Computer assisted assessment in elementary algebra: experiences and points of view from the APLUSIX project

by

Denis Bouhineau* (email: denis.bouhineau@imag.fr), Alain Bronner**(email: alain.bronner@montpellier.iufm.fr), Hamid Chaachoua* (email: hamid.chaachoua@imag.fr), Sophie Mezerette* (email: sophie.mezerette@imag.fr) and Jean-François Nicaud*(email: françois.nicaud@imag.fr)

* Lab. Leibniz-IMAG, Grenoble, France

http://aplusix.imag.fr/dir-en/

** Lab. LIRDEF - IUFM de Montpellier, France

Introduction

Since 2000, we have been working on a project for a global computer system dedicated to the teaching and learning of formal elementary algebra at secondary level schools. During a first phase, from 2000 to 2003, most of our work has been employed with the conception and realisation of a microworld for doing actively elementary algebra. This period has lead to the aplusix environment [Nicaud 2004], a tool dedicated to the first activities concerning algebra (resolution of equations, factorisation and expansion of polynomials). It was principally a piece of software devoted to students, based on the concept of a microworld, which means that a set of objects and relations were defined internally to represent algebraic expressions and represented externally in the most natural way, allowing activities in elementary algebra with direct-manipulation and intuitive advanced editing. The main principles used to develop this environment were, first, to let students freely and easily build and transform algebraic expressions, as they can do on paper, and second, to give permanent but not intrusive epistemic feedback on the syntactic and semantic correctness of the reasoning followed by the students. Experiments conducted in classes from K7 to K11 have shown that learning occurs with the use of theaplusix microworld [Nicaud 2005]. A screen shot of the environment is shown Fig. 1.

During this first phase we concentrated on the microworld paradigm and our main concern was for students as our principal users. The second phase introduced two shifts in our preoccupations. Firstly, besides the fact that we tried to continue to improve the microworld aspect of our environment, we have worked towards nesting it into some kind of "exerciser" where activities could be automatically generated, solutions could be automatically found and work done by students automatically analysed and scored. Secondly, we have worked specifically with teachers in mind, trying to provide them with tools for managing, improving and facilitating their work, for example, developing tools for the administration of classes and students, tools for editing of specific exercises, list of exercises, or production of richer exercises (for word problems, or problems with many linked sections) and tools for the statistical analysis of student's results. Many tools linked to, or separate from, the microworld have been developed during this phase and many are linked to problems belonging to the Computer Assisted Assessment domain.

In the following sections we will focus on our work related to the Computer Assisted Assessment domain. We will deal with the following subjects:

- automatic generation of exercises and list of exercises;
- definition of possible different student's activities related to CAA; and,
- automatic production and verification of solutions;
- scoring and statistical analysis.

Notice that these subjects are almost independent from each other. We will also try to gain some perspective from the last 5 years, drawing on our experience in developing our system, to find some guiding principles. We reflect on the evolution we have observed from the initial definition of the previously listed elements to the current state of these elements and we will try to identify the main final characteristics of these elements.

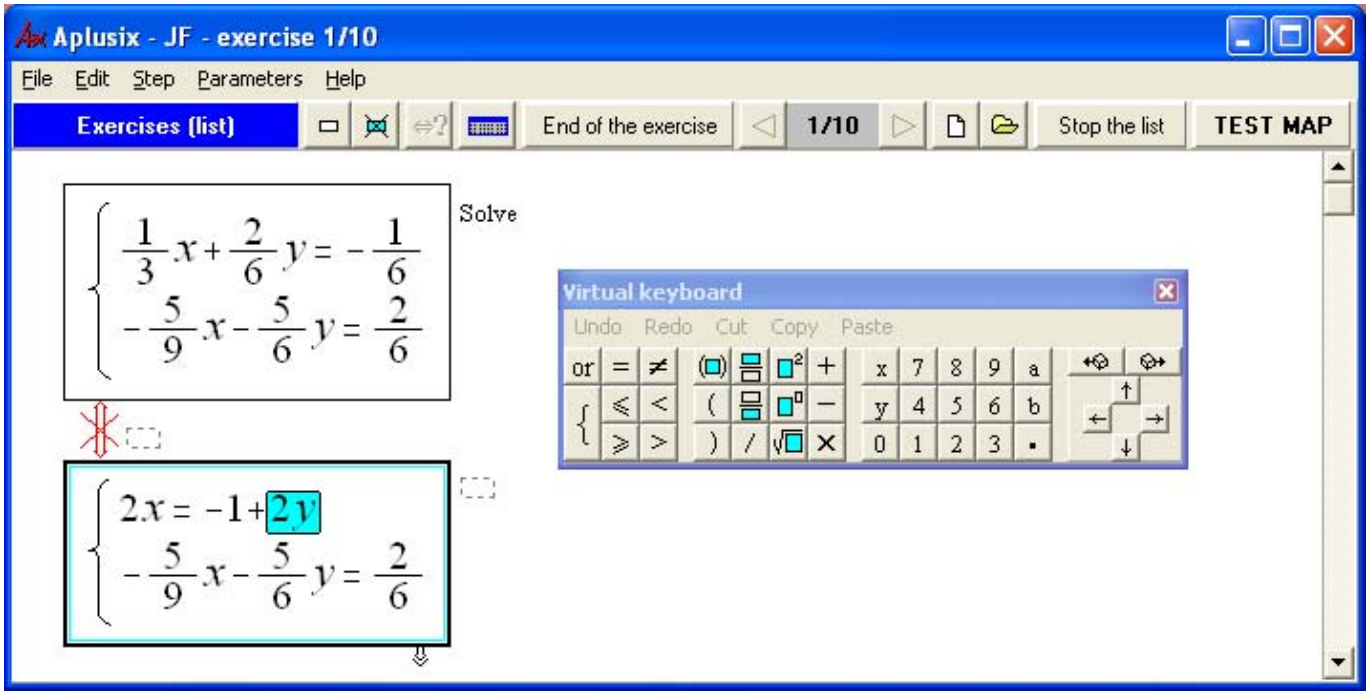

Fig 1 -The aplusix microworld

Delivering exercises

When the teacher defines and organises exercises

The first step in transforming a microworld into an exerciser is to provide exercises for students. The simplest way involves: first asking teachers to record exercises in some structure and then to arrange these exercises into lists. Secondly, functionality has to be added to the microworld which permits students to work on the list. These steps were used in our first version and used for our first experiments.

We went on to develop a second version with complex scenarios combining the list of exercises with general system parameters for the microworld (important parameters including didactical parameters) to be merged with the student and class general parameters for the microworld. Other parameters govern the execution of the scenario e.g. time management, order of the exercise, message set for exercises and temporal condition for executing the next series of exercises. This version allowed us to perform more complex experiments, but was abandoned

later, because of its complexity. Apart from an academic experiment, the complexity of this tool was too costly to be used and did not fit well with the actual needs of teachers.

We came back to working on a simpler version based on the list of exercises, which was just a bit more elaborate than our first version. But we have drawn three lessons from that failure:

1. Not to conceive tools and situations that are too complex
2. Two principal situations need to be considered: the training and test situations
3. Some system parameters can be fixed without loss of generality, some can be ignored by the system, masked to teachers and user access given to the students.

After this episode and the lessons learned, we continued to develop the exercise editor for teachers such that a richer kind of exercise could be proposed, allowing problems with several sections and word problems.

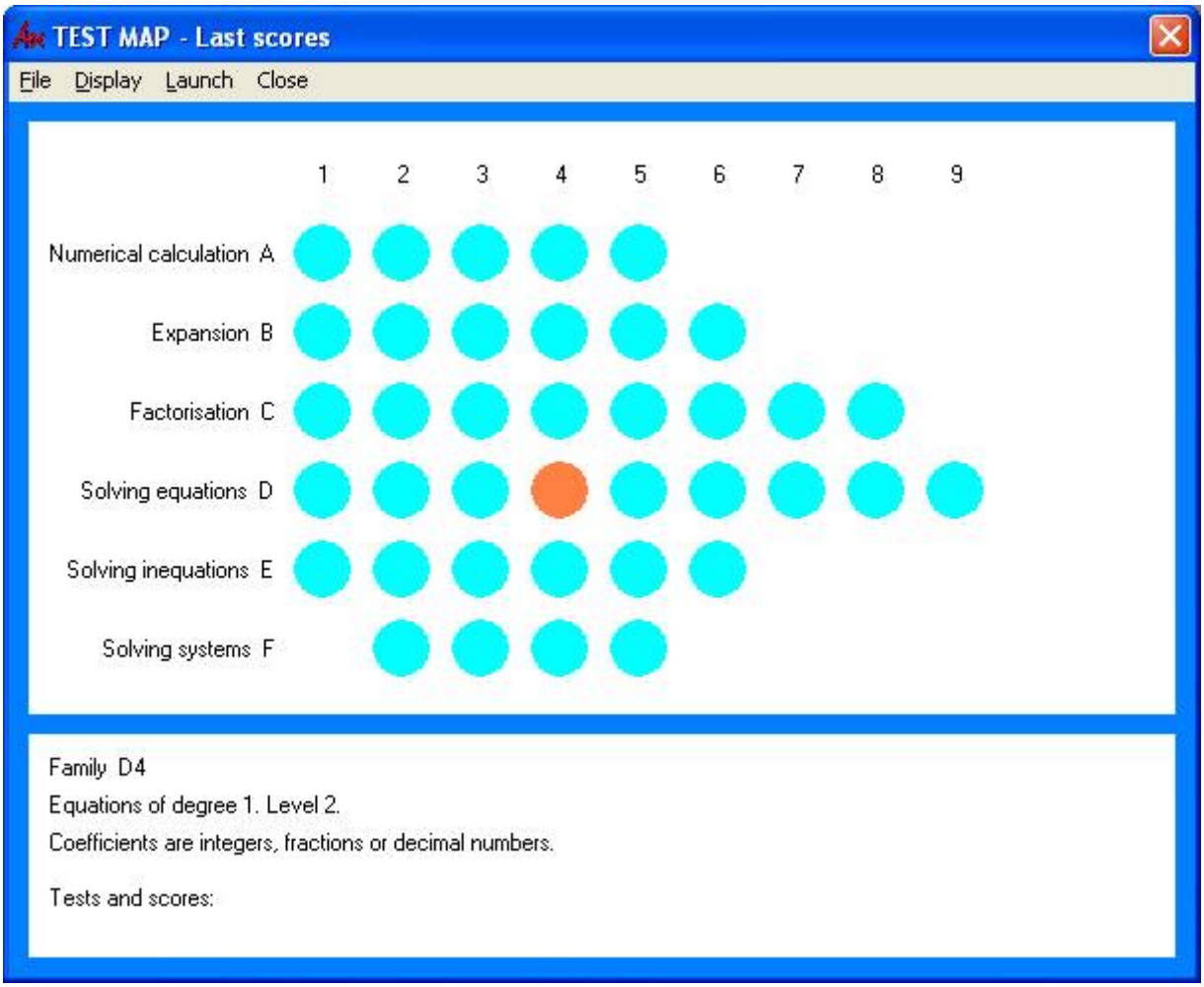

Fig 2 -The map of exercises or test map presents the families of exercises as dots organised in a plane. Each horizontal line corresponds to an exercise type, with difficulty increasing from left to right.

When a computer defines and organises exercises

At the same time we began to work on the automatic generation of exercises. The goal was twofold, first to have a “ready to use” map of exercises allowing students to work without the need for teachers and second to facilitate the preparation of exercise lists for the teachers. The result is a set of about 40 families of exercises organized by type of exercise and level of

difficulty, for example, first degree equations with fractions and square roots. Each family contains about 10 patterns of exercises having variables for the numbers and constraints. The exercises are generated from patterns by random choices of the numbers and verification of the constraints.

The families of exercises and the patterns of exercises used in the map of exercises have been defined by two didacticians in mathematics, with whom we have worked to define an ad hoc simple language to design families of exercises and patterns of exercises. Patterns belong to a hierarchic organisation with inheritance. Each pattern is an object with a dozen attributes; most of these attributes are defined by inheritance, e.g. mean time to solve the exercise, exercise type, hints to describe the pattern or general family of pattern, pseudo-frequency to set the number of apparitions of the pattern with respect to other pseudo-frequency of the other patterns of the same family. The main other attributes are the pattern itself and the domain for the value of parameters and constraints to be tested out to validate an exercise.

Parameters are to be chosen in their domain, but it is well known that "*parameters produced must be realistic, in that the questions should be neither overly tedious nor trivial*" [Griffin 2004]. Most of the time, or even always as far as we know, it is the responsibility of the producer to avoid a wrong value for the parameters, and it means that developing new questions can involve significant programming effort and tricky formulation. We have chosen another solution to avoid such incorrect instantiated exercises; we have introduced constraints to be respected when patterns are instantiated (e.g. values for two expressions must be different, value for an expression must not be a square, value for one expression must not be a factor for the value of another expression). . As a consequence the instantiation mechanism is a non trivial randomised choice where constraints are to be verified to validate an exercise, and which deal with problems when not too many instantiations are possible or other problems resulting from the rejection of instantiated exercises. The introduction of constrains to be checked is an innovative aspect of our simple language. Compared to patterns defined for Wims [Perrin-Riou 2004], MacQTeX [Griffin 2004] and others, our approach permits a rather efficient, powerful and expressive language.

Obtaining all those patterns (more than 400) has been a satisfying result, considering that the providers were two didacticians, conversant with mathematics, but with almost no skills in computer science. Our goal was to find a way to have didacticians working with computers within a not too complicated framework, but nevertheless powerful enough so that they could work easily. Complex frameworks like those introduced by [IMS-QTI 2004] and MathML or OpenMath were avoided, even syntax relying on TeX language or similar languages was rejected. We did not believe that bringing the complexity of computer science into the hands of a non-specialist would be a success.

Once the exercises have been obtained, algebraic pretty-print processes are executed to suppress neutral elements from algebraic expressions e.g. (-1)3x=1*3 is transformed into –3x=3 and finally exercises are locally randomly rearranged in the list, and associated such that the global duration (sum of the exercises durations) is close to an arbitrary duration of 30 minutes.

```
{[name FactorDistSTD23Z]
[KindOf FactorDistSTD]
[pattern <<ax^2+bx>>]
[domain ((c integer+ small)(d integer+ small)(e integer* small))]
[with ((<> c 1) (:= a (* d c)) (:= b (* e c)))]}
```

Fig 3 - Example of a pattern: x remains an abstract variable, a and b are instantiated with products d.c and e.c where c and d are small positive integers, e is a non-null, non-unitary integer. For example, that pattern can produce the following expression $18x^2+30x$, $12x-9x$ and $14x^2+7x$.

Possible activities related to CAA

At the beginning, recall that our system was a microworld whose goal was to help students solve exercises and problems in elementary algebra. Students could perform their own calculations by typing in expressions and making the steps they want, as easily as on paper. Students were supposed to work by equivalent steps with aplusix providing the fundamental feedback that indicates whether the calculations are correct or not (equivalent to the initial expression), and at the end, for the final step, whether the exercises are correctly solved or not with respect to one of the following predefined types of exercise: Calculate (for numerical calculations), Expand, Factor or Solve (equation, inequation or system of equations). The shift from microworld to exerciser has been possible because of the existence of the reduced set of problem types which concern almost all problems of elementary algebra at secondary school. Apart from these simple exercises, there are problems composed of sections providing long text full of information and asking questions. Answers are algebraic expressions that are compared to the one given by the designer of the problem. For example, a problem can have a text in natural language with a first question asking the student to input an equation and a second question asking them to solve this equation.

In the latest versions of our system, we organise student's work within 5 activities: microworld activity, exercises activity, test activity, self-correction activity and observation activity.

Microworld activity

In the microworld activity students have to input the first expression of their sheet and can change that first expression, whereas in the exercise activity and the test activity the first expression is defined by the exercise in the file or produced by the map of exercises and could not be changed. The first expression on the sheet is important; it defines the exercise and what will be the answer.

Training activity

In both the training and the microworld activities the system provides epistemic feedback in order to help students learn the domain: verification of the equivalence and the correct end of the exercise. This feedback is also a way for the students to self evaluate their skill in algebra alongside solving the problems. Additionally while solving an exercise, students can ask for their score or even ask to see the solution at any moment. These possibilities are disabled for the test activity.

Test activity

The Test activity provides a 30-minute test on exercises chosen from the map of exercises or from a file (with time limitation possibly defined in the file) without any feedback on: the correctness of the student's steps, the end of an exercise solution and score.

At the end of a test, aplusix gives a score and provides students with an opportunity to correct their solution using a "Self-correction" activity. This is a new kind of activity which reinforces student's self evaluation and allows students to overcome their mistakes.

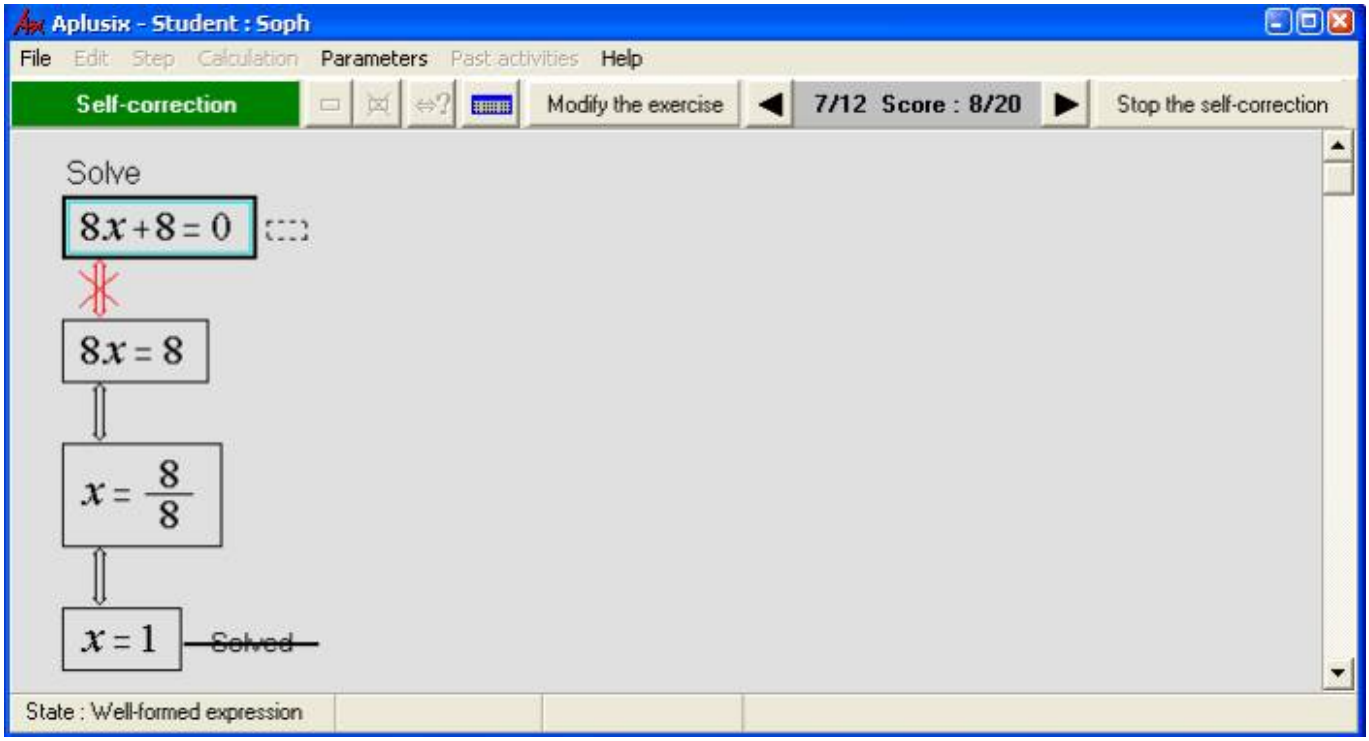

Fig 4 - Screen copy of the self-correction activity. The incorrect calculations are marked in red. The word "Solved" is crossed out because there are incorrect calculations.

Self-correction activity

In the self-correction mode (which is available at the end of a test and also by means of a menu of past activities) aplusix presents the student's work in its final form with indications of the correct/incorrect steps, of the correct/incorrect end, and of the score. Students look for their errors and then they can modify each exercise to correct their errors with the help of the training mode feedback (i.e. feedback that was not available during the test mode).

Observation activity

The observation mode is a more elaborate version of the visualisation of past activities, where students can launch a replay system to see their work action by action. The observation mode was initially intended for teachers and especially for didacticians, so that they can see exactly what have been done by students, but usage reported shows some teachers working with the replay system to help students analyse their own work.

Activities versus system parameters

The organisation of work with aplusix into activities has helped to reduce use of the system parameters. A set of system parameters allows the system to be customised for each class and situation. Parameters continue to exist and can define e.g. the mode and scope of the verification of the equivalence, and how the system must manage an incorrect or ill-formed step; access to the solutions and to the CAS-like commands, the order of exercises obtained from a list (randomised or not), the introduction of a strong invitation encouraging students to comment their steps etc.

Activities do not set all the system parameters, but the most important ones, and reduce the number of those which still need to be set. Because of the complexity of using system parameters (a set of parameters can be assigned to each class and each session), there was a real need to find a way of creating a customised version of the system without too much effort. Activities have been our solution, a style followed in developing the different versions for the production of exercises and focusing on trying to make the system easier to use.

About solutions

After the exercises have been produced, delivered to students, and attempted by students, solutions have to be checked, and if no solution has been found by the student, maybe the system should propose one?

Two main situations are to be considered, the first one is for ‘simple’ exercises produced by teachers or by the map of exercises i.e. an exercise with an initial expression and a well defined type of problem based on Calculate, Factor out, Expand, and Solve. These exercises rely only on algebra, and solutions can be computed from the initial expression straightforwardly. As a consequence, for the exercises given by teachers, no solution is given by teachers, but in the “Exercises” and “Self-correction” activities, the system can give the solution to students on their request, when the system parameter for giving a solution is enabled.

The second situation concerns complex problems given by teachers (for example, word problems and problems beginning with finding an algebraic expression representing the problem). For that kind of situation, teachers must provide the desired solution. Additionally, teachers can specify how solutions given by students will be compared to the desired solution. Three comparisons are possible.

1. Comparison succeeds only for identical expressions;
2. Comparison succeeds for expressions which are semantically equivalent and syntactically close to each other i.e. expressions must be equals modulo elimination of neutral elements, commutative operators and associative operators;
3. Teachers can specify if students can have different names for the variables.

But what for the evaluation of solutions for the first situation, the ‘simple’ exercise case? The use of a Computer Algebra System (CAS) is a classic solution [Sangwin 2005] but introduces drawbacks, especially for elementary algebra. In particular, the syntactic and semantic parts of the analysis of the solutions are not always distinctly defined; except for Mathematica, no CAS allows quantifier elimination which is the most rigorous method for assessing the algebraic semantic equivalence and last of this incomplete list, the use of a CAS often introduces disturbing problems with the syntax of inputs asked from students and outputs shown to students. Hence we chose, from the very beginning of our project, to avoid the use of a CAS for as long as possible. We hoped that elementary algebra was a domain that was not too complex, and it has shown to be in fact not too complex but not that easy. Below is the complete analysis of reasoning, separated into two important parts: the syntactic part and the semantic part.

When is this solved?

First, the reasoning path leading to the solution must be semantically valid. The reasoning is valid when each step in the reasoning is equivalent to the initial one. aplusix verifies that two connected steps contain equal numerical expressions, equal algebraic expressions, equivalent equations, equivalent inequalities, or equivalent systems of equations. In training mode, the student can see the results of the verification with the drawn link between steps: a double black

line means “The expressions are equal” and with arrows added “The equations, inequalities, systems of equations are equivalent”, a double blue crossed line means “One expression is achieved or undefined”, a double red crossed line “The expressions are not equal” and with arrows added “The equations, inequalities, systems of equations are not equivalent”. In test mode a single black line is drawn between steps, it means that “No verification is done”.

Notice that all the algebraic expressions are considered over the set of real numbers and that our system verifies the calculations in the following ways: for the exercise types Calculate, Expand, Factor, canonical forms of the expressions are calculated, for the exercise type Solve, canonical forms of the sets of solutions of the equations, inequations, systems of equations are calculated. Equivalence is then decided when two canonical forms are identical. The calculations of the canonical forms are approached by calculation. The equality of a number is judged with 10 digits.

Second, the final expression must be syntactically checked.

An exercise of the type “Calculate” is solved when the expression is a number written in a canonical form (integer, decimal, rational or irrational).

An exercise of the type “Expand”, with an expression that does not contain square roots, is solved when the expression has no parentheses (neither explicit nor implicit like a sum in the numerator of a fraction) and is simplified.

An exercise of the type “Expand” with an expression that contains square roots is solved when the expression is a canonical polynomial form or when the expression has no parentheses (neither explicit nor implicit like a sum in the numerator of a fraction) and is simplified.

An exercise of the type “Factor”, with an expression representing a polynomial of degree 1, is solved if the constants have been factored.

An exercise of the type “Factor”, with an expression representing a polynomial of one variable of degree greater than 1, is solved if the expression is a product of simplified prime polynomials (prime polynomials are first-degree polynomials and second degree polynomials without roots. Note that constant factorisation in an expression representing a polynomial of degree greater than 1 are not necessary.

An exercise of the type “Solve”, except in the special cases described below, is solved: for an equation, when the expression is “x=a”, where a is a simplified writing of a number, or where “x=a1 or x=a2”, a1, a2 are simplified writings of distinct numbers; for an inequation, when the expression is “x<a” or “x>a” or “x·a” or “x·a”, where “a” is a simplified writing of a number; for a system of equations, when the expression is “x=a1 and y=a2”, where a1and a2 are simplified writings of numbers.

The special cases for “Solve” are first when there is no solution. In this case, no particular form is expected and students have to click on the “End of the exercise” button and choose “No solution”. Second, when any number, for each variable, is a solution, no particular form is expected and students have to click on the “End of the exercise” button and choose “Any number is a solution”. Third, when a system of equations with N unknowns leads to less than N equations, but more than zero, it must have a form in which some variables are expressed as functions of others, for example, for the system x+y=3 and 2x+2y=6, one can give the expression x=3-y.

Scoring and statistics

Both students and teachers need to an analysis of the work done. For students, the first basic solution is to provide a score about the work. For teachers, computations about scores obtained by students can be done to give a statistical or global view of classes. One can imagine some more complex analysis, and we are in fact working on some computerised didactical analysis of student's work, but at present it is only for us to develop a greater perspective.

Just before the description of our scoring process, we have to highlight that according to French legislation, a score automatically computed (without human intervention) from exercises done by students on a computer cannot be used to evaluate students.

Scoring

Scores are calculated considering the progress of the reasoning made with correct calculations. When nearly in solved form, with correct calculations, the score is close to the maximum. When a solved form has been given but "Solved" has not been indicated, the score will not be the maximum.

If there are incorrect calculations, aplusix considers the situation before the first incorrect calculation. If the score is not high, aplusix looks at the calculations after the incorrect one and may increase the score.

Statistics

The Statistics window displays a table and a graphic (histogram or time curve) about a selected period, selected activities, selected population with information (total, average, standard deviation) about: number of attempted exercises, number of well-solved exercises, number of calculation errors, scores, time spent. The table can be sorted by clicking on the head of the column to be sorted. This action has repercussions on the histogram. When students are working, the window is updated each 30 seconds so that teachers can view the progress of student's work.

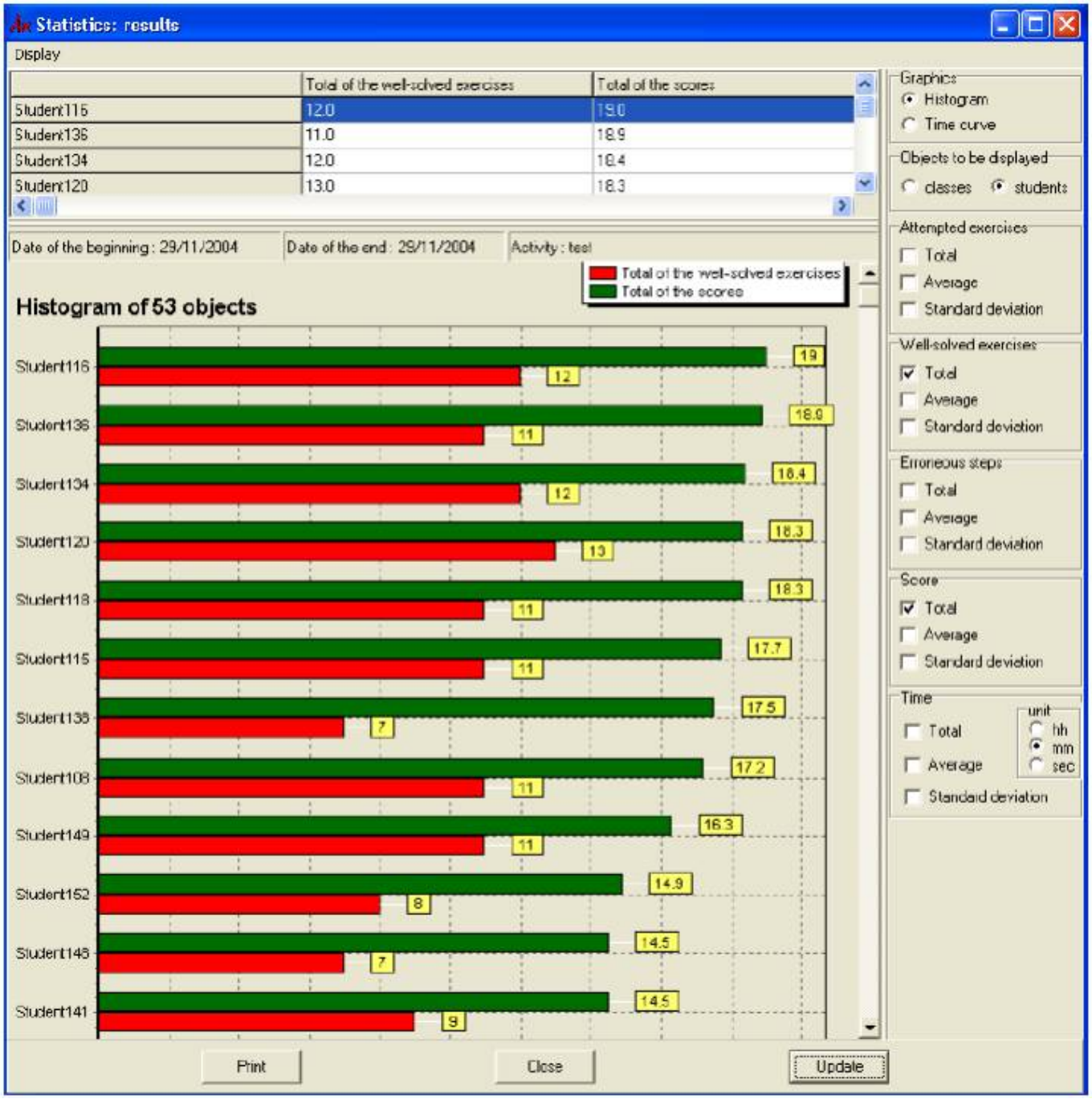

Fig 5 - Screen copy of the statistic window

Conclusion, result and perspective

Our choices have been for simplicity. We have tried to simplify our system for students (K7-K11, so that their work deals with mathematics and not with computer science), for teachers (who are not always fond of computers) and for us computer scientists and math didacticians. With the evolution of our system, from the first version, which was a microworld to the latest version in which the microworld is embedded within an exerciser, we have added components and modified previous components to favour simplicity. The evolution itself can be seen as a search for simplicity, so that teachers and students can work on algebra with our system easily.

As a consequence, we do not have use a CAS, nor IMS-LD, nor MathML; we have introduced different activities, some are particularly devoted to CAA; we have forged a map of exercises

which can produce exercises for almost 40 families of situation (i.e. for almost all the curricula in elementary algebra); we have added components to score student's attempts and to give teachers an overview of the work done in their class.

The next step could be to add some Artificial Intelligence components to give students and teachers some higher level information and help. We are already working on a remedial tutor for helping students with difficulties, a companion for providing examples of pedagogical solutions to every student and didactical analysis of student's work for teachers which could give them a good, global and understandable high level diagnostic about student's conceptual understanding of elementary algebra.

References

1. [Griffin 2004], Griffin F. "MacQTeX Randomised Quiz System for Mathematics" in Maths CAA Series, LTSN Maths, Stats & OR Network, University of Birmingham, UK, January 2004.
2. [IMS-QTI 2004] IMS Question and Test Interoperability: "Item Implementation Guide" & "Item Information Model", V2.0 Editor Steve Lay, University of Cambridge, http://www.imsglobal.org/question/qti_item_v2p0pd written in June 2004
3. [-Nicaud 2004] Nicaud, J.F., Bouhineau, D., and Chaachoua H. "Mixing microworld and CAS features in building computer systems that help students learn algebra", in International Journal of Computers for Mathematical Learning, Volume 9, Issue 2, Springer-Verlag, 2004.
4. [Nicaud 2005] Nicaud J.F., Bittar M., Chaachoua H., Inamdar P., Maffei L. "Experiments of APLUSIX in four countries", proceedings of the International Conference on Teaching Mathematics with Technology, ICTMT'7, Bristol, UK, July 2005.
5. [Perrin-Riou 2004] Perrin-Riou, B. "Programmation d'exercices OEF (open exercise format)" http://wims.auto.u-psud.fr/wims/ written in January 2004.
6. [Sangwin 2005] Sangwin C.J. "Making Mathematical Distinction in CAA with Computer Algebra" proceedings of the International Conference on Teaching Mathematics with Technology, ICTMT'7, Bristol, UK, July 2005.