Taking the Teacher's Perspective for User Modeling in Complex Domains

Christian P. Janssen & Hedderik van Rijn
Department of Artificial Intelligence, University of Groningen
Grote Kruisstraat 2/1, 9712 TS Groningen, The Netherlands
cjanssen@ai.rug.nl, D.H.van.Rijn@rug.nl

Abstract

Serious games that should adapt training to the individual might benefit from methods that are developed for intelligent tutoring systems. One method, model tracing, might be used for domains with a strict hierarchy, while complex domains that lack such a hierarchy might use the method we introduce in this paper as teacher modeling. It takes the perspective of a teacher, who does not always know what leads a student to an answer, but who does know when the answer is (in-) correct and who can assess the capacities of the student over time. The assessment of this tutoring system is based on the amount of training and the amount of positive and negative completed exercises of identified training categories. Principles from a cognitive architecture are used to keep close to aspects of human learning, namely frequency and recency of usage. Simulation runs demonstrate a successful adaptation of training: exercises of categories with which students have most difficulties are presented most.

1 Introduction

Recently it has been advocated that games might be a good training environment, for example in [Aldrich, 2005; Gee, 2003; Michael and Chen, 2005]. These games are referred to as serious games. Serious games are developed for several domains, ranging from high school mathematics to social skills. Over two hundred examples can be found on the website of social impact games [Prensky, 2007]. The new research field of serious games might benefit from techniques that are developed in other fields, and focus on information personalization and digital learning: user modeling and intelligent tutoring systems.

This paper will give an introduction to intelligent tutoring systems, in the context of serious games. Two methods will be explained, that might be used for different domains. If the training domain has a hierarchical structure, the intelligent tutoring method of model tracing might be used. On the contrast, if such a hierarchical structure is missing, other methods have to be found. We will therefore introduce an alternative method that we call *teacher modeling*. We will motivate why such a method might be beneficial for games, and then outline the mechanism and the role that principles from a cognitive architecture play in the method. Our discussion will end with a set of simulations that illustrate that teacher modeling successfully adapts its trainings to the performance of the individual: the more difficulties a student has with completing exercises of a category correct, the more exercises of this category are selected for future training.

In this paper we will often refer to students and teachers, but we do not want to restrict our discussion to them. Therefore one can consider "student" as referring to any person who is learning something, for example a pupil, trainee, or a patient. Additionally, "teacher" can be considered as referring to any person who trains the student, for example an expert, a trainer, or a therapist.

2 Tutoring systems

Intelligent tutoring systems can be defined as computational instruction systems that *adapt* instruction dynamically to the learner, ideally both in form of instruction and in content [Ohlsson, 1986]. Being adaptive has as an advantage that the student can be presented with exercises that best fit individual challenges. Or, to state it differently, the student does not have to spend time on topics that he or she already has mastered.

Several steps can be distinguished in intelligent tutoring systems. In Figure 1, we illustrate a skeleton model of intelligent tutoring systems, to which additional components (e.g., a data-base with exercises, or a natural language generator) could be added for specific systems. A basic tutoring system, as illustrated, starts with a student working on an exercise. The behavior parser continually tracks the behavior of the student, and this information is used to update the view that the system has about a student's skills and performance. For example, eye gaze could be used to derive what item is currently attended, and what items are ignored [Anderson and Gluck, 2001].

The knowledge from the behavior parser can be added to the user model. This is an abstract model that keeps track of some specific (mental) aspects of the user [Johnson and Taatgen, 2005]. The model might contain general characteristics of the student, for example "likes math, but not linguistics", but might also contain more detailed knowledge, for example "makes mistakes in math additions where one of the addends is greater than ten".

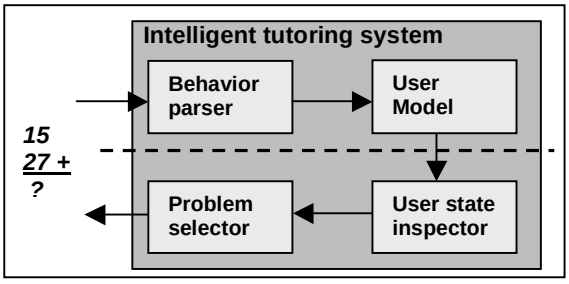

Figure 1: A skeleton model of intelligent tutoring systems.

Note that the information that the user model contains about the student might be more than the student herself is explicitly aware of, as this information might be implicitly and indirectly derived as an *interpretation* of her behavior.

The user model can be observed by the user state inspector to get a general impression about the *type* of exercises the training should focus on. Given this impression, a specific exercise can be selected by the problem selector and given to the student. Observations of the student's performance on this exercise can then be used to update the user model and to keep on selecting new exercises, until the course objectives are completed.

3 Model tracing

3.1 Outline of model tracing

One method for assessing the capabilities of users of intelligent tutoring systems is model tracing. This method tries to assess the mental processes that might have lead to a certain reaction of the student in a detailed manner. To be able to do this, a large set of possible mental processes that can be used in the task at hand are incorporated. These involve both correct rules and facts (e.g., the fact "3+4=7") and incorrect rules and facts (e.g., "3+4=8"). Based on the student's answer to an exercise, the *model* can be *traced* to identify mental processes that might have lead to that answer [Anderson and Gluck, 2001].

Model tracing requires a theory of cognition and learning, and a detailed understanding of the training domain at hand. This domain should be decomposable into a set of components, and the overall learning should be explainable by the learning of those individual components [Anderson and Gluck, 2001]. Because of this hierarchical structure, training can also be hierarchical structured: first the basic levels are trained, and once these are learned correct, more advanced levels can be trained.

An example domain for model tracing is mathematics [Anderson and Gluck, 2001]. If a student has to solve 315 + 216 and answers 521, the user model can be traced to find the possible mistake. In this case, the carry rule that the ten from 5 + 6 = 10 + 1 carries over to the addition of 10 + 10 is probably not used. A tutoring system might thus decide to start focusing on this specific rule.

In order to use model tracing, the decomposition of the domain, and the coupling to possible mental states should be possible and uncontroversial. Incorrect decomposition might lead to incorrect models and therefore to incorrect tracing of performance, and incorrect training adaptation.

3.2 Challenges of model tracing in serious games

Implementing model tracing in serious games has challenges, as games are rich environments in which a player often has a broad set of interaction possibilities. It might be difficult to map each specific action to a specific mental process. For example, in a role-playing game a player might talk to a non-player character because he wants specific information from that character, but he might also talk to the character merely to gather general intelligence.

To restrict the challenge of correctly mapping all actions to mental processes, assumptions and simplifications can be made on the mapping, with the risk of making incorrect assumptions that lead to errors in the interpretation of a player's behavior. Alternatively, the amount of interaction possibilities that the player has in the game could also be restricted. But this is at odds with both the intended flexibility of intelligent tutoring systems to adapt to an individual's needs [Ohlsson, 1986], and with principles of game design that a player should be able to play a game in several ways, thereby developing their own strategies and game-play [see discussion of the multiple routes principle in Gee, 2003].

Despite this challenge, we should not give up on games as training mechanisms, as they might have several training advantages, depending on the exact implementation of the game. Gee [2003] mentions amongst others: (1) young people are used to playing games, which makes it a natural medium for them to use, (2) the fun might encourage longer learning, and (3) players can learn by doing, instead of learning theory outside of its context.

We will therefore now introduce a method that can be used in a game setting and in which the training can be adapted to the individual without using model tracing. As this method does not require an exact mapping between single game actions and mental states, it can be applied in domains that do not have such a detailed theory, for example in depression prevention [Janssen *et al.*, 2007].

4 Teacher modeling

4.1 Motivation for teacher modeling

The method that we will outline is based on a general assessment of the capacities of the student. Educational researchers such as Shepard [2000] have identified and motivated the critical role assessments can play in normal (classroom) learning settings: a good assessment can give an indication how instruction should be adapted to the level of the individual. This conclusion can also be extended to the domains of e-learning and serious games, and therefore assessments play a central role in the method that we outline here. The method can be considered as looking at the student through the eyes of a teacher, as we try to model aspects of the assessments of the teacher: although a teacher does not always explicitly know *what* a student is thinking in order to derive an answer, the teacher does know *when* a student is making a mistake, and can make an *assessment over time* that states what skills the student often performs correct or incorrect. Moreover, the teacher continuously needs *observations of the performance* of a student, to prevent forgetting at what level the student performs. Because of these similarities with real teachers, we call our method teacher modeling.

So what information do we need to develop a system with teacher modeling? First of all, the designer of the system and a group of domain experts should define a set of training objectives. As our method is to be applied in cases were more fine-grained methods such as model tracing are inapplicable, these categories will often be very broad. Examples are "has said no in an assertive way" [in case of a depression prevention tool, Janssen *et al.*, 2007], and "has correctly answered a question about the content of chapter 10" (in a textbook tutor). We will call these categories training dimensions from now on.

For each of these training dimensions, several exercises should be available. Exercises should have multiple outcomes, leaving the student with a choice, but can take several forms, for example dialogues or multiple-choice questions. Experts should be able to indicate for each outcome, and each dimension whether the student's choice (leading to the outcome) was positive (or correct), nega-

tive (or incorrect), or not at all focusing on an aspect of that dimension (thus, training aspects of other dimensions).

This information can be stored in the behavior parser (see Figure 1). Using this information about how to interpret students' answers, the user state inspector can update its assessment of the student, this way acting as a teacher. The assessment is based on three types of information for each dimension: the amount of completed exercises, the amount of positive completed exercises and the amount of negative completed exercises. The next section will explain how this information is stored and how the assessment of the virtual teacher is made.

4.2 Using principles from a cognitive architecture

Although the information of a student's behavior could perhaps be stored in different formats (for example, an absolute counter), we use principles from the cognitive architecture ACT-R [Anderson *et al.*, 2004; Anderson and Lebiere, 1998]. Using principles from a cognitive architecture is advantageous, as this keeps us close to aspects of human cognition and learning.

We use chunks to store information on performance for each dimension. Chunks are the building blocks of ACT-R and maintain an activation level, that represents its usefulness in the past, and that is updated using the base-level learning equation [Anderson *et al.*, 2004; Anderson and Lebiere, 1998]:

$$B_i = \ln\left(\sum_{j=1}^{n} t^{-d}\right) \qquad \text{(Equation 1)}$$

In this equation t_j indicates the time since the jth practice of an item, and d indicates a time-based decay parameter (set to 0.5 by default). The formula has been shown to be good at modeling effects of recency and frequency of usage [Anderson and Schooler, 1991], and, in a slightly modified version, at predicting correct spacing of training [Pavlik and Anderson, 2005]. Being able to incorporate such aspects of learning in the system is beneficial, as our system is used in an environment where people learn.

Each dimension is modeled using three chunks. The "amount of training chunk" reflects the total amount of training, and is updated each time a player has completed an exercise of the dimension of that chunk. The second and third chunk are the "positive chunk" and the "negative chunk" which are updated each time that the student completes an exercise in respectively a positive (or correct) or a negative (or incorrect) way for that dimension.

The score that is indicated by the behavior parser after completion of an exercise is used to decide *if* the activation value of the chunks should update. However, as can be seen in Figure 2, the size with which the activation value increases depends on the amount of previous updates: the more recent and frequent chunks were updated, the smaller the increase of activation value at each update. If chunks are not updated due to completing an exercise of that dimension at a time stamp, their activation value decreases, as it is subject to decay. In our metaphor of the observing teacher that estimates the success of a student, decay can be interpreted as the teacher becoming less confident that a student performs good or bad in a certain dimension, if no new training observations are made.

Interpreting an activation score as a probability, as we did informally for our teacher metaphor, can be formally expressed using the ACT-R retrieval probability equation [Anderson *et al.*, 2004; Anderson and Lebiere, 1998]:

$$P_i = \frac{1}{1 + e^{-(A_i - \tau)/s}} \qquad \text{(Equation 2)}$$

In which A_i is the activation value of chunk i (calculated using Equation 1), τ is the threshold value after which a chunk is selected and s controls the noise (set to about 0.4 by default, or set to 1 if no noise should be included). The function transforms the activation value in a sigmoidal manner [Anderson and Lebiere, 1998]. Therefore, a small increase or decrease in activation value for a chunk that already has a very high or very low activation value will lead to only a small difference in probability value, while in other cases, it can lead to a relatively big difference.

In the ideal situation, where a student has fully mastered a dimension, the probability of the positive chunk of that dimension will be high, and the probability of the negative chunk will be low (due to decay). Situations that differ from these ideal standards should be trained. However, for an application, a more strict definition for selecting training dimensions is necessary.

In the simulations of the next section, we used the following equation that reflects the certainty of our teacher that a player is good at performing a skill of a dimension:

$$Total_j = P_{pos(j)} + \left(1 - P_{neg(j)}\right) + P_{train(j)} \qquad \text{(Equation 3)}$$

Where $Total_j$ is the total score for dimension j, ranging between 0 and 3, and the probability values are the values of the retrieval probability equation (Equation 2) for the three chunks of dimension j: the positive chunk, the negative chunk and the amount of training chunk. The function increases if a student has a high probability value for the positive chunk, and a low value for the negative chunk. This reflects that an observing teacher is more certain that a student is good at the skill associated with that dimension if the student completes exercises correctly and *not* incorrectly. Thus, lower values of the total score indicate

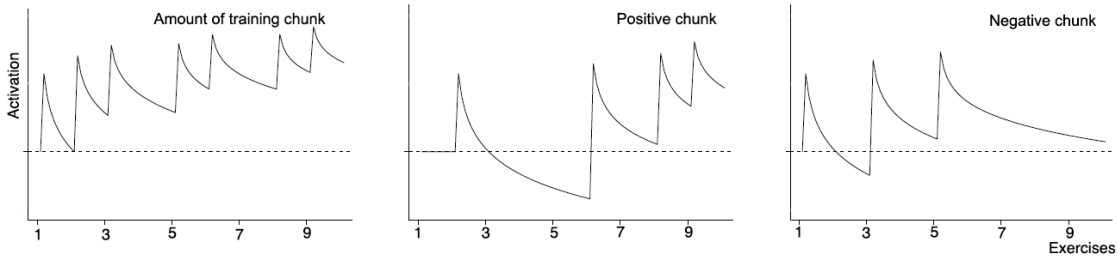

Figure 2: Simulation of the activation values for three chunks of a training dimension. There are positive outcomes for exercises 2, 6, 8 and 9 and negative outcomes for exercises 1, 3 and 5. Notice that the activation of the amount of training chunk increases with both positive and negative outcomes, and that exercises 4 and 7 do not train this dimension.

that the corresponding dimension needs training.

A teacher can only make a good assessment of a student's skills in a certain dimension, when the student has received enough exercises. To stimulate that enough observations of each dimension are made, the probability value of the amount of training chunk is included in Equation 3. With all other values being equal, a dimension with a lower probability of the amount of training chunk will have a lower total score. Thus, in these cases one will first observe a student's behavior in the dimension of which one is least certain about the performance, due to a lack of observations of that dimension. If a student completes the exercise of this dimension, the teacher will become more confident about its observation. If the exercise is completed in a positive way, the probability of the positive chunk of that dimension will increase and therefore, exercises of this dimension will become *less* likely to be presented to the student. However, if the exercise is completed in a negative way, the probability of the negative chunk will increase, and consequently the total score according to Equation 3 will decrease, compared to positive completed situations. Therefore, exercises of this dimension will now become *more* likely to be trained.

4.3 Example simulations

To test if the tutoring mechanism selected the appropriate type of training (i.e., those that a student has most difficulties with), we ran a set of simulations. In these simulations four dimensions were trained in two hundred exercises. Each exercise reflected an interaction that dealt with one dimension, and had two possible outcomes, either positive or negative for that dimension. The exercises were abstract representations for the different ways one can complete an exercise of a dimension. The time between exercises was set to a constant value. Several runs, not discussed here, have shown that results stay fairly similar for different values for this time interval.

The dimension of which an exercise should be presented was dynamically selected using the scores of the dimension according to Equation 3. However, each time we excluded the dimension that had the highest value on the amount of training chunk. Most of the time this was the dimension of which a student had made an exercise last. This guaranteed that the program would not present exercises of just one dimension constantly. For the parameters of the equations we used the following values: $d = 0.5$, $\tau = 0$, and $s = 1$ (no noise).

We tested the mechanism for four different types of students. Each character had a different set of likelihoods for completing an exercise of a dimension in a positive way. The *novice* never completed exercises positive, reflecting inexperience with the skill. The *intermediate character* completed 50% of the exercises of all dimensions positive, reflecting that part of the skills is known, but none has been mastered fully. The *partial expert* answered 25% of the exercises on two dimensions correct (A and B in Figure 3), 75% on another dimension (C) and 100% on the fourth dimension (D). This reflects that the partial expert already performs (almost) perfect on some of the skills, being an expert on these sub domains, but still has difficulties with other skills. The fourth character is the *expert*, who completed 100% of the exercises correctly for three dimensions (B, C and D), while answering 75% correct on another dimension (A). Thus the expert has almost completely mastered all skills.

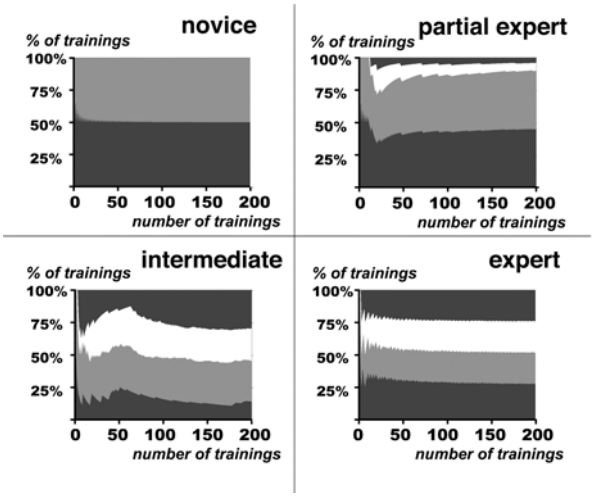

Figure 3: Division of the presented exercises of four dimensions as a function of the total number of presented trainings. The dimensions are: A (lower black area), B (grey area), C (white area) and D (upper black area). Each character (novice, intermediate, partial expert and expert) has different levels of skill for the different dimensions, and therefore a different training. Exercises of dimensions in which they perform worst are presented most.

In Figure 3 the percentage of the exercises that is devoted to each dimension is plotted for the four characters as a function of the amount of presented exercises. At the beginning of the training this pattern fluctuates a lot, as the characters are presented with exercises from different dimensions to assess what dimensions they are not good at. A more stable pattern develops as more exercises are presented. Each of the characters has a different distribution of the dimensions that are trained, which reflects that the user model adapts to individual differences. We will now discuss how the individual patterns can be related to each of the characters of our simulations.

The novice character is only presented with exercises of two dimensions, as the novice fails to complete any of them successfully. This way, the negative chunks of dimensions of which exercises are presented at least once get a high probability value, making the overall performance score for these dimensions low. This way, exercises of these dimensions will be more likely to be picked next time. As the mechanism places a penalty on the dimension with the highest activation value for the amount of training chunk, no two exercises of the same dimension can directly succeed each other. Therefore, the mechanism alternates between exercises of different, but of only *two* dimensions, as the novice already performs very badly on exercises of these dimensions.

The intermediate character has equal skills for each dimension, completing half of the exercises of each dimension correct. The positive reactions temporarily give the teacher an impression that the student is good at the skill of the corresponding dimension. Thus, the mechanism will then select an exercise of another dimension, which makes sure that the student is presented with exercises of *all* dimensions at least once. Over time, the system will become less certain that a student is good at a skill of a previously positively completed dimension, due to decay, and it will once more present an exercise of that dimension. The combination of different responses (both positive and negative) to exercises and decay of activation values lets the mechanism continuously alternate between

exercises of different dimensions. As the intermediate student has the same level of skills for each dimension, all dimensions get about the same amount of training. However, as this character answers random (answering half of the exercises correct and the other half incorrect), some dimensions are slightly more trained than others.

The partial expert and the expert are also presented with exercises of all dimensions, as there is not one dimension where they continuously fail. In contrast to the other characters, the partial expert and expert have different levels of skill for different dimensions: exercises of some dimensions are always completed positive, in others they make some mistakes. The partial expert almost always positively completes exercises of dimension C (75% of the exercises) and D (100% of the exercises), but almost never (in 25% of the cases) completes exercises of dimensions A and B correct. Therefore, the system most of the time presents exercises of dimensions A and B.

As with the other characters, the expert is presented with most training on the dimension that it has least developed skills in: dimension A. However, the likelihood of completing dimension A positive (75% of the exercises) does not differ that much from that of the other dimensions (100% correct). Therefore, the amount of presented exercises of dimension A is only slightly higher than that of the other dimensions.

Figure 3 illustrates that the mechanism adapts to the skills of different types of students. However, if the training exercises are good, students will improve their skills while playing. Due to this improvement, they will answer exercises more often correctly. The training selection mechanism has to react to these changes, by offering exercises of other dimensions. To test this situation, a new set of simulations was run using the same initial characters as used for Figure 3. However, this time they learned during training. Learning was simulated by increasing the likelihood of completing an exercise positive with one percent each time that the characters were presented with an exercise of that dimension. For example, if the expert has been presented with ten exercises of dimension A, the likelihood of answering an exercise of that dimension positive has increased with ten percent from 75% to 85%.

The distribution of trainings is illustrated in Figure 4, and has a different shape than in Figure 3. During training, performance on exercises in each of the dimensions increases. The system still mostly selects exercises of dimensions in which a student performs worst, and in time students will become equally good (that is: perfect) in performance on all dimensions. Therefore the shape of the distribution goes towards that of the expert in Figure 4: all dimensions get an equal amount of training. As the novice gets better at some dimensions (and sometimes answers exercises positive) already in the beginning, but is at that moment worse at other dimensions, training is now focused on all four dimensions instead of on only two.

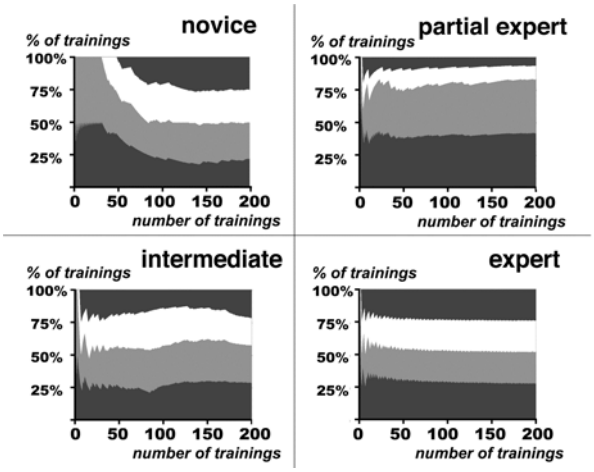

Figure 4: Division of the presented exercises of four dimensions when the characters learn during the training. The better the characters start performing on all dimensions, the more evenly spread the distribution of trainings becomes.

The higher the initial difference is between performances on the different dimensions for a character, the longer it will take before this scenario of continuous equal distributed training on all dimensions will take. Therefore, the partial expert mostly trains two dimensions, while the intermediate and novice are presented with an almost equal amount of exercises of all dimensions.

5 Discussion

In this paper a method for intelligent tutoring in complex contexts such as serious games was introduced as an alternative for the method of model tracing. An alternative was necessary, because model tracing requires a strict decomposable domain [Anderson and Gluck, 2001], which might be unavailable for some training domains. Moreover, it might be difficult to trace the mental states for all of the actions that are possible in games.

The model we introduced requires neither a strict hierarchical domain, nor a detailed understanding of the mental processes that lead a user to his or her actions. Rather, it mimics aspects of an observer, or teacher, who can assess when a student is doing something correct or incorrect, and over time can develop an impression about the student's strengths and challenges. This is not to say that the mechanism acts *exactly* like a teacher. This would be a difficult objective, as games are different settings than classrooms and therefore require at least slightly different approaches. However, the introduced method does include a continuous skill assessment that is used for training selection. This is an important aspect for good teaching [Shepard, 2000], hence the name teacher modeling.

Due to space constraints, we compared teacher modeling only with model tracing, but of course there are other methods for intelligent tutoring. Constraint-based modeling [Mitrovic and Martin, 2007], dynamic bayesian networks [see Manske and Conati, 2005, for an implementation in a game], and predictive learning curves [Baker *et al.*, 2007] are alternatives. However, these methods can only be applied for hierarchical domains, while teacher modeling can be applied in non-hierarchical domains.

Teacher modeling requires a set of broad training dimensions, and a set of exercises with multiple outcomes, each of which has been labeled by experts as being completed good or bad for a specific dimension, or as not making use of skills from that dimension. In the approach we used principles from a cognitive architecture to keep close to aspects of human cognition and learning. In our case, the mechanisms that we used are claimed to be good at modeling effects of recency and frequency of occurrence [Anderson and Schooler, 1991]. Therefore, the type of exercises that the system will select depends not only on the success of completion, but also on the frequency and recency at which the exercises have been presented. As the system can adapt training to the individual, it is preferred over simpler strategies such as level based

games, in which students possibly also have to complete exercises that they are already good at and that therefore might be boring.

However, adaptive training is not a guarantee for a motivating training. As can be seen with the novice in Figure 3, the training mechanism successfully offers training on only two dimensions if a user shows repeated failure on corresponding exercises. However, training only aspects in which you fail to improve might be frustrating, and can indicate that the training is not appropriate for the player. These cases can be detected by looking at the dimension scores: these will be low and have a small variance. If detected, the system should respond to them, for example by calling in an expert.

These advanced features were not included in the simulations that we discussed, which have shown how the introduced mechanism adaptively selects training exercises. Mostly the dimensions in which students have difficulties were selected for training, also in cases where students improve their performance during game play. The effect of the training mechanism still has to be tested in real life, and each specific implementation will require a user study for its effect. The success of the test will crucially depend on the quality of the exercises that are developed, and on the interrater-reliability for the labels that experts assigned to the outcomes of the exercises: the higher the reliability, the more certain one can be that this label is correct for the behavior of students who chose this outcome, thereby decreasing the risk of an incorrect assessment.

If more exercises need to be added to the game, this is relatively easy with the teacher modeling approach. This only requires a set of exercises of which the outcomes have been labeled in broad terms; it does not require a strict decomposition of all interactions and mental states as is required in model tracing. Still it can be challenging to develop exercises, as they also have to fit game constraints. The number of game constraints and the amount of impact they might have on the difficulty for developing exercises will depend on the type of game at hand. For example, in role-playing games where a student is free to interact with virtual characters, every exercise has to be developed for a virtual character, and every virtual character must have enough exercises available for the intelligent tutoring system to make a good selection.

Although we introduced our method in the context of intelligent tutoring systems, it might also be considered as a more general method for recommender systems, because it can be used to produce individual recommendations as output [Burke, 2002]. In the context of serious games these individual recommendations can be exercises.

To further test the mechanism of teacher modeling, it is currently being implemented in a serious game. The goal of this game is to train a player's assertive skills. More on this specific project can be read in [Janssen *et al.*, 2007].

Acknowledgments

This work is funded by the Regional programme of innovative actions in Groningen (project UBAssertiv).

References

[Aldrich, 2005] Aldrich, C. *Learning by doing: A comprehensive guide to simulations, computer games, and pedagogy in e-learning and other educational experiences*. Pfeiffer, San Francisco, 2005.

[Anderson *et al.*, 2004] Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., and Qin, Y. An integrated theory of the mind. *Psychological Review*, 111(4): 1036-1060, 2004.

[Anderson and Gluck, 2001] Anderson, J. R., and Gluck, K. What role do cognitive architectures play in intelligent tutoring systems? In S. M. Carver and D. Klahr (Eds.), *Cognition & instruction: Twenty-five years of progress*, pages 227-262, Erlbaum, Mahwah, NJ, 2001.

[Anderson and Lebiere, 1998] Anderson, J. R., and Lebiere, C. J. *The atomic components of thought*. Erlbaum, Mahwah, NJ, 1998.

[Anderson and Schooler, 1991] Anderson, J. R., and Schooler, L. J. Reflections of the environment in memory. *Psychological Science*, 2(6): 396-408, 1991.

[Baker *et al.*, 2007] Baker, R. S. J. de, Habgood, M. P. J., Ainsworth, S. E., and Corbett, A. T. Modeling the acquisition of fluent skill in educational action games. In *Proceedings of the 11th International Conference on User Modeling*, pages 17-26, Corfu, Greece.

[Burke, 2002] Burke, R. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4): 331-370, 2002.

[Gee, 2003] Gee, J. P. *What video games have to teach us about learning and literacy*. Palgrave Macmillan, New York, 2003.

[Janssen *et al.*, 2007] Janssen, C. P., Van Rijn, H., Van Liempd, G., and Van der Pompe, G. User modeling for training recommendation in a depression prevention game. In *Proceedings of the first NSVKI student conference*, pages 29-35, Nijmegen, The Netherlands.

[Johnson and Taatgen, 2005] Johnson, A., and Taatgen, N. A. User modeling. In R. W. Proctor and K. L. Vu (Eds.), *The handbook of human factors in web design*, pages 424-438, Erlbaum, Mahwah, NJ, 2005.

[Manske and Conati, 2005] Manske, M., and Conati, C. Modelling learning in an educational game. In *Proceedings of the 12th International Conference on Artificial Intelligence in Education*, Amsterdam, The Netherlands, 2005

[Michael and Chen, 2005] Michael, D. R., and Chen, S. L. *Serious games: Games that educate, train, and inform*. Thomson course technology, Boston, 2005.

[Mitrovic and Martin, 2007] Mitrovic, A., and Martin, B. Evaluating the effect of open student models on self-assessment. *International Journal of Artificial Intelligence in Education*, 17(2): 121-144, 2007.

[Ohlsson, 1986] Ohlsson, S. Some principles of intelligent tutoring. *Instructional Science*, 14(3): 293-326, 1986.

[Pavlik and Anderson, 2005] Pavlik, P. I., and Anderson, J. R. Practice and forgetting effects on vocabulary memory: An activation-based model of the spacing effect. *Cognitive Science*, 29: 559-586, 2005.

[Prensky, 2007] Prensky, M. Social impact games website (www.socialimpactgames.com). Retrieved 19-01-2007

[Shepard, 2000] Shepard, L. A. The role of assessment in a learning culture. *Educational Researcher*, 29(7): 4-14, 2000.