

AMBRE-teacher: a module helping teachers to facilitate the integration of AMBRE in classrooms

Stéphanie Jean-Daubias, Nathalie Guin

Université de Lyon, France

Université Lyon 1 – LIRIS, CNRS, UMR5205

43 bd du 11 novembre 1918, 69622 Villeurbanne Cedex, France

{Stephanie.Jean-Daubias, Nathalie.Guin}@liris.univ-lyon1.fr

Abstract. If teachers use few ILEs (Interactive Learning Environments) with their students, it may be because they don't have the opportunity to act upon the available environments. To permit the adoption of ILEs by teachers, these systems must be adaptable to the learning context and to the teacher's pedagogical approach. To achieve this we propose a module dedicated to the teacher in the context of the AMBRE project. This module, AMBRE-teacher, allows the user to configure the ILE he wants to use. We identified the functionalities necessary to adapt an AMBRE ILE. We notably designed a knowledge based system allowing the teacher to generate the problems he wants to propose to his students in the ILE. Our approach is implemented for AMBRE-add, an ILE proposing word additive problems in elementary school.

Keywords. adaptation of ILE, role of teacher, problem generation, knowledge.

1. Introduction

The AMBRE project is a multidisciplinary study in computer science, cognitive sciences, mathematical didactics and educational science, which aims at designing learning environments for the acquisition of a specific method in problem solving. In each application domain, a method is based on the classification by the learner of problems and solving tools. We propose, to help the learner acquire such methods, to use Case-Based Reasoning, a paradigm developed in Artificial Intelligence and inspired by research in psychology on reasoning by analogy. After the evaluation of a first prototype for the numbering problems domain (final scientific year level, 18 year-old students), we implemented and tested a complete system for additive word problems solving which are studied in primary school: AMBRE-add [13].

In order to facilitate the integration of AMBRE in education, we conceived a module dedicated to teachers, AMBRE-teacher. In this paper, after presenting the functionalities of this module, we describe the functionality dedicated to problem generation and present the architecture of the knowledge based system allowing to implement it.

2. AMBRE: “which role for the teacher?”

Most ILEs (Interactive Learning Environments) focus on the computer/learner couple, thus often masking the important role of teachers in these environments [17]. ILEs rarely consider the teacher as a potential user distinct from the learner. Besides, we can notice that only a very small part of ILEs intelligence is devoted to teachers. Now it seems essential to consider that teacher’s place is distinct from learners’ one in ILEs, and that a part of the system should be developed for him [1] [10]. For this, it is necessary to identify the different roles that teachers may play in ILEs: designer or design partners, prescribers or user.

Designer or design partner The teacher can take part directly into the ILE design, this can be achieved by taking him into account in different manners [7]. Participatory design [16] allows teachers to be actors of the design: they are not only observed and questioned about their practices, but also integrated in the design process, where they can make innovative propositions and even participate directly in the design choices, thus allowing the creation of a software really matching their needs and real practices. Informant design [15] can be defined as an approach calling users as informants during design, without restricting them to a passive role, but not either considering them as full partners. For example they can work with the designers on prototypes, but don’t take part in final decisions.

Author The teacher himself can be an ILE designer as user of an authoring tool. Such tools allow teachers to build educational software. As ILEs are very complicated software, authoring tools are generally limited to the creation of courses resources or intelligent tutoring systems. GenDoc, developed in the context of the ARIADNE project, is an example of such systems [4].

Prescriber In most ILEs, the teacher has the role of prescriber: he chooses the system to be used by his pupils, according to his needs and pedagogical choices.

Secondary user This is the most frequent role of teachers in ILEs, mostly implicit. In that case, the ILE is mainly centered on the learner, but leaves a place to the teacher to handle the use of the ILE. The teacher can thus adapt and personalize the system dedicated to learners, add pedagogical situations, interact with learners during the use of the ILE, but also make a report of the learners’ session. The teacher thus uses the ILE to adapt it into his pedagogical strategies, and to its main users, the learners, as permitted for example by the systems Roboteach [9] and Aplusix [11].

Main user Some systems are dedicated directly to the teacher by proposing him tools helping him in his work. ILEs of this type have an approach centered on the teacher and not on the learner. This is the case of Eprofilea environment which proposes a set of tools to facilitate the following up of learners by the teacher [8].

In the AMBRE project, if several teachers took part as **design partners** in the framework of differentiated design [7], teachers' major role is **main user** of a module dedicated exclusively to them. Thus they explicitly take their role of **secondary user** of the AMBRE learner module through the proposed environment, by adapting and defining the parameters of the learner environment. Finally, teachers keep their role of **prescribers**, by choosing the ILE used in their classroom.

3. AMBRE-teacher

In the context of the AMBRE project, which gives rise to the creation of ILEs intended to learning methods, we want to propose to the teacher an environment allowing him to tune the learner environment, to create sequences for the whole class or specifically for some learners, and more particularly to generate problems to be solved. This environment, specifically intended to teachers has to match their needs and to permit them to integrate and adapt an AMBRE ILE to their approach, to their pedagogical strategies, but also to the contexts in which they work. For this, AMBRE-teacher allows them to generate problems, to create learning sequences (sets of activities intended to learners) and exercises themes, but also to tune the learner environment, to create lists of pupils and to assign the work (sequences and exercises) to classes or learners.

The **learner environment tuning** consists of the personalization by the teacher of the learner software interface: mainly colors and language choice. AMBRE-teacher must also permit teachers to establish the list of learners for their classes.

Some exercises, created by the designers, are furnished with the ILE. But in order to allow teachers to propose problems with characteristics that they define for their learners, AMBRE-teacher includes a **problems generation module**. This generation is a possibility offered to teachers, not an obligation. This approach seems therefore to be necessary to increase and vary the set of exercises of AMBRE ILE, and to adapt the wordings to the learners. To generate a problem to be solved in the AMBRE ILE consists of proposing a wording in natural language and a description of the problem consistent with the solver used by the learner software. From the teacher point of view, generating the problem consists of specifying some characteristics of the problem. The problem generation can be more or less automated depending on the teacher's choice: he can either specify all the problem characteristics, only some of them, or none of them.

We also want to offer the teacher the possibility to **create learning sequences** (sets of problems to be solved in one or several sessions of software use) by using the learning material (the problems) that he created. This functionality can permit him to instantiate his strategy and his pedagogical approach, by integrating, as he wants, the problems of his choice in a sequence. For this, we propose to the user two ways to create a sequence: manual or automated. For the manual creation, the teacher selects the exercises he wants to integrate in his sequence. He can choose to make them appear chronologically or randomly, depending on his strategy. For example, he can create a sequence in which the difficulty level increases progressively with the problems to be solved. In that case, he will choose by himself the order of the problems presentation in

the sequence. The teacher can then define the behavior of the learner software for the exercises sequence: number of attempts allowed for the learner for each exercise step (and what to do if this number is reached), number of verification or help functionalities uses allowed during one exercise, behavior of the diagnosis (always diagnose or let the learner make mistakes at certain resolution steps), etc. We define the default behavior for all exercises of the sequence. However, if the teacher wants to integrate a progression in the sequence, he may define a different behavior depending on the exercises (he may thus for example allow the highest level of diagnosis at the beginning of the sequence and reduce it progressively). For the automated creation of a sequence, the teacher chooses only the exercises folder in which the system will take the exercises and the number of exercises he wants in the sequence. The system then chooses randomly the exercises. The teacher can then modify the system's propositions (delete, add or replace a problem), or reorder exercises differently, define a specific behavior, etc.

AMBRE-teacher must also allow to **assign the work** to learners, that is to say to associate the whole class or each learner with one or several sequences. The teacher can propose the same sequences to all of his students or choose to individualize learning by proposing specific sequences to certain learners.

To allow the teacher to entirely adapt the generated problems to his environment and to his students' context, AMBRE-teacher will permit to **manage surface features** that will be used in the exercises. The teacher may then create, modify and delete exercises themes and surface features linked to the themes (for example specific objects and characters, associated actions, etc.).

4. A problem generator for AMBRE

We have chosen a semi-automatic generators approach [6] [14] [5], which builds the wording of the problems, but let the user intervene in the creation process. Actually, on the one hand, automatic generators [1] [12], permitting no interaction with users, are not suitable to our aim. On the other hand, manual generators [3], like authoring tools, are not able to solve the problems they allow to create nor to propose any diagnosis of the learner's answers or help functionalities.

An AMBRE ILE is based on a knowledge based system which relies on a problem solver and allows to provide the learner with help, a diagnosis of his answers and explanations concerning his errors [5]. The problems proposed to the learner must be understandable by the solver to allow the ILE to provide these functionalities.

With AMBRE-teacher, teachers can influence the problems to be generated, by specifying a set of constraints on the exercises to generate. As the problems are built by the system from these constraints, the result of the generation will not only be a wording in natural language, but also a model of the problem usable by the solver.

4.1. The problem generation environment for the teacher

We designed and implemented a tool for problem generation dedicated to teachers, for the word additive problems domain suited to AMBRE-add used in primary school. Problems of this domain describe concrete situations, for example a marbles play: “Alex had 32 marbles. At the end of play, he has 45. How many marbles did he win?”. This domain has been widely studied in mathematical didactics and several problems classifications have been established. The one we use in AMBRE-add is presented in [5].

For word additive problems, the teacher can define constraints of four types: structure features, surface features, values and complication (Figure 1 gives an overview of constraints available in AMBRE-teacher).

Figure 1. Report screen of the teacher environment of AMBRE-add, with the preview of an exercise generated by the system.

The **structure** of a problem to be generated corresponds to the class of the problem. This class is defined by several attributes that can be set or not.

Surface features are the elements that complete the produced wording. The teacher can specify some elements of this category, for example themes, objects and characters.

He can also choose the **values** of the data that will be used in the problems or define an interval for each required values and the wanted difference between min and max values, allowing the carrying over or not, etc.

Complication concerns all options proposing to complicate the wording of the problem to adapt it to the students' level. Designing this part required a close collaboration with teachers to identify their needs. The environment proposes language complications and complications of the wording itself. For word additive problems,

complication takes the form of vocabulary used and turn of phrases complexity, writing of numbers in full, modification of the sentences order, addition of distractor sentences, addition of non pertinent data.

Not all constraints are mandatory for the exercises creation. Constraints not specified by the teacher will be randomly defined by the system. Figure 1 presents the report screen of the teacher module of AMBRE-add. This screen sums up the constraints defined by the teacher for the four categories of constraints and presents an example of problem which could be generated from these constraints.

The four categories defined for word additive problems are not reusable as it for another application domain of AMBRE. Nevertheless, structure features, surface features, and probably complication will still be necessary. Values will only be present for numerical domains.

4.2. The GenAMBRE architecture

The problem generation process that we established in the GENAMBRE architecture takes as input the set of constraints specified by the teacher and gives as output two elements: the wording of the problem in natural language for the learner and a computer-usable formulation of the wording named descriptive model of the problem, for AMBRE problems solver.

The problem generator architecture for AMBRE is presented in **Figure 2** and each of its components is presented in the following of the section. For a D domain (for example the additive word problems domain), the five knowledge bases of the *domain level* required by the *generation level* must be defined by using the domain independent formalisms of knowledge representation. The problem generation process is done in two stages: the system builds a problem generation model, then builds the wording in natural language and the descriptive model of the problem. Both these processes are domain independent. Both processes and the knowledge bases of the D domain, constitute together a problem generator for the D domain: GenAMBRE-D.

Classification knowledge For each application domain, an expert gives to AMBRE solver, and consequently to GenAMBRE generator, a problems classification graph. This hierarchy of classes is used by the solver to classify the problem. This is a domain dependant class hierarchy, but its representation is the same for all domains.

Knowledge of the themes To generate a problem, it is necessary to know the concerned theme and the associated surface features (for example objects, characters and actions). Knowledge of the themes is given by the expert, or created by the teacher himself, through the surface features management module of AMBRE-teacher.

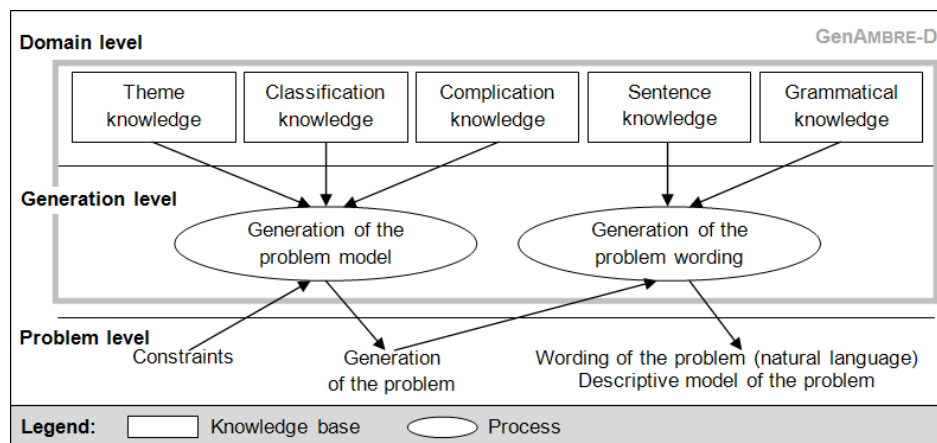


Figure 2. The GenAMBRE architecture.

Complication knowledge For additive problems, complicating a wording mainly consists in changing the sentences order and adding distractor sentences. So complication knowledge answers the following questions: how can one modify the order of the sentences of the problem? What distractor sentences can we add to problems and where can we place them?

Generation of the problem model process From the three knowledge bases previously described and from the constraints keyed in by the teacher, the system has to generate what we call a generation of the problem model. This model is an extensive descriptive model, because it also details the problem class and its theme. To build this model, the process fulfills the constraints defined by the teacher, notably by choosing random values for the undefined constraints.

Grammatical knowledge The domain expert must furnish a grammar to the generation system, that is to say a set of sentences structures that could be used in the domain.

Knowledge on the sentences The generated sentences, notably their structure, depend on the class of the problem. It is therefore necessary to know what sentences structures (from the grammar) could be used for the problem to be generated to permit to generate the wording in natural language. So, knowledge on the sentences allows to associate the class of the problem to the usable sentences structures, and the associated elements of the problem.

Generation of the problem wording process To generate a wording in natural language, the process uses knowledge on the sentences and domain grammar, as well as the generation of the problem model previously created. Knowledge on the sentences allows the system to take conceptual decisions (deciding what to tell), then the process goes to the text generation step, and establishes syntactical treatments (deciding how to tell it), lexical and morphological treatments (deciding how to write it).

4.3. Implementation of AMBRE-teacher for AMBRE-add

We implemented this generator for the domain of word additive problems, by supplying our generic architecture with all the knowledge bases necessary for this domain. The AMBRE-teacher interface is done in Delphi. Prolog is used for the knowledge management part and all AI treatments. XML files are used for the communication between Delphi and Prolog.

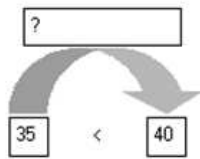
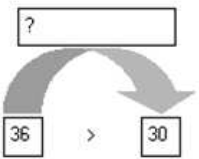
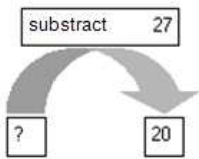
John is at school with Paul. Before playtime, he had thirty-five marbles. Now he has forty. How many marbles did he win or receive from Paul.	Emily and Anthony are in the playground. Emily had thirty-six green marbles before playing. After, she has thirty. Look for the number of green marbles lost or won by Emily during the play.	Its four pm. Ethan had red marbles before playtime. He lost twenty-seven of them by playing with his friends. Now he has twenty. Look for the initial number of Ethan's red marbles.
		
The problem is written: 35+?=40 The calculation is written: 40-35=? The solution is: 5 The answer is: John won 5 marbles	The problem is written: 36-?=30 The calculation is written: 36-30=? The solution is: 6 The answer is: Emily lost 6 green marbles	The problem is written: ?-27=20 The calculation is written: 20+27=? The solution is: 47 The answer is: Ethan had 47 red marbles

Figure 3. Examples of exercises generated by Ambre-teacher for Ambre-add: problems and solutions (translated from French).

The system allows to create problems for AMBRE-add. On one hand AMBRE-teacher generates problems wordings in linguistically correct natural language, matching the constraints defined by the teacher and suitable to teachers' expectations (cf. Figure 3 for examples of generated problems and solutions). On the other hand, it generates descriptive models of the problems (cf. Figure 4 to see the Descriptive model of the third problem of Figure 3) suited AMBRE-add's requirements (Figure 5 shows the use of this problem in the learner module).

<p>wording_natural(EXres_sub_val,'Its four pm. Ethan had red marbles before playtime. He lost twenty-seven of them by playing with his friends. Now he has twenty. Look for the initial number of Ethan's red marbles.').</p> <p>wording_plan(EXres_sub_val,[[['number of Ethan's red marbles', 'initial']], 'lost', 'twenty-seven of them', 'has twenty']).</p> <p>distractor_sentences(EXres_sub_val,['Its four o'clock pm.']).</p> <p>theme_pb(EXres_sub_val,game).</p>		
<p>fact(problem(EXres_sub_val)).</p> <p>fact(verb(EXres_sub_val,'lose')).</p> <p>fact(action(EXres_sub_val,p)).</p> <p>fact(is_a(p,play)).</p> <p>fact(participants(p,'Ethan',i)).</p> <p>fact(is_a('Ethan',person)).</p> <p>fact(play(p,t1,t2)).</p> <p>fact(has('Ethan',t1,e)).</p>	<p>fact(is_a(e,set)).</p> <p>fact(all_element(e,z)).</p> <p>fact(is_a(z,'blue marble')).</p> <p>fact(size(e,t)).</p> <p>fact(lost('Ethan',p,m27)).</p> <p>fact(is_a(m27,set)).</p> <p>fact(all_element(m27,y)).</p> <p>fact(is_a(y,'blue marble')).</p>	<p>fact(size(m27,27)).</p> <p>fact(has('Ethan',t2,m20)).</p> <p>fact(is_a(m20,set)).</p> <p>fact(all_element(m20,x)).</p> <p>fact(is_a(x,'blue marble')).</p> <p>fact(size(m20,20)).</p> <p>fact(to_compute(EXres_sub_val,t)).</p>

Figure 4. Example of descriptive model of a problem generated by AMBRE-teacher for AMBRE-add (translated from French).

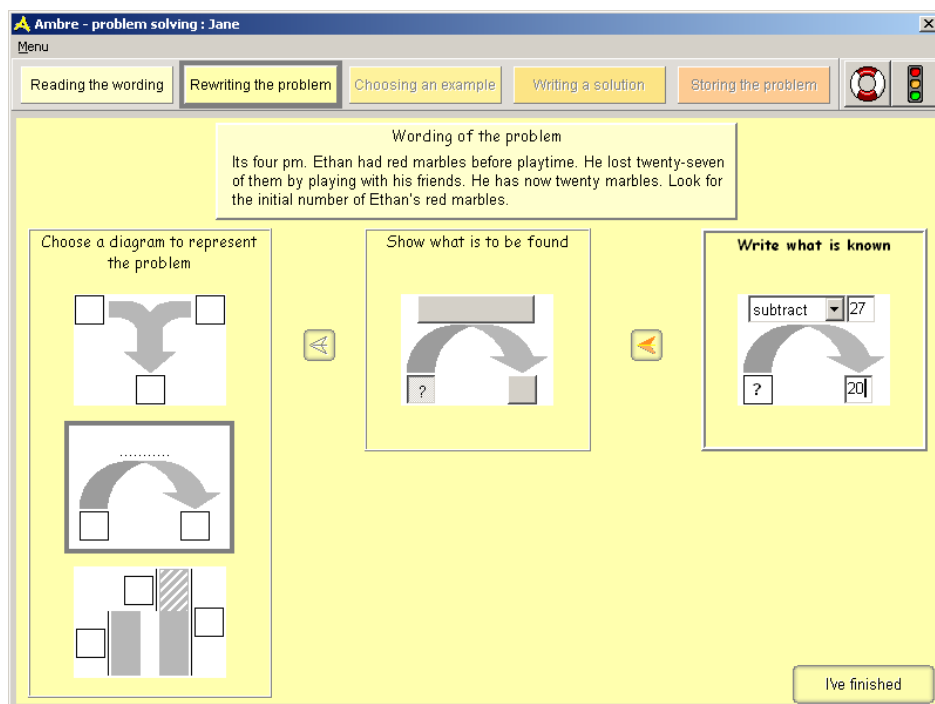


Figure 5. Example of a problem generated by AMBRE-teacher used in AMBRE-add, the learner module.

5. Conclusion

In this paper, we presented how we designed a module dedicated to teachers for the AMBRE-add ILE, to allow them to adapt the ILE to the learning context and to their pedagogical approach. By adopting a generic approach, we identified, with the help from teachers, functionalities that a teacher module must propose for an AMBRE ILE (AMBRE-teacher). We have also enabled the teacher to configure the environment, generating problems suited to his needs, creating learning sequences suited to his students by choosing the problems and the behavior of the ILE, assigning these sequences to his students, and creating new themes of exercises.

These functionalities are implemented for the word additive problems domain. For this, we designed a problem generation system whose architecture is domain independent. Even if we integrated teachers in the design of AMBRE-teacher, it is now necessary to evaluate it in actual classroom situations with a significant number of teachers. Finally, we must also validate the functionalities defined for AMBRE-teacher and the genericity of the GenAMBRE architecture by implementing a teacher module for an AMBRE ILE for another domain.

6. References

- [1] Bouhineau, D., Channac, S. (1996). La programmation logique par contraintes pour l'aide à l'enseignant, Intelligent Tutoring Systems (ITS'1996), Canada.
- [2] Burton, R. R. (1982). Diagnosing bugs in a simple procedural skill. Intelligent Tutoring Systems. London, Academic Press, 157-184.
- [3] David, J.-P., Cogne, A. and Dutel, A. (1996). Hypermedia exercises prototyping and modelising. Computer Aided Learning and Instruction in Science and Engineering. S. B. Heidelberg, 252-260.
- [4] David, J.P., Guilloux, C., Flament, A. (2002). A learning objects generator with xml-xslt technology, Conférence TICE2002, session ARIADNE.
- [5] Duclosson, N. (2004). Représentation des connaissances dans l'EIAH AMBRE-add, TICE'2004 conference, France, 164-171.
- [6] Giroire, H. (1989). Un système à base de connaissances pour la génération d'exercices dans des domaines liés au monde réel, PhD Thesis, Université Paris 6.
- [7] Jean-Daubias, S. (2009). Differentiated design: a design method for ILE. Research report RR-LIRIS-2009-015.
- [8] Jean-Daubias, S. and Eyssautier-Bavay, C. (2005). An environment helping teachers to track students' competencies, Workshop Learner Modelling for Reflection, Artificial Intelligence in Education (AIED'2005), Netherlands, 19-23.
- [9] Leroux, P. (1997). ROBOTEACH : un assistant pédagogique logiciel dédié à l'alphabétisation en technologie, Actes du cinquième Colloque International sur la Robotique Pédagogique, Montréal, Canada, 45-63.

- [10] Leroux, P. (2002). *Machines partenaires des apprenants et des enseignants – Étude dans la cadre d’environnements supports de projets pédagogiques*, Habilitation à Diriger des Recherches en Informatique de l’Université du Maine, Le Mans.
- [11] Nicaud J.F., Bouhineau, D., Chaachoua, H. (2004). Mixing microworld and cas features in building computer systems that help students learn algebra, *International Journal of Computers for Mathematical Learning*, vol. 9, 169-211.
- [12] Mitrovic, A., Stoinemov, L., Djordjevic-Kajan, S. (1996). INSTRUCT: Modelling Students by asking questions, *User Modelling and User-Adapted Interaction*, vol.6-4, 273-301.
- [13] Nogry, S., Jean-Daubias, S., Duclosson, N. (2004). ITS Evaluation in Classroom: The Case of AMBRE-AWP, ITS’2004 conference, 511-520.
- [14] Pecego, G. (1998). SYGEP, un Système de Génération d’Énoncés de Problèmes dans des domaines variés, PhD Thesis, Université Paris 6.
- [15] Scaife, M., Rogers, Y. (1999). Kids as Informants: Telling Us What We Didn’t Know or Confirming What We Knew Already?. *The design of Children’s Technology*, Allison Druin (ed.), Morgan Kaufmann, 28-50.
- [16] Schuler, D., Namioka, A. (1993). *Participatory design: Principles and practices*, Lawrence Erlbaum Associates, Hillsdale.
- [17] Vivet, M. (1990). Uses of ITS: Which role for the teacher?, *New Directions for Intelligent Tutoring Systems*, NATO ASI series, Vol. F91, Springer-verlag, Sintra.