Electronic Algebra and Calculus Tutor

Larissa FRADKIN and Victor ZERNOV[1]
Sound Mathematics Ltd.
11 Mulberry Close, Cambridge, CB4 2AS, UK

[1]Now at SmartOdds, 53-79 Highgate Road, London, NW5 1TL, UK

ABSTRACT

Modern undergraduates join science and engineering courses with poorer mathematical background than most contemporaries of the current faculty had when they were freshers. The problem is very acute in the United Kingdom but more and more countries adopt less resource intensive models of teaching and the problem spreads. University tutors and lecturers spend more and more time covering the basics. However, most of them still rely on traditional methods of delivery which presuppose that learners have a good memory and considerable time to practice, so that they can memorize disjointed facts and discover for themselves various connections between the underlying concepts. These suppositions are particularly unrealistic when dealing with a large number of undergraduates who are ordinary learners with limited mathematics background. The first author has developed a teaching system that allows such adult learners achieve relatively deep learning of mathematics – and remarkably quickly – through a teacher-guided (often called Socratic) dialog, which aims at the frequent reinforcement of basic mathematical abstractions through Eulerian sequencing. These ideas have been applied to create a prototype of a Cognitive Mathematics Tutoring System aimed at teaching basic mathematics to University freshers., an electronic Personal Algebra and Calculus Tutor (e-PACT).

Keywords: Undergraduate Mathematics, Socratic Dialog, Eulerian Sequencing, Cognitive Tutor.

1. INTRODUCTION

Modern undergraduates join science and engineering courses with poorer mathematical background than most contemporaries of the current faculty had when they were freshers. The problem is very acute in the United Kingdom but with more and more countries adopting less resource intensive models of teaching mathematics in schools the problem spreads. University tutors and lecturers spend more and more time delivering stepping up courses and support classes, covering the basics. However, most of them still rely on traditional methods of delivery which presuppose that the learners have a good memory and considerable time to practice, so that they can memorize disjointed facts and discover for themselves various connections between the underlying concepts. These suppositions are particularly unrealistic when dealing with a large number of undergraduates who are ordinary learners with limited mathematics background, limited memory, limited proficiency in explanatory reasoning, limited interest in the subject and limited time to cover a large amount of material, all exacerbated by limited contact with teachers and limited study skills.

The first author has developed a teaching system that allows such adult learners achieve relatively deep learning of mathematics – and remarkably quickly – through a teacher-guided (often called Socratic) dialog, which aims at frequent reinforcement of basic mathematical abstractions through Eulerian sequencing. The latter is a name for a systematic approach to mathematics as a language, which allows students to analyze (sequence) given mathematical expressions and thus find relevant solution algorithm (sequence of solution steps). It teaches learners to generate self-explanations, that is, argue why various steps are to be taken and not just what these steps are. This is important, because the amount learned is proportional to the number of self-explanations generated. Thus, the Eulerian Socratic dialog involves a teacher asking a series of questions surrounding a mathematical concept or solution step, and answering such questions posed by students. In addition, the teacher often asks what questions the students should ask themselves to proceed with the solution process. The dialogs are conducted in a friendly and sometimes humorous manner. Below we refer to the system as ESD (Eulerian Socratic Dialog).

While the teaching methodology described above puts a great emphasis on explanation of abstract mathematical concepts, it still requires students to do a reasonable number of exercises and have their understanding of concepts and deep-level reasoning skills reinforced every time they make a mistake. This part of the educational process can be automated with a Cognitive Tutor, a piece of software containing an artificial intelligence component to track students' work and tailor its feedback and hints, which captures teaching expertise, creating an artificial teaching expert. Two most prominent and relevant systems of this nature are AutoTutor that is designed to conduct a Socratic dialog with freshers studying Newtonian mechanics or IT and a Carnegie Learning System that employs a similar approach to teaching school algebra and geometry.

2. COGNITIVE TUTORING SYSTEMS

e-PACT has been conceived as a Cognitive Tutor to be integrated into the ESD mathematics teaching system. A cognitive tutor is an example of an expert system, an application that captures expertise of a specialist in a particular domain. Areas of expertise used in systems of this nature include the diagnosis of infectious diseases, the exploration for oil and minerals, the analysis of organic compounds, income tax planning and calculation, the operation of an air defence system, the configuration of

complex computer systems, and fault diagnosis in a modern automobile [1]. They all use a knowledge base - a list of fundamental facts about the domain and the rules that the human experts use, an inference engine (core) and a graphical user interface (GUI). Given an uncertainty in building the knowledge bases there is a danger in relying completely on the advice of expert systems in making life and death decisions. However, when a poor decision cannot put anybody or any business in jeopardy use of expert systems becomes less controversial.

Despite the initial high hopes use of expert systems is not that widely spread, because there is more to human expertise than facts and rules. In particular, most practical educators do not believe that computers can replace a teacher. Neither do we. However, there are aspects of teaching that can be automated. In particular, when learning mathematics there are some unequivocal facts that have to be mastered, such as algebraic or calculus rules. The connection between them can be explained by a teacher but to be internalized by a student the explanations need much reinforcing. Thus, there is a room for an expert system that can generate a large number of relatively simple mathematical exercises and comment on student mistakes in their solution, constantly reminding them of rules and definitions. Such a system would operate in a limited world of basic mathematics and even if some of its responses were imperfect it could prove a useful teaching tool. We see no need for such tool once the basics have been mastered. At this stage students can begin to use traditional textbooks.

Quite a few expert systems of pedagogical nature have been developed in recent years, with Pittsburgh University and Carnegie Melon University leading the way. The Pittsburgh University research revolves around cognitive tutors such as Andes and later AutoTutor which engage freshers in a Socratic dialog based on the natural language, simulating the dialog moves of human tutors. They indicate to students whether their answers are correct, can generate hints, divide the problem into different steps and provide proper feedback for each. The current versions of these tutors are designed to help college students learn topics in physics and computer literacy [2] and references therein. The research by Carnegie Melon University into cognitive mathematics tutors for the middle school led to creation of the Carnegie Learning System. Principles behind some of their design ideas can be found in [3].

Both groups are widely known for their interdisciplinary approach that combines cognitive psychology with artificial intelligence. There are other Intelligent Tutoring Systems based on the Socratic dialog, such as the CIRCSIM-Tutor designed to teach the first-year medical students the reflex control of blood pressure. Recently, the Pearson Publishing House began to offer a comprehensive tutoring system MyLab [4] which covers many subjects and makes an extensive use of hinting facilities. It contains a mathematical module called MyMathLab, but this relies on the traditional approach to mathematical teaching and makes use of neither Socratic dialog nor Eulerian sequencing. It thus cannot promote explanatory reasoning as e-PACT is meant to do.

Practical development of cognitive tutors is now gaining momentum, e.g. a general Hinting module has been developed that can be incorporated into any cognitive tutor [5]. It is designed to help students by giving suggestions, recommendations *etc.* It is not optimized for any specific study area such as mathematics, but implements hinting strategies, such as the maximum number of hints to select, meta-information for students, scoring that takes into account hints *etc*. A prototype has been created for the generation of adaptive hints based on the Semantic Web technologies.

Other modern developments in technology enhanced learning revolve around web based learning environments and agent based architectures, including animated pedagogical agents which are used to model social and emotional interactions.

3. ARCHITECTURES OF CURRENT MATHEMATICS TUTORS

Three major paradigms are used by designers of current mathematics tutors, Computer-aided assessment (CAA), Computer-aided instruction (CAI) and Intelligent computer-aided instruction (ICAI). The CAI type tutors still represent an overwhelming majority of hundreds of mathematics software packages that can be found or advertised on the web. A good example of CAA is *Mathletics* [6] which produces a large supply of questions generated at runtime, each with very complete feedback, including a fully-worked solution if a student gets an answer wrong. Examples of CAI applications include guided drill and practice exercises, computer visualization and computer-facilitated communication between students and teachers. Well known current CAI type mathematics tutors for engineering students are available through HELM [7]. They offer digitized lecture notes enhanced with hypertext, worksheets and multiple choice tests. Many students report enjoying the immediate responsiveness of computer interactions and appreciate the self-paced and private learning environment. Moreover, computer-learning experiences often engage the students, motivate them to learn and increase their independence and personal responsibility for education.

However, in some applications, especially those involving abstract reasoning and problem-solving processes, CAI has not been very effective. Critics claim that poorly designed CAI systems can even dehumanize or regiment the educational experience and thus diminish student interest and motivation. This is not surprising, since digitized lecture notes offer only a marginal advance on textbooks, simply alleviating search for relevant material. However, if students have no proficiency in deep-level reasoning and lecture notes offer no such reasoning themselves, it is difficult to expect much pedagogical gain from such notes, whether they are hardcopy or digitized.

Open object oriented learning environments are a newer development in CAI aiming to provide users with one platform which allows them an easy access to various graphical, modeling and pedagogical tools (agents) as well as easy interaction between different learners and learners and human tutors. While an exciting challenge to computer science and potentially an interesting tool to use in a classroom, they are subject to the same criticisms as offered above of old style CAI.

The ICIA tutors come closer to implementing constructivist epistemology. The architectures of classical cognitive tutors include procedural representations, conceptual structures and

production rules while newer architectures also have multiple soft constraints (e.g. neural networks, fuzzy production systems) as well as dialog moves generators. Propositional representations, neural networks and fuzzy production systems are relevant only to tutors involved in processing natural language. Procedural representations are used when the ordering of reasoning steps is important, as it is when teaching mathematics. Production rules are relevant to all cognitive tutors, since according to ACT-R theories [8], cognitive skills are based on production rules. These two representations have been thoroughly discussed by designers of two major Cognitive Tutors developed to instruct in technical subjects and mentioned above, AutoTutor and the Carnegie Learning System.

Both groups emphasize that the main difficulty lies in designing a Cognitive Model, that is, the part of the tutor that is charged with the task of interpreting the student performance and making instructional decisions based on this interpretation. To clarify the concept of a Cognitive Model, the one used in AutoTutor is based on the idea of curriculum scripts. These are "well-defined, loosely structured lesson plans that include important concepts, questions, cases, and problems" to be covered in a particular lesson. For example, the curriculum script for AutoTutor on computer literacy currently covers three macrotopics, hardware, the operating system, and the Internet. There are 12 topics within each of the 3 macrotopics (36 in total). The script includes *36 computer literacy questions and/or problems and 36 topic related questions/problems.* It also includes 36 *Ideal Answers* that correspond to each of the 36 topics. The *quality of any given learner contribution is determined by matching the learner contribution to each aspect* and all possible combinations of aspects in a particular Ideal Answer. Additional information contained in the curriculum scripts includes: (1) *anticipated bad answers for each of the 36 topics*, (2) *corrective splices (i.e., correct answers) for each anticipated bad answer*, and (3) numerous dialog moves (i.e., elaborations, hints, prompts, prompt responses and summaries) that are related to aspects of the Ideal Answers. All content in the curriculum scripts is written in English, as opposed to computer code. Therefore, a teacher or other individual who is not an expert programmer can easily author a curriculum script.

While holding an exciting promise, the current ICAI technologies, even as advanced as AutoTutor and the Carnegie Learning System, are still immature when it comes to teaching and learning, both because their repertoire is very limited and because the dialog they generate often leaves a lot to be desired.

4. ARCHITECTURE OF E-PACT

The natural language processing that hampers the dialog in such systems as AutoTutor is not to be addressed by e-PACT and therefore, it is to utilize only the following architectural features: Cognitive Model, Procedural representation (based on Decision Trees), Object-Oriented Design, Production Rules (if – then or condition-action pairs) and Dialog Moves.

Cognitive Model and Procedural Representation

e-PACT is meant to utilize the EDS lecture/tutorial model of teaching and make use of a scaffolding tool not usually adopted in mathematics instructions, a Decision Tree. The Lectures and Summaries of lectures are to be included under Help as texts containing didactic descriptions supported by examples, but some conversational aspects of the EDS methodology built around the first author's experience of how students learn and think are to be automated. To widen the Tutor repertoire the idea of a database/back-end is abandoned: e-PACT is meant to generate a large number of problems at random and is to be programmed to "discuss" any of them. Thus, on the one hand, the e-PACT's cognitive model is to be simpler than any of those used in AutoTutor or CLS but on the other hand, unlike them, it is to generate and discuss a large number of possible exercises (of prescribed types). This can run into hundreds.

e-PACT is meant to contain Intelligent Context Aware Parsers that recognize common errors and misconceptions and dynamic Decision Trees that sequence solution steps and guide the student through them with prompts and comments. e-PACT is to tailor its hints and responses to specific student mistakes. Whatever student's turn, e-PACT is to present relevant and effective comments to build his/her mastery of the subject. By constructing human-like dialogs and using correct verbalization of mathematical processes e-PACT is meant to emphasize communication skills.

Whether commenting on a particular answer or engaging student in a Decision Tree based dialog, creators of e-PACT creators to strive to ascertain that an intelligent feedback is provided in all cases, and that the feedback is always delivered in an understandable and conversational manner. This is a challenge, because e-PACT is meant to interpret many different styles of mathematical input, allowing students to use a variety of conventions and be tolerant of many low-level mistakes. This is a conscious choice, since "in tutorial systems, effective progress in teaching the problem-solving target is frequently hindered by expressive sloppiness and low-level errors made by the student, especially in conventionalized expressions such as formulas." [9]. Thus, the current e-PACT prototype is tolerant to a number of spaces used by the user, it can interpret a function whether the argument is bracketed or not (such inputs as *sin(x), sinx* and *sin x* would be all acceptable); if an expression is bracketed more than once, the prototype just generates a gentle warning that the input contains extra brackets; and if a bracket is missing here or there, this is also handled through warnings rather than error messages. If an error is of the type expected of a dyslexic student, say the input is ex rather than e^x the prototype sends a detailed message on the corresponding mathematical convention, drawing the student's attention to the fact that in mathematics the order and position of symbols is often imbued with a particular meaning. The e-PACT prototype already "knows" enough algebra to be able to comment on such input as p^{-1} whether it is given in that form or as $\frac{1}{p}$ and in its messages use the language appropriate to the form chosen by student.

While this tolerance provides for better usability it makes interpreting the student performance and arriving at instructional decisions based on this interpretation so much a harder task.

Object Oriented Design

e-PACT's object oriented design is based on the classes that model mathematical objects. For example, in the current prototype the class Function Elementary contains the same elements as elementary functions used in undergraduate mathematics, *function name, argument* and *power index* (if applicable); the class Sum of Two Functions models *addition of functions etc*. The e-PACT's architecture as engineered at the highest level is shown in Figure 1: the Random Problem Generators are meant to generate a variety of exercises, Solvers are meant to solve them rather than have solutions stored in a database, Intelligent Input Readers are meant to interpret a large class of possible inputs, including the ones containing extraneous symbols and Solution Comparators are meant to compare the expected answers with the ones provided by students. The e-PACT Controller is meant to manage the GUI interaction between the user and Core, including interaction with the dynamic Decision Trees to be affected via a Decision Tree Based Dialog module (see Figure 2). It is planned that in the full version of e-PACT the Controller takes into account the particular difficulties experienced by the user and adapts by taking him/her to the topics that require extra revision.

Production Rules

Comparators implemented in e-PACT are not meant to involve any statistical analysis but compare identifiable parts (aspects) of the mathematical objects, such as *sign, coefficient, function name, argument, term factor, sum, product etc.* All instructional decisions and messages are meant to be based on such comparisons and therefore, be context-dependent.

For this reason, e-PACT messages are meant to utilize information from different pieces generated by the Compare methods in different classes, such as Elementary Function, Simple Function, Sum of Two Functions, Product of Two Functions *etc.,* each responsible for its own portion of the mathematical input. The messages are meant to be formatted to alleviate comprehension, using indents and helpful connectives, depending on the number of identifiable aspects in the student answer and the number of mistakes in each.

When creating a Cognitive Tutor, the advantages of using the object-oriented design rather than a database are two-fold: we can deal with relatively large classes of problems at once and extra levels of difficulty and extra complexity can be added without affecting the design functionality. This implies that the Cognitive Tutors designed in this manner are easily maintained and enhanced. The clarity of design is also an advantage: the Core classes model traditional mathematical concepts and thus new researchers can be easily integrated into the team.

Dialog moves

The types of dialog moves used in e-PACT are similar to the ones used in other Cognitive Tutors but since responses are only half pre-programmed their implementation is more involved. The types of moves used in the current prototype are positive feedback, negative feedback, splice, elaboration, explanation, summary, prompt, guess and connection.

A sample screen shot is presented in Figure 3.

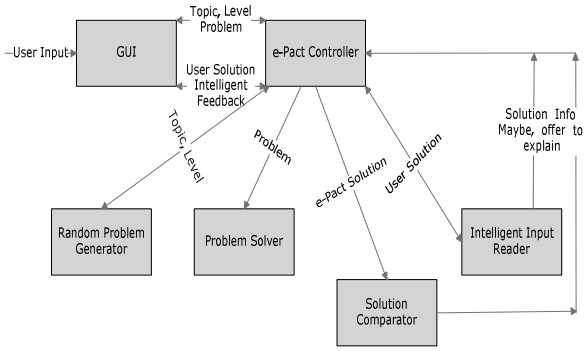

Figure. 1. e-PACT Architecture: assessment of student's final solution.

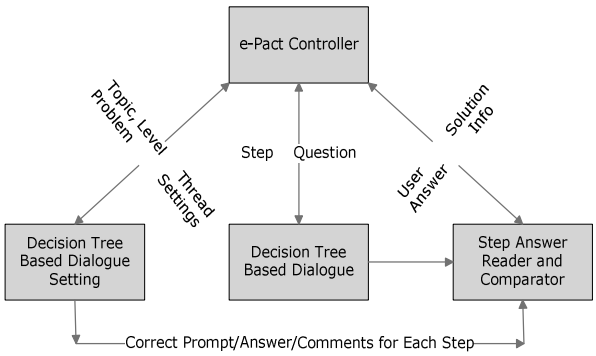

Figure. 2. Architecture of the Decision Tree Based Dialog.

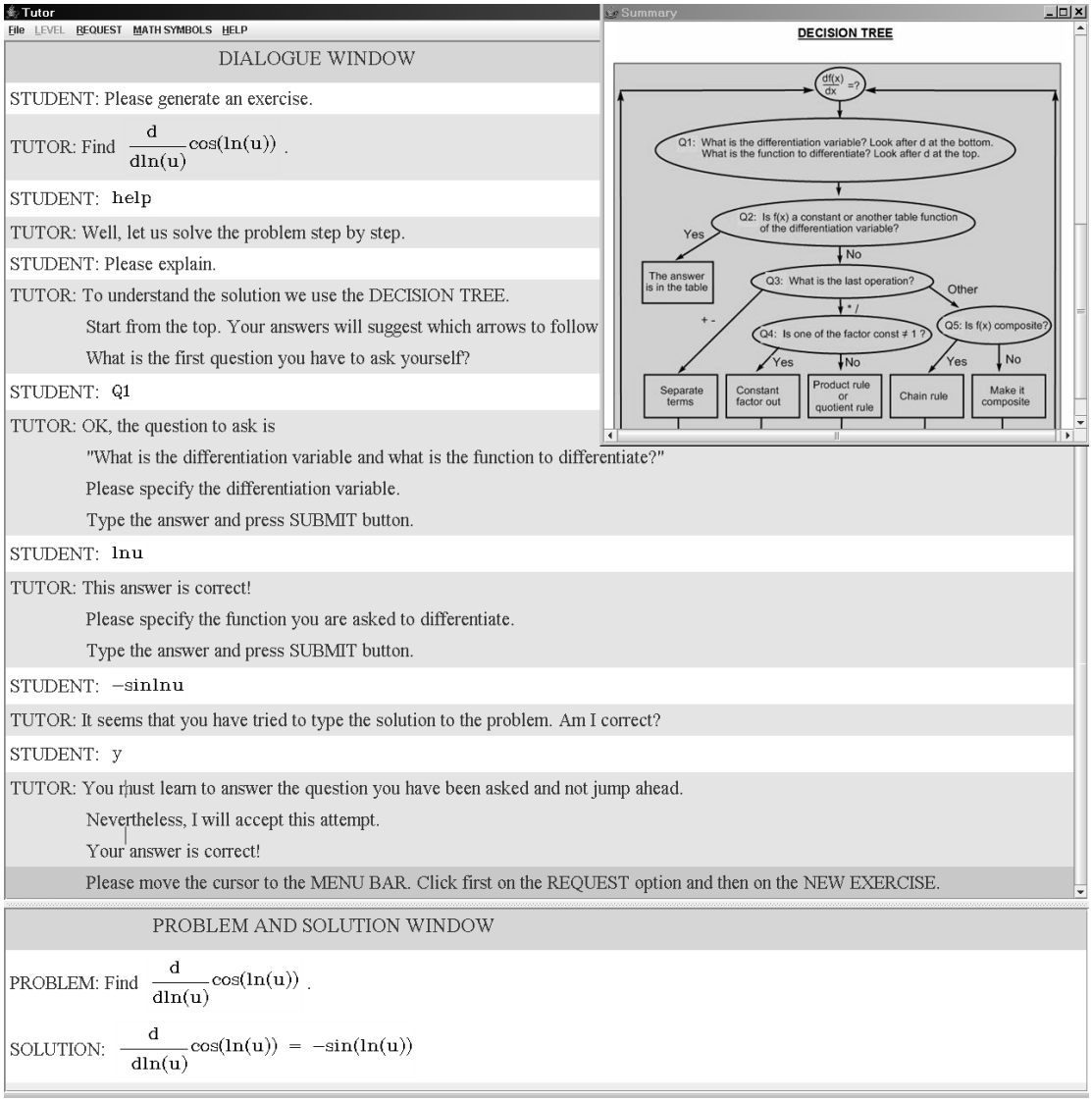

Figure 3. A screen shot of a recent Differentiation Tutor with a sample dialog on Level 2

5. THE USER REQUIREMENTS FOR E-PACT

The user requirements for e-PACT are similar to the requirements for the Carnegie Learning System:

Simple, Straightforward GUI

1) Straightforward presentation of mathematical symbols
2) No distracting images or photos, minimalist presentation
3) Intuitive
4) Easy to use

Just-in-time Feedback

1) Hints are contextual and oriented towards helping the student to follow key steps in the problem.
2) Immediate feedback enables the student to self-correct and leads to more effective learning and applying of the mathematical rules
3) e-PACT recognizes the most common student errors and responds appropriately.

Achievement monitor

1) As a student becomes more proficient in a skill, e-PACT progresses him/her to a higher level of difficulty.
2) Teachers can view off-line snapshots of each dialog.
3) Students receive an immediate and precise feedback, which is a strong motivator to succeed.

6. TESTING E-PACT

The e-PACT testing can be separated into feature testing, testing of its reaction to correct answers and testing of its reaction to incorrect answers. Features to address revolve around mathematical conventions, usability, data typing and GUI. Both alpha and beta testing are required.

Alpha Testing
A considerable amount of alpha-testing can be carried out during the implementation process. Testing of the Generate, Print and Solve methods is easily automated to ascertain that all problems and solutions are generated and printed correctly. It is more difficult to test Parsers and Compare methods. Only partial automation of this testing process can be achieved: it is impossible to envisage all possible erroneous inputs.

Here are several examples of issues that were addressed during the alpha feature testing of the current prototype.

Mathematical conventions In order for e-PACT to help students concentrate on mathematics rather than learn various artificial ways of communicating with the application developers addressed various issues revolving around algebraic rules:

1) The *factor* 1 and the *term* 0 can always be dropped but if a student keeps them in his/her answers e-PACT does not treat this as an error.
2) e-PACT is aware of *commutativity* of both *addition* and *multiplication*.
3) It is a mathematical convention to drop *power index* 1. In view of this, a power index 1 should not be printed when composing a message on *power* and by the same token, while in its responses e-PACT normally refers to both the *function* name (*power*) and *power index, w*hen the *power index* is 1 the *function* name changes to *linear* and the *power index* is not mentioned.

Usability To make e-PACT attractive to users its prompts and responses are presented in a readable and engaging manner employing simple yet technically correct language:

1) If the correct answer involves a *power* but the student inputs a different *function* instead, the error message specifies that the correct function is *power* but does not mention the *power index*. The index is mentioned only if the student answer is a *power* but his/her *index* is incorrect.
2) The *argument* of an *exponent* is referred to as a *power* and the *argument* of *power* is referred to as an *independent variable* - unless the power is -1 and the student presents the answer as "*one over*", in which case the *argument* is referred to as a *denominator*.
3) When the student answer contains many mistakes, they are all described in a formatted and easily readable manner.
4) If the student enters the Decision Tree Based dialog but half way through realizes what the final answer is, in order not to loose his/her interest the student this final answer is allowed and commented.

Data typing General usability issues common to all AI projects are addressed in e-PACT as well:

1) If the student types a character where a number is expected this can cause software failure as data typed translations will not match. The problem is dealt with by using intermediary objects which test for appropriate input contents prior to conducting a typecast.
2) If the student types an extraneous input in front, or at the end of legitimate input e-PACT sends a message advising them to double-check input in front or at the end or both.
3) If the student input misses an argument, either through oversight or due to dyslexia or dyscalculia the problem is spotted and commented on.

GUI issues Many GUI tests have been conducted to ascertain that editing facilities are intuitive, allowing use of the cursor and arrows as well as DELETE key and only the uncorrupted final version of the student input that is passed to the Core. It is well known that while some users prefer to submit their input by pressing ENTER others look for a SUBMIT button. Both solutions have been implemented. It is also well known that some users prefer to move the mouse to the menu bar options while others prefer hot keys. Both solutions have been implemented.

Beta Testing
By way of beta testing, a number of first authors' students were observed by independent researchers using the software [10]. Even when exposed to it for the first time, most needed very little assistance with the technical aspects of communicating with e-PACT, and even less assistance with the mathematics. They found the system easy to use, since it follows the approach adopted in their lectures and tutorials. *"It doesn't tell you what to do"*, one student said, and then went on to explain that the software does not show the next step in the solution or the answer, instead *"the feedback encourages you to think about what you need to do; (Differentiation Module of – LF) e-PACT takes you back to the Decision Tree, so you will be able to differentiate any function"*.

The only issue identified was the fact that inexperienced users do not read instructions issued by e-PACT but scan the page for a button to push. In order to resolve this issue

1) the wording of many of e-PACT instructions has been simplified,
2) the manual type instructions are presented on a grey background, mathematical instructions are presented on the yellow background, while the Answer Box employs the white background,
3) after each Tutor prompt users are directed to the Answer Box At the moment this is achieved by implementing a flashing square. Other attention focusing devices will be deployed in future.

7. CONCLUSIONS

Two modules of e-PACT, a Differentiation Tutor and Tutor for Solving Algebraic Equations are included into the current prototype of e-PACT. It is planned to use the experience gained to redesign e-PACT and implement several other modules that would allow students work on mathematical topics they need to exercise most.

8. REFERENCES

[1] K. Devlin **Goodbye Descartes: The End of Logic and the Search for a New Cosmology of the Mind**, New York: John Wiley, 1997.

[2] A.C. Graesser, D. S. McNamara and K. VanLehn "Scaffolding Deep Comprehension Strategies Through Point&Query, AutoTutor, and iSTART", **Educational Psychologist**, Vol. 40, No.4 2005, pp. 225-234.

[3] V. Aleven, B. Mclaren, I. Roll, and K. Koedinger, "Toward Meta-cognitive Tutoring: A Model of Help Seeking with a Cognitive Tutor", **Int. J. Artif. Intell. Ed.**, Vol. 1, No.2, 2006, pp. 101–128.

[4] http://www.coursecompass.com/

[5] P.J. Muñoz-Merino, and C.D. Kloos (2009). "An architecture for Combining Semantic Web Techniques with Intelligent Tutoring Systems", **Intelligent Tutoring Systems Conference,** Vol. 5091, 2009, pp. 540–550.

[6] M. Greenhow "Mathletics – a suite of computer-assisted assessments", **MSOR Connections**, Vol. 8, No 3, 2008.

[7] http://www.lboro.ac.uk/research/helm/

[8] R.J. Anderson and C Libiere, **The Atomic Components of Thought**, Maywah, New Jersey: Lawrence Erlbaum, 1998.

[9] H. Horacek, M. Wolska "Handling Errors in Mathematical Formulas", **Lecture Notes in Computer Science**, Vol. 4053, 2006, pp. 339-348, Berlin/Heidelberg: Springer.

[10] L. Fradkin, V. Zernov, C. Crisan and S. Lerman "First Year Engineering Mathematics: The London South Bank University experience", **MSOR Connections**, Vol. 10, No. 10, 2010.