. Boukachour, "GE3D: a virtual campus for technology-enhanced learning," *IEEE Engineering Education Conference*, Madrid, Spain, 2010.

[15] M. Wolf, "An Intelligent Artificial Player for the Game of Risk," Diploma Thesis, Knowledge Engineering Group, Darmstadt University of Technology, Darmstadt, Germany, 2005.

[16] O. Lyne, "Risk FAQ 5.61," `http://www.kent.ac.uk/IMS/personal/odl/riskfaq.htm`, 2006.

[17] P. Person, H. Boukachour, M. Coletta, T. Galinho, and F. Serin, "From Three MultiAgent Systems to One Decision Support System," *IICAI05*, Pune, India, 2005.

[18] "RoboCupRescue Simulation Project," `http://www.robocuprescue.org/`, 2008.

[19] F. Kebair, F. Serin, and C. Bertelle, "Agent-based perception of an environment in an emergency situation," *International Conference of Computational Intelligence and Intelligent Systems (ICCIIS), World Congress of Engineering (WCE)*, London, UK, 2008.