Tech., Inst., Cognition and Learning, Vol. 1, pp. xx-xx
Reprints available directly from the publisher
Photocopying permitted by license only

Memory Features in Simulated Students to Improve the Software Engineering Process and the Performance of Intelligent Tutoring Systems

KONSTANTINOS S. MANOS AND MARIA VIRVOU

Department of Informatics, University of Piraeus, Piraeus 18534, Greece
konstantinos@kman.gr; mvirvou@unipi.gr

This paper describes how memory features have been incorporated into the student modelling process of educational software to create simulated students. These simulated students are agents that simulate the way that students learn new facts from the domain being taught with respect to the amount of time that they can remember these facts. For this purpose, the student modelling process is based on principles of cognitive psychology. The resulting simulated students can be used to improve the software engineering process of the educational application and its performance. The software engineering process can be improved by the use of simulated students to evaluate educational software before it is delivered to real students. Moreover, the performance of the educational application can be improved since the simulated students can indicate how much new knowledge the real students may learn at a specific time and what needs to be reviewed. Thus, the system can plan the tutoring of new material and the revisions to be presented to students dynamically.

Keywords: Student modelling, Intelligent Tutoring Systems, Simulated students, Retention Abilities, Instructional Design

INTRODUCTION

Educational software is a special kind of software that aims at facilitating the difficult cognitive process of students' learning. In this respect, educational software has to combine many qualities to make the most of the interactive means provided by computers and be educationally beneficial. Such

qualities include attractive multimedia presentations, the individualization of tutoring, reasoning abilities, user-friendly interfaces, etc. To achieve all these qualities there has to be a combination of educational software technologies and ideas, such as the combination of multimedia educational software technology with the underlying reasoning mechanisms of Intelligent Tutoring Systems (ITSs). Moreover, the coexistence of these qualities demands a lot of effort during the software engineering process, which has to be iterative so that it may allow many evaluations of the educational application life cycle.

Indeed, an iterative software engineering process that allows many evaluations of the software is recommended for any kind of software. For this reason there have been older software life cycle models, such as the spiral model (Sommerville, 1992), and newer, very successful ones, such as the Rational Unified Process (Kruchten, 99; Quatrani, 98), which advocate multiple iterations of the developmental process. Multiple iterations of the software engineering process can be very beneficial for educational software as well. This means that there may be a first prototype which has to be evaluated, and subsequently there can be an improved executable release based on the results of the evaluation and so on. As Dix et al. (1993) point out, evaluation is an integral part of the design process and should take place throughout the design phase of the life cycle. However, evaluations have often been neglected in educational software development. For example, Gilbert (1999) analyses the Teaching and Learning Technology Programme (TLTP) evaluation report by the Higher Education Funding Council for England. Among other things, he notes that the programme seriously underestimated the complexity of designing materials that could be considered, in any sense of the word, “intelligent”, and that there has been a serious lack of evaluation.

In view of the high demands on the reasoning abilities of educational software and the apparent need for the improvement of the software engineering process, we have created simulated students that can be used to improve the performance of educational applications dynamically (on the fly) and can be used as evaluation agents in an iterative software engineering process with these applications. In particular, we are going to describe how a student modelling process has been incorporated into simulated students. This particular student modelling process focuses on keeping track of what a student is being taught and will actually remember after the end of the lesson (Virvou & Manos 2003a). This is achieved by the adaptation and application of models of cognitive psychology to the particular circum-

stances of the educational software application.

Student modelling is an important process for ITSs since it may provide detailed reasoning concerning the students' needs and progress and thus make the applications highly individualised. Indeed, student modelling has become a core or even defining issue for ITSs (Cumming & McDougall 2000). In the research described in this paper, the student models give input for the creation of simulated students. Then the simulated students are used during the execution of the educational applications to give insight to the system as to how much a student has learnt from the material that has been taught to him or her and what needs to be reviewed. If something needs to be reviewed the ITS reschedules the teaching material and presents the topic to the student to be reviewed. Moreover, the simulated students may be used by instructors and ITS designers to evaluate the courses that they have created before these are delivered to real students. Thus, designers are given the opportunity to fine-tune the ITSs so as to achieve better results with the real students.

Simulated students have been created and used in past ITSs, mainly to assist the learning process of students. For example, the mode of the simulated co-learner has been considered quite important by many researchers for the purpose of improving the educational benefit of tutoring systems. One reason for this is the fact that the simulated student can simultaneously be an expert and a co-learner and can thus scaffold and guide the human's learning in subtle ways (VanLehn et al. 1994). However, simulated students have not been used as evaluation components in the software engineering process of ITSs. Such evaluation components may be very useful because they allow and encourage multiple iterations of the design process, which in turn may ensure that the resulting educational software applications are of better quality.

THE SIMULATED STUDENTS AS EVALUATION AGENTS IN THE SOFTWARE ENGINEERING PROCESS OF AN ITS

As a test-bed for our research we have used an ITS that operates as a Virtual Reality game. The VR-game is called VR-ENGAGE (Virvou et al. 2002) and teaches geography. VR-ENGAGE has been enhanced by the addition of a module that may measure/simulate the way students learn and possibly forget throughout the process of a game/lesson.

Description of the ITS

VR-ENGAGE is an educational application that has the reasoning mechanisms of an ITS, such as student modelling and adaptive tutoring, and can be operated by students as a VR adventure game. The reason for the selection of a VR game as the medium of operation of the ITS is to make the educational application more engaging and motivating for students who are actually acting as players. Indeed, recently a lot of researchers have shown themselves to be convinced that education may benefit a lot from the incorporation of computer games. For example, Muntaz (2001) notes that a range of cognitive skills are practised in computer game playing due to the sheer number of decisions children make as they weave their way through various games. In addition, other researchers have already incorporated computer games into the educational software they have created (e.g. Amory et al. 1998; Conati & Zhou 2002).

In VR-ENGAGE, the ultimate goal of a student-player is to navigate through a virtual world and find the hidden book of wisdom. While players are navigating through the virtual world, they meet animated agents who lead them to places where they can read lessons about the domain being taught. Moreover, each player has an inventory list which may contain certain objects that could be of help during the game.

The player also finds keys, which are guarded by dragons. A guard dragon poses a question to the player from the domain of geography. If the player provides a correct answer then the dragon allows him or her to take the key. Each of these keys opens a door which leads the player closer to the "book of wisdom". If the player provides an erroneous answer which is close to the correct one a virtual companion shows up who tries to help the student find the correct answer. In case a student provides a wrong answer he or she may have the possibility of bypassing the guard by using a key from the inventory (in case he or she has a key there).

The system also has the ability to adapt teaching to the specific needs of each student to maximise the amount of knowledge that the student learns. For example, it may dynamically select which part of the theory the student is going to see and when. For this purpose there is a student modelling component that keeps track of what the student has already seen (and when he or she saw it), what the student seems to have learnt given his or her answers to questions and what the student is likely to remember by the end of the lesson. This information is stored in the long-term student model to be used in subsequent lessons and for the creation of simulated students. During the whole process the student model continuously examines the student's mem-

ory capabilities. This is accomplished by measuring time intervals between a student's reading of a piece of theory and answering of the corresponding riddle or by gathering statistical information about the student's reactions.

Iterative Software Engineering Process

The simulated students can be used to evaluate the ITS throughout the software engineering process. For this purpose, during the software engineering process an ITS designer has to collaborate with human experts (in this case, teachers) to complete a first version of the system. Then, the first time that the system is executed, it needs to be used by real students so as to gather some initial retention statistics on which to base the simulated student. The retention statistics concern the student's ability to remember things that he or she has been taught during the game/lesson. Hence, a student model is created and stored inside the system's knowledge base for each real student-player. This preliminary process is illustrated in Figure 1.

During the preliminary process, it is a good practice to choose students of different retention capabilities (e.g. good students, moderate and bad ones), so as to have better comparative results at the end of the process. In this way, it can be ensured that the final lesson will be appropriate for the whole range of students rather than just one category of them. Each student model that has been constructed during this preliminary process is used to create a simulated student.

As a second step, the ITS designer may ask the simulated student-players, which act as evaluation agents, to "play" the virtual game using different student models. These student models have been stored in the students' PCs and may be different from those in the designer's PC. However, they can be collected through the internet. In the end, the human teacher views the results and may choose to modify the virtual world's content so as to emphasise some parts of the theory more than others, or he or she may find a mistake in the flow of the lesson which he or she may wish to correct. With this tool, the teacher and ITS designer may actually have a measure of the virtual lesson's efficiency before taking it to class. This allows an iteration of the software engineering process of the ITS and thus ensures better quality of the resulting educational application. The process of evaluating the ITS using simulated students is illustrated in Figure 2.

The simulated student is an agent that, given a student model, starts "playing" the virtual lesson inside the ITS by simulating the user's reactions. The agent incorporates a cognitive model which is based on cognitive psychology and gives the teacher insight on the portion of the information that

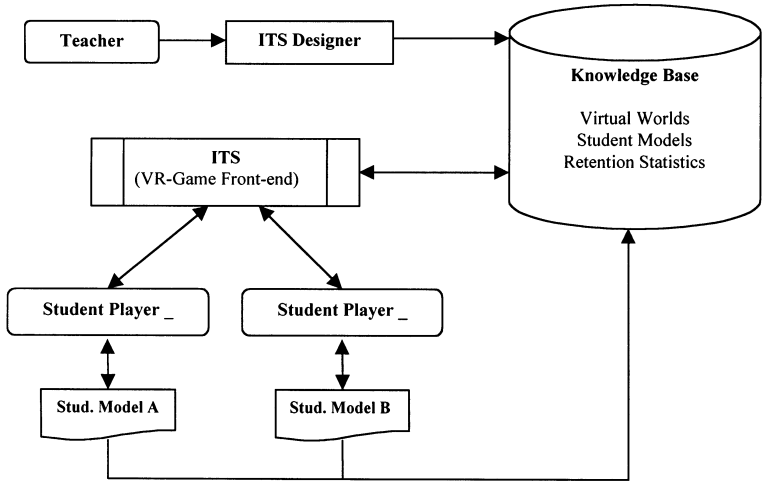

FIGURE 1
Preliminary process

is actually learnt by a student-player during the Virtual Lesson. This model calculates and simulates the retention and memorisation capabilities of a student and will be explained in more detail in the following sections.

Thus, the simulated student-player at first starts "walking" inside the virtual world. When it encounters a part of the theory, it uses the cognitive model to store that information inside its "mental" library. It then continues its "walk". When it faces one of the "guards" it tries to answer the riddle. To do this, it checks its "mental" library for the part of the theory that is needed for that riddle, it then checks the information with the cognitive model, deciding whether it "remembers" the fact or not. To decide whether it actually remembers the fact, the retention factor of the specific information should be higher than the student's retention factor. If it is, a correct answer is generated; otherwise a wrong one is given. In the case of a wrong answer, the simulated student-player acts according to the student's profile, which indicates whether the student would go back to revise the theory or use one of the available items in his or her inventory to bypass the guard. Eventually, the agent finishes the virtual lesson.

During the virtual lesson, the system monitors the agent's "play" just as it would monitor an actual student. As far as the system is concerned, it does not "know" the existence of the agent, and thus it reacts in exactly the same way it would were an actual student playing. This means that in the end the

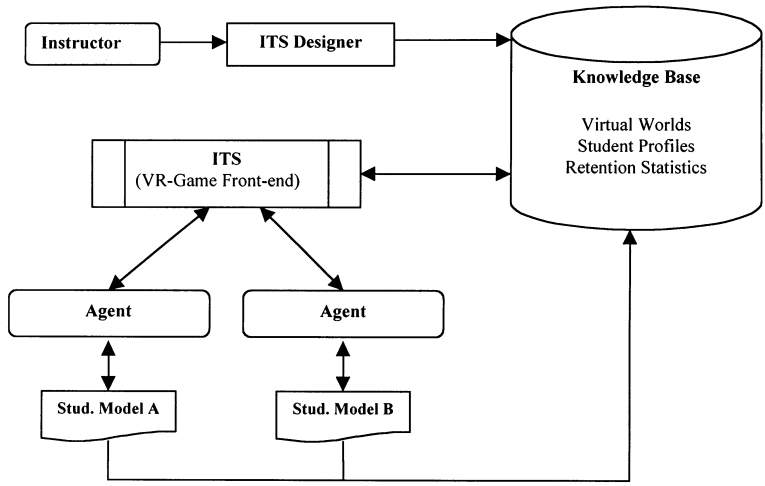

FIGURE 2
Evaluation of the ITS using simulated students

system ends up with many statistical results and a new user model created by the input provided by the agent. In the same manner, the cognitive model monitoring the agent's play ends up with its own results concerning the student's retention factor for each of the portions of the theory encountered.

These results are not far from the ones that would come from the real students' play. At first, as far as the system's student modelling mechanism is concerned and given the fact that the agent operates on a specific, previously constructed student model, the resulting model is exactly the same as the one used by the agent. In this way, we ensure that the agent actually played its role as it was meant to, thus effectively simulating the student. As far as the cognitive module's results are concerned, these are pretty close to the ones that the actual player would generate with his or her own play.

In this way the teachers, in collaboration with the ITS designer, may evaluate the efficiency of the lessons they have created. If the agent's results are not the desired ones, they may change the lesson's layout, theory content and/or quantity to better highlight certain parts of the theory. This process leads to iterations of the software life-cycle which may result in a high quality ITS that can be educationally beneficial to real students. The multiple iterations of the software engineering process are illustrated in Figure 3.

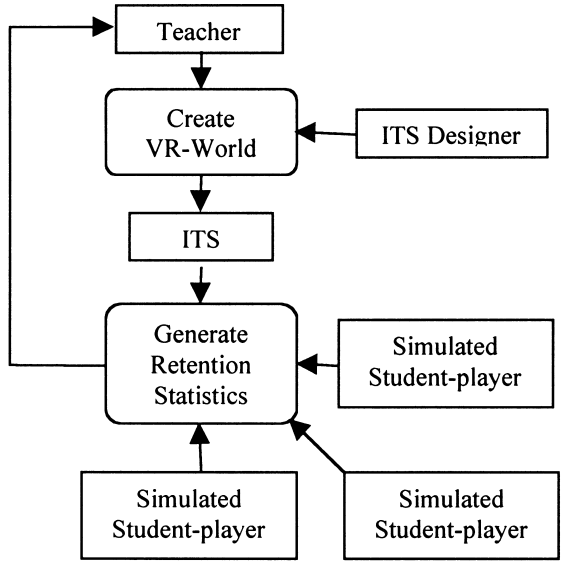

FIGURE 3
Iterations of the software engineering process

STUDENT MODELLING IN THE ITS

Student modelling in VR-ENGAGE is based on the overlay technique. The overlay model was invented by Stansfield, Carr and Goldstein (1976) and has been used in many early user-modelling systems (Goldstein, 1982) and more recent systems (e.g. Matthews et al. 2000). The main assumption underlying the overlay model is that a user may have incomplete knowledge of the domain. Therefore, the user model may be constructed as a subset of the domain knowledge. This subset represents the user's partial knowledge of a domain, enabling the system to know which parts of the theory the user knows and which he or she does not know. However, as Rivers (1989) points out, overlay models are inadequate for sophisticated modelling because they do not take into account the way users make inferences, how they integrate new knowledge with knowledge they already have or how their own representational structures change with learning. One additional problem with the overlay technique is that it assumes for the student an "all or nothing" knowledge of each part of the domain (either a student does or does not know something).

The overlay technique has to be used in conjunction with inference mechanisms about the students' knowledge. The inference mechanisms that have been employed so far in the literature have been mainly based on actions students make in assessment tests that show evidence of their knowing or not knowing something. However, even in cases when the student shows evidence of knowing something at a particular time, he or she may forget it after a while. Therefore, in our research we take into account what parts of the theory the student has been shown, how often this has happened and what the student is likely to remember. For this purpose, the overlay technique has been extended to include degrees of knowledge for each fact. Each degree represents the possibility of a student knowing and remembering something given the time at which it was learnt. For this purpose, we use a forgetting model.

There are two popular views on forgetting (Anderson, 2000). One of them, the decay theory, supports the view that memory traces simply fade with time if they are not "called up" now and then. The second view states that once some material is learned, it remains forever in one's mental library, but for various reasons it may be difficult to retrieve. These may seem to be "conflicting" theories, but when someone has "forgotten" something, there is really no way for us to tell whether it has been completely removed from his or her mental library or is simply very (almost impossibly) difficult for him or her to retrieve it. For our study, both theories have practically the same meaning: If a student finds it hard to remember a fact that he or she has learnt (either due to memory fading or difficulty of retrieval) then the learning process was not good enough and should be modified.

A classic approach on how people forget is based on research conducted by Herman Ebbinghaus and appears in a reprinted form in (Ebbinghaus, 1998). Ebbinghaus worked for a period of one month and showed that memory loss was rapid soon after initial learning and then tapered off. In particular, Ebbinghaus' empirical research led him to create a mathematical formula which calculates an approximation of how much may be remembered by an individual in relation to how much time has passed since the end of learning (Equation 1).

$$b = \frac{100 * k}{(\log t)^c + k} \tag{1}$$

In Equation 1:

- t: is the time in minutes, starting one minute before the end of learning
- b: is the equivalent of the amount remembered from the first learning. As it is evident from the logarithmic nature of the formula, b decreases greatly at the beginning and starts to stabilise as time passes.
- c and k : are two constants with the following calculated values: k = 1.84 and c = 1.25

Linton (1979) also conducted research on the retention of knowledge and worked for a period of six years. Linton's results were similar to Ebbinghaus' results. Finally, Klatzky (1980) also reports the results of a study that consisted of experiments on retention. These experiments involved repetitions of a memorised list of words after a pre-specified break length, typically up to a few days. This study showed that memory decay is a power function of the break length. For example, subjects forget 55% of the words within a six hour break time and 80% percent within 72 hours. However, these results are very close to Ebbinghaus' results. Indeed, if Ebbinghaus' formula was used, one would find that subjects forget 60% of the words within a six hour break and 75% within 72 hours. Such differences in the results have little importance for the purpose of incorporating a forgetting model into an educational application. Therefore Ebbinghaus' mathematical formula has been used in VR-ENGAGE to give the system insight on the students' learning and forgetfulness.

In our model there is a database that simulates the mental library of the student. Each fact a student encounters during the game/lesson is stored in this database as a record. In addition to the fact, the database also stores the date and time the fact was last used along with a numerical factor describing the likelihood of the student's recalling the given fact. The smaller the factor the less likely it is that the pupil will remember the fact after the end of the game/lesson.

LEARNING AND REMEMBERING

Our research goal is to make the educational game more effective in teaching the student. This will happen if the student actually ends up with many facts with high factors in his or her mental library after the course. To model this, we assume that the student has a blank mental library on the subject being taught, meaning that during the first lesson there is nothing in the mental library of the student to be retrieved.

While the student plays the educational game, he or she encounters a "tutor" that provides him or her with a piece of information to be taught. This is the first encounter with this information, and it is thus added to the memory database. The data saved in the database are:

- **ID**: a string ID of the fact being taught
- **TeachDate**: the date and time of the first occurrence of the fact
- **RetentionFactor(RF)**: a number showing how likely it is that the student will actually remember the given fact after the end of a "game lesson"

When a fact is inserted into the database, the TeachDate is set to the current date and time, while the RF is set to a base number. The RF stored in the "mental" database for each fact is the one representing the student's memory state at the time shown by the TeachDate field.

Having saved the above data, one may calculate the percentage of retention of a given fact that a particular student is likely to have at a particular time. We call this Retention Percentage (RP). Whenever we need to know the current RP of the fact, equation 2 is used.

$$RP = \frac{b}{100} * RF \tag{2}$$

Where:

- b: is Ebbinghaus' power function result (Equation 1), setting t=Now-TeachDate
- RF: is the Retention Factor stored in our database.

The retention factor is used to individualise this equation for the particular circumstances of each student by taking into account evidence from his or her own actions. If the system does not take into account this evidence from the individual students' actions then the Retention Factor may be set to 100, in which case the result is identical to Ebbinghaus' generic calculations concerning human memory in general. However, if the system has collected sufficient evidence for a particular student the Retention Factor is set to 95 when a fact is first encountered by this student and then modified accord-

ingly, as will be described in detail in the following sections.

The mathematical formula (1) by Ebbinghaus gives an estimation of how students learn and forget which applies to all kinds of students and does not take into account any individual characteristics. However, the information stored in each individual student model may provide more information about each student's ability to learn and memorise new facts. This kind of information has been used to individualise the results provided by the Ebbinghaus formula (Virvou & Manos 2003b). In particular, we have used what we call the Personal Base Retention Percentage, the Memorisation Ability factor and the Response Quality factor, which will be explained in detail in the following subsections.

Base Retention Percentage

To estimate whether a student has learnt a fact that has been taught to him or her during a lesson, the student's RP for that particular fact has to be calculated at the end of the lesson. Then the RP has to be compared with a number that represents a threshold of learning for students. We call this number the Base Retention Percentage (BRP). This number is set by default to 70. This is because any number below 70 corresponds to a forgotten fact according to the Ebbinghaus formula. If the calculated RP of a particular student for a particular fact is greater than the BRP the student is assumed to have learnt the fact, otherwise he or she needs to review it.

A BRP of 70 may give more or less accurate results for a wide variety of students. However each individual student is a unique entity and has his or her own personal BRP. For example, students with strong memorisation abilities tend to have a lower BRP while on the other hand weak students tend to have a higher BRP. For example, there may be cases where a student is believed to remember 60% of a fact. This percentage is below 70%, and thus the student is by default believed to have forgotten the fact. However, the student may answer a question correctly that concerns this fact. This shows that the student knows the fact although he or she is believed to remember only 60% of it. If this happens for many facts for that particular student, then the student has stronger memorisation abilities than the average student modelled by the Ebbinghaus formula. In such cases, the BRP for this kind of student should be lower than 70, which is the average BRP. The unique BRP for each student is what we call the student's Personal BRP.

Unfortunately, there is no automatic or mathematical way of calculating the personal BRP for every student. One way to calculate it is by giving the student a sequence of tests. First, we need to have the student answer all the

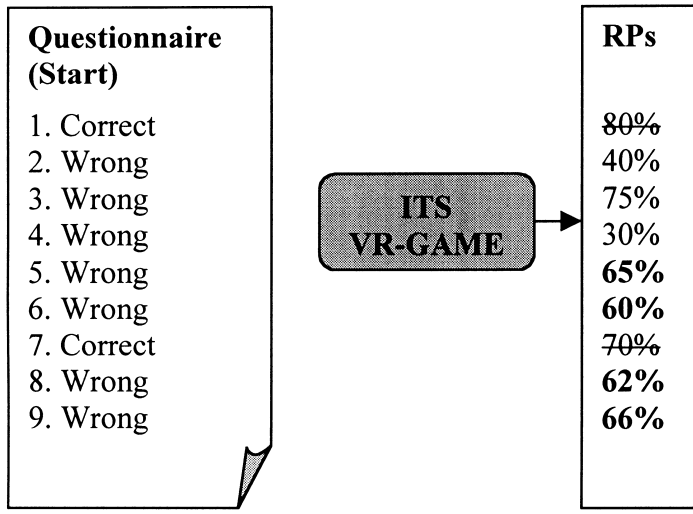

FIGURE 4
Calculating the personal Base Retention Percentage

questions he or she will encounter in the virtual world. These answers should be given by the student before he or she plays the game and without his or her having read the theory. In this way, the system may find out what the student already knows before he or she reads the theory to be taught. Then, the student is left to play the game. When he or she finishes playing the game, the ITS calculates an RP for each and every part of the theory. Last, we give the student the same questionnaire as at the beginning and mark the results (Figure 4). The answers that the student provides to the questions at the start are compared with those that he or she provides at the end. Moreover, it is examined whether the RP corresponding to every fact associated with each question is less or greater than 70, which is the default BRP.

First, during the comparison of the student's answers at the start with those at the end, we cross out all the facts that correspond to a correctly answered question in the first questionnaire. This is because we assume that the student already knew these facts and that they thus might tamper with our calculations. Indeed, these facts in the example in Figure 3 have been crossed out. Next, we find all the RPs that correspond to correctly answered questions which were not answered correctly in the first questionnaire and for which the calculated RP is below 70 (for our example these are facts 5,

6, 8 and 9, with RPs 65, 60, 62 and 66 respectively). These questions correspond to facts that were not known by the student before the game/lesson and seem to have been learnt by the student during the game/lesson since he or she answered the questions correctly after the end of it. Moreover, the facts are found to have been learnt by the student although the RPs that correspond to them are lower than 70. If such RPs are more than or equal to 4 (to limit the possibility of such a case being an "one-time" event), we take the highest of them (for our case that is 66), and that is one metric (RP1).

After this, we find all the RPs that correspond to wrong answers for which the calculated RP is above 70. Again, if such RPs are more than or equal to 4, we take the lowest of them and use it as the final metric (RP2). In our example we do not have an RP2 metric.

To define the student's personal BRP, we need to consider the following issues:

1. If no RP1 exists we examine RP2. If RP2 exists then this is the student's personal BRP; otherwise we keep the value of 70.
2. If RP1 exists but no RP2 was found, then this is the student's personal BRP; otherwise we keep the value of 70. For our example the student's personal BRP is set to 66%.
3. If both metrics exist then our experiments have shown that this case is extremely rare, and the student has probably been answering the questions (during the phases of the test) by chance.

For the rest of this study whenever we talk about the BRP we are going to assume that its value is 70. Given the fact that any RP below 70 corresponds to a "forgotten" fact, using Ebbinghaus' power function we may calculate the "lifespan" of any given fact.

Individual Memorisation Ability

One important individual student characteristic that is taken into account is the ability of each student to memorise new facts. Some students have to repeat a fact many times to learn it while others may remember it from the first occurrence with no repetition. To take into account these differences, we have introduced the student's Memorisation Ability factor (MA). The values of this factor range from 0 and 4. The value 0 corresponds to "very weak memory," 1 to "weak memory," 2 to "moderate memory," 3 to "strong memory" and 4 to "very strong memory."

During the course of a virtual game there are many different clues that

can give insight on the student's MA. One important hint can be found in the interval of time between a student's having read about a fact and his or her answering a question concerning that fact. For example, if the student has given a wrong answer about a fact that he or she has just read about then he or she is considered to have a weak memory. On the other hand, if he or she gives a correct answer concerning something he or she read about a long time ago then he or she is considered to have a strong memory.

Taking into consideration such evidence, one may calculate the student's MA value. Using MA, the Retention Factor is modified according to the MA value of the student in the manner illustrated in Table 1. As mentioned earlier, every fact inserted in the database has an initial RF of 95.

TABLE 1
Retention Factor modification depending on Memorisation Ability

Memorisation Ability	**Memorisation Ability Value**	**Retention Factor Modification**
Very Weak Memory	0	RF` = RF – 5
Weak Memory	1	RF` = RF – 2
Moderate Memory	2	RF` = RF
Strong Memory	3	RF` = RF + 2
Very Strong Memory	4	RF` = RF + 5

After the modifications, which are based on the MA factor, the student's personal RF ranges from 90 (very weak memory) to 100 (very strong memory), depending on the his or her profile. Taking as a fact that any RP below 70 corresponds to a "forgotten" fact, one may calculate the "lifespan" of any given fact for the MA mentioned above using Equation 2. Thus, a student with a very weak memory would remember a fact for 3 minutes while a student with a very strong memory would remember it for 6.

Individual Response Quality

During the game, the student also faces question-riddles (which require the "recall" of some facts to be answered correctly). In that case the RP of the fact is updated according to the student's answer. An additional factor, the Response Quality (RQ) factor, is used for this modification. This factor ranges from 0 to 3 and reflects the "quality" of the student's answer. In particular, 0 represents "no memory of the fact," 1 represents an "incorrect

response; but the student was close to the answer," 2 represents "correct response; but the student hesitated" and 3 represents a "perfect response." The formulae for the calculation of the new RF depending on the Response Quality Factor are illustrated in Table 2.

TABLE 2
Response Quality Factor, reflecting the quality of the student's answer

Response Quality	**RQ Value**	**Modification**
No memory of the fact	0	RF' = RP – 10, set TeachDate=Now
Close Answer	1	RF' = RP – 5, set TeachDate = Now
Correct but with hesitation	2	RF' = RF + (MA + 1) * 3
Perfect Response	3	RF' = RF + (MA + 1) * 4

When a student gives an incorrect answer, the TeachDate is reset, so that Ebbinghaus' power function is restarted. This is the case both when the student gives a completely incorrect answer (RQ value = 0) and when the student gives an incorrect answer which is close to the correct one (RQ value = 1). When a student gives a correct answer, the increase of his or her Retention Factor depends on his or her profile and more specifically on his or her Memorisation Ability factor. In particular, if the student's RQ is 2 and he or she has a very weak memory then the RF will be increased by 3 points (extending the lifespan of the memory of a fact by about a minute), while if he or she has a very strong memory the RF will be increased by 15 (extending the lifespan by over 6 minutes). These formulae for the calculation of the RF give the cognitive model a more "personal" aspect since they are not generic but based on the student's profile.

The previously mentioned individualisations were made and refined based on empirical research data. In the case of an RQ of 0 or 1 (wrong answer) there is strong evidence that the student has forgotten the fact, and thus we calculate the RP as if the student had first seen the fact at the time he or she gave the answer, then we lower it and assign the new value as the RF. Finally, we also reset the time.

Indeed, in the case of an RQ of 0 or 1 we achieve a rapid loss of retention by resetting the time in the formulae for the calculation of RF. This is due to the logarithmic nature of the Ebbinghaus power function. Moreover, the RP is also decreased by 10 and 5 respectively. As a consequence of these modifications, the resulting RP will almost definitely correspond to a "for-

gotten" fact (one with a final RP lower than 70). After these modifications, the only time the system may come up with a fact of an RP value greater than 70 is if the student has an MA of 4 (very strong memory) and the answer is among the latest ones given (thus the time difference that is applied to the Ebbinghaus function is small and as a consequence the retention decline smaller too). Our experiments have shown that such cases are very rare and when they do occur the student has accidentally answered incorrectly.

If a student has an RQ of 2 and 3 then we know that he or she has answered correctly, and we can thus raise the RF value in accordance to the student's personal MA value. The stronger the student's memorisation abilities are the higher we can raise the RF. There are cases when a correct answer may be given as the result of a "lucky" choice, but the system has adequate information about the student's profile to track such cases down and remove them from the retention process.

CONCLUSIONS

In this paper, we have shown how simulated students can be created and used for the enhancement of the performance and the software engineering process of an ITS. For this purpose the student modelling component of the ITS was used to allow a simulated student to be created with the characteristics from each individual student model as input. In particular, it takes into account what the student has been able to remember from the material taught as this has been recorded in his or her performance on tests. This information is combined with principles of cognitive psychology, giving the ITS insight on what students may remember from the material being taught to them.

This memory information is used by the system to adapt the teaching process accordingly. Depending on what a student does or does not remember, the system proceeds by presenting new course material or repeating certain parts of the course material that have already been taught. In this way, the educational software application becomes more personalised and adaptive by responding appropriately to each individual student's needs regarding the way the course material is being taught to him or her.

Moreover, and most importantly, the simulated students are used instead of real students for the evaluation of the ITS. In this way, the developers of the ITS (teachers and ITS designers) may find out what needs to be correct-

ed in a subsequent version of the ITS without cost to the educational process. Indeed, evaluating a course on real students is not fair for them since they would have to suffer the consequences of all the possible mistakes that the developers may have made and would have corrected if they had discovered them earlier. As a result, the simulated students may make a major contribution to the software engineering process of educational applications by encouraging and facilitating many iterations of the software life-cycle and many evaluations. This process can guarantee that the end result will be of higher quality than it would be without the use of simulated students.

REFERENCES

Amory, A., Naicker, K., Vincent, J. & Claudia, A. (1998). Computer Games as a Learning Resource. *Proceedings of ED-MEDIA, ED-TELECOM 98*, World Conference on Education Multimedia and Educational Telecommunications, Vol. 1, pp. 50-55.

Anderson J.R. (2000). *Learning and Memory: An Integrated Approach* (2nd ed.). John Wiley & Sons, Inc.

Conati, C. & Zhou, X. (2002). Modeling Students' Emotions from Cognitive Appraisal in Educational Games. In S. A. Cerri, G. Gouarderes and F. Paraguacu (Eds.): *Intelligent Tutoring Systems 2002, LNCS, 2363*, pp. 944-954, Springer-Verlag Berlin Heidelberg 2002.

Cumming, G. & McDougall, A. (2000). Mainstreaming AIED into Education? *International Journal of Artificial Intelligence in Education, 11*, 197-207

Dix, A., Finlay, J., Abowd, G., & Beale, R. (1993). *Human-Computer Interaction.* New York: Prentice-Hall.

Ebbinghaus, H. (1998). Classics in Psychology, 1885: Vol. 20, *Memory*, R.H. Wozniak (Ed.), Thoemmes Press, 1998

Gilbert, L. (1999). Some Valuable Lessons from the Teaching and Learning Technology Programme in the U.K, *Journal of Interactive Learning Research, 10,* 1, 67-85.

Goldstein, I. (1982). The Genetic Graph: A Representation for the Evolution of Procedural Knowledge. In D. Sleeman & L. Brown (Eds.), *Intelligent Tutoring Systems.* London: Academic Press.

Klatzky, R.L. (1980). *Human Memory – Structure and Processes*, W.H. Freedman and Co., San Francisco.

Kruchten, P. (1999). *Rational Unified Process-An Introduction*, Addison-Wesley.

Laurillard D. (1995). Multimedia and the Changing Experience of the Learner, *British Journal of Educational Technology, 26*(3), 179-189.

Linton, M. (1979). Real-world Memory after 6 Years- Invivo Study of Very Long-term-memory. *Bulletin of the British Psychological Society, 32*, (Feb): 80.

Matthews, M., Pharr, W., Biswas G. & Neelakandan, (2000). USCSH: An Active Intelligent Assistance System, *Artificial Intelligence Review, 14*, 121-141.

Muntaz, S. (2001), Children's Enjoyment and Perception of Computer Use in the Home and the School. *Computers & Education, 36*, 347-362.

Quatrani, T. (1998), Visual Modeling with Rational Rose and UML, Addison-Wesley.

Rivers, R. (1989). Embedded User Models – Where Next? *Interacting with Computers, 1*, 14-30.

Sommerville, I. (1992). *Software Engineering*, Reading, Mass.: Addison-Wesley Pub. Co.

Stansfield, J.C., Carr, B., & Goldstein, I.P. (1976). Wumpus Advisor I: A First Implementation of a Program that Tutors Logical and Probabilistic Reasoning Skills. At Lab Memo 381. Massachusetts Institute of Technology, Cambridge, Massachusetts.

Virvou, M. & Manos, K. (2003a). A Simulated Student-player in Support of the Authoring Process in a Knowledge-based Authoring Tool for Educational Games. *Proceedings of the 3rd IEEE International Conference on Advanced Learning Technologies 2003*, Athens, Greece, July 9-11, 2002.

Virvou, M. & Manos, K. (2003b). Individualising a Cognitive Model of Students' Memory in Intelligent Tutoring Systems. To appear in *Lecture Notes in Artificial Intelligence*, Editors V. Palade, R.J. Howlett, L.C. Jain: *Knowledge-based Intelligent Information and Engineering Systems*, 2003, Springer-Verlag.

Virvou, M., Manos, C., Katsionis, G. & Tourtoglou K. (2002). VR-ENGAGE: A Virtual Reality Educational Game that Incorporates Intelligence. *Proceedings of the 2nd IEEE International Conference on Advanced Learning Technologies 2002*, Kazan, Russia.

VanLehn, K., Ohlsson, S. & Nason, R. (1994). Applications of Simulated Students: An Exploration, *Journal of Artificial Intelligence in Education, 8*, 262-283.