Exercise generation by group models for autonomous web-based learning

Michael Sonntag
Institute for Information Processing and Microprocessor Technology,
Johannes Kepler University Linz
Altenbergerstr. 69, 4040 Linz, Austria
e-mail: sonntag@fim.uni-linz.ac.at

***Abstract*— Creating exercises for learners requires significant time. This is one reason, beside difficulties of discussing individualized tasks in a classroom setting, why often only few exercises are created and posed to all learners alike. In web-based autonomous learning the setting problem is removed, while simultaneously demand for individualized exercises, comparable to adaptive learning material, increases.**

A model is proposed for assembling exercises from independent elements and creating an exemplary solution alongside. Here the cold-start problem is especially problematic, as opposed to learning material no sensible default-view exists. This difficulty can be reduced by integrating a group model, i.e. the history of results or actions of other learners.

This paper presents an implementation of the model for generating cases for learning in the legal area. The web-based user interface of such an online system is very important to render comparing the learners' solution with the exemplary one simple, allowing correction by peers as well. The current implementation state as well as planned extensions is described.

Adaptivity, web-based learning, case generation

I. INTRODUCTION

Learning is an individual activity of each person. Yet typically tasks and examples as basis for learning are given to everyone identically. This is especially prevalent at universities and schools, where online "classes" consist of many students and few teachers: the higher the ratio of learners per teachers is, the less individualized teaching, and therefore learning too, becomes.

One possibility to reduce this problem is adapting learning content to the individual person, called adaptivity [2]. Many approaches have been examined or are currently under development or in testing for adapting the learning material or its delivery: showing or hiding specific elements for which prerequisites have not been completed or which are already known well or might be especially interesting for this particular learner. The difficulty is, that in this area only "negative" adaptation is possible. Material which does exist can be restructured or removed, but no new content can be generated automatically if a lack is discovered by the system.

However, adaptivity is also possible in the area of exercises, where the focus lies on the production of individualized content, producing different tasks for every student to work on to test or extend their knowledge. While in conventional teaching in a class setting every-one must receive the same task so discussions of the result can be understood by everyone, this does not apply to E-Learning where each learner works individually, usually through a web interface. Examples for this are adaptive tests, quizzes, etc. Regarding exercises problems are similar, but in contrast to the learning material itself also new elements may be generated at least in some cases; e.g. when training "adding small numbers" the example might be parameterized to allow arbitrary numbers. Contrastingly the explanation how to add two numbers suffers from the problem described above. As exercise, two numbers may be generated randomly and must be added by the learner. The computer can easily verify whether the answer is correct according to its own internal representation of the formula and perhaps even identify common pitfalls (and explain them with other pre-defined textual content).

While parameterizing examples is useful, it still suffers from the pedagogical shortcoming that for best learning results the task difficulty should match the learner's knowledge. Only then full adaptivity has been reached: selection of an example, variation of its content, and a matching level of difficulty in both. In the mathematical area this could mean automatic selection of count and "difficulty" of operators present in an equation.

Adaptivity usually suffers from the important cold start problem. At the beginning of a course no formal and explicit information about knowledge, preferences etc. of the participants exists. For instance questionnaires or tests may be used for obtaining such data, but are rather disliked by users. Another option is transferring a general competency level ("good/average/weak student") from previous courses or other information sources. This difficulty occurs for each learner separately, even if others have already much experience in this course, as adaptivity is based on an individualized view. In web-based learning this is especially common, as asynchronicity is one of its key advantages.

Reducing this problem is possible through basing adaptivity in addition on a "group model". While for the first user the cold start problem still exists, users entering the course later will find a generalized base for adaptivity based on their predecessors. Obviously, this will not match them perfectly, but it does model specific aspects of a course applying to all learners of a course alike:

- Weaknesses in the material or the presentation by the teacher, resulting in general difficulties understanding a specific area
- Identification of sections which are hard and should therefore be practiced extensively
- General progress of learners, allowing to give students hints whether they are falling behind
- Aggregated proficiency of the class, showing the teacher the learners "readiness" for closing the class or a final examination

While models of individual users cannot be reused between classes, a group model does apply to different learners in the same course, further reducing the cold start problem in subsequent classes.

In this paper a system for generating exercises for individual learning by students based on a combination of a user and a group model is described. The next section describes the approach generically and presents an example where the system has been implemented and is currently undergoing evaluation in a real course. Ideas for further work and conclusions follow a section on related work at the end.

II. GENERATING EXERCISES BASED ON MODELS OF SINGLE USERS AND GROUPS

The basic approach adopted here is closely related to Intelligent Tutoring Systems [5] and can be described as follows: all exercises consist of elements which are assigned an exemplary solution and a difficulty label. The latter metadata is derived from the user and the group models as described below. These elements are then sorted according to their difficulty and a function selects several of them. Finally they are combined to form the complete task for the learner. In parallel to the generation of the task its solution is assembled to provide feedback to learners.

A. Exercise selection based on the individual learning history of a learner

To select or assemble exercises based on this approach, some prerequisites must be met. Firstly, elements should be freely combinable, at least generally. This means, they are independent and selecting one does not preclude or require selecting others. If this cannot be guaranteed fully, the approach may still be possible but must take place stepwise, verifying after each step whether a complete task can still be generated, respectively whether the current selection set remains consistent.

Secondly, every element must be annotated with a difficulty level for the specific user (not generally for all users!) the task is being generated for. This includes elements not yet seen or worked on by the user. For those a default value, perhaps set by the designer, may be useful. This is a significant problem, especially for areas where learners will complete only few larger exercises as then little basic information exists. To reduce especially this difficulty a group model should be integrated.

Thirdly, difficulty should be additive. If e.g. four instead of two problems are selected, the resulting task should approximately double in difficulty. If not, task generation is still possible but the difficulty must be modeled separately and it may be impossible to generate tasks of a specific target level.

Finally, the result of the generated exercise must be calculable through the problem selection. If no exemplary solution can be shown to users, such a system would not be very useful as a human teacher would have to correct the learners' solutions. This is undesirable in computer education and even more difficult than conventional teaching as each exercise would be different.

Elements are selected for inclusion from the sorted list randomly. This is important so learners actually receive individual cases: an incentive for discussing them between learners and preventing copying solutions. To integrate the difficulty and generate "easy" or "difficult" exercises, the distribution of the selection must be modified from a flat uniform random function to a weighted one. Its shape can integrate a general difficulty level: if didactics requires generating a simple or complex task, problems should be selected predominantly from the matching end of the list.

The probability can for example be linearly distributed (see Fig. 1: lines 1 and 3 for generating easy respectively difficult tasks, and line 2 for a completely random selection). Based on experiences from extensive testing it became apparent, that these linear distributions result in quite good coverage of the problem area, but simultaneously exhibit relatively little concentration on problematic areas. So if more focusing is desired, exponential probability (see lines 4 and 5) should be considered. On the other hand this may lead to an incorrect focus. If the first task is marked as difficult for a user, regardless of the reason, only very similar tasks will be created from then on, potentially leaving out important other areas which might then not be tested at all. Balancing this probability curve is therefore an important and pedagogical decision and depends on the subject area and the variability of available elements.

B. Integrating group learning history

When no difficulty level for an element for a certain user can be ascertained, the group history becomes important. If other users have already worked on it in exercises they completed, a general difficulty level exists. It measures how complicated the element is or how much/little the typical learner knows about it and is therefore a good first approximation.

Another advantage of the group model is, that learners rarely take the same course twice, so the detailed information on their knowledge is largely useless after its successful completion. Through the group model aggregated data can be transferred to the next batch of learners and improves continuously. Care must be taken, however, when the course content changes: whether the group model is still at least partially (recalculation from individual results) valid, or whether it must be restarted from scratch.

The integration of the group model into the element selection process could be used as a backup only, i.e. when no individual assessment exists yet, like for the first exercises (cold start problem). However, it can be useful always, as e.g. a good result in a difficult area might have been just single a lucky guess. Continuous integration, although to a lesser degree, ensures that problematic areas are covered in detail even for knowledgeable students, improving the validity of the individual user models.

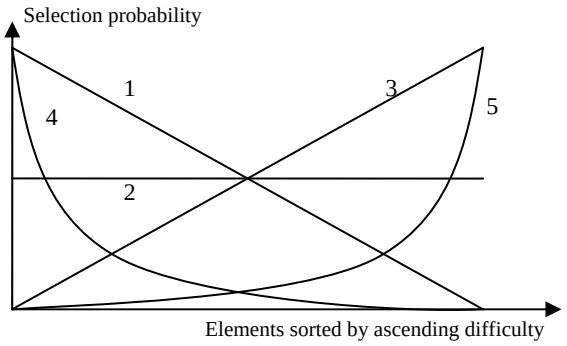

Figure 1. Selection probability for elements

The danger of incorrect focusing is larger here as compared to using a single user model only. A self-reinforcing cycle occurs faster because of the larger number of tasks generated and the fact, that examples are often generated in batches. E.g. at the course start all students will more or less simultaneously generate their first task with the same group model. So if these are very similar (strong focusing; see Fig. 1 lines 4 and 5), after completion the group model will cover only a small part of the element space, but this quite well. Therefore this model's influence should be less compared to the user model. An exception is the start of a new, and therefore empty, user model: the results from the first tasks allow only a rough assessment and so the group model (based on a larger number of feedback circles) should take precedence there.

$$D = F * D_U * (1 + \frac{SD_G - SD_U}{SD_G + SD_U}) + (1 - F) * D_G * (1 + \frac{SD_U - SD_G}{SD_G + SD_U}) \quad (1)$$

One possibility to achieve this is integrating not only an average difficulty level (D_U and D_G for user and group difficulty; value between 0.0 and 1.0), but also its standard deviation (SD_U and SD_G). The difference of these two deviations can be used for weighting the values as for example in (1), resulting in a total difficulty level D between 0.0 (easy) and 1.0 (very difficult), taking into account the "certainty" of the models. The influence of the group model can be adjusted with the factor F. A value of 0.5 results in both user and group model having the same weight, while 1.0 or 0.0 would mean that the group, respectively the user, model is ignored completely.

C. *Overall difficulty*

This approach allows adjusting the resulting difficulty of the exercises generated as well: the number of elements combined into an exercise determines its difficulty (see additivity of difficulty above). Obviously, a lower bound is necessary to avoid creating trivial tasks. But an upper bound is required as well to prevent overly complex exercises and reduce the probability of two elements producing unforeseen consequences when occurring together, e.g. when elements cannot be guaranteed to be always independent. Similarly didactic decisions can be incorporated, like defining a difficulty curve; for instance simple at the start and a constant higher level later. Even meta-adaptivity is possible, determining the number of problems from the (lack of) success at previous tasks.

D. *Self-assessment as feedback cycle*

This method of individualized exercises requires a correction element in the feedback cycle, as just generating problems, perhaps accompanied by an exemplary solution, cannot update the user model. Typically correction by an expert (teacher, coach etc.) is not an option, so only self- or peer-assessment can be expected. In both instances a computer-generated solution is a prerequisite and support for the correction is necessary. The other option for the feedback cycle would be observing the user's behavior. However, here no relevant action to monitor occurs. Users read the exemplary solution and compare it mentally with their own. No indicative action takes place. As explicitly asking users is typically disliked by them (additional work they see little reason for), the user interface is very important here. It must be as simple and fast as possible to perform and enter this assessment, or users will evade it.

III. EXEMPLARY IMPLEMENTATION: LEGAL CASE GENERATOR

Creating legal cases for students to solve requires a lot of work, so typically only few are developed and given to all learners as an exercise alike. This can be ameliorated through a generator for producing individualized cases for learning on demand based on user and group models. The problem space, an international proceeding regarding the ownership of domain names (UDRP, [3]), has been modeled hierarchically in an ontology. The generator adaptively composes an exercise, a case plus its exemplary solution, from several small individual elements; in this case text fragments. Its inner working has been described in [6] and is therefore omitted here.

Regarding the requirements defined above, the following difficulties can be identified for this case study: independence of elements does not exist fully, as not every text fragments may occur with every other in one case. For instance, a domain holder cannot request two different amounts of money simultaneously. Because of this, generation must employ a backtracking algorithm. Whenever a new element is selected for inclusion, it is verified whether the resulting case is still valid. If not, it is removed from both the case and the list of available elements and another one is chosen from the latter. The second requirement of full difficulty annotation is fulfilled through the user and group models. Additionally, as the text fragments are categorized in a hierarchy, generalization is possible (see below). The third element, additive difficulty, applies too. As each text fragment is an independent problem with a separate solution the overall difficulty of a case is the sum of the difficulties of the elements it consists of.

The generator was implemented in Java and was integrated into the learning platform Sakai as a "tool".

A. *Use case: Learners*

Usage by learners is planned as follows:

1. Users initiate case generation based on the user and group model through the web interface. Actual generation of the case is delayed as long as possible (e.g. no automatic generation of the first case in advance) to take advantage of the group model.
2. The user reads the case as a complete single text (which parts are separate elements is not disclosed) and can solve it directly - or copy it to another location for offline solving. Whether the solution developed afterwards consists of full sentences or just a few words is up to the learner.
3. The solution is entered/copied into the platform and stored. Later changes are not allowed, as this would require retroactively modifying user and group models. Personal annotations are a possibility to ameliorate this restriction.
4. The exemplary solution is made available to the user, who compares it to his own solution. He may align both texts through drag&drop (see below) and assigns correctness and completeness values for each constituent problem element.

5. The user can now generate a new case. Alternatively she can review old cases, seeing her own and the generated solution as well as her markings and the alignment.

B. User and group models

Both user and group models are overlays of the problem structure. This means, for every problem class (from which text fragments may be selected) a representation exists in every user model and once in the global group model. For each class correctness and completeness in the form of mean value and standard deviation is stored. Correctness represents the absence of errors in the solutions created by the learners. It is used to calculate the difficulty according to the proposition that a class is the more difficult, the less elements from it have been solved correctly in the past. This is not necessarily equivalent, as e.g. the duration required for solving is not taken into account. A learner might always produce perfect results, but for some classes it might take much longer → Therefore these are more difficult. However, time is a very difficult aspect to measure in web applications, as students might read the case, work on other things and then solve it, work on it offline etc., rendering any measurement suspect. Completeness is not used in the selection algorithm currently, but is requested for investigation as an alternative or future enhancement of the standard deviation.

Because of the structural equality between problem space and user/group model, the difficulty of a text element for a certain user can be derived as follows: from the text element the class it belongs to is looked up. For this class the data in both the user and group models are retrieved. From these then the final difficulty is calculated (see (1) above). If for the class in question no matching value is present in the user or group model, generalization takes place.

Generalization is performed by aggregating the values of all child classes and assigning the result temporarily to the parent class, i.e. difficulties of detail classes determine the difficulty of their general parent class. In the current version there is no "attenuation", i.e. the mean value of all child classes is the derived value of the parent class. A more complex formula would be possible, taking into account e.g. the increasing uncertainty of multi-level generalization or the coverage of the children (like reducing completeness for children from which few/no elements had been included in exercises yet or artificially increasing their standard deviation).

A by-product of generalization is a kind of overall self-assessment of the learner. At the top of the class hierarchy a value emerges describing the whole tree below, i.e. correctness and completeness of solving all kinds of problems in this subject area. While obviously this cannot be used for assigning marks, it provides an approximate general assessment for the learner and allows comparisons between them, in this way showing a rough measure of knowledge as compared to them.

C. Web user interface

E-Learning typically takes place over the web. So to prevent media breaks, presentation of the tasks, solving them, and their correction should take place there too. While this seems to be simple, the approach described here requires an elaborate interface as exercises and solutions consist of several independent elements. Each of the text fragments should be assessed separately, and ideally the whole exercise in addition.

Consequently a different presentation of the exercise at various stages of its lifecycle follows. While at first it is shown only as a composite (full case in a single paragraph; theoretically sentences from several fragment might be interleaved), for correction the individual parts must be identified and separated from their integration (a single paragraph per fragment) so their independent marking becomes possible. Obviously, this can only be done for the task itself and the generated solution (for both of which a formal model exists), but not the learner's solution.

It may be didactically useful to allow learners to align their solution with the exemplary one, e.g. by matching segments of their solution to the corresponding elements of the exemplary one. Drag&drop is a useful metaphor here. Users mark a part of their result and drag it to the area showing the matching part of the task or the exemplary solution. Additionally they assign each element (on the exemplary solution side, not on theirs) a value for correctness of the answer. This can then be used to update user and group models. Rendering options for this value are sliders (adjusting correctness), text fields (percentage entry; this one is used in the evaluation), or radio buttons (selecting a rough correctness level).

The user interface of the implementation is exemplified here by the life cycle of a learning case. First a new case is generated and shown for solving (see Fig. 2, top). After entering the learner's own solution, an applet for the comparison is available. This technique was chosen as it is much simpler there to implement drag&drop as compared to JavaScript, especially in a browser-independent, efficient, and fast way (see Fig. 2, center; applet only excluding the surrounding Sakai UI). Additionally, further processing, like natural language parsing, can be added there in the future easily through libraries.

Whenever the user clicks on an element in the case or the exemplary solution area, the two textboxes in the middle change content and color. These are used for marking correctness and completeness of each fragment. Additionally the corresponding parts in the case and the exemplary solution are highlighted in bold (as well as in the student solution if already assigned by the learner). In this way the "current" element is always clearly indicated. If the general part of the solution is clicked, the textboxes are disabled. Later this state may be used for values describing the case as a whole (which has not been used in the evaluation to reduce the work for learners). After completion the result can be reviewed in HTML (Fig. 2, bottom) or again in the applet (link in top row), but then only in a read-only version where no further modifications are possible. This facilitates reviewing previous solutions; however to keep the user and group models consistent, no changes are possible any more.

IV. Further work

Although the system has been completed, work on enhancements is continuing. In the summer term an evaluation is being performed and extensions are planned or already under development.

A. *Evaluation*

The generator within the learning platform is currently being evaluated in a course. There participants are split in two groups: the first one learns the subject area in a conventional way by receiving full real decisions, while the second one uses generated cases. Based on a detailed logging of the activities, a questionnaire after the course, and the results of the final examination, which will include a hand-crafted case from the generator's subject area, the usefulness of this system for learning will be evaluated. Through the exam the learning outcome (effectiveness) can be distinguished, while the questionnaire evaluates acceptance and provides hints for improvement, especially regarding the web-based user interface.

B. *User interface improvements*

The user interface will be improved by the ability of users to review their own user model. Whether manual modifications of it will be allowed has not yet been decided. Visualization will show the basic data, i.e. the class hierarchy annotated with the direct/derived values. To improve one-look comparison between the own and the group model, color coding is planned: if in the same range

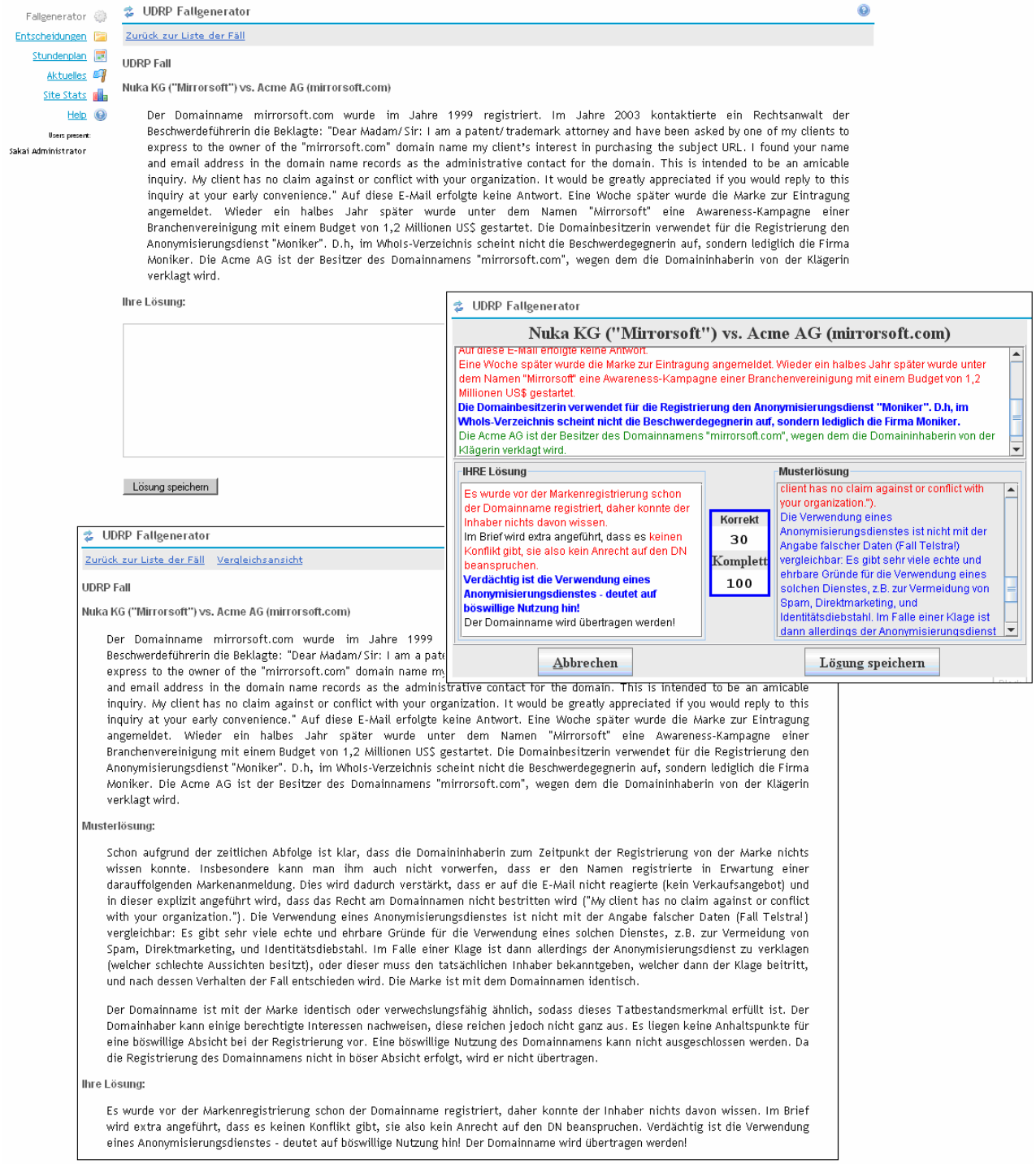

Figure 2. Display of generated case and area for entering the learner's solution (top), applet for comparing own and exemplary solution (center), and view of case, exemplary, and learner solution (bottom)

as the group model, yellow color can be used, for better areas green, and for worse ones red. Combined with the generalization through the class hierarchy the top-most color therefore provides a quick assessment of overall progress. For this only elements covered in exercises completed by this user may be included. Otherwise a group model transferred from a previous term would always mark all students as lacking at the beginning.

In addition graphical feedback is planned: one possibility is a "smiley" with the degree of happiness/sadness determined by the correctness and its transparency by the completeness (coverage of the area). This allows a very fast and informative feedback. E.g. a perfectly completed course should result in a clear smile, while a problematic area not yet covered in depth would be represented by a sad and faded image.

Finally, display of the completed case (Fig 2. bottom) will be extended. The text fragments a case consists of already contain metadata on their source, i.e. from which real-world decision they were excerpted from. This will be integrated so learners can access further information if desired after completing the assessment of their own solution. The alignment of learner and exemplary solution from the comparison applet will be shown through coloring there as well. Correctness and completeness assigned to the own solution can be added to the exemplary solution – as there each constituent element is a single paragraph, while in the learner solution it might consist of several independent parts.

V. Related work

A similar approach to the one presented here is PROSA [4] (PROblem Situations in Administrative law). This is an intelligent tutoring system to train the matching of legal rules to actual cases, i.e. the case solving process. Cases for learning are developed manually and selected by the student according to a difficulty rating. Students then select legal rules or precedent cases, decompose them into components, and match facts from the case description to them. The system provides information on whether the solution is already complete or which elements are incomplete or incorrect. However, no generation of cases takes place. In contrast to the approach described here, cases are not created automatically and there is no real solution: The focus is on the procedure of handling a case.

Another approach is described in [8], which covers the comparatively tiny area of "the gain of property by a third party in good faith" as a subject. In includes a case generator, which select some facts according to the competency level of the student. A case text generator then derives the descriptions in natural language from them. Accordingly, every case has the same structure and only differs in the facts. Learning is then performed by the system asking questions about the case, the learner selecting questions for the computer to answer, or stepwise solving. This differs significantly from the approach here, where users must solve the case independently and where a much wider variety of problems is covered through employing an ontology of a larger subject area.

A somewhat similar area is arguing with cases [1]. There cases are annotated according to whether they are similar to another one, can serve as (counter-)examples to a different case, whether they are bad, medium or good arguments etc. Students then receive a case and must select cases to cite for various arguments (for, against, best case …) in a dialogic setting. Here again cases are created manually from real decisions. An advantage is, that because of the extensive classification and the limited questions automatic correction is possible.

VI. Conclusions

We have described a general method for automatically generating exercises for learners to work on individually. The selection is based on the learning history of the individual the task is generated for as well as a collective history. Such a system is especially useful with a web interface, as then the comparison of the learner solution with an exemplary one need not be done by the learner himself, but could also be performed by someone else, e.g. another learner (peer coaching [7]). The system has been implemented and is currently under evaluation, where a special focus is put on the integration of the group model (whether it brings advantages and which kind of parameters, e.g. selection probability distribution, works best) and the web-based user interface.

Further research is necessary in the following areas: generation of cases can be improved through better integration of the separate elements, i.e. a more natural flow of cross references like "he", "the complainant" etc. between them, as well as the integration of more grammar elements, reducing the need for specially crafted wording to be correct in every possible combination. Another aspect offering room for improvement is the (technically simple) adjustment of the overall difficulty: whether it should be static, dynamic according to the user and group learning progress or success, predefined according to a didactic model, or perhaps adjustable by the learner.

An interesting research topic would be to investigate, how a single group model compares to separate smaller group models, perhaps created or derived according to the general proficiency level ("weak students" group) or their learning styles (visual/auditory/tactile learners).

Natural language parsing might be used for providing first hints which part - sentence, phrase, or paragraph – could match a certain part of the exemplary solution. Through this the effort needed for aligning user and exemplary solution could be reduced. This would be easier if the learner solution were structured as well, or if a structure could be derived from it. In that area existing approaches of case analysis could help.

Acknowledgment

This publication is a result of the research project "ASCOLLA – Adaptive Support for Collaborative E-Learning", which is being funded by the Austrian Science Fund (FWF; P20260-N15).

References

[1] K. D. Ashley and V. Aleven, Towards an Intelligent Tutoring System for Teaching Law Students to Argue with Cases. Proceedings of the 3rd international conference on Artificial intelligence and law. ACM 1991, pp. 42-52

[2] F. Kareal and J. Klema, Adaptivity in e-learning, in A. Méndez-Vilas, A. Solano, J. Mesa, and J. A. Mesa (Eds.), Current Developments in Technology-Assisted Education, Formatex 2006, pp. 260-264

[3] ICANN: Uniform Domain-Name Dispute-Resolution Policy. http://www.icann.org/udrp/udrp.htm

[4] A. J. Muntjewerff and J. A. Breuker, Evaluating PROSA, a system to train solving legal cases, in J.D. Moore, C. L. Redfield and W. L. Johnson (Eds.), Artificial Intelligence in Education. AI-ED in the Wired and Wireless Future. IOS Press 2001, pp. 278 - 285.

[5] T. Murray, Authoring Intelligent Tutoring Systems: An Analysis of the State of the Art. International J. of Artificial Intelligence in Education (1999), 10, pp. 98-129

[6] M. Sonntag, Adaptive Legal Case Generator for Self-Study, in J. Luca, and E. Weippl (Eds.), Proceedings of ED-MEDIA 2008. World Conference on Educational Multimedia, Hypermedia & Telecommunications. Chesapeake: AACE 2008, pp. 5432-5437

[7] M. Sonntag and A. Paramythis, Adaptive feedback for legal E-Learning, in M. Auer (Ed.), The Future of Learning - Globalizing in Education. Wien: Kassel University Press 2008

[8] G. Span, LITES, an intelligent tutoring system for legal problem solving in the domain of Dutch Civil law, in Proceedings of the 4th international conference on Artificial intelligence and law. ACM 1993, pp. 76-81