An Authoring Tool to Elicit Knowledge to be Taught without Programming

Awa Diattara[1,2], Nathalie Guin[1], Vanda Luengo[3] and Amélie Cordier[1]
[1]Université de Lyon, CNRS, Université Lyon 1, LIRIS, UMR5205, F-69622, France
[2]Univ. Grenoble Alpes, LIG, F-38000 Grenoble, France
[3]Sorbonne Université, UPMC Université Paris 6, F-75005, CNRS, UMR 7606, LIP6, F-75005, Paris, France

Keywords: Authoring Tool, Knowledge Acquisition, Problem Solving Methods, Intelligent Tutoring Systems.

Abstract: Knowledge acquisition is a crucial problem for the design of Intelligent Tutoring Systems (ITSs). To overcome this problem, authoring tools have been proposed. Over two dozen of authoring tools have been built since the earliest days of ITS, but each of them focuses on a particular kind of ITSs such as constraint-based tutors or model-tracing tutors. In the context of the AMBRE project, we are interested in ITSs teaching problem-solving methods. Such ITSs enable learners to acquire a specific method in problem-solving. Despite of the variety of existing authoring tools, these tools do not meet our needs either because approach adopted do not match to AMBRE principle or because they do not allow to represent all knowledge needed to design an AMBRE ITS. We propose AMBRE-KB, an authoring tool to help authors to elicit knowledge needed for the design of AMBRE ITSs. This tool supports the acquisition of knowledge to be taught, and the description of problems to be solved. We present the authoring process and illustrate it using French verb conjugation domain. A preliminary evaluation shows that AMBRE-KB is successful in producing domains models but more thorough evaluation is planned.

1 INTRODUCTION

The work we present in this paper takes its origin in the AMBRE project (Guin-Duclosson et al. 2002) The purpose of this project is to design Intelligent Tutoring Systems (ITSs) for teaching problem solving methods (Schoenfeld, A. 1988; Schoenfeld 1985). Such ITSs enable the learner to acquire a specific method in problem-solving. In each application domain, a method is based on a classification of problems and solving tools.
To assist the learner in building his/her own classification, the AMBRE project proposes a learning process based on reasoning by analogy, called the AMBRE cycle (Nogry et al. 2008). The AMBRE cycle consists in showing first solved problems (serving as cases-base initialization) to the learner and then encouraging him/her to apply analogical reasoning to solve other problems. AMBRE ITSs are based on a knowledge-based system coded in Prolog, a programming language for knowledge representation. This knowledge-based system relies on a problem solver and uses two main types of knowledge: knowledge about the method to be taught and knowledge to guide the learner when he/she solves problems, providing assistance and diagnosing his/her answers.

Building an AMBRE ITS, like any other ITS, is a labor-intensive process that requires expertise in the domain application and in programming (Murray 2003; Woolf & Cunningham 1987). Our goal is to reduce the effort of making AMBRE ITSs by building an authoring tool that can generate the domain models. This tool should provide assistance to the authors during knowledge elicitation process. Our intention is to enable authors with no Prolog programming expertise to build their own ITS in the domain they are interested in.

Many authoring tools have been proposed in the literature (Blessing et al. 2007; Mitrovic et al.; Murray 2003a), but as far as we know, none of these tools meet our need, because either they do not match to AMBRE principle or do not allow to represent all knowledge needed for the design of an AMBRE ITS. This is why we propose to design AMBRE-KB, an authoring tool for the AMBRE project.

This paper presents AMBRE-KB, and illustrates the process of creating an ITS using this tool. This

Diattara, A., Guin, N., Luengo, V. and Cordier, A.
An Authoring Tool to Elicit Knowledge to be Taught without Programming.
DOI: 10.5220/0006315100820091
In *Proceedings of the 9th International Conference on Computer Supported Education (CSEDU 2017) - Volume 1*, pages 82-91
ISBN: 978-989-758-239-4

authoring tool assists the author in the process of knowledge acquisition and generates a Prolog version of this knowledge which is directly usable by the ITS.

The paper starts with a brief introduction related to knowledge acquisition methods and techniques used within ITSs authoring tools. Section 3 details the AMBRE-KB authoring tool and the knowledge acquisition process proposed. We also include a preliminary evaluation about how AMBRE-KB can be used to elicit knowledge needed for a given domain of learning. Section 4 presents conclusion and directions of future work.

2 RELATED WORK

In the literature, ITSs authoring tools have been classified into two main groups: pedagogy-oriented and performance-oriented (Murray 2003a). Pedagogy-oriented systems are those that focus on instructional planning and teach relatively fixed content. On the other hand, performance-oriented tools focus on providing rich learning environments in which students can learn skills by practicing them and receiving feedback.

REDEEM (Ainsworth et al. 2003; Ainsworth & Grimshaw 2003), ISD-Expert (Tennyson & Breuer 1994) and DNA (Shute et al. 1998) are examples of the first category of authoring tool. To acquire knowledge, these systems use an interactive dialogue box. But, one of the notorious limits of these systems is that generally, experts feel constrained by the fixed sequence of data entry.

Further work focused on the acquisition of procedural knowledge. Several systems have been developed. We may mention for example DISCIPLE (Tecuci & Keeling 1999), DEMONSTR8 (Murray 2003b). In DEMONSTR8, for example, the system infers production rules using programming-by-demonstration techniques, coupled with methods to further abstract the generated production.

Nevertheless, these systems are limited to domains where the knowledge can be represented step by step.

ASPIRE (Mitrovic et al. 2006), an authoring that enables the design of constraint-based tutors, belongs to the category of performance-oriented tools. To acquire knowledge in this system, the authors use ontologies: (1) construction of the domain ontology, (2) acquisition of syntactic constraints directly from the ontology, and (3) use of a dialog box with the expert in order to infer semantic constraints. But, this tool is limited to models based on constraints.

CREAM-Tools (Nkambou et al., 2003) also belongs to the category of performance-oriented tools since it allows the connection between skills and how to acquire them. For example, specific learning materials are linked to specific skills to support their learning. But approach adopted with this tool does not match to AMBRE principle.

Another authoring tool in this category is CTAT (Cognitive Tutor Authoring Tools). CTAT assists authors in the creation and the delivery of two kinds of tutors: cognitive tutors (Koedinger & Corbett 2006) and example-tracing tutors (Aleven et al. 2009; Aleven et al. 2016). Cognitive tutors are based on a cognitive model with rules of production, and concern generally tasks of problem resolution. The resolution is led step by step, and the behavior of the student at each stage is analyzed and corrected if it deviates from the planned procedure. However, cognitive tutors are limited to domain where the task of problem resolution is made step by step and where all the knowledge of the domain can be represented in the form of production rules. Moreover, the design of such tutors requires programming skills. Example-tracing tutors, the second type of tutors built by CTAT, have the advantage to be quickly developed without programming. To build an example-tracing tutor, the author builds at first the student interface using graphic tools, then he/she defines a graph of resolution of the strategies of resolution of the learners and their misconceptions. Example-tracing tutors present the same inconvenient as model-tracing tutors concerning the resolution of problems which is led step by step.

ITSs designed within the AMBRE project belongs also to the category of performance-oriented tools, since they provide a learning environment in which students can learn how to solve problems in various domains and receive feedback about their answers. In particular, AMBRE ITSs are based on a knowledge-based system coded in Prolog. This system relies on a solver which uses two categories of knowledge: knowledge about the method to be taught and knowledge to guide the learner providing him/her assistance and diagnosis of his/her answers. The knowledge to be taught is constituted by three types of knowledge: classification knowledge, reformulation knowledge and resolution knowledge. Problems are given to the system as a model we call *descriptive model* (presented in section in section 3.1). To solve a problem for a given domain of learning, the solver uses the classification knowledge and the reformulation knowledge to (i) determine the class of the problem and (ii) to build a new model of the problem called *operational model*. Then, the resolution itself consists in applying to the operational

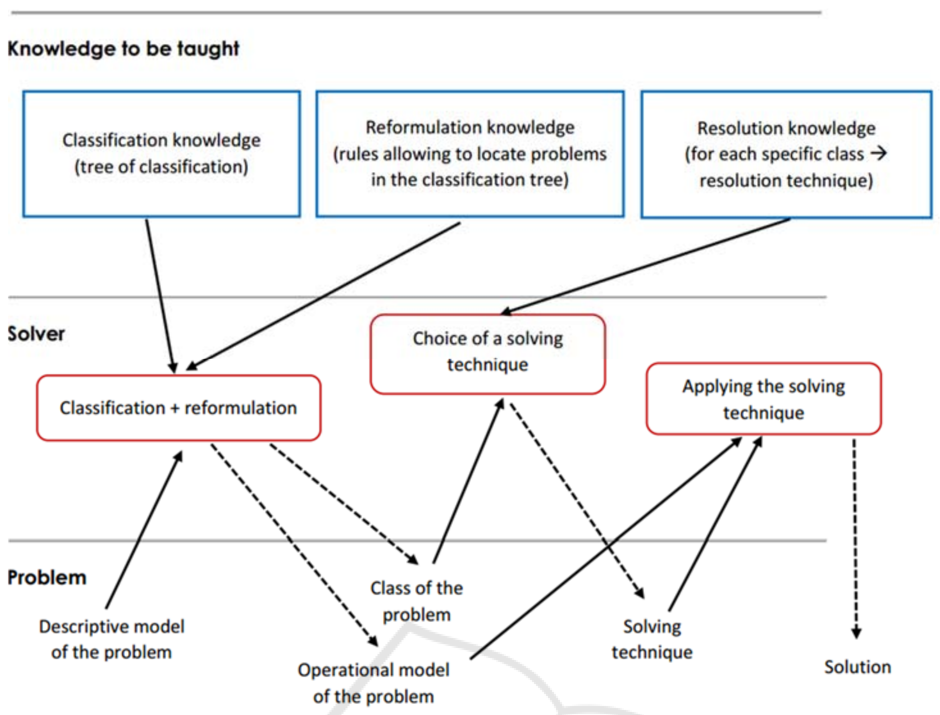

Knowledge to be taught
Classification knowledge (tree of classification)
Reformulation knowledge (rules allowing to locate problems in the classification tree)
Resolution knowledge (for each specific class → resolution technique)
Solver
Choice of a solving technique
Classification + reformulation
Applying the solving technique
Problem
Descriptive model of the problem
Class of the problem
Operational model of the problem
Solving technique
Solution

Figure 1: Functioning of AMBRE solver.

model the solving technique associated with the class, to find the solution of the problem (Figure 1).

Classification knowledge is in the form of a classification tree of problems to be solved. Reformulation knowledge is constituted by a set of rules. These rules enable, from the statement of the problem, to identify the most specific class of the classification tree to which the problem belongs. Resolution knowledge is constituted by the solving techniques associated to each specific class of the classification tree. We considered existing authoring tools defined below, but they do not meet our needs either because they do not match to AMBRE principle or because techniques used do not enable to represent all knowledge needed by an AMBRE ITS. That is why we propose to define AMBRE-KB.

3 OVERVIEW OF AMBRE-KB

The goal of AMBRE-KB (AMBRE-Knowledge Builder) is to assist authors in the in creation of ITSs teaching problem-solving methods in the domain they are interested in. Our intention is specially to allow the description of knowledge needed by the system without any Prolog programming.

AMBRE-KB enables the author to explicit knowledge needed by an ITS and generates a Prolog version of this knowledge in order to constitute the knowledge bases of the ITS.

The acquisition of knowledge in AMBRE-KB is an automated process based on meta-models. In order to do so, we defined a meta-model for each type of knowledge to be acquired. These meta-models define the form of knowledge to be defined and constrain the design process enabling to do so.

3.1 Authoring Process

The authoring process in AMBRE-KB consists of nine steps summarized on Table 1.

Table 1: Authoring process.

1.	Choice of a domain of learning
2.	Choice of the types of exercises to be solved
3.	Definition of the vocabulary for the domain of learning
4.	Description of problems to be solved by the system and the learner
5.	Description of knowledge to be taught (classification knowledge, reformulation knowledge and resolution knowledge)
6.	Design of the interface for students
7.	Description of knowledge to guide the learner (proving him/her help and diagnosing his/her answers)
8.	Generation of knowledge models by AMBRE-KB
9.	Test of generated models by the solver

Phase 1: Choice of a Domain of Learning

The first task to do when designing an AMBRE ITS is to choose the application domain. AMBRE ITSs are based on a classification of problems to be solved and solving techniques. Thus, the author has to choose a domain in which he/she can establish a classification of problems and solving techniques.

Figure 2 shows an example of classification for the domain: French verbs conjugation.

Phase 2: Choice of the Types of Exercises to be Solved

It is important before beginning the knowledge elicitation process, that the author think about the type of problems to propose to the learner. This second step will help him/her to define the vocabulary (phase 3) - which play a central role in definition of knowledge - and the classification tree (phase 4).

Phase 3: Definition of the Vocabulary for the Domain of Learning

The third step consists in the definition of the vocabulary. We need a vocabulary because, in the AMBRE project, problems are given to the system as a model we call *descriptive model*. This model describes a concrete situation which is the one represented in the statement of the problem. Objects that appear in the statement are concrete elements that constitute the contextual aspect of the problem. For example, in geometry, the objects can concern the characteristics of the geometric figure (dimensions of the segments, angles, and so on).

The objects are connected by relations to form a concrete situation. There are properties and relations on those objects. Each object is characterized by an identifier and a set of characteristics. The vocabulary is constituted by all the objects needed to describe problems and the question to be answered.

Phase 4: Description of Problems to be Solved by the System and the Learner

In the fourth step, the author uses the vocabulary to define problems to be solved by the system and the learner. For each problem, the author has to define:

- the *statement* of the problem in natural language, so that the learner can understand what to do.
- the *descriptive model* of the problem, for the system. For that, he/she chooses in the vocabulary, objects concerned, he/she then instantiates objects (giving a name and a type for
- each characteristic of the object). He/she finally specifies the question to be answered.

Phase 5: Description of Knowledge to be Taught

Using AMBRE-KB, the author defines the knowledge to be taught: classification knowledge, reformulation knowledge, and resolution knowledge (Diattara et al. 2016)

Classification Knowledge is in the form of a classification tree where a class C2 is subclass of a class C1 if any problem of C2 is also a problem of C1. The root class is defined as the most general class, and the leaves, the most specific ones. For each class a *discriminating attribute* is defined. This attribute must have different values in each subclass. Non-discriminating attributes - called *problem attributes* – can also be defined if they make sense for problems

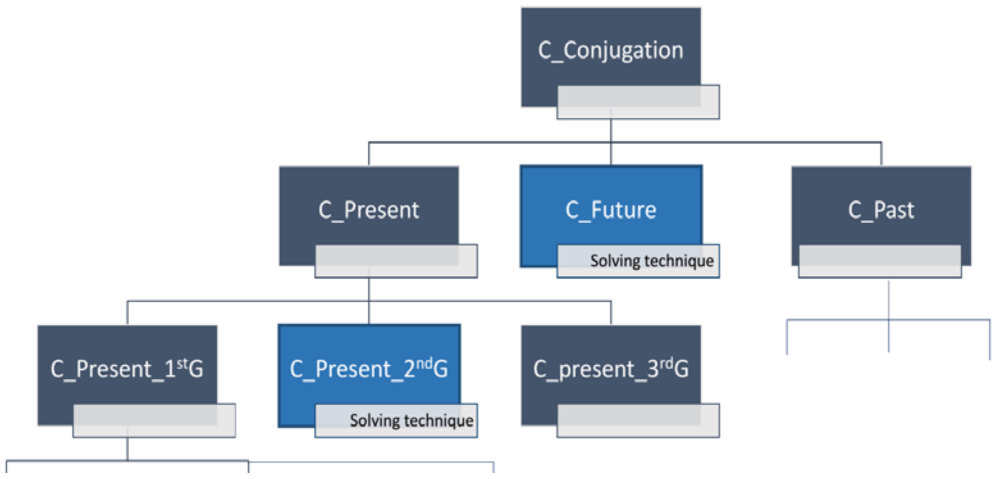


Figure 2: Conjugation domain classification.

of the class. Problem attributes are useful for the resolution and their values depend on the problem to solve. Classes that are specific enough so that we can assign them a resolution technique are called *operational classes*.

Figure 3 shows a part of the classification graph for conjugation, defined with AMBRE-KB.

To define the classification tree, the author can choose to build the tree from the root to the leaves or vice-versa. He/she has the possibility to define all classes, and then organize them as a hierarchy. He/she can also organize the classes into a hierarchy as the definition of classes progresses. Some classes can be defined adapting other classes.

The system checks that all non-operational classes have at least one subclass. Operational classes can have subclasses which are more specific. When two classes have the same discriminating attribute, the system suggests to the author to define the attribute at the level of their lowest common ancestor class.

Reformulation Knowledge. In order to solve a given problem, the solver needs first to determine the class this problem belongs to. For that, the solver needs rules allowing to calculate the value of attributes (discriminating or not), thus enabling to locate the problem in the classification tree. All the rules constitute the reformulation knowledge. A rule is defined by its *name*, a *set of premises* related to the elements of the statement (objects of the problems), and a *set of conclusions* enabling to calculate or to modify the values of the attributes.

For example, Rule 1 enables to conclude about the value of the attribute *group*.

Rule 1:
If for a given problem
- there is a verb V
- the suffix of V is “er”
- V is different from the verb “aller”

Then, the value of the attribute *group* is *1st group*.

To define rules with AMBRE-KB, the author can choose, for example, to define rules as the definition of the classes progresses. In this case, for each class defined in the tree, he/she has to define the attributes (discriminating or not), and then the rules that enable to first define the whole classification tree and then, process to the definition of rules.

The system checks if the expert has associated rules to each attribute defined in the classification tree in order to calculate their value.

For each rule, the system also checks if the conclusion part of the rule provides information about the attribute.

Figure 4 is a representation with AMBRE-KB of rule 1.

On (1) we have the list of attributes, and for each of them the rules enabling to determine its different values.

On (2), we have the premises of the rule.

(3) shows the conclusion of the rule which enable to conclude about the value of the concerned attribute.

Resolution Knowledge. Each operational class in the classification tree has an associated technique.

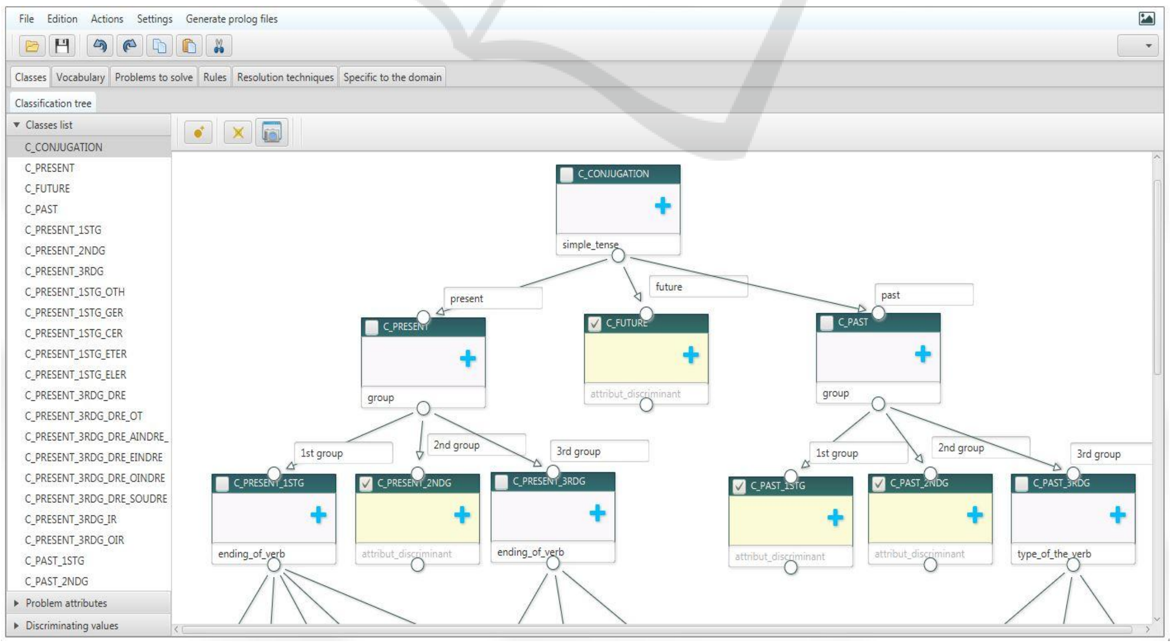
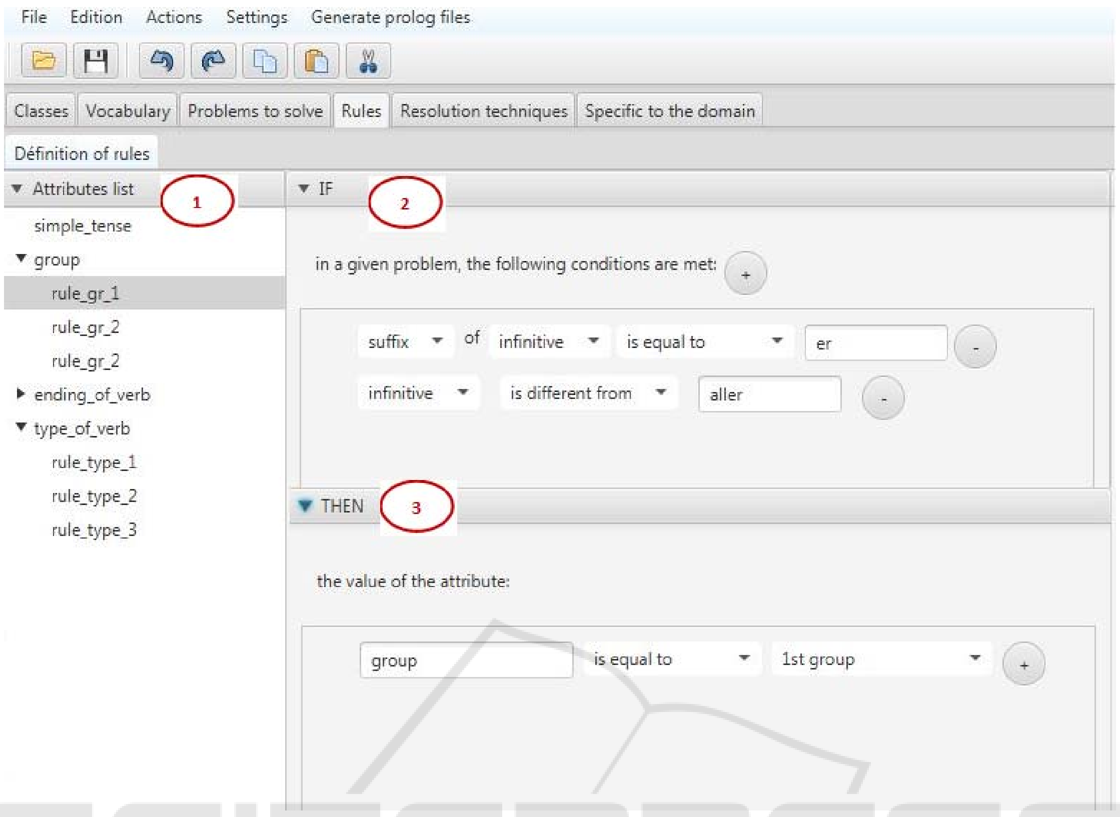


Figure 3: A part of conjugation tree defined with AMBRE-KB.



simple_tense
group
rule_gr_1
rule_gr_2
rule_gr_2
ending_of_verb
type_of_verb
rule_type_1
rule_type_2
rule_type_3

in a given problem, the following conditions are met:

suffix of infinitive is equal to er

infinitive is different from aller

the value of the attribute:

group is equal to 1st group

Figure 4: Definition of rule 1 using AMBRE-KB.

These techniques constitute the resolution knowledge. They are specific to each application domain. For example, in the domain of arithmetic problems, a resolution technique provides a plan for solving an exercise and a formula for calculating its numerical solution. For example, *Technique 1* shows the technique to apply to conjugate verbs of the class *C_present_1*st*G_Reg* to which the problem P1 belongs.

Technique 1

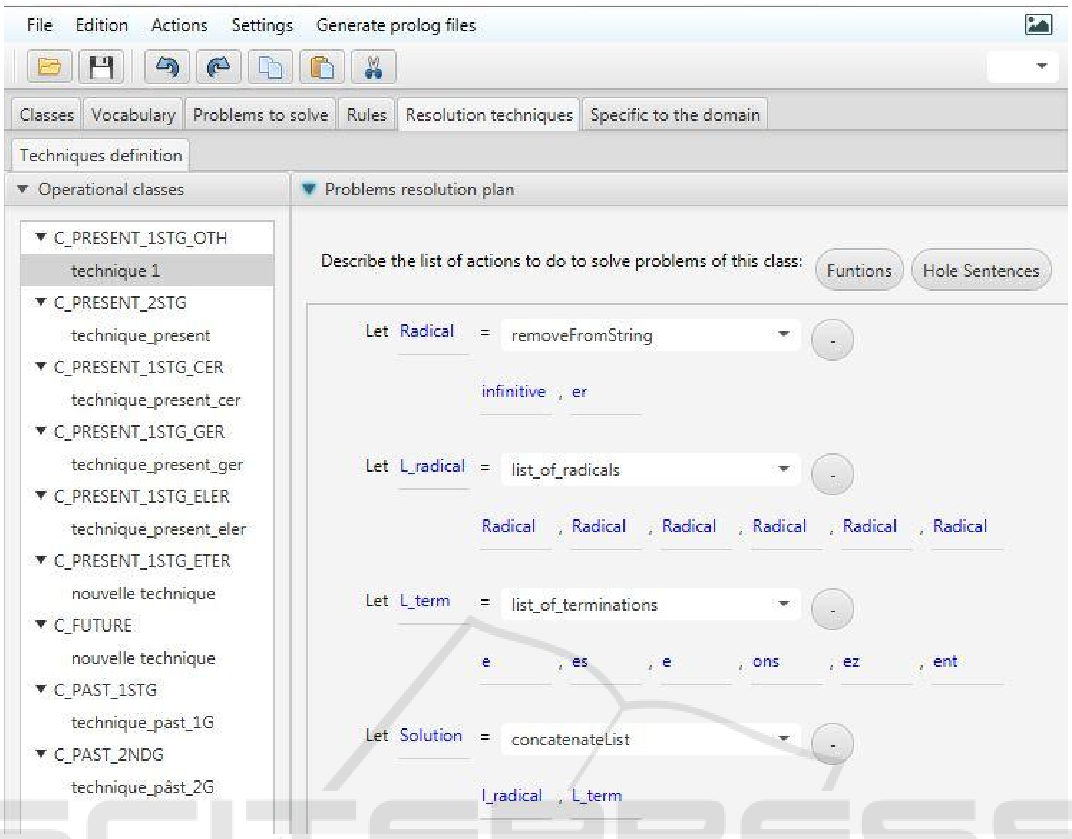
- determine the *radical* of the verb by removing the suffix "*er*" from the verb
- build a list of six elements with the [radical]
- build a list with the following elements [*e*, *es*, *e*, *ons*, *ez*, *ent*]
- concatenate the two lists in order to find the solution.

To explicit resolution knowledge using AMBRE-KB, the author has two possibilities: he/she can directly associate to each operational class of the classification tree the corresponding resolution technique or, he/she can define all classes first, and then define all resolution techniques, and finally connect each resolution technique to corresponding operational class (or classes).

Figure 5 is a representation with AMBRE-KB of technique 1 defined on section 2.3.

Phase 6: Design of the Interface for Students

The sixth step consists in designing the interface of the ITS, and especially the tasks the student must perform to solve problems, based on the AMBRE cycle. But, as AMBRE-KB does not yet support the design of the interface, the development of this interface must be implemented by an IT specialist.



- C_PRESENT_1STG_OTH
  - technique 1
- C_PRESENT_2STG
  - technique_present
- C_PRESENT_1STG_CER
  - technique_present_cer
- C_PRESENT_1STG_GER
  - technique_present_ger
- C_PRESENT_1STG_ELER
  - technique_present_eler
- C_PRESENT_1STG_ETER
  - nouvelle technique
- C_FUTURE
  - nouvelle technique
- C_PAST_1STG
  - technique_past_1G
- C_PAST_2NDG
  - technique_pâst_2G

Describe the list of actions to do to solve problems of this class:

Let Radical = removeFromString
infinitive , er

Let L_radical = list_of_radicals
Radical , Radical , Radical , Radical , Radical , Radical

Let L_term = list_of_terminations
e , es , e , ons , ez , ent

Let Solution = concatenateList
l_radical , L_term

Figure 5: Definition of Technique 1 using AMBRE-KB.

Phase 7: Description of Knowledge to Guide the Learner

Using AMBRE-KB, and based on knowledge to be taught defined in phase 5, and the steps of resolution defined in phase 6, the author defines the knowledge to guide the learner.

Phase 8: Generation of Knowledge Models by AMBRE-KB

For each type of knowledge, AMBRE-KB generates a Prolog version of this knowledge as a model. All the knowledge models generated constitute the knowledge bases of the ITS that the solver can use to solve problems.

Phase 9: Test of Generated Knowledge Models by the Solver

In the last phase (ninth step), the author tests if knowledge generated by AMBRE-KB enable the solver to solve the different problems. For each problem, he/she tests the resolution with the solver; if the solver is able to solve it, it gives the solution of the problem. Otherwise, it sends an error message which should enable the author to understand and to correct the error.

4 PRELIMINARY EVALUATION

We conducted a first experiment to verify if AMBRE-KB is apt to encode correctly knowledge to be taught (classification knowledge, reformulation knowledge and resolution knowledge) so that the solver can use it to solve problems.

4.1 Evaluation Protocol

We choose to perform the experiment with two existing AMBRE ITSs.

Two domains were tested: arithmetic problems for seven-year-old to nine-year-old pupils and French verb conjugation.

The experiment procedure consisted in three stages. We began by describing all knowledges on paper, independently from Prolog. For each of the two domains, we first defined the different problems to be solved. We then proposed a vocabulary to describe these problems. Next, we defined the classification tree. For each problem, we defined rules allowing to locate this problem in the classification tree. Next, we defined the resolution technique for each operational class of the classification tree.

In the second step, we used AMBRE-KB to elicit these knowledges. Once we finished with knowledge elicitation, the system generated the knowledge models in a Prolog version. Finally, we tested each problem with the solver. If the solver was able to solve the problem, it showed the class the problem belongs to, the lists of rules executed by the system and the solution of the problem. When the solver was not able to solve a problem, it sent an error message enabling to know what did not work and how to correct it.

4.2 Results and Discussion

For French verbs conjugation, we defined with AMBRE-KB:

- 21 classes, among which 13 are operational,
- 15 rules,
- 13 solving techniques,
- 80 problems to solve.

At the end of the experiment, AMBRE-KB generated the Prolog version of this knowledge. The solver was able to solve 100% of the problems. We can deduce that AMBRE-KB has correctly generated knowledge models so that they can be used by the solver to solve problems.

For arithmetic problems, we defined with AMBRE-KB:

- 21 classes, among which 14 are operational,
- 18 rules,
- 14 solving techniques,
- 98 problems to solve.

For 93% of the problems, the solver was able to solve problems correctly using generated knowledge models. For 7% of problems, the solver first sent an error message to explain what did not work. For 4% of the problems, we made an error on the value of the discriminating attribute. The solver was not able to find a reformulation rule to use, thus it was not able to locate the problem in the classification tree. For the 3% remaining problems, the solver sent an error message because it failed in applying the solving techniques, because of lack of knowledge.

Once the missed or erroneous knowledge were fixed, the solver was able to solve the 7% remaining problems.

The objective of this first experiment was to ensure that AMBRE-KB was able to correctly generate knowledge models that the solver can use to solve problems. The next step, which is the most important for us, is to test the utility and usability of AMBRE-KB by authors who are not IT experts. For this experiment, we plan to process in three steps:

- The first step will consist in presenting to the author the principle of the AMBRE project, and the meta-models of knowledge to be acquired (classification knowledge, reformulation knowledge and resolution knowledge). We will also present the objective of AMBRE-KB. At the end of this first step, we will invite the author to think about the domain in which he/she wants to design an AMBRE ITS.
- In the second step, we will ask him/her to choose a domain. Next, based on knowledge models, we will ask him/her to elicit all knowledge needed on paper. He/she will begin by defining the types of problems to be solved. Next, he/she will define a vocabulary for the domain. He/she then will define the problems using the vocabulary. Finally, he/she will define the classification knowledge, the reformulation knowledge and the resolution knowledge. At the end of this step, the author will have described all knowledge needed to solve problems.
- Finally, the author will use AMBRE-KB to elicit this knowledge. Once the knowledge elicitation is complete, the system will generate the knowledge models and the author will be able to test each problem with the solver. For each of them, the solver will send the solution if all knowledge needed is correctly elicited, otherwise, it will send an error message enabling the author to understand what was wrong and to correct or complete missed knowledge.

During the experiment, the author will be filmed and the verbal exchanges will also be recorded. During the third step, we will observe his/her interaction with the software and take notes on the time passed on every type of knowledge, the difficulties met during the knowledge elicitation, gestures and non-verbal behavior of the author, his/her remarks and questions. When the questions asked by the author concern the functioning of the tool, we will first orient him/her on the help proposed by the system. However, if the help proposed by the system is not sufficient, we will give him/her the necessary information, so that he/she can continue the process of knowledge elicitation.

If we notice that the author does not move forward in knowledge elicitation, we will consider that he/she feels difficulties about the functioning of the tool. In this case, we will ask him/her a question to know what he/she wants to do. If the answer of this question corresponds to a feature taken into account by the tool, we will guide him/her so that he/she can continue the knowledge elicitation process. Otherwise, when what he/she wants to do is not taken into account by the tool, we will consider that the concerned task cannot be finished, and he/she will continue the elicitation with others types of knowledges.

The experiment will end when the author finishes testing all problems with the solver. He/she will complete a questionnaire composed of many sections about the functioning of AMBRE-KB, the proposed interfaces, the assistance proposed during knowledge elicitation, the feedback of the solver, and their profile (how he/she frequently uses a computer for example).

5 CONCLUSIONS

We provided in this article an overview of AMBRE-KB, an authoring tool that assists authors in building ITSs teaching problem solving methods. This tool enables the user to elicit all knowledge needed by the system. AMBRE-KB follows an automated process based on knowledge meta-models. The paper presents the knowledge acquisition process and how can AMBRE-KB can be used to build an ITS in a given application domain.

We conducted a first experiment to verify if AMBRE-KB can correctly generate knowledge models in a Prolog version, so that the solver can use them to solve problems. The results of this experiment were satisfactory, but thorough evaluation is planned with non-IT experts in order to test the usability and utility of AMBRE-KB.

This work focused on the acquisition of knowledge about the method to be taught. The next stage will concern the acquisition of knowledge intended to guide the learner during his/her learning. This knowledge will enable to propose to the learner help and explanations of various natures according to the step of its resolution or committed errors.

In addition, a relevant track in the continuation of our work is the integration of features of generalization of knowledge from cases. This feature will enable users to define an example, rather than an abstract knowledge, the system proposing them a generalization of the knowledge that they can validate. The SimStudent (Matsuda et al. 2010) approach which is based on learning abstract knowledge from examples would be appropriate in the context of this work.

ACKNOWLEDGEMENTS

We thank the Auvergne-Rhône Alpes region for granting the scholarship that supports this thesis work.

REFERENCES

Ainsworth, S. et al., 2003. REDEEM: Simple Intelligent Tutoring Systems From Usable Tools. In T. Murray, S. B. Blessing, & S. Ainsworth, eds. *Authoring Tools for Advanced Technology Learning Environments: Toward Cost-Effective Adaptive, Interactive and Intelligent Educational Software*. Dordrecht: Kluwer Academic, pp. 205–232.

Ainsworth, S. E. & Grimshaw, S., 2003. Evaluating the REDEEM Authoring Tool: Can Teachers Create Effective Learning Environments? *International Journal of Artificial Intelligence in Education*, 14(3/4), pp.279–312.

Aleven, V. et al., 2009. A new paradigm for intelligent tutoring systems: Example-tracing tutors. *International Journal of Artificial Intelligence in Education*, 19(2), pp.105–154.

Aleven, V. et al., 2016. Example-Tracing Tutors: Intelligent Tutor Development for Non-programmers. *International Journal of Artificial Intelligence in Education*, 26(1), pp.224–269.

Blessing, S. B. et al., 2007. Authoring Model-Tracing Cognitive Tutors. *International Artificial Intelligence in Education Society (IJAIED)*, 19(2), pp.189–210.

Diattara, A. et al., 2016. Towards an Authoring Tool to Acquire Knowledge for ITSs Teaching Problem Solving Methods. In *EC-TEL*, 2016, Lyon, France, pp. 575-578.

Guin-Duclosson, N., Jean-Daubias, S. & Nogry, S., 2002. The Ambre ILE: How to Use Case-Based Reasoning to Teach Methods. In *Conference: Intelligent Tutoring Systems, 6th International Conference, ITS 2002, Biarritz, France and San Sebastian, Spain*. Biarritz, pp. 782–791. Available at: http://link.springer.com/ 10.1007/3-540-47987-2_78.

Koedinger, K. R. & Corbett, A. T., 2006. Cognitive tutors: technology bringing learning science to the classroom. In R. K. Sawyer, ed. *The Cambridge Handbook of the Learning Sciences*. Cambridge Handbooks in Psychology. Cambridge University Press, pp. 61–77.

Matsuda, N., Cohen, W. W. & Koedinger, K. R., 2010. Learning by teaching SimStudent: technical accomplishments and an initial use with students. In *Intelligent Tutoring Systems: 10th International Conference, ITS 2010, Pittsburgh, PA, USA, June 14-18, 2010, Proceedings, Part I.* Pittsburgh, USA, pp. 317–326.

Mitrovic, A. et al., ASPIRE: An Authoring System and Environment for Constraint-Based Tutors Deployment. Available at: http://hdl.handle.net/10092/3478.

Mitrovic, A. et al., 2006. Authoring Constraint-Based Tutors in ASPIRE. In Springer Berlin Heidelberg, pp. 41–50.

Murray, T., 2003. An Overview of Intelligent Tutoring System Authoring Tools: Updated analysis of the state of the art. In *Authoring tools for advanced technology learning*. Available at: http://link.springer.com/chapter/10.1007/978-94-017-0819-7_17 [Accessed December 12, 2014].

Murray, T., 2003a. An Overview of Intelligent Tutoring System Authoring Tools: Updated analysis of the state of the art. In T. Murray, S. B. Blessing, & S. Ainsworth, eds. *Authoring Tools for Advanced Technology Learning Environments: Toward Cost-Effective Adaptive, Interactive and Intelligent Educational Software*. Dordrecht: Springer Netherlands, pp. 491–544. Available at: http://dx.doi.org/10.1007/978-94-017-0819-7_17 [Accessed December 12, 2014].

Murray, T., 2003b. Authoring intelligent tutoring systems: An analysis of the state of the art. In T. Murray, B. Stephen, & A. Shaaron, eds. *Authoring Tools for Advanced Technology Learning Environments*. Kluwer Academic Publishers, pp. 98–129.

Nkambou, R., Frasson, C. & Gauthier, G., 2003. CREAM-Tools: An authoring Environment for Knowledge Engineering in Intelligent Tutoring Systems. In T. Murray, S. B. Blessing, & S. Ainsworth, eds. *Authoring Tools for Advanced Technology Learning Environments*. Dordrecht: Springer Netherlands, pp. 269–308.

Nogry, S., Guin, N. & Jean-Daubias, S., 2008. AMBRE-add: An ITS to Teach Solving Arithmetic Word Problems. *Technology, Instruction, Cognition and Learning*, 6(1), pp.53–61. Available at: http://liris.cnrs.fr/publis/?id=3548.

Schoenfeld, A., H., 1988. Problem solving in context. In I. Charles, R & A. Silver, E., eds. *The teaching and assessing of mathematical problem solving*. Reston, VA: The National Council of Teachers of Mathematics., pp. 82–92.

Schoenfeld, A. H., 1985. *Mathematical Problem Solving*, Academic Press.

Shute, V., Torreano, L. & Willis, R., 1998. DNA — Uncorking the Bottleneck in Knowledge Elicitation and Organization. In B. Goettl et al., eds. *Intelligent Tutoring Systems SE- 20*. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 146–155.

Tecuci, G. & Keeling, H., 1999. Developing an intelligent educational agent with disciple. *International Journal of Artificial Intelligence in Education*, 10(3–4), pp.221–237.

Tennyson, R. & Breuer, K., 1994. ISD EXPERT: An Automated Approach to Instructional Design. In R. Tennyson, ed. *Automating Instructional Design, Development, and Delivery SE - 9*. NATO ASI Series. Springer Berlin Heidelberg, pp. 139–161. Available at: http://dx.doi.org/10.1007/978-3-642-78389-0_9.

Woolf, B. & Cunningham, P.A., 1987. Multiple Knowledge Sources in Intelligent Teaching Systems. *IEEE Expert*, 2(2), pp.41–54.