Development of an Intelligent Tutorial System to Enhance Students' Mathematical Competence in Problem Solving

Simon El-Khoury
Département d'informatique et de recherche opérationnelle
Université de Montréal
Canada
elkhous@iro.umontreal.ca

Esma Aïmeur
Département d'informatique et de recherche opérationnelle
Université de Montréal
Canada
aimeur@iro.umontreal.ca

Philippe R. Richard
Département de didactique
Université de Montréal
Canada
philippe.r.richard@umontreal.ca

Josep M. Fortuny
Departament de Didàctica de la Matemàtica i de les Ciències Experimentals
Universitat Autònoma de Barcelona
Espanya
JosepMaria.Fortuny@uab.es

Abstract. This paper presents work related to the development of an intelligent tutoring system called TURING (French acronym of «TUtoRiel INtelligent en Géométrie»). TURING is a multidisciplinary project, that joints recent research in didactic of mathematics with the possibilities of computer-based learning environments. From the *educational point* of view, the system is helpful for the secondary school student to improve problem solving aptitudes, mathematical reasoning abilities and communication skills using natural and mathematical language. In addition, the system is helpful to assist the teacher in his responsibility of attending the diversity of the development of mathematical competences in a whole class. From the *technical point* of view, the TURING architecture is a multi-agent system conceived in a flexible approach so that a teacher can adjust the heuristic and discursive features according to specific practices with real students. In particular, for the discovery of a conjecture or for the realization of a mathematical proof, the system consents to adapt, in the problem solving process, the space of meaningful actions and, in argumentative process, the set of strategic messages with pedagogical *agents*.

1 Introduction

One of the purposes of research in Didactics of Mathematics is to improve the *teaching* and *learning* processes and to increase the quality of interactions between the different poles taking part in the educational system (student, teacher, knowledge and milieu), which has cognitive and social benefits for the students. This increase in the quality of interactions can be achieved by designing, implementing and validating those learning environments.

The use of virtual, interactive and collaborative systems of tutoring allows students to work in an adaptable way, either autonomously, in interaction with the virtual tutor, or with the teacher or other students. Moreover, it encourages the great quality of the learning, which supports the acquisition of training attitudes, values and habits that it will have need in the future to enter the world of work (Techno-mathematical Literacy in the Workplace). Besides, the flexibility of these systems allows meeting the educational and training demands of groups of people whose needs could not have been met due to several reasons, namely students with marked cognitive differences (e.g., students who need personalized support outside the classroom due to several reasons). With the intention of meeting the demands of this situation, the main interests of our system are the following:

- Developing an intelligent tutoring system TURING (Aïmeur *& al*., 2005) in order to improve students' competence when solving mathematical problems and those teachers' competence needed in order to focus on understanding how their students solve problems in an e-learning environment.
- Analyzing the influence and effects of interactions with TURING system on the development of the students' strategic competences when solving problems and characterizing the competence levels generated.

The development of the TURING system also includes aspects related to teacher training and the teachers' performance as tutors. These tutors could carry out teacher's tasks in both classroom and virtually (eTutor), as well as the dissemination and exploitation of results in the education sector. The project is innovative by the integration of the theoretical approaches on which it is based. The algorithms and the warehouse structure gives the TURING system the ability to a simulation that is very close to the reality of teaching and learning during the problem solving.

Our TURING system shows the application of a model of interactions that is based on research into didactic of mathematics. This research is valuable for the development of intelligent virtual pedagogical agents in an artificial tutoring system. The system offers to the students a meta-cognitive support to help them in their development of mathematical competences (e.g. problem solving, mathematical reasoning, communication with natural and mathematical languages). The agent architecture of TURING system is conceived as a hybrid multi-agent architecture made up of four distinct agents. Each agent has intelligence to support and help the student, the teacher and the *didactic expert* using an open architecture and flexible structure.

This paper is organized as follows: Section 1 presents an introduction of this paper, and in Section 2 we briefly discuss an overview of the TURING system. Following that, in Section 3, we outline our approach and we describe the architecture and the functionality of the system and we illustrate a sample scenario. In section 4, we describe the implementation of the TURING *User Interfaces.* Section 5 discusses other applications similar to the TURING. Finally, in Section 6, we draw some conclusions and present future work planned for the TURING.

2 Overview of the TURING

TURING is based on the integration of the social debate simulated in a computer-based environment of human training for the problem solving and mathematical reasoning (Aïmeur *& al*., 2005). The disciplinary contents (concepts, processes and attitudes) come from the compulsory curriculum of mathematics. The theoretical framework combine the theory of *didactic situations* of Guy Brousseau (1998), the proofs and refutations dialectical of Imre Lakatos (1984) and the theory of the *functions of the language* of Raymond Duval (1995). It also combines the application of an intelligent tutoring system in the development of mathematical competence and acquisition of knowledge by the learner. The system aims to extend and verify the field of validity from the preceding framework in a computer-based environment.

3 The Approach of TURING

Contrary to the approaches which introduce an agent tutor to simulate the role of the teacher – with, possibly, an agent companion or a disturbing agent (Aïmeur & Frasson, 1996) to simulate the attitude of peers –, we formulate the assumption that a virtual pedagogical agent is an autonomous being of nature compared to what should be a human model. In other words, the same virtual pedagogical agent can assume several traditional roles (tutor, companion or disturber, within the meaning of Aïmeur, 1998) while being defined in a distinct way compared to each one of these roles. This assumption is strong, but it makes it possible to ensure jointly the epistemological validity of the collaborative process solving while respecting the solving strategies to which a student would have recourse in the particular context of a given problem. We must underline here that we do not claim to replace regular teaching, but we want to provide a personalized help for the pupils who need some extra help (Fortuny, Giménez *& al*., 2002). Thus, the TURING system needs the flexibility to build virtual pedagogical agents that give the students a cognitive and meta-cognitive support using strategic messages in an argumentative process that jointly considers the meaningful actions of the student, i.e. in our case the meaning of the adopted propositions (discursive and graphic, within the meaning of Richard, 2004a) associated with the discovery of a conjecture and the realization of a proof.

3.1 The TURING Architecture

The architecture of TURING system consists of four main agents (Figure 1):

- *Student Agent*. Converts graphic action to discursive action and vice versa. Validates or autocorrect discursive text action. Allow student to solve and write a problem. Return message to the student in some situations.
- *Teacher Agent*. Helps the teacher to modify the system using the messages content. Shows the strategy for a selected problem. Makes easier for the teacher to learn the student's behaviour. Modifies the message any time, even during the problem solving process. Allows teacher to supervise any selected student and shows him witch strategic path the student has used to solve the problem.

- *Didactic Expert Agent*. Allows didactic experts to create their own strategy, modifies and update the strategy according to changes after didactic study and experiment review. This agent assists the expert during the creation of a strategy. It helps to create the strategy automatically using a graph model. The strategy modification can use the data related to a student during the solving process.
- *Tutor Agent*. The Tutor agent is the engine of our TURING system. It works on the client and the server-side (for complex problem). The Tutor Agent takes the action and returns the best matching message to the student. It can offer help to the didactic expert to create a strategy or to show all the available possibilities. It generates messages using an intelligent approach, e.g., collects data (Request, Actions or Inferences) and generates invariants according to some characteristics managed by the didactic expert or the teacher (such as latency of student actions, level of tolerance, etc.).

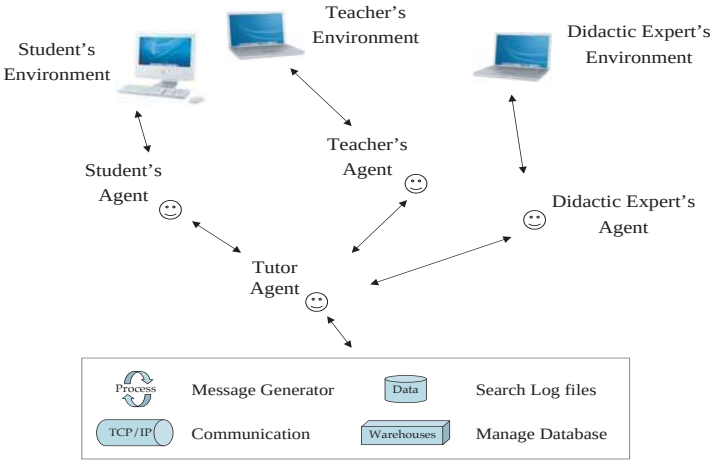


Figure 1. The TURING Architecture

3.2 Warehouses

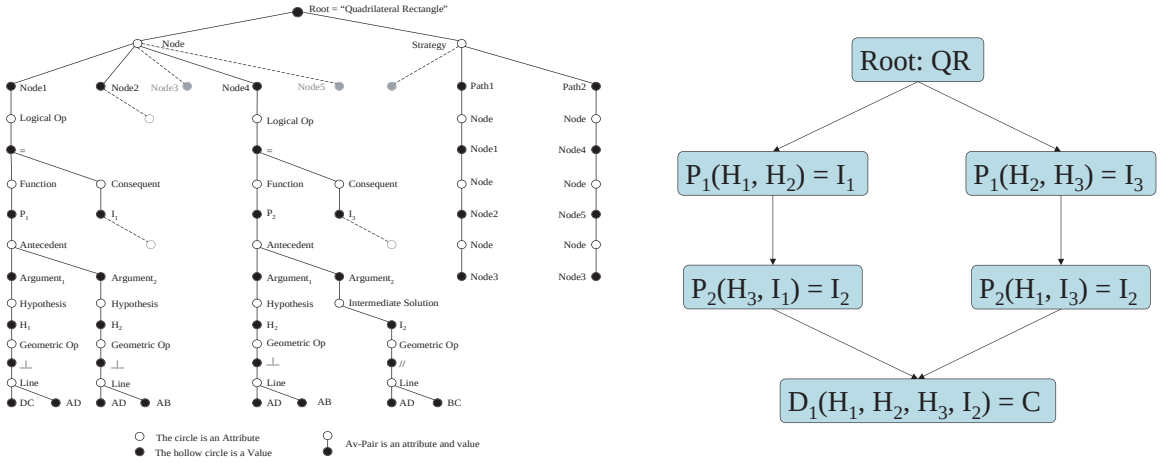
The warehouses are the central data structure or the database in the TURING system. They achieve suppleness and are based on a hierarchy of attributes and values. An attribute is a category in which an object can be classified, for example its «Line». A value is the object classification within that category like «AB». Together, an attribute and its associated value form an *attribute-value* pair or *av-pair*. The hierarchy of *av-pairs* allows the TURING provide a help to the student and teacher to precisely describe the strategy for each problem in different fields it might be used. While several complex request languages exist in the literature, our approach, based on attributes and values, is particularly simple. We also design the warehouse structure to be independent of the specific language used to perform requests (for example a student can solve the problem in any language he wants, the request will be indexed, and the system will process the request without any dependent with the language of the student), so that it can also be used in the context of other description languages.

The structure of a warehouse is a hierarchical arrangement of av-pairs, such that an av-pair is a descendent of another av-pair in the hierarchy when it is dependent on it (see Figure 2). There can be multiple values per attribute and each value can in turn be refined by multiple attributes. In comparison, the Figure 3 presents the strategy created by the didactic expert represented in a graph structure (mathematical proof in Euclidean geometry). The Graph is a set of nodes or inferences, where each node is a set of actions, linked by logical operators. Thus, in «Function(Argument$_1$, Argument$_2$) = Intermediate Result», Function can be a mathematical Property or Definition (P or D), while argument is an Hypothesis or Intermediate Result (H or I). The graph can be generated automatically using a deductive approach similar to *Baghera*, but the didactic expert with the TURING can create, modify and update it, especially, he can program semantic and discursive inferences for argumentative purposes.

The nodes in the graph have pointers to the next node (except the last one), which represents the passage from a cognitive state to another in the strategy of the didactic expert. Using functional notation, each node consists of three parts: *antecedent* (i.e. Argument$_1$, Argument$_2$), *consequent* (i.e. Intermediate Result or Conclusion), and *justification* (Function itself) and might be divided into multiple nodes with the same structure depending on the complexity of the cognitive state. Each path of the graph (composition of functions) represents a possible strategy in the problem solving.

3.3 Messages during the Solving Process

One of the most important issues in the TURING system concerns messages. For each action that the student can produce, a message must be available to improve mathematical competence. The messages are attached to each possible action and all its attributes, like «change node», «action state», «tolerance level», «action wait», «action true», «wait time» and many other parameters related to this action and the problem strategy (Aïmeur *& al*., 2005). Messages are generated using a combination of standard string text (created the didactic expert according to each problem) and the action attributes.



Figures 2 and 3. Warehouse structure and Graph structure

3.4 Student's Request

Students validate requests to identify their proposition during the process of solving a problem. Requests are built using arrangements of attributes and values, related by relational operators, which we call *av-relations* (for example, *av-relation* [Attribute: Line Value: AB]). A request consists of a simple action or a complex action (set of well-formed actions, e.g., a simple action [Attribute: Line Value: AB] // [Attribute: Line Value: CD]). Requests can be one or more inferences, one or more actions or any text, but separate agents will treat them. Student's requests are described using the following grammar:

```
<proposition> ::= <inference> | <action> | <av-relation> ;
<inference> ::= 'Because' '(' <action> 'and' <action> ')' 'according to' 'property' 'Then' '(' <action> ')' ;
<action> ::= '(' <av-relation> ')' <gop> '(' <av-relation> ')';
<av-relation> ::= attribute <lop> value ;
<lop> ::= '=' | '<=' | '>=' | '<' | '>' | '<>' ;
<gop> ::= '//' | '_|_' | 'E' | 'C' | '>' | '<>' ;
```

The *Validate* algorithm selects the student's answer and validates it using the grammar shown before. This algorithm processed by the *student agent*. It takes the proposition text and converts it to a format that would be understandable by the TURING system. The *validate* algorithm is based on the result returned from the auto-correct algorithm. The *auto-correct* algorithm selects the proposition text entered by the student (finite sequence of characters) and forms this proposition to a well-known format

3.5 Processing a Student's Request

The *Request* algorithm selects the answers that best fit the student's request. This algorithm is at the heart of the TURING system during the solving problem. It returns the message corresponding to the action's characteristics. The algorithm first starts by searching the warehouses corresponding to the student agent. If the characteristics of the action are found, the process returns the related message; this is the best-case situation. Otherwise, the algorithm sends the action to the tutor agent that runs on the client side until it finds the related answers; this is the intermediate case. If the request was not fulfilled, i.e., the related message is not found than, it sends the actions to the tutor agent on the server side that has the suitable message related to the requested action; this latter situation is the worst case

situation. The search starts in the student agent that communicates with the tutor agent if needed. The student agent treats the actions that are out of context (off the point) or to formalize the request to an understandable formula by the strategy. Only the tutor algorithm searches the didactic graph to choose the best path, which directs the student using a message strategy to follow the most efficient way that helps the student in his reasoning to solve the problem. To anticipate a future work, this algorithm will have the ability to decide what step will go or from where to start. It will be general to all problems and any strategy. It will be tested and the performance will be evaluated. The algorithm treats the invariants (cognitive and meta-cognitive), thus decreasing the number of messages associated with the matrices (see below).

Formally, we define *Request* and *Validate* algorithms as follows:

Algorithm 1 *Request*

Let *Act:* action or an elementary step of reasoning.
Let *Act[]:*array of actions
Let *Sp:* Step is a stage of reasoning. It is a sequence of actions. It is represented as a node or in the didactic graph.
Let *Pr*: Proposition that represents a finite sequence of characters entered by the user
Let *SAW*: Student agent warehouse content. It contains all the information related to the student agent
Let *LW*: local warehouse content. It contains all the information that concern small problems
Let *RW*: remote warehouse, similar to *LW content* but on remote server and contain all the information.
Let *Compare(Act, Sp):* return true if the characteristics of the action are found, if not it return false
Let *Ans:* list of string message filled by the *compare* function;

Act[] ← *Validate(Pr)* /* *See Algorithm 2 "Validate"* */

A-Best case
For each *Act* **in** *Act[]*
  If *Compare(Act, Sp, SAW)*
    Return *Ans*
  Else
    Go To "B-Intermediate case"
  End-If
End-For

B-Intermediate case
For each *Act* **in** *Act[]*
  If *Compare(Act, Sp, LW)*
    Return *Ans*
  Else
    Go To "C-Worst case"
  End-If
End-For

C-Worst case
For each *Act* **in** *Act[]*
  If *Compare(Act, Sp, RW)*
    Return *Ans*
  Else
    Ans ← "out off point" message
    Return Ans
  End-If
End-For

Algorithm 2 *Validate*

Let *Act:* action or an elementary step of reasoning
Let *Act[]:* array of actions
Let *Pr*: Proposition that represents a finite sequence of characters entered by the user
Let *GetStudentText()*: return the text entered by the student in the text zone
Let *WaitAct*: represent an action generated with wait characteristics
Let *Timeout():* waits for an arbitrary period of time
Let *ErrMsg()*: message related to a syntax error in the student's proposition
Let *Break(Pr)*: break up the proposition and return a list of actions. It returns null if proposition is understandable
Let *Send(Pr)*: return Boolean variable equal true if the student send the proposition
Let *Sentence(Act)*: return the type of the action;

Pr ← *GetStudentText()*

A-Check proposition syntax

If *Timeout()*
  Pr ← *Pr + WaitAct*
End-If

If Not *Grammar(Pr)*
  Return *ErrMsg()*
  If *Send(Pr)*
    Go To "B-Break up Proposition"
  End-If
  Validate(Pr)
End-If

B-Break up Proposition

Act[]← *Break(Pr)*

For each *Act* **in** *Act[]*
  Select *Sentence(Act):*
    Case Sentence(Act) : Return *Msg(Sentence(Act))*
End-For

Return *Act[]*

3.6 Sample Scenario

The following scenario is an example of how the TURING system might be used and how it works. In this scenario, we use five different cases, which allow us to demonstrate the suppleness of our implementation using the *Validate* and *Request* algorithms. We start with the first example: a student wants to solve a quadrilateral rectangle problem. To initiate the proposition, the student just needs to fill out the discursive text field in user interface (see Figure 4) with proposition (i.e. Pr = «Because H_1[(AB) // (CD)] and H_2[(AB) ⊥ (AD)], Then I_1[(AD) ⊥ (CD)]»). In this scenario we supposed that the student use the correct syntax for the proposition (the auto correct algorithm is responsible of the correction of the proposition's syntax).

Figure 4 shows *Content of Warehouses* and how the didactic expert can create the problem strategy and save the related information in the warehouses. The information described in Figure 2 and Figure 3 helps us to understand the scenario we are describing.

The student utilizes student interface of the TURING system to send the proposition. In this example we use the didactic expert strategies shown in Figure 3 and the algorithm *Request* described in Section 3.5 *Processing a Student's Request*. The example investigates the following cases:

Case 1: *Pr* = "I am beautiful". Action is processed as being off the point. *Request* algorithm returns a random message from an *off the point* table of messages.
Case 2: *Pr* = "(AB) = 4 cm". In this case where *Pr* is correct, the *Request* algorithm return a message showing that this answer is not helpful in solving this problem.
Case 3: *Pr* = "(AB) // (CD)". This action is not available in the strategy. *Request* algorithm returns a strategic message indicating the wrong proposition.
Case 4: *Pr* = "Because (AB) // (CD) and (AB) ⊥ (AD)". The actions in this proposition are correct but still something is something to complete the inference. In this case a strategic message returned by the *Request* algorithm asking for the missing actions.
Case 5: *Pr* = "Because (AB) // (CD) and (AB) ⊥ (AD), Then (AD) ⊥ (CD)". The proposition *Pr is correct*. In this case the *Request* algorithm returns a congratulation message.

Although this scenario is simplified for the sake of brevity, it illustrates how to use the TURING *student interface*, and how to ask for a proposition. The examples describe, step by step, the manner in which the proposition is processed using the *request* algorithm. Currently, by means of this implementation, the learning in problem solving is becoming increasingly efficient and flexible.

4 Implementation

4.1 The TURING User Interfaces

TURING is an intelligent tutoring system that is made up of three main constituents: student, didactic expert and teacher (see Figure 1). The *student interface* is used to solve and write a given problem. Figure 4 shows a dynamic Cabri figure. This figure is pre-built in Cabri Geometry II plus. In this interface, the student can validate an action like the discursive writing of «AB = 5 cm», the graphic selection of (AB) // (CD) or a request like a set of actions to make an inference. These actions are the meaningful actions in the context of the problem, i.e. a part of graph. The validate option is a part of the student agent; it takes the correct request and validate it with the available data concerning the problem and the student. The student interface is based on the student agent that also allows the translation between the graphic part and the discursive text. It has the ability to correct the student's text and gives help to choose the best expression or request. Actually, the system seeks to check syntax (well-formed statement) so that it is usable by the system (standardization in the internal). The student interface offers many other options that help the student to track his activities or to ask a question related to the solving process. In comparison, Figure 5 shows the *didactic expert interface*, which is based on the didactic expert agent; this interface allows the expert to update or create a new problem and its strategy in the form of a graph. The graph is the structure to represent an adapted version of the *basic space of the problem* in the sense of Cobo & Fortuny (2000). The tutor agent uses the strategy entered by the didactic expert as the base to compare or process the student's action and returs to him suitable messages or others if necessary.

The *teacher interface* is based on the teacher agent so that the real teacher can adjust the system and modify the strategic messages according to the practices of its students. This interface gives the teacher the ability to change some invariants that change and adjust the TURING message strategy; it also allows the teacher to send messages to

the student according to what it sees when the process takes place during the class time. The student's path is in red to indicate the state of the solving process of a selected student for a specific problem. This interface allows the teacher watching many students at the same time, being able to consult the history of their activities, a log file that records each move and the conversation activities between the student and the agent.

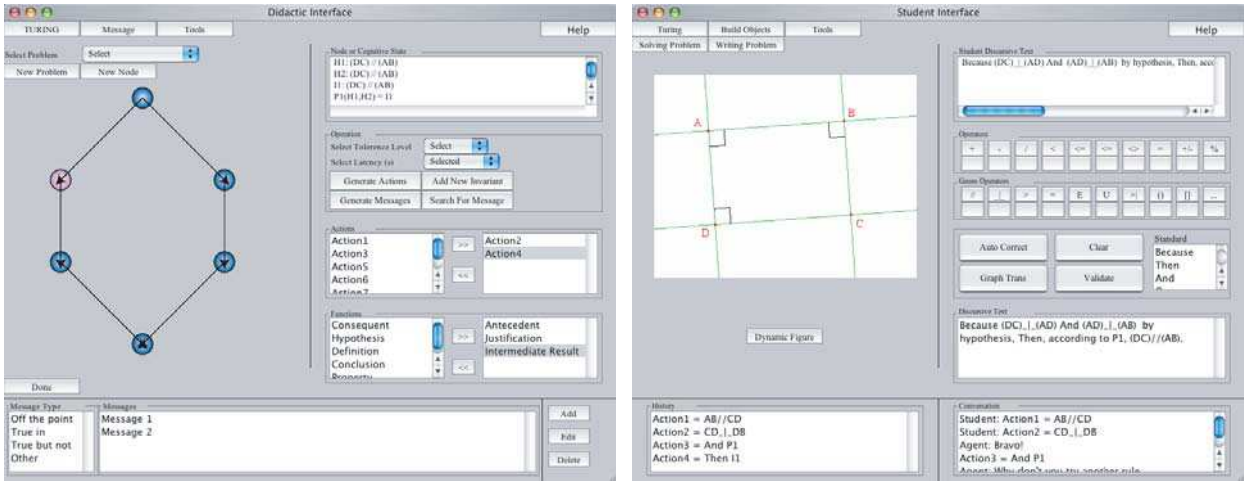


Figure 4 and 5. The TURING didactic expert's interface and student's interface to solve and write a problem

5 Related Work

Improve teaching and learning process, the main objective of TURING system, is also the goal of many other projects currently under way at several research centers. Like immediate antecedent of the TURING, the *Baghera* project, directed by the researcher Nicolas Balacheff within the Leibniz Laboratory (Laboratoire Leibniz, 2003). *Baghera* uses algorithmic models in proving geometric problem to check the strategy automatically with a «deductive engine». The *AgentGeom* (Cobo *& al*., 2005) is a first prototype of an intelligent tutor system that uses the construction of a geometrical figure to generate interactions. Just like *Baghera*, *AgentGeom* is conceived as a multi-agent tutorial system of diagnosis that can identify knowledge of the students through their interactions with the system that comprises an adaptation of the system to the cognitive characteristics of the students and to the evolution of their mathematical knowledge (see Richard, Fortuny *& al.*, 2004). *Exploragraph* (Dufresne & Paquette, 2000) focuses on the design of interface for learners taking distance courses and also to integrate in it adaptive functions and advices to support them but they dont use argumentative messages in their support. The *Cabri* (Cabri-géomètre, 2005) is a software aimed to create geometric figures based on the geometric logic. Unlike these projects, the TURING may be distinguished not only by the new integration of theoretical approaches, but especially by the adaptable nature of the heuristic strategy and the strategic messages (discursive and argumentative). Jointly with *AgentGeom*, TURING system uses an implementation approach starting from didactic studies with real students for each problem. The implementation can be improved after testing and evaluating it into a socio-cultural reality. The strategy is not related to a pre-existent model of solving process, but it could be adapted so that the student can realize the distinctiveness of each problem; in addition the TURING is a real time system. Our current research can benefit from the messages, the strategy and the algorithms we propose in our context.

6 Conclusion and Future Work

This paper describes the aspects related to the technological characteristics of the TURING system. We believe that our work is a suitable approach to enhance and improve Mathematical Competence In Problem Solving trough interactions with a multi-agent system. The TURING system is based on both studies of human behaviour and system's processes. It simulates the collaborative behaviour in the class with a student, using strategic messages for a given problem. The system agents' architecture and the warehouses structure assure the suppleness and the intelligence of our system. Our implementation is in Java to take advantage of its cross-platform portability. The warehouse structure is developed in XML; access to the warehouse structure is performed in Java using the XML package developed by IBM (W3C, 2004).

Following the work we presented above, we anticipate that TURING may become an advancement compared to other technologies in the field. Nevertheless, there are many aspects that we would like to investigate. In the following we list some of them:

- Define a process to limit the size of stored information (warehouses) by removing unused data according to some priority rule. This involves studying the notion a gradually forgetting about rarely used information.
- Analyzing the performance and the complexity of the algorithms and optimize them.
- Analyzing the influence and effects of using TURING system on the development of the students' strategic competences and Validate our system using real student environment in a class at high school.
- Generalize our system to be able to work in any e-Learning domain such as medical, physics, etc.

References

Aïmeur, E. (1998). Application and Assessment of Cognitive Dissonance Theory in The Learning Process. *Journal of Universal Computer Science, 4*(3), 216-247.

Aïmeur, E. et Frasson, C. (1996). Analyzing a New Learning Strategy According to Different Knowledge Levels, Computer and Education, An International Journal, 27(2), 115-127.

Aïmeur, E., Cobo, P., Fortuny, J.M. & Richard, P.R. (2003). Stratégie argumentative et système tutoriel pour l'apprentissage interactif de la géométrie. In *Actes de l'EMF-2003 (Espace mathématique francophone)*, Tozeur.

Aïmeur, E., El-Khoury, S., Fortuny, J.M. & Richard, P.R. (2005). An 0pen Architecture to Improve the Mathematical Competences in High School. *Proceedings of the ED-Media World Conference of Association for the Advancement of Computing in Education*, Montréal.

Allen, R. J., & Trilling, L. (1997). Dynamic Geometry and Declarative Geometric Programming. In J. R. King & D. Schattschneider (Ed.), *Geometry Turned On. Dynamic software in learning, teaching, and research* (pp. 193-197). Washington: MAA Notes Series.

Brousseau, G. (1998). *Théorie des situations didactiques*. Grenoble: La Pensée Sauvage.

Cabri-géomètre (2005). *Proceedings of the CabriWorld 2004*, Rome.

Cobo, P. & Fortuny, J.M. (2000). Social interactions and cognitive effects in contexts of area-comparison problem solving. *Educational Studies in Mathematics, 42*, 115-140.

Cobo, P., Fortuny, J.M., Puertas, E. & Richard, P.R. (2005). AgentGeom: a Multiagent System for Pedagogical Support in a Geometric Proof Problem. *International Journal of Computers for Mathematical Learning*. Manuscript in evaluation.

Dufresne, A., & Paquette, G. (2000). *ExploraGraph: A Flexible and Adaptive Interface to Support Distance Learning*. World Conference on Educational Multimedia, Hypermedia and Telecommunications 2000(1), 304-309.

Duval, R. (1995). *Sémiosis et pensée humaine*. Berne: Peter Lang.

Fortuny, J.M., Giménez, J. & Richard, P.R. (2002). Distance Education at Secondary Levels: Contexts and Norms for the Learning of Mathematics. *Proceedings of the E-Learn 2002 World Conference of Association for the Advancement of Computing in Education*.

Kieran, C. (2001). The Mathematical Discourse of 13-year-old Partnered Problem Solving and Its Relation to the Mathematics that Emerges. *Educational Studies in Mathematics, 42*, 115-140.

Laboratoire Leibniz (2003). Baghera Assessment Project: Designing an hybrid and emergent educational society. In Soury-Lavergne S. (Ed.), *Rapport pour la commission européenne, Programme IST, Les Cahiers du Laboratoire Leibniz nº 81*, Grenoble.

Lakatos, I. (1984). *Preuves et réfutations. Essai sur la logique de la découverte mathématique*. Paris: Hermann.

Moschkovich, J. N. (2004). Appropriating mathematical practices: a case study of learning to use and explore functions through interaction with a tutor, *Educational Studies in Mathematics, 55*, 49-80.

Richard, P.R. (2004a). L'inférence figurale: un pas de raisonnement discursivo-graphique. *Educational Studies in Mathematics, 57*, 229–263.

Richard, P.R. (2004b). *Raisonnement et stratégies de preuve dans l'enseignement des mathématiques*. Berne: Peter Lang.

Richard, P.R., Fortuny, J. M., Cobo, P. et Puertas, E. (2004). Pedagogical Support in the Solving and Proving of a Geometry Problem trough Interactions with an Agent Tutor. *Proceedings of the E-Learn 2004 World Conference of Association for the Advancement of Computing in Education*. Washington.

W3C (2004, August 27) *Extensible Markup Language (XML)*. Retrieved March 18, 2005, from http://www.w3.org/ XML/.

Acknowledgement

Prof. Philippe R. Richard and Prof. Esma Aïmeur could contribute to the development of this research thanks to a subvention granted by the *Fonds québécois de la recherche sur la société et la culture* (FQRSC 2005-AI-97435, Gouvernement du Québec) and the logistic support of the *Laboratoire Turing* (http://turing.scedu.umontreal.ca).