Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Research Report
RR-02-01

Evaluators and Suggestion Mechanisms for ACTIVEMATH

Erica Melis and Eric Andrès

November 2002

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Postfach 20 80
67608 Kaiserslautern, FRG
Tel.: + 49 (631) 205-3211
Fax: + 49 (631) 205-3210
E-Mail: info@dfki.uni-kl.de

Stuhlsatzenhausweg 3
66123 Saarbrücken, FRG
Tel.: + 49 (681) 302-5252
Fax: + 49 (681) 302-5341
E-Mail: info@dfki.de

WWW: http://www.dfki.de

Deutsches Forschungszentrum für Künstliche Intelligenz

DFKI GmbH

German Research Center for Artificial Intelligence

Founded in 1988, DFKI today is one of the largest nonprofit contract research institutes in the field of innovative software technology based on Artificial Intelligence (AI) methods. DFKI is focusing on the complete cycle of innovation — from world-class basic research and technology development through leading-edge demonstrators and prototypes to product functions and commercialization.

Based in Kaiserslautern and Saarbrücken, the German Research Center for Artificial Intelligence ranks among the important “Centers of Excellence” worldwide.

An important element of DFKI’s mission is to move innovations as quickly as possible from the lab into the marketplace. Only by maintaining research projects at the forefront of science can DFKI have the strength to meet its technology transfer goals.

DFKI has about 165 full-time employees, including 141 research scientists with advanced degrees. There are also around 95 part-time research assistants.

Revenues for DFKI were about 30 million DM in 2000, half from government contract work and half from commercial clients. The annual increase in contracts from commercial clients was greater than 20% during the last three years.

At DFKI, all work is organized in the form of clearly focused research or development projects with planned deliverables, various milestones, and a duration from several months up to three years.

DFKI benefits from interaction with the faculty of the Universities of Saarbrücken and Kaiserslautern and in turn provides opportunities for research and Ph.D. thesis supervision to students from these universities, which have an outstanding reputation in Computer Science.

The key directors of DFKI are Prof. Wolfgang Wahlster (CEO) and Dr. Walter Olthoff (CFO).

DFKI’s five research departments are directed by internationally recognized research scientists:

- ❏ Knowledge Management (Director: Prof. A. Dengel)
- ❏ Intelligent Visualization and Simulation Systems (Director: Prof. H. Hagen)
- ❏ Deduction and Multiagent Systems (Director: Prof. J. Siekmann)
- ❏ Language Technology (Director: Prof. H. Uszkoreit)
- ❏ Intelligent User Interfaces (Director: Prof. W. Wahlster)

In this series, DFKI publishes research reports, technical memos, documents (eg. workshop proceedings), and final project reports. The aim is to make new results, ideas, and software available as quickly as possible.

Prof. Wolfgang Wahlster
Director

Evaluators and Suggestion Mechanisms for ActiveMath

Erica Melis and Eric Andrès

DFKI-RR-02-01

This work has been supported by a grant from The Federal Ministry of Education, Science, Research, and Technology (FKZ ITW-).

© Deutsches Forschungszentrum für Künstliche Intelligenz 2002

This work may not be copied or reproduced in whole or part for any commercial purpose. Permission to copy in whole or part without payment of fee is granted for nonprofit educational and research purposes provided that all such whole or partial copies include the following: a notice that such copying is by permission of the Deutsche Forschungszentrum für Künstliche Intelligenz, Kaiserslautern, Federal Republic of Germany; an acknowledgement of the authors and individual contributors to the work; all applicable portions of this copyright notice. Copying, reproducing, or republishing for any other purpose shall require a licence with payment of fee to Deutsches Forschungszentrum für Künstliche Intelligenz.

ISSN 0946-008X

Evaluators and Suggestion Mechanisms for ACTIVEMATH

Erica Melis and Eric Andrès

Contents

Abstract

This report describes the feedback in ActiveMath, a learning environment for mathematics. In particular, we distinguish local and global feedbacks. For the global feedback we describe the blackboard-architecture with its reusable and easily modifiable components, specify the current evaluators of the learners actions, and present the user-adaptive global learning suggestions implemented so far. The report also describes the design of the suggestion user interface that is based on empirical results about feedback described in the literature. Finally, remaining questions that need to be clarified by empirical research are discussed.

1 Introduction

ActiveMath is a user-adaptive web-based learning environment for mathematics. It generates learning material for the individual learner according to her learning goals, preferences, and mastery of concepts as well as to the choosen learning scenario [20].
ActiveMath's user model consists of the components history, profile, and a database of mastery-level for the learning objects. The history contains information about the user's activities (reading time for items, exercise success rate, editing of user model). The user profile contains her preferences, scenario, goals submitted for the session. To represent the concept mastery assessment, the user model contains values for a subset of the competences of Bloom's mastery taxonomy [6]:

- Knowledge
- Comprehension
- Application.

We assume that knowledge-mastery is reached by reading a text but also from examples and exercises. Bloom defines the skills needed for 'knowing' as recall of information, knowledge of major ideas; Comprehension-mastery can be achieved by relating several concepts and then answer questions and understand examples. Bloom defines the needed skills as understanding information, translating knowledge into a new context, predicting consequences; Application-mastery relates to solving problems by applying the concept. Bloom defines the needed skills as usage of the concept (in new situations), solving problems using the required skills or concepts.[1] Similar categorizations of mastery have been employed by several researchers, e.g., Merrill and Shute [25] in their taxonomies of outcome types. Finishing an exercise or going to another page triggers an updating of the user model in ActiveMath. Different types of user actions influence the values of corresponding competencies because each reflects a different competency [2]. In

[1] Transfer-mastery could be the most advanced mastery but currently, we subsume this in application currently.

[2] The implicit connection between competencies is also considered but will be better formalized in Bayesian Net user model which is in progress.

particular, reading concepts corresponds to 'Knowledge', following examples corresponds to 'Comprehension', and solving exercises corresponds to 'Application'. So far, ActiveMath primarily served the adaptive (static) document generation, the integration of service systems, and local feedback in exercises. This report addresses an extension and discusses different types of feedback in ActiveMath. We suggest to distinguish local and global feedback. The global feedback is a new service of ActiveMath which is described in this report. In particular, we describe the architecture, the kinds of evaluators of student action, and some of the user-adaptive global learning suggestions implemented so far.

2 Separation of Local and Global Feedback

Traditionally, user-adaptive feedback and help in tutor systems (ITSs) has been designed for a direct response to students' problem solving actions and the feedback is designed to help students to accomplish a solution, e.g., in the PACT tutors [5] or in CAPIT [19]. Frequently, this feedback is an explicitly authored text that reflects the author's experience with typical errors in a specific problem solving context. In some sense, the specific feedback is questionable because authoring all the specific feedback is a very laborious task (see e.g., [29]) and often requires an extreme authoring effort for explicitly authoring what can go wrong and what the reason is for each erroneous action in each exercising. Therefore, we try to avoid such a kind of diagnosis and corresponding feedback in ActiveMath. Moreover, the usage and benefit of more and more detailed help may strongly depend on the individual user [1] and some users might even dislike frequent suggestions and intrusion [8].

Although in most ITSs the feedback is a direct answer to single problem solving steps, some systems provide feedback targeting meta-cognitive skills of the student. For instance, [13, 2] try to support self-explanation of worked-out examples; SciWise [31] provides feedback for planning and reflecting activities in experimental science and for collaboration.

Generally, two kinds of feedback and guidance can be provided by an ITS, a *local* response to student activities which is supposed to coach the recognition of errors and the correction of single problem solving steps of the learner and a *global* feedback targeting the entire learning process. This differentiation somewhat resembles the distinction of task-level and high-level described in the b4-process model of [3].

These two kinds of feedback differ with respect to time, realm, content, and aim. For instance, local feedback is provided immediately after each problem solving action of the user and local feedback should be given directly attached to the problem solving activity and possibly presented in the exercise window. Instead, the global feedback and suggestions can be provided independently and may be delayed, i.e. delivered, when the user has finished reading a text, studying an example, or working on an exercise.

Many ITSs do not provide global feedback at all. And even if they do, such as SQL-Tutor [21] and CAPIT [19], they do not separate local and global feedback, say architecturally. external e.g.,

In ACTIVEMATH, local and global feedback is distinguished because of their different aims, different foci, different learning dimensions, and different mechanisms. In addition, the employment of service systems for the check of the correctness of a problem solving step and for the generation of local problem solving feedback [7] is a practical reason for separating local and global feedback. The local feedback such as 'syntax error', 'step not correct, because...', 'task not finished yet', or 'step not applicable' is computed by a system and related to a problem solving step in an exercise or to the final achievement in an exercise. The current implementation of local feedback is explained in [7] and more technically in `http://www.ags.uni-sb.de/~adrianf/activemath/`.
As opposed to the local feedback, the global feedback scaffolds the student's navigation, her meta-cognition, and dynamically suggests appropriate presentations of content (including examples and exercises). The global feedback and suggestions may concern, e.g., what to learn, to repeat, to look at, or to do next, navigating the content, reflecting, monitoring.
The computation of global feedback requires diagnostic information of several user activities. The information about the student's navigation, reading, understanding, and problem solving actions, e.g. their duration, and success rate, has to be used as a basis for user-adaptive suggestions. That is, information about the history of the learner's actions and information from her user model is necessary to generate useful suggestions.
In what follows we deal with global feedback only.

Notation. Note that K/C/A-present(c) is an abbreviation for: present content contributing to the concept c in a K-, C-, or A-oriented way respectively. K/C/A-present(c) are functions of the ACTIVEMATH' existing course generator. K-oriented means present just concepts and possibly explanations; C-oriented means present concepts and examples; A-oriented means present the full spectrum of content including concepts, examples, and exercises.

3 Blackboard Architecture for Global Feedback

The architecture for the global suggestion mechanism in ACTIVEMATH clearly separates diagnoses and suggestions as shown in in Figure 1. An advantage of the separation of evaluation and suggestions is that the same evaluation results can be used by different suggestion mechanisms, in different pedagogical strategies, and later also by a dialog system. For instance, if the diagnosis yields a `seen(example, insufficient)`[3], then `example` is presented again in a strict-guidance strategy but not in a weak-guidance strategy.

Evaluators Some evaluators provide a diagnosis immediately from one of a user's action while other evaluators infer a diagnosis from the immediate diagnoses and additional information In Figure 1, several *immediate* and one *intermediate* evaluator are displayed. New immediate and intermediate diagnosis agents can be easily added, e.g., an evaluator for the individual average reading time.

[3] i.e., the time for reading the example is less than a threshold

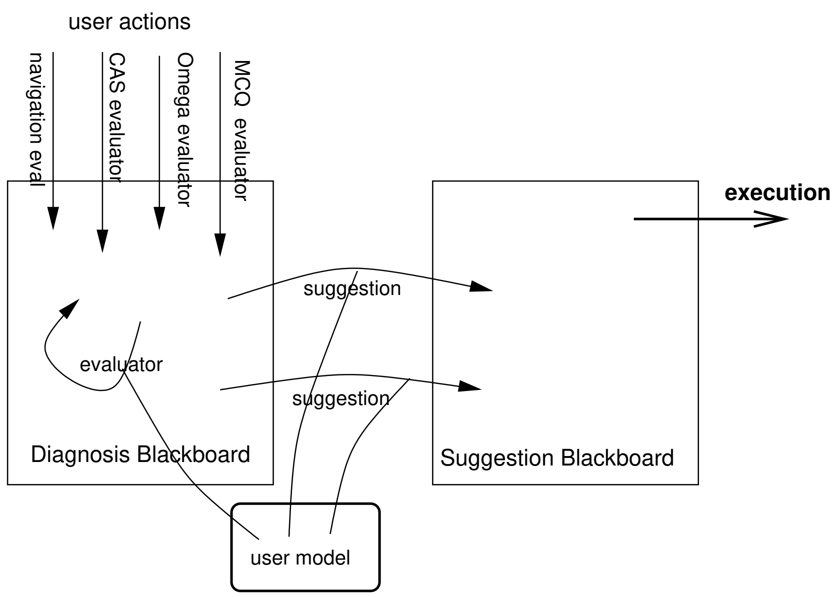

Figure 1: The architecture of evaluator and suggestion mechanisms

The immediate evaluators each watch one of the following types of activities

- navigation
- reading (time)
- problem solving (assessed performance)
  - MCQ exercises
  - exercises with a Computer Algebra System
  - exercises with the Omega proof planner

The immediate evaluators pass their results to a *diagnosis blackboard* (DBB) and to the user model (user's mastery-level of concepts and to the activity history). The current updating mechanism for mastery-values in the user model is described in [20]. Essentially, K-mastery values are triggered by reading, C-mastery values by dealing with examples, and A-mastery values by dealing with exercises. A Bayesian Net user model including its updating mechanism is future work.
Intermediate diagnoses are computed by other evaluators from the information on the DBB and in the user model. These diagnosis are written on the DBB too. As displayed in Figure 1, several suggestors compute global feedback from the diagnoses on the DBB and write on the *suggestion blackboard* (SBB). If necessary, the results are sent to a ConflictManager that rates the different suggestions on the SBB. Then the best rated suggestions are executed.

4 Evaluators/Diagnoses

Currently, ActiveMath has the following evaluators, each one watching a particular type of activity of the user or the DBB.

- navigation
- reading
- exercises
  - MCQ exercises
  - exercises with a CAS
  - exercises with the Omega proof planner
- intermediate evaluators

These evaluators return diagnoses, among them

- mastery with values `poor` or `okay`. Poor means that the success rating was below 30%. This evaluator acts when the user has finished an exercise.
- `navigation` with values `okay/(irrational, ?start)`, where `irrational` is diagnosed in case the user navigated from one concept to another without following any dependency.
- `focus-concept`(?c)
- `seen` with arguments ID and one of the values `okay/notSeen/insufficient` for evaluated reading time of an item ID.
- `K,C,A-mastery` of a concept with values that correspond to a percentage. This is passed to the user model for updating and written onto the DBB.
- `solution` with the arguments ID (of exercise) and one of the values `correct/incorrect`.
- `teacherDiagnosis(?x)`, where ?x is the name of a typical error. If the learner's input matches with a typical error, typicalError(?x), pre-defined by the author of an exercise, the evaluator returns to the DBB a corresponding tuple `(teacherDiagnosis(?x), teacherFeedback(?x), teacherReaction(?x))` that is pre-defined by the author. If `teacherDiagnosis(?x)` provides a mastery diagnosis, then it is passed to the user model as well.

The following intermediate features can be inferred from information in the DBB and in the user model:

- `missingPrerequisite(?concept, ?level)`, where `?level` can be K, C, or A currently. This fact means that `?concept` is a prerequisite concept of the focus-concept and its `?level`-mastery in the user model is insufficient. This fact is inferred only in case of an insufficient result of the user actions related to the focus-concept.
- `SeenAndKnown(?level)`, `notSeenButKnown(?level)`,`SeenButUnknown(?level)`, `notSeenAndUnknown(?level)`, all meant for the focus-concept, where `?level` is one of the values K, C, or A. These intermediate features can be inferred from the `focus-concept`, from the `seen` diagnoses of the items

contributing to the `focus-concept`, and from the `K,C,A-mastery` for the `focus-concept`.

to

The following diagnostic features may be added later.

- self-guidedness
- average reading speed
- CAS-ability
- Omega-ability
- the user's motivational state.
- percentage of guess and/or slip

In the following, we describe the evaluators for reading, navigation and for the intermediate features in more detail. The evaluators for exercises are directly implemented in the exercise environment and pass their results to the DBB.

4.1 Evaluators for Reading

The device the reading evaluator is based upon is a *poor man's eye tracker* [28]. This is a software component that, based on mouse movement, can approximately measure the time a user spends looking at a unit, i.e., a definition, a motivation, a theorem, an exercise, or an example.
A reading evaluator calculates its results from the actual reading time for a unit returned by ActiveMath' poor man's eye tracker and from the `averageReadingSpeed`. The evaluator can provide the values okay, notSeen, insufficient, and unclear of the feature `seen`. Note that the required threshold for a sufficient reading time might vary between a definition, a theorem, an explanation, an example, and an exercise. Note also that the reading speed for formulas may individually differ from the reading time for text.
A read-evaluator updates the user model's K-value for a concept (i.e., the value for knowledge-level mastery) after the user has looked at an instructional item that contributes to learning the concept and C-values (for comprehension-level mastery) after reading examples in particular.

4.2 Evaluators for Navigation

The navigation behaviour of a student has to be monitored because a user with little experience may get lost in a (collection of) hypermedia document [22]. ActiveMath's user tracking device, DFKeye [28], communicates to the reading evaluator the page and the time the student spent on it. The evaluator diagnoses, whether the navigation behaviour makes sense according to

- the position of the page relative to the previous one in the user's history

- the relation of the focus concept on the current page with the focus concept of the preceeding one.

If the current page is immediately following or preceeding the previous page, the navigation is rated okay. In this case, the user is either following the curriculum that was proposed, or is going back to see the last concept that was presented again. If the new focus is in any relation with the old one (e.g., a precondition), then the navigation is also okay. These conditions define the *positive* navigation actions. If a navigational action is not positive, we call it *negative*. The evaluator keeps track of both types of navigational actions. If the user seems to be lost, that is, she performs many negative navigational actions, it diagnoses *irrational navigation* and passes this to the DBB. This can lead to the generation of a hint that helps the user on her way through the hypertext. Conversely, if the user's navigational actions are rated positively, then it diagnoses *navigation okay* and this can cause the generation of reassuring feedback.
A more complicated evaluation function is planned that results in the diagnose *self-guided navigation* for a navigation behaviour that does not fully follow the sequencing suggested by the system (static book or dynamic suggestions) but is in line with the user's mastery level. That is, if the user discovers that she does not know enough about several prerequisites and navigates back to pages concerned with the prerequisite-concepts or -items and then returns to the current page, this is taken as an evidence for her awareness of the course structure and dependencies as well as for self-guidedness.

4.3 Evaluators for Intermediate Features

Each evaluator for intermediate diagnosis watches the DBB-entries and the user model and infers intermediate diagnoses. Currently, these are the following:

- `missingPrerequisite(concept, level)`, where `level` can be **K**(nowledge), **C**(omprehension), or **A**(pplication). `missingPrerequisite` is inferred only in case of an insufficient result for an exercise related to the concept which is in focus. This fact means that `concept` is a prerequisite concept or item of the insufficiently mastered focus-concept and its `level`-mastery in the user model is also insufficient. (The mastery of the focus-concept itself is not intermediate but immediate.)

- `SeenAndKnown(level)`, `notSeenButKnown(level)`, `SeenButUnknown(level)`, `notSeenAndUnknown(level)` are all stated for the focus-concept, where `level` is one of the values K, C, or A. `xxUnknown(K)` means that in the user model the K-value (for the focus-concept) is insufficient (say, less than 80%). `xxUnknown(C)` means that in the user model the C-value (for the focus-concept) is insufficient but not the K-value. `xxUnknown(A)` means that in the user model the A-value (for the focus-concept) is insufficient but not the K- and C-values.

  The intermediate features are inferred for the `focus-concept`, using the `Seen` diagnoses of those items that contribute to the `focus-concept` in the learning material and the `K,C,A-mastery` for the `focus-concept`.

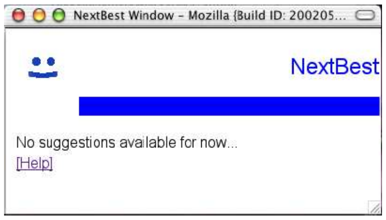

Figure 2: The prototypical user interface for the global suggestions (TODO: image with a suggestion.)

5 Suggestion Mechanisms

A suggestor evaluates rules and writes its results on the SBB. Based on the facts on the SBB, the actual suggestion presentation is generated to deliver

- a smile/noSmile puzzled, neutral, happy (smile), or sad (noSmile) face (which is thought as a shortcut). The default expression is neutral.
- the more detailed and personalized feedback that consists of a verbal feedback (only abbreviated in the tables below) and
- one or several presentation-actions.

We specified the reassuring 'smile' feedback as part of the GUI presented in Figure 2 because we feel that reassurance and positive feedback is important for motivational reasons [18] and for avoiding situations in which the learner feels insecure.

5.1 Suggestors

The presentation-actions include

1. navigation help
2. content suggestions
    - present new or skipped example
    - present similar example
    - present new exercise
    - present same exercise
    - present again the focus-concept maybe also examples, exercises
    - present (missed) instructional items[4]
    - present certain prerequisites maybe together with examples and exercises

[4] if misunderstanding of concept not attributed to prerequisites or reading time

- K-present a concept. This means a presentation of items that intends the improvement of the user's knowlege of a concept. Mainly definitions and elaborations will be shown.
- C-present a concept. The goal of this action is to lead the user to an understanding of the concept. Currently, this is tried by presenting items like examples and understanding questions.
- A-present a concept, i.e., present some exercises and other items that help the user to learn how to apply the concept.

For each learning-goal level (K-, C-, or A), a suggestion strategy can be designed as a set of suggestors. In what follows, we present the suggestors of an *A-level* oriented suggestion strategy which (1) reacts to navigation problems and (2) suggests content. First, we explain the essence of each rule and then the actual rules follow.
Note that x in the expression `seenButUnknown`(x) denotes the missing level of mastery of the focus-concept.

Rules for Navigation Suggestions are needed because ActiveMath delivers a hypertext learning document and it is known that navigation in hypertexts needs special attention [22] and being lost in hyperspace puts an additional load on the learner.

- if the user navigated appropriately, then provide reassuring feedback (abbreviated by 'smile').

  ```
  IF Navig(okay) THEN - smile -
  ```

- if an irrational navigation is diagnosed that started at point `?start` of the table of contents (TOC), then two pointers show the current position in the TOC and the `?start` position. In this case, the user can click the `?start` position to return to a 'useful' learning path.

  ```
  IF Navig(irrational,?start) THEN - noSmile -
                                   ``did you get lost by chance?''
                                   pointer(current) and pointer(?start)
  ```

Rules for Content Suggestions are needed because if the goal-level of mastery is not yet reached by the learner, then the presentation of appropriate content might help to improve. As opposed to the local feedback that corrects single problem solving steps, the global feedback described below prompts and supports the learner in activities such as reading, repeating, self-explaining, comparing, varying, information gathering that are known to improve learning, see, e.g., [11, 2, 24, 27, 12].

- If the last exercise was solved correctly and the concept in focus is mastered, then provide reassurance.

```
IF Known(A ?focus)                          THEN - smile -
```

A second rule belongs to a more elaborate strategy that assumes that even if the mastery-level is reached, more exercising can strengthen the mastery of the concept and the confidence for less confident learners. This rule requires that, if there is an exercise that is more difficult than those solved already, then present it.

```
IF Known(A ?focus)                          THEN - smile -
AND solution(?id correct)                        ``see more''
AND exerciseFor(?focus ?id)                      present(?exc1)
AND exerciseFor(?focus moreDiff(?id ?exc1))
AND notSeen(?exc1)
```

- If seenButUnknown(A) holds for the focus-concept, then present examples similar to the failed exercise of the focus-concept, unseen simpler exercises, and then again the incorrectly solved exercise of the focus-concept. This suggestion is made because A-mastery is the learning goal but not yet achieved, and therefore another exercise for the focus-concept should be offered to be solved. This exercise should be a bit simpler in order to keep the user's motivation up (in the proximal zone of development [30]). Then an example similar to the exercise should be shown for comparison. Finally, the originally failed exercises should be presented again.

```
IF SeenButUnknown(A ?focus)       - noSmile -
AND solution(?id incorrect) THEN  ``go deeper into the problem''
AND exerciseFor(?focus ?id)       exerciseFor(?focus lessDiff(?id ?exc1))
                                  exampleFor(?focus simTo(?id ?exm1))
                                  present(?exm1 ?exc1 ?id)
```

- If seenButUnknown(C) holds for the focus-concept, then show not yet sufficiently seen examples and counter-examples (if available) and ask for an explanation of the examples.

  This suggestion is made because C-mastery of the focus-concept is not yet achieved and therefore, this comprehension has to be supported. This is tried by showing more examples and prompting the learner to engage herself in self-explanation.

```
IF SeenButUnknown(C ?focus)        - noSmile -
AND exampleFor (?focus ?exm) THEN  ``please explain example''
AND NOT Seen(okay ?exm)            present(?exm)
```

- If seenButUnknown(C) holds for the focus-concept then prompt the learner to explain and vary an example of the focus-concept. This suggestion is made because C-mastery of the focus-concept is not yet achieved and therefore, this comprehension has to be supported. This is tried by showing more examples and prompting the learner to engage herself in self-explanation.

```
IF SeenButUnknown(C ?focus)        - noSmile -
AND exampleFor (?focus ?exm) THEN  ``please explain and vary example''
AND  Seen(okay ?exm)               present(?exm)
```

- if seenButUnknown(K) holds for the focus-concept, then the user should re-read the last page on which the concept has been elaborated. This suggestion is made because K-mastery (and also A- and C-mastery) is not yet achieved. Since A-mastery is the learning goal in the overall strategy, everything (reading, examples, and exercises) for the focus-concept, except any solved exercises, needs to be repeated.

```
IF seenButUnknown(K ?focus) THEN - noSmile -
                                 ``please return to last page''
                                  present(lastPage)
```

- if notSeenAndUnknown(K/C/A) holds for the focus-concept, then K/C/A-present, respectively. This suggestion is made because A-mastery is the learning goal and so content for every level not mastered yet is suggested again.

```
IF notSeenAndUnknown(K/C/A) THEN ``repeat''
                                  K/C/A-present(focus)
```

- if there is a missing K/C/A-prerequisite c (that is, a prerequisite concept that whose mastery level has not reached K/C/A), then K/C/A-present that prerequisite c according to what is missing for the mastery. No further substantiation needed because the failure is likely to be caused by the missing mastery of the prerequisites of the focus-concept.

```
IF missingPre(?c, ?level)   THEN -noSmile -
                                 ``missing prerequisite''
                                 ?level-present(?c)
```

- if an item ID has been seen sufficiently and more than twice, then do not present automatically ID again. Otherwise the motivation might drop. This selection is made in the suggestion generator.

5.2 Conflict Management

It is possible that conflicting suggestions occur on the SBB, in case a user action triggers several rules. For instance, when a new page is selected by the user, this can trigger a navigation suggestion as well as a conflicting concept presentation suggestion. If not all of the suggestions can be presented at the same time, then a rating has to indicate the priorities, and only the rules with the highest rating will generate their suggestions.

For the decision it makes a difference whether two suggestions are both of the same class, say navigation suggestion, or not. Therefore, the rules are classified as *navigation*, *content*, and *exercises* rules, currently. A conflict between two rules from the same class is resolved by a default resolution (e.g., more specific rules

have higher rating). A conflict between rules from different classes is resolved based on a particular conflict resolution strategy. The simplest strategy decides according to a fixed priority rating of the classes. If the highest rated rule cannot generate a suggestion, e.g., because there is no counter example, then the it gives up and the next best rated rule takes the turn.
For the implementation of a conflict resolution strategy in ActiveMath the Jess [15]-interface `strategy` and its method `compare` are employed. The `strategy` can be easily configured in the ActiveMath property list.

6 Implementation

The described mechanisms are implemented by a Jess-application. This section covers the interfaces to other components of ActiveMath as well as internal reasoning mechanisms. The suggestion mechanism runs in three phases. First, the evaluators communicate their results to a diagnosis component. This potentially triggers the inference of intermediate diagnoses/facts (1. phase). Then, suggestions are generated from the intermediate diagnoses and put on the SBB (2. phase). The third phase resolves existing conflicts and realizes the presentation of content if needed.
The next subsections go into more detail.

6.1 Interfaces

We need to consider several interfaces. First we have to look at the evaluators, because their diagnosis blackboard generates the diagnoses needed by suggestion mechanism to make the suggestions. Then, we will have a closer look at the suggestion mechanism's interface and the functionality it offers. Finally, we'll briefly discuss the presentation servlet.

6.1.1 Evaluator Interface

As the evaluators have to be notified about the user's actions their interface is based on methods designed to do this job. There are some central actions a user can perform in ActiveMath currently. Each is reported using a corresponding method:

`startPageBrowsed` the user has come to a new page.

`endPageBrowsed` the user has finished a page.

`startExercise` the user started an exercise.

`endExercise` the user finished an exercise.

`enableEyeTracker` the user is using the poor man's eyetracker

`elementHasBeenSeen` the user has seen an item on the current page (this is used only when the poor man's eyetracker is enabled).

This interface provides access the different evaluators, it will propagate the method calls to those evaluators registered for each action. For instance, the information in the `endExercise` method is only relevant to the exercise evaluator, so the other evaluators need not to be notified of this. In contrast to the `endExercise` method the output of `endPageBrowsed` is needed both, by the read and the navigation evaluator.
For a more technical description of these methods, see
`http://www.activemath.org/~ilo/java-doc/org/activemath/abstractcontent/AbstractUserModel.htm`.

6.1.2 The Suggestion Mechanism Interface

The suggestion mechanism has to be notified, when an evaluator delivers a diagnose. This is realized by the `addDiagnose` method, which will insert the diagnose into the suggestion blackboard. When the mechanism generates a suggestion, it notifies the servlet that deals with displaying and rendering. This servlet receives the suggestion that is to be presented by the method `getNextBestAdvice`.
Since the user can also actively request some advice, the servlet needs a possibility to tell suggestion mechanism to generate a suggestion. `requestNextBestAdvice` is the method to be called in this case. Summarizing, we have the following methods in the interface

`addDiagnose` to inform suggestion mechanism about new diagnoses

`getNextBestAdvice` to allow the presentation servlet to access the suggestion

`requestNextBestAdvice` to explicitely request a suggestion.

6.1.3 Presentation Interface

The presentation servlet is accessible for other modules via `HTTP`. It realizes the following functionalities

- visual notification that a new suggestion is available
- display of the suggestion text
- display of the suggestion content
- pass an active user request for the generation of a suggestion.

6.2 The Diagnosis Blackboard

There is one DBB per user. This allows to keep the Jess fact base clean and to speed up the suggestion generation. A DBB receives input from the evaluators, reasons about this input, and then transmits its diagnoses to the SBB as described in the following.
Once the evaluators have processed the raw input coming from the user-tracking servlets, they pass first diagnoses to the DBB. For instance, if the student finished an exercise, the `ExerciseEvaluator` is notified by the user-tracking device with

a call to `endExercise`, it performs some simple evaluations, and then inserts a new fact into the DBB, namely

```
(solution <exercise> <rating>),
```

where `<exercise>` is the exercise's ID and `<rating>` is one of the values `POOR`, `OK` or `GOOD`.
The DBB contains a Jess rule engine to reason about the incoming facts. Its rules infer the intermediate facts in the DBB. For instance, if the performance was rated `POOR`, the rule engine will try to find out why the student performed poorly. A reason could be that one of the prerequisites of this exercise was unknown to the student. In this case, a rule will fire and insert a new fact into the DBB fact base. The concrete fact is

```
(missing-prerequisite <dimension> <exercise> <item>),
```

where `<dimension>` is the target mastery level, `<exercise>` is the ID of the exercise for which the missing prerequisite was found, and `<item>` is the ID of the missing prerequisite. All the currently available DBB-facts can be found in table 1.

Technical Description The Java code in the DBB serves as a wrapper for the rule engine Jess, a Java Expert System Shell. In the following, we present selected rules that run on the DBB, namely those needed to infer that the student missed a prerequisite to solve an exercise.
To continue the example, the search for missing prerequisites starts, when the fact `(solution exercise POOR)` is added to the fact base.
This triggers a rule that queries the user model for the student's mastery values of the prerequisites of the exercise. Both, the concepts that the exercise is `for` as well as the concepts the exercise `depends-on`, are needed in the general diagnosis procedure. In our example, we only need the mastery values of concepts that the exercise depends on (these are precisely the prerequisites) to diagnose prerequisites that are insufficiently known.

Fact	Options	Interpretation
PRIMITIVE FACTS		
navigation	`OKAY`	student is aware of her position in the course
	`IRRATIONAL`, `start`	student navigates without following any dependencies or following the outline of the course since position `start`
seen	`item`, `NOTSEEN`	`item` was on the last page, but the student did not read it at all.
	`item`, `SHORT`	`item` was on the last page, and the student did not read it long enough.
	`item`, `OK`	`item` was on the last page, and the student has read it long enough.
	`item`, `LONG`	`item` was on the last page, and the student has read it far longer than the average reading time.
solution	`exercise`, `POOR`	the student performed poorly in an exercise `exercise`.
	`exercise`, `OK`	the student performed neither well nor poorly in an exercise `exercise`.
	`exercise`, `GOOD`	the student performed well in an exercise `exercise`.
INTERMEDIATE FACTS		
missing-prerequisite	`K`, `exercise`, `item`	`item` is a missing prerequisite for the student to solve `exercise` because she does not know it well enough.
	`C`, `exercise`, `item`	`item` is a missing prerequisite for the student to solve `exercise` because she does not truly understand it.
	`A`, `exercise`, `item`	`item` is a missing prerequisite for the student to solve `exercise` because she cannot apply it properly.
seen-but-unknown	`K. concept`	`concept` has been seen several times but the user still does not know it.
	`C. concept`	`concept` has been seen several times but the user still does not understand it.
	`A. concept`	`concept` has been seen several times but the user still cannot apply it.
not-seen-and-unknown	`item`	The user has finished with a page but its focus-concept hasn't been seen and is not even known.

Table 1: The DBB facts with their parameters

The JESS rule that collects the concepts the exercise depends on is given in the following.[5]

```
(defrule start-performance-diagnosis
 ;; IF
 ;; Student has solved exercise ?exc with success rating ?r
 (solution ?exc ?r)

 ;; AND
 ;; The reference to the user model is ?model
 (model ?model)

=>

 ;; THEN
 ;; Load the mastery-values of ?user for exercise ?exc was for
 ;; from ?model into the the fact base
 (get-for-what ?exc ?user ?model)

 ;; AND
 ;; Load the mastery-values of ?user for prerequisites of exercise ?exc
 ;; from ?model into the the fact base
 (get-prerequisites ?exc ?user ?model)
)
```

The rule calls the functions `get-prerequisites` and `get-for-what` which load the student's mastery values of concepts into the JESS fact base.
Specifically, `get-prerequisites` queries the knowledge base MBASE for the concepts `exercise` depends on, that is, its prerequisites, and inserts facts such as `(dependency ON MONOID-EXERCISE DEF-MONOID)` which expresses that the `MONOID-EXERCISE` needs the concept defined in `DEF-MONOID` as a prerequisite. `get-for-what` fetches the concepts `exercise` is for. We say that those concepts are the *goals* of the exercise.
Then, the DBB queries the user model for the user's mastery values and new facts such as `(mastery-value K DEF-MONOID 0.4)` are asserted. This fact states that the likelyhood that the learner `K`nows the concept Monoid is 0.4. Once the fact base contains the needed information about the student's mastery of the related concepts, the next rule that fires is `poor-perf-poor-application` that inserts new intermediate diagnoses into the DBB.

```
(defrule poor-perf-poor-application
 ;; IF
 ;; The student has given a POOR solution to exercise ?exc
 (solution ?exc POOR)

 ;; AND
```

[5]The syntax is simplified. ?xx indicates that ?xx is a variable, ;; declares comments.

```
 ;; exercise ?exc depends on concept ?id
 (dependency ON ?id ?exc)
 ;; AND
 ;; The application value for item ?id is low (below 0.3)
  (mastery-value A ?id ?val &: (< ?val 0.3))

=>

 ;; THEN
 ;; assert that the missing prerequisite item ?id  has been found for ?exc.
 ;; There was a lack at the application-level.
 (assert ("missingPrerequisite" "A" ?id ?exc))
)
```

Now those diagnoses can be processed by the knowledge sources that insert facts into the suggestion blackboard. The following rule performs such a transition to the SBB.

```
(defrule diagnosed-missing-prerequisite

  ;;NotSeenAndUnkown holds for the focus concept
  ;; Our missing prerequisite
  ("missingPrerequisite" "A" ?id ?exc)
  ;; The username
  (user ?user-name)
  ;; Reference to the suggestion blackboard.
  (nextbest ?sbb)

=>

  ;;Send a suggestion to a-present the ?id for ?user because it is a missing precondition
  (suggest ?user ?sbb ("A-present" ?id "You should have a look at some items that are
needed to solve this exercise which you are not comfortable enough with" "MISSING_PRECONDITION"))
)
```

6.3 The Suggestion Blackboard

The suggestion rules watch the diagnoses on the DBB and put new facts on the SBB. The rules compute the suggestions and their presentation via ActiveMath with a specialized course generator. Moreover, the SBB can resolve potential conflicts between suggestions.
Let us continue the example from the last section. Once the fact

```
  ("A-present" ?id "You should have a look at some items that are
 needed to solve this exercise which you are not comfortable enough
                   with" "MISSING-PRECONDITION")
```

has been put onto the SBB, a suggestion rule fires that calls presentation functions for content generation:

```
(defrule do-a-present
 ;;IF
 ;;there is a diagnose yielding the need to a-present ?id
 ("A-Present" ?id ?message ?type)

 ;;AND
 ;;The Nextbest object is ?nextbest. This is needed to pass the resulting
 ;; suggestion.
 (nextbest ?nextbest)

 ;;AND
 ;;the username is ?user
 (user-name ?user)

=>

 ;;THEN

 ;;set the correct presentation scenario
 (bind ?scenario "A-present")

 ;;create a nextbest presentation generator for the item ?what,using the
scenario ?scenario
 (bind ?nextbest-pp (new NextBestPP ?id ?user ?scenario ))

 ;;run it
 (call ?nextbest-pp run)

 ;;get the generated document
 (bind ?document (call ?nextbestb-pp getStaticBook))

 ;;generate the Suggestion Object
 (bind ?suggestion (call ?nextbest createSuggestion))

 ;;set the expression of the little face
 (call ?suggestion setMood "unhappy")

 ;;set the message that should be displayed
 (call ?suggestion setMessage ?message)

 ;;set the ?document as content
 (call ?suggestion setPresentation ?document)

 ;;set the type of the suggestion to be ?type
 (call ?suggestion setType ?type))

 ;;tell nextbest to send the suggestion to the presentation servlet.
```

```
 (call ?nextbest makeSuggestion ?suggestion)
)
```

When the generation process is finished, suggestion mechanism's graphical user interface displays a text saying <"You should have a look at some items that are needed to solve this exercise which you are not comfortable enough with"> followed by a generated `HTML` link to the suggestion's content.

Technical Description The SBB is implemented using a JESS engine wrapped by Java code that controls the access. The interface of the SBB is the following.

```
interface AbstractNextBestCentral
{
  getNextBestAdvice(id, user);

  createNextBestFor(user);

  requestNextBestAdvice(user, topic);

  addDiagnose(user, factname, args);

  public void suggestionFollowed(String user, int suggestion_type);

  public void suggestionIgnored(String user, int suggestion_type);
}
```

7 Design of the Suggestion User Interface

The primary user interface of the global suggestion mechanism consists of a face/companion with a variety of possible types of states. This face moves regularly even if the user does well and there is no new suggestion. This is implemented in order to prevent the user wondering about the mechanism being stuck. This face informs the user whether the system wants to offer a suggestion or not. If not, it reassures the learner by looking happily.
When new coaching can be suggested by the system, the face changes and the student can now choose to follow or to ignore the suggestion. The offer expires after some time and then disappears. The suggestions are personalized in that the messages address the learner by the first name she has introduced at login.
To follow the offer, the learner presses a button for requesting the detailed suggestions. Then, a new browser window pops up that contains the generated content.
The user interface also includes buttons for requesting more specific information and content as described in 7.

Active vs. Passive Feedback Several empirical investigations [10, 23] found that feedback on demand (we shall call it *active* feedback here) typically yields longer learning time than automatic feedback. However, feedback on demand yields deeper knowledge, i.e., better transfer ability and more self-regulation, -monitoring, and motivation.
Therefore, the global feedback GUI in ActiveMath has two different components: general feedback is provided that the learner receives rather passively as well as buttons for more specialized requests and for more specific feedback on demand. ActiveMath realizes the active feedback by buttons that the user can click on, whereas passive feedback means the user does not request the information but receives it automatically in the NextBest window through an initiative of the system.
777user chooses time and type of information and therefore needs to think about her learning requirements more actively in case she requests more specific information. Not invasisve in both cases
In the passive mode, if the user does not click a specific button (i.e., she is passive), there are two alternatives. Either the system presents the highest rated amongst all suggestions to the user or a dialog is started to find out what would be the appropriate and desired help. The latter way will be explored in the future only.
The first button for a specific global feedback request implemented in ActiveMath is the giveMeMore button. The learner can use it to ask for more exercises, similar to one she worked on recently, or for a similar example.
We plan to design other buttons with a telling name for each of the suggestion mechanisms , e.g., whereAmI?, DoIKnow?, CorrectSolution? LookUpYourUserModel!, WhichPrerequisites?, explainMore! This is a first step towards a tutorial dialog.

8 Future Work

Some future work has been mentioned already. In this section, we describe the plans more systematically. In particular, we mention the extensions and improvements of the diagnosis and suggestion mechanisms including the consideration of the learner's motivation and different suggestion strategies for different learning goal-levels and for different learning scenarios. Moreover, we discuss the empirical investigations that will be conducted in cooperation with psychologists.

8.1 More Elaborate Diagnoses

The duration of reading is one of the data from which diagnoses can be inferred. However, an individual learner may have an individual average reading speed. Therefore, the *optimal reading* time has to be determined for each learner during the time the student uses ActiveMath in order for the reading evaluations to become more precise.
We plan to develop the more specific diagnostic features of a proof attempt with the proof planner Omega and with Computer Algebra systems.

- More elaborate `A-mastery` values for all the concepts involved in solving an exercise might be determined from the user's detailed proof attempt and from the difficulty of the exercise. This requires to model the problem solving process, e.g., by a dynamic Bayesian Net which contains the concepts and methods involved in the exercise as nodes and its various results as symptom nodes.

- From such a dynamic Bayesian Net the missing prerequisites may be diagnosed more properly.

- The ability-value of using Omega or a particular CAS will be updated (e.g., from the number of syntax errors).

- An update of the `guess` and `slip` values could be learned.

- If the author provides a tuple (typicalError, teacherDiagnosis, teacherFeedback, teacherReaction), then an evaluator can pass the (teacherDiagnosis, teacherFeedback, teacherReaction) tuple to the diagnosis blackboard in case the user's input matches typicalError.

With more effort for authors, a more detailed diagnosis of the user's solution for multiple choice questions could be provided in the future. For instance, (1) if the author of the alternative answers in MCQs follows a particular Boolean schema with mastery variables for several concepts, then this schema can be used for an automatic diagnosis that computes a mastery diagnosis for several involved concepts rather than for one concept only. Such a schema consists of Boolean combinations of the mastery[6] of the involved concepts for each answer. (2) If the author provided the information `(teacherDiagnosis, teacherAction)` for each answer of an MCQ, then the proxy can pass this information to the diagnosis blackboard.

8.2 More Suggestions

Our next investigations will focus on the design of a personalized and gender-specific verbalization of the feedback and on the design of new validated rules as well as on other kinds of global feedback including

- exploration tasks with system support (dependencies, more information, e.g.,historic information)

- meta-reasoning exercises, for instance
  - structure a task or solution
  - self-explaine
  - exercise/example variation
  - devise examples supported by the system
  - compare solutions supported by the system

[6] k-,c-,or a-mastery

- draw concept map

- feedback for improving motivation.

8.3 Reactions to the Learner's Motivational State

Few tutor systems take the learner's motivational state into consideration currently. For instance, WEST [9] follows the principle *Do not tutor on two succcessive moves, no matter what* which tries to avoid demotivation by too much interference. SOLA [4] structures content according to the learner's style (holistic vs. serial and low vs. high confidence). *Less confident students receive content focussed on one major topic, whereas highly confident students may focus on several topics; a serialist will rather focus on one topic whereas a holist could pursue several topics at the same time.* For motivational reasons, MENO [32] delivers a particularly positive feedback, when a student eventually succeeds who has previously failed in order to restore her confidence. This is – as a tendency – particularly important for female students.
Those and similar principles such as *Do not present new material, when the learner is engaged already*, *Once a less confident learner does a task well, activities that are likely to succeed should be presented by the system*, and *Highly motivated students deserve praise even if their performance is not optimal* [14] have to be tested empirically and will be formalized in ACTIVEMATH' suggestion rules.
Several techniques to enhance the learner's motivation are known. For instance, curiosity can be stimulated if necessary for increasing the learner's motivation. Perceptual curiosity techniques deliver audio and visual efects such as highlighting a detail, i.e., focussing the learner's attention to the detail. Also bright colors and animations focus attention. Cognitive curiosity can be stimulated by puzzling questions, incomplete information, relevant analogies, counter-examples, unexpected or paradoxical questions, or a topic that requires an explanation. In order to reduce the learner's frustration, forthcoming difficulties should be signalled. An anti-help suggestion button should allow the student to control the provision of suggestions.

Diagnosis of Motivational State In order to be able to react to the learner's motivational state it has to be diagnosed in the first place. A reliable indication for the student's intrinsic motivation is the student's effort rather than her performance [26]. A motivation diagnosis can also be obtained by self-evaluation questions combined with inferences. For instance, low confidence is inferred, if the student repeatedly asks for help and high confidence is inferred in case of total absence of help requests. The absence of help requests may indicate that problem is too easy. Motivated students may use hints or not even request help. Students who ask for help without exploring the hints provided by the system are likely to be demotivated.

8.4 Different Pedagogical Strategies

The computation, rating, and eventual choice of a suggestion depends on the pedagogical strategy whose choice, in turn, may depend on the user's ability, goals, scenario, and motivation. For instance, a good learner will more likely succeed with a few hints and active learning and benefit from it, whereas a weak learner has to be corrected immediately and strongly guided.
A strategy (represented as a set of rules) determines, e.g., when, how often, and which types of exercises and examples to present, how many exercises and examples to suggest, the difficulty of exercises, in which sequence items will be shown, which feedback will be given.
There are the classical pedagogical strategies Didactic, Socratic, Inquiry, and Discovery learning [17]. Another dimension for devising strategies is its learning goal-level (K/C/A/T). For instance, [25] suggests different reactions according to different targeted mastery levels.
In the tutoring literature, mostly strategies for *local* feedback are described. And even if these strategies are relatively general, the realization of a strategy is pretty specific for a particular learning area so far. For instance, the tutor system Miss Lindquist [16] has four strategies for teaching problem solving: concrete-articulation, decompose-into-subgoals, translate-representation from text into computation and abstract formula and vice versa, and present model solutions, The concrete-articulation strategy for solving algebra word problems consists of the rules:

- if wrong answer, then present simpler exercise about a sub-formula which requires a number as the answer
- then ask to provide the computation steps
- then present variablized steps
- then present full formula.

In ActiveMath, the first strategies will be related to the four learning goal-levels: knowledge, comprehension, application, and transfer. The strategy formed by the rules suggested in §5.1 is a strategy for A-level learning with a relatively strong guidance.

8.5 Empirical Questions

Most importantly, the diagnosis and suggestion mechanisms have to be evaluated empirically with actual users. Then, the mechanisms will be improved according to the test results.
For some of the specifications there exist no empirical basis yet. For instance, the following questions could be investigated empirically:

- what is the individual optimal reading time for instructional items?
- how can motivational features diagnosed reliably?

- when to provide immediate and when delayed feedback?
- when to repeat a typical example?
- when to try an exercise again?
- when to present a counter example?
- when to ask for reasons for a step (stimulate meta reasoning)?
- how should the global suggestion rules be rated?
- should the correct solution be shown in the global feedback?
- when to show seen and when show unseen examples and exercises

Design The design of the multi-modal suggestions has to be investigated. Among others, design questions comprise the following: how to personalize a suggestion, which number and position of feedback and suggestion windows is appropriate for learning? Should the content be presented in a separate window? Should an example solution or a correct exercise solution be shown next to a faulty solution of a failed exercise? Moreover, we need to experiment with the buttons in order to find out about their usability.

Diagnoses for Reading Speed Apart from recognizing the optimal individual average reading speed there will be more subtle differences between reading different types of instructional elements. For instance, reading time for examples should include time for self-explanation for a better comprehension. Note also that the reading speed for formulas may individually differ from the reading time for normal text.
ActiveMath can serve as a tool for controled experiments determining dependencies of user characteristics and for benefits of many other instructional decisions.

9 Conclusion

The web-based learning environment ActiveMath presents content, worked-out examples and exercises to the student rather than exercises only or examples only. This presentation may include incomplete examples, elaborations and examples with built-in questions, etc. An on-line demo of ActiveMath is available at `http://www.activemath.org`.
This article mainly describes our research on global feedback in ActiveMath and beyond. It also includes the distinction between local and global feedback as well as the separation of diagnosis and global suggestion mechanisms by an architecture with two blackboards. The suggestion mechanism is implemented in Java.

9.1 Acknowledgement

We thank the ActiveMath group, in particular, Carsten Ullich and and Sabine Hauser for the involved discussions as well as Bernhard Jacobs for his supply of empirical knowledge.

References

[1] V. Aleven and K.R. Koedinger. Limitations of student control: Do student know when they need help? In G. Gauthier, C. Frasson, and K. VanLehn, editors, *International Conference on Intelligent Tutoring Systems, ITS 2000*, pages 292–303. Springer-Verlag, 2000.

[2] V. Aleven, K.R. Koedinger, and K. Cross. Tutoring answer explanation fosters learning with understanding. In S.P. Lajoie and M. Vivet, editors, *Artificial Intelligence in Education*, pages 199–206. IOS Press, 1999.

[3] R.G. Almond, L.S. Steinberg, and R.J. Mislevy. A sample assessment using the four process framework. Educational Testing Service, 1999. http://www.education.umd.edu/EDMS/mislevy/papers/FourProcess.pdf.

[4] F. Arshad. *The Design of Knowledge-Based Advisors for Learning.* PhD thesis, School of Education, University of Leeds, 1990.

[5] A.T.Corbett, K.R. Koedinger, and J.R. Anderson. Intelligent tutoring systems. In Landauer Helander, M. G., T.K., and P.V. Prabhu, editors, *Handbook of Human-Computer Interaction*, pages 849–874. Elsevier Science, The Netherlands, 1997.

[6] B.S. Bloom, editor. *Taxonomy of Educational Objectives: The Classification of Educational Goals: Handbook I, Cognitive Domain.* Longmans, Green, Totonto, 1956.

[7] J. Buedenbender, E. Andres, Adrian Frischauf, G. Goguadze, P. Libbrecht, E. Melis, and C. Ullrich. Using computer algebra systems as cognitive tools. In S.A. Cerri, G. Gouarderes, and F. Paraguacu, editors, *6th International Conference on Intelligent Tutor Systems (ITS-2002)*, number 2363 in Lecture Notes in Computer Science, pages 802–810. Springer-Verlag, 2002.

[8] A. Bunt, C. Conati, M. Huggett, and Kasia Muldner. On improving the effectiveness of open learning environments through tailored support for exploration. In J.D. Moore, C.L. Redfield, and W.L. Johnson, editors, *International Conference on Artificial Intelligence and Education*, pages 365–376. IOS Press, 2001.

[9] R.R. Burton and J.S. Brown. An investigation of computer coaching for informal learning activities. In D. Sleeman and J. Brown, editors, *Intelligent Tutor Systems.* Academic Press, 1982.

[10] D. Butler and P. Winne. Feedback and self-regulated learning: A theoretical synthesis. *Review of Educational Research*, 65(3):245–281, 1995.

[11] M.T.H. Chi, N. De Leeuw, M. Chiu, and C. Lavancher. Eliciting self-explanations improves understanding. *Cognitive Science*, 18:439–477, 1994.

[12] M.T.H. Chi, S.A. Siler, H. Jeong, T. Yamauchi, and R.G. Hausmann. Learning from tutoring. *Cognitive Science*, 25:471–533, 2001.

[13] C. Conati and K. VanLehn. Teaching meta-cognitive skills: Implementation and evaluation of a tutoring system to guide self-explanation while learning from examples. In S.P. Lajoie and M. Vivet, editors, *Artificial Intelligence in Education*, pages 297–304. IOS Press, 1999.

[14] T. del Soldato. Detecting and reacting to the learner's motivational state. In D. Biermann, J. Breuker, and J. Sandberg, editors, *Proceedings of AIED 89*, pages 567–574, Amsterdam, Springfield, Tokyo, 1989. IOS.

[15] E. Friedman-Hill. *Jess, the Java Expert System Shell*, 2000. `http:/herzberg.ca.sandia.gov/jess/`.

[16] N.T. Heffernan. *Intelligent Tutoring Systems have Forgotten the Tutor: Adding a Cognitive Model of Human Tutors*. PhD thesis, School of Computer Science, Carnegie Mellon University, 2001.

[17] M. Keegan. Optimizing the instructional moment. *Educational Technology*, pages 17–22, April 1993.

[18] H. Mandl, H. Gruber, and A. Renkl. *Enzyklopädie der Psychologie*, volume 4, chapter Lernen und Lehren mit dem Computer, pages 436–467. Hogrefe, 1997.

[19] M. Mayo, A. Mitrovich, and J. McKenzie. An intelligent tutoring system for capitalisation and punctuation. In M. Kinshuk, C. Jesshope, and T. Okamoto, editors, *Adavanced Learning Technology: Design and Development Issues*, pages 151–154. IEEE Computer Science, 2000.

[20] E. Melis, J. Buedenbender E. Andres, Adrian Frischauf, G. Goguadse, P. Libbrecht, M. Pollet, and C. Ullrich. ActiveMath: A generic and adaptive web-based learning environment. *Artificial Intelligence and Education*, 12(4):385–407, winter 2001.

[21] A. Mitrovich and S. Ohlsson. Evaluationof a constraint-based tutor for database language. *International Journal on Artificial Intelligence in Education*, 10(3-4):238–256, 1999.

[22] T. Murray, J. Piemonte, S. Khan, T. Shen, and C. Condit. Evaluating the need for intelligence in an adaptive hypermedia system. In G. Gauthier, Claude Frasson, and K. VanLehn, editors, *Intelligent Tutoring Systems, Proceedings of the 5th International Conference ITS-2000*, volume 1839 of *LNCS*, pages 373–382. Springer-Verlag, 2000.

[23] B.J. Reiser, W.A. Copen, M. Ranney, A. Hamid, and D.Y. Kimberg. Cognitive and motivational consequences of tutoring and discovery learning. Technical Report 54, The Institute for the Learning Sciences, Northwestern University, 1994.

[24] A. Renkl. Worked-out examples: Instructional explanations support learning by self-explanations. *Learning and Instruction*, 2001.

[25] V.J. Shute. SMART: Student modeling approach for responsive tutoring. *User Modeling and User-Adapted Interaction*, 5:1–44, 1995.

[26] Teresa Del Soldato. Detecting and reacting to the learner's motivational state. In , editor, *Proceedings of the International Conference on Intelligent Tutoring Systems*, volume 608 of *Lecture Notes in Computer Science*, pages 567–574. Springer, 1992.

[27] R. Stark. Instruktionale Effekte beim Lernen mit unvollständigen Lösungen. Forschungsberichte 117, Ludwigs-Maximilian-Universität, München, Lehrstuhl für Empirische Pädagogik und Pädagogische Psychologie, 2000.

[28] C. Ullrich and E. Melis. The poor man's eyetracker in ActiveMath. In *Proceedings of the World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education (eLearn-2002).*

[29] K. VanLehn. Bayesian student modeling, user interfaces and feedback: A sensitivity analysis. *International Journal of Artificial Intelligence in Education*, 12:.., 2001.

[30] L. Vygotsky. *The Development of Higher Psychological Processes*. Harvard University Press, Cambridge, 1978.

[31] B.Y. White and T.A. Shimoda. Enabling students to construct theories of collaborative inquiry and reflective learning: Computer support for metacognitive development. *International Journal of Artificial Intelligence in Education*, 10:151–182, 1999.

[32] B.P. Woolf. *Context-Depending Planning in a Machine Tutor*. PhD thesis, Department of Computer and Information Science, University of Massachusets, 1984.

Evaluators and Suggestion Mechanisms for ActiveMath

Erica Melis and Eric Andrès

RR-02-01
Research Report