

Personalized Multi-Student Improvement Based on Bayesian Cybernetics

Vassilis G. Kaburlasos^{*1}, Catherine C. Marinagi², and Vassilis Th. Tsoukalas¹

¹Technological Educational Institution of Kavala

Department of Industrial Informatics

GR-65404 Kavala, Greece

Emails: {vgkabs,vtsouk}@teikav.edu.gr

²Technological Educational Institution of Halkida

Department of Logistics

GR-32200 Thiva, Greece

Email: marinagi@teihal.gr

Corresponding Author

Name: Vassilis G. Kaburlasos

Postal Address: Department of Industrial Informatics
Technological Educational Institution of Kavala
GR 654 04 Kavala
Greece

Phone Number: +30 (2510) 462-320

Fax Number: +30 (2510) 462-348

E-mail Address: vgkabs@teikav.edu.gr

* Corresponding Author Fax. no. + 30 (2510) 462-348, E-mail address: vgkabs@teikav.edu.gr

Personalized Multi-Student Improvement Based on Bayesian Cybernetics

Abstract

This work presents innovative cybernetics (feedback) techniques based on Bayesian statistics for drawing questions from an Item Bank towards personalized multi-student improvement. A novel software tool, namely *Module for Adaptive Assessment of Students* (or, *MAAS* for short), implements the proposed (feedback) techniques. In conclusion, a pilot application to two Computer Science courses during a period of four years demonstrates the effectiveness of the proposed techniques. Statistical evidence strongly suggests that the proposed techniques can improve student performance. The benefits of automating a quicker delivery of University quality education to a large body of students can be substantial as discussed here.

Keywords: Architectures for educational technology systems; Intelligent tutoring systems; Interactive learning environments; Teaching/learning strategies

1. Introduction

The total number as well as the distribution of students admitted (from secondary) to tertiary education in Greece is currently regulated by central government policies. In this context, a number of higher educational institutions face the challenge of providing education to large numbers of students given limited teaching resources. For instance, the fairly new department of Industrial Informatics at Technological Educational Institution of Kavala has started its operation in academic year 1999-2000 by admitting a total number of 226 students. In the following years the number of admitted students steadily increased to reach the not-easily manageable, due to a disadvantageous student+teacher ratio, number of 349 students in academic year 2002-2003 (Kaburlasos et al., 2003). Currently, the aforementioned number has stabilized to over 300 students per year.

A similar situation also arises elsewhere, where it was reported that the number of students in higher education may rise faster than an increase in teaching resources including academic staff (Daly & Horgan, 2004; Davies, 1999; Smailes, 2003; Valenti, 2003). Hence, the aforementioned challenge remains. A more ambitious challenge would be to improve, in addition, the quality of supplied education while, at the same time, increase *student throughput*; the latter is defined here as the percentage of students who pass the final exam.

The benefits, including economic ones, of automating a quicker delivery of high quality education to a large body of students can be substantial. In particular, a quicker delivery of University quality education can imply substantial financial savings; moreover, the graduating students can contribute to production sooner. This work reports on both the development and pilot application of innovative techniques to meet the aforementioned challenges.

One popular instrument for dealing with the problem of academic staff shortage is *Computer-Based Assessment (CBA)*, also referred to as *Computer Assisted Assessment (CAA)*. CBA/CAA uses computerized testing for assessing student knowledge level and, thus, reducing scoring workload. More specifically, instead of receiving a test with a fixed set of *Items* (i.e. questions), students receive a set of Items randomly selected from an Item Bank (Brown et al., 1999; Thelwall, 2000; Sclater & Howie, 2003). Several pedagogic as well as strategic advantages of CBA over *Paper-and-Pencil Tests* have been presented comparatively including both unbiased and accurate scoring, containment of cheating, and immediate feedback regarding student performance (Harvey & Moge, 1999; Russell & Haney, 2000). However, other studies, regarding equivalence of CBA with paper-and-pencil exams, give conflicting results. In particular, individual differences may still affect the aforementioned equivalence, even though student familiarity with computers is increasing rapidly (McDonald, 2002). Therefore, CBA tests have not thoroughly replaced paper-and-pencil tests. Of particular interest here is an employment of feedback techniques towards student improvement. Consider the following definition regarding *cybernetics* (Wiener, 1965).

Cybernetics is the study of *feedback* and derived concepts such as *communication* and *control* in living organisms, machines and organizations. Its focus is how an entity (digital, mechanical, or biological) processes information, reacts to information, and changes to better accomplish its tasks.

Inspired from the above definition, we developed feedback techniques for student improvement as follows. We presented the “learning state” of an individual student by a vector of probabilities computed by Bayesian statistics as detailed below. Then, the “learning state” vector was used to compute a (student) customized *focus-of-attention* vector of probabilities. The latter (vector) was used to randomly draw questions from an Item Bank which (questions) were fed back to an individual student towards improving incrementally a student’s “learning state”. We review potentially useful technology for student testing, next.

In *Adaptive Testing*, which is also referred to as *tailored testing*, tests are adjusted to the competence level of individual students (Lord, 1980). In *Computer Adaptive Testing (CAT)*, in particular, adaptive tests are administered by computers (Lilley et al., 2004; Linacre, 2000).

Items are administered one-at-a-time such that the selection of an Item depends on student response to previous Items.

Adaptive assessment retains the advantages of CBA; additional benefits include accurate as well as reliable estimation of student competence, reduction of testing time, and avoidance of easy/difficult Items that may cause boredom/stress of students. It is recognized that CAT may facilitate integration of assessment with learning. Also, CAT has been used as a diagnostic module in personalized e-learning environments such as in KOD (Sampson et al., 2002), INSPIRE (Papanikolaou et al., 2002), PEL-IRT (Chen et al., 2004) as well as to Web-based tutoring systems including ELM-ART (Weber & Brusilovsky, 2001) and SEATS (Davidovic et al., 2003). Of particular interest here is *self-assessment* towards student improvement. Note that the employment of self-assessment for student improvement was pursued lately, but not conclusively (Pérez et al., 2005). Furthermore, different authors have reported encouraging results regarding student improvement by self-assessment, nevertheless without feedback (Guzmán et al., 2007).

Critical in the design of CAT for large-scale educational tests is *content balancing*, the latter is a prespecified percentage of Items from each (different) content area, including the popular *constrained CAT (CCAT)* procedure for Item selection (Kingsbury and Zara, 1989). In particular, CCAT selects an “optimal” Item from the content area whose current exposure rate is the furthest under its corresponding (prespecified) percentage. The *modified constrained CAT (MCCAT)* was proposed for eliminating predictability of the sequence of content areas of CCAT by selecting an optimal Item from all the content areas that still have quota not fully used up (Leung et al., 2000). A more recent work has presented a study that compares those content balancing methods with the *modified multinomial model (MMM)*, which incorporates randomness (Leung et al., 2003). The results indicate a systematic effect of content balancing method on Item Bank (or pool) utilization, whereas MMM appears to reduce both the predictability of Item contents sequence and the number of overexposed Items.

Frequently, CAT employs *Item Response Theory (IRT)* as a supportive model (Lord, 1980). For instance, IRT was used to determine the probability of a correct response, update student level, and select the next Item. A three parameter logistic (3-PL) IRT model was used to calculate the probability of a correct response to a given Item, based on both student competence and three Item parameters, namely Item difficulty, Item discrimination, and a pseudo-guessing parameter. IRT was also applied to personalized e-learning environments. During the last two decades, commercially available software for implementing IRT-based CAT was developed including FastTEST Pro and MicroCAT (Assessment Systems Corporation, 2001). Furthermore, the feasibility of IRT in the Graduate Record Examination (GRE) was investigated (McKinley & Kingston, 1987).

Adaptive Mastery Testing (AMT) is a CAT that measures the mastery of a set of skills. The decision is to either classify an individual into one of two “status” categories, i.e. master or nonmaster, or to continue testing. Both the Item selection and the stopping rule are adaptive. At least two IRT-based strategies were primarily used in AMT for Item selection. First, the Bayesian Item selection strategy selects an Item that minimizes posterior variance of a student’s last ability estimate (Owen, 1975) – Note that posterior variance may be obtained via Owen’s Bayesian scoring algorithm, or other numerical techniques developed later. Second, the Item to be selected next is the one that maximizes the amount of information in a student’s last ability estimate (Kingsbury & Weiss, 1983). Both aforementioned IRT-based Item selection strategies use confidence intervals of student’s latent ability either to decide student categorization or to continue testing.

Sequential Mastery Testing (SMT) is another type of mastery testing, where the next Item is selected randomly, moreover the stopping rule is adaptive. A binomial distribution was proposed assuming that Items have identical properties (Ferguson, 1969). Moreover, a varying Item difficulty and discrimination was proposed with an IRT-model (Reckase, 1983). Errors in SMT can be of either *Type-I* (i.e. classify a master as nonmaster) or *Type-II* (i.e. classify a nonmaster as master) – Note that a Type-II error is considered more serious than a Type-I error. Various studies have proposed different “expert-defined” Type- I and II error costs (Ferguson 1969; Lewis & Sheehan 1990; Reckase 1979). In case that more than two student categories are considered, the cost of assigning a student of category j to category i can be given by function $|i-j|$ for all i,j (Rudner, 2002). The latter error cost function was employed in this work because we also consider several student categories.

Another approach in SMT applies *Bayesian Decision Theory* (BDT) (Lewis & Sheehan, 1990; Vos, 2000; Rudner, 2002, 2003). BDT assumes prior probabilities for a student before starting a test. Then, each time a single Item is answered, a new student competence level is estimated “a posteriori” using the aforementioned Item answer as well as prior probability values. Note also that response behaviour was modelled in this context using a 3-PL model from IRT (Lewis & Sheehan, 1990). An alternative application of BDT was proposed such that a random- rather than a ITR- based adaptive Item selection procedure is employed (Vos, 2000; Rudner, 2002, 2003).

Using simulated Item response data, adaptive BDT testing procedures were compared with IRT, in terms of classification accuracy, with favourable results for BDT (Rudner, 2002). Furthermore, a pilot study using real world data has demonstrated that simple BDT using *Maximum A posteriori Probability* (MAP) – for selecting the category with the highest posterior probability – as a decision rule can yield high classification accuracies compared to IRT (Rudner, 2003). *Minimum expected cost* defines the optimal Item to be selected next as

the Item with the lowest expected cost. The latter approach is often employed in SMT (Lewis & Sheehan, 1990; Vos, 2000; Rudner, 2002, 2003). An important comparative advantage of BDT is that the latter (BDT) does not need IRT's formalities such as confidence intervals of student ability and prespecified cut-offs, furthermore BDT does not assume student competence to be normally distributed. The only limitation of BDT is *Item independence*¹, which, however, is outweighed by the simplicity of BDT.

Bayesian decision-making is instrumental in this work. Note that a number of education-related publications have employed Bayes formula, e.g. for modelling student behaviour (Xenos, 2004) as well as for detecting student learning styles (García et al., 2007) based on Bayesian networks. The latter (networks) were also used together with CAT for improving estimation of a student's *cognitive state*, which is assumed to be "stationary" (Millán and Pérez-de-la-Cruz, 2002). Moreover, Bayes formula was used for (intelligent) classification that supports Web-based thematic learning (Huang et al., 2007). Nevertheless, the employment of Bayes formula here is different since it is used for computing the entries of a student "learning state" vector, which (vector) directs student learning as detailed below.

This work reports on the development and application of a novel software tool, namely *Module for Adaptive Assessment of Students* (MAAS), for personalized adaptive multi-student improvement based on innovative Bayesian cybernetics, i.e. feedback, techniques. MAAS is embedded in a software platform, namely *Platform for Adaptive and Reliable Evaluation of Students* (PARES). A preliminary version of PARES, was implemented in the Delphi programming language (Kaburlasos et al., 2003). The first version of PARES was developed in the Java programming language (Kaburlasos et al., 2004; Marinagi et al., 2004). The second version of PARES here incorporates MAAS.

The paper is organized as follows. Section 2 presents terminology and definitions. Section 3, after introducing a simplifying index notation, describes different types of student assessment. Section 4 outlines software platform PARES with emphasis on (module) MAAS. Section 5 presents a pilot application and statistical hypothesis testing results. Section 6 concludes by summarizing the contribution of this work including future work. Appendix A details initialization of useful parameter values, and Appendix B details the computation of two useful vectors.

¹ In particular, it is assumed that the response to an Item is *independent* of the responses to other Items.

2. Preliminaries and terminology

This section summarizes clearly the definitions of a number of notions to be employed in the following chapter for describing our proposed student assessment techniques.

The person who takes a test is often called *examinee*, *participant*, *user*, or *learner*. Here we, alternatively, employ the term *student* because higher education students are involved.

Definition 1. An *Item* is a piece of information including both a *Question* and a set of alternative *Answers*.

Fig. 1 shows an Item example regarding a course in *Software Engineering*.

Item Type may be *multiple-choice*, *multiple-response*, *matching*, *true/false*, *fill-in-the-blanks*, etc. This work employed solely multiple-choice and multiple-response Item Types. In particular, first, for a multiple-choice Item Type we assume that a student response is “right” when the unique correct answer is selected, whereas it is “wrong” otherwise. Second, for a multiple-response Item Type we assume that a student response is “right” when all the correct answers are selected, whereas it is “wrong” otherwise. A “blank” response is also acceptable and treated differently from a “wrong” one. In conclusion, a student response a_n to an Item τ_n may take on three values, namely $a_n=R$ for right, $a_n=W$ for wrong, and $a_n=B$ for blank.

Item Difficulty quantifies how difficult it is considered to provide the correct answer to an Item. Item Difficulty values range in the interval $[0,100]$, where 0 and 100 correspond, respectively, to the easiest and most difficult Items to answer as explained in Appendix A.2.

Experts, i.e. (typically) course instructors, compose Items and store them in an Item Bank (IB).

Definition 2. An *Item Bank (IB)* is a repository of Items regarding an *educational subject*.

We remark that an *educational subject* may alternatively be called *course*.

Items in an IB are grouped in *Units* “uniform” in the sense that all the Items in a Unit are both thematically similar and have similar difficulty values calculated as detailed in Appendix A.2. Moreover, Units are grouped in *Chapters*. Hence, a *tree-structured* IB is produced per course.

Fig. 1, Fig. 2, and Fig. 3 illustrate how an Item may appear under different circumstances. In particular, Fig. 1 shows how an Item is stored in the Item Bank (IB). Note that Fig. 1 displays a Question with four alternative Answers; for each answer the corresponding correctness value (yes/no) is specified followed by a Comment including documentation and/or a URL address for further information. An Item has two properties, namely *Item Type* and *Item Difficulty* shown at the top of Fig. 1. Fig. 2 shows how the Item of Fig. 1 is displayed on a student monitor. Finally, Fig. 3 shows what the software feeds (back) when a student provides

a “wrong” answer during self-assessment; more specifically, Fig. 3 displays the correct answer, the (wrong) student answer, and a comment regarding the student’s wrong answer.

We have assumed J levels (intervals) of *student competence* denoted by L_1, \dots, L_J . For example, given a range of student scores in the interval $[0, 10]$, and assuming $J = 20$, level L_j may be the interval $L_j = [0.5(j-1), 0.5j]$ for $j=1, \dots, 19$ and level L_{20} may be the interval $L_{20} = [9.5, 10]$.

Before introducing additional terminology, recall *Bayes* formula next.

$$p(m_j / d) = \frac{p(d / m_j) p(m_j)}{p(d)}, \quad (1)$$

where $m_j, j=1, \dots, J$ is a model which may produce datum d ; $p(d/m_j)$, namely *likelihood*, is the probability of observing datum d given model m_j ; $p(m_j)$, namely *prior*, represents all that is known before any data becomes available; $p(d)$, namely *evidence*, describes the probability of observing datum d . In conclusion, Bayes formula computes *posterior* $p(m_j/d)$, i.e. the probability of a specific model $m_j, j=1, \dots, J$ given datum d . In this work, a model m_j in Bayes formula above corresponds to a student competence level (interval) L_j . For convenience we introduce the following definitions.

Definition 3. *Likelihood* $p(a_n | L_j)$ is the probability that a student, of competence level $L_j, j \in \{1, \dots, J\}$, responds $a_n \in \{R, W, B\}$ to Item τ_n .

Definition 4. *Prior* $p(L_j)$ is the probability that a student is of competence level $L_j, j \in \{1, \dots, J\}$.

We assume that the events “a student is of competence level $L_j, j \in \{1, \dots, J\}$ ” are mutually exclusive, moreover their corresponding probabilities sum up to one.

Definition 5. *Evidence* $p(a_n)$ is the probability that a student responds $a_n \in \{R, W, B\}$ to Item τ_n .

Definition 6. *Posterior* $p(L_j | a_n)$ is the probability a student is of competence level $L_j, j \in \{1, \dots, J\}$, given the student’s response $a_n \in \{R, W, B\}$ to Item τ_n .

Let $\mathbf{a} = a_{n_1}, \dots, a_{n_K}$ be a series of student responses to Items $\tau_{n_1}, \dots, \tau_{n_K}$, respectively. Then, $p(L_j | \mathbf{a})$ denotes the (posterior) probability that the student is of competence level $L_j, j \in \{1, \dots, J\}$ given \mathbf{a} . Furthermore, $p(\mathbf{a} | L_j)$ denotes the (likelihood) probability that a student produces a series \mathbf{a} of responses a_{n_1}, \dots, a_{n_K} to Items $\tau_{n_1}, \dots, \tau_{n_K}$, respectively, given that the aforementioned student is of competence level $L_j, j \in \{1, \dots, J\}$. In this work, we assume

independence of student responses to different Items; hence, $p(\mathbf{a} | L_j) = \prod_{k=1}^K p(a_{n_k} | L_j)$.

Definition 7. Let S_e be the number of students tested in a course e . Then, the *competence state* vector of a student $i \in \{1, \dots, S_e\}$ is defined as $s_e(i) = (p(L_1|\mathbf{a}), \dots, p(L_J|\mathbf{a}))$.

We remark that the entries of vector $s_e(i)$ in definition 7 are computed by Eq. (B1), in Appendix B, and they quantify a student's competence by the probabilities the student in question belongs to different competence levels.

Definition 8. Let $s_e(i) = (p(L_1|\mathbf{a}), \dots, p(L_J|\mathbf{a}))$ be a student competence state vector. Then, the corresponding student *score* is calculated as $g_e(i) = \sum_{j=1}^J \omega_j p(L_j|\mathbf{a})$, where $\beta_j \in L_j$.

We remark that in the context of this work we have used $\beta_j = \sup\{x: x \in L_j\}$, where \sup denotes the *supremum* (*least upper bound*) of the set $\{x: x \in L_j\}$; e.g. $\sup\{x: x \in [3.5, 4)\} = 4$.

Definition 9. *Unit-content* $\pi(u_i)$ is the percentage of Items in a test to be selected from Unit u_i .

3. Student assessment techniques

In the first place, we introduce a simplifying index notation regarding both Items and Units.

Let the Item Bank IB_e in a specific educational subject (i.e. course) e include C_e Chapters. Moreover, let a Chapter κ_c , $c \in \{1, \dots, C_e\}$ include U_c Units. Finally, let a Unit $\mu_{c,u}$, $c \in \{1, \dots, C_e\}$, $u \in \{1, \dots, U_c\}$ include $I_{c,u}$ Items.

First, a Unit $\mu_{c,u}$ is uniquely identified by two indices, namely $c \in \{1, \dots, C_e\}$ and $u \in \{1, \dots, U_c\}$.

Therefore, a single index m results in for a Unit $\mu_{c,u}$ as follows

$$m = u + \sum_{k=1}^{c-1} U_k . \quad (2)$$

Second, an Item $\tau_{c,u,i}$ is uniquely identified by three indices, namely $c \in \{1, \dots, C_e\}$, $u \in \{1, \dots, U_c\}$ and $i \in \{1, \dots, I_{c,u}\}$. Hence, a single index n results in for an Item $\tau_{c,u,i}$ as follows

$$n = i + \sum_{j=1}^{u-1} I_{c,j} + \sum_{k=1}^{c-1} \sum_{j=1}^{U_k} I_{k,j} . \quad (3)$$

In conclusion, an Item will be denoted as τ_n , where $n \in \{1, \dots, N_e = \sum_{k=1}^{C_e} \sum_{j=1}^{U_k} I_{k,j}\}$, whereas a Unit

will be denoted as μ_m , where $m \in \{1, \dots, M_e = \sum_{k=1}^{C_e} U_k\}$.

The following example demonstrates the computation of indices m and n .

Example

Let a tree-structured Item Bank IB_e in a course e include three Chapters, namely κ_1 , κ_2 , and κ_3 (hence, $C_e=3$). Furthermore, let Chapter κ_1 include three Units $\mu_{1,1}$, $\mu_{1,2}$ and $\mu_{1,3}$ (hence, $U_1=3$); let Chapter κ_2 include two Units $\mu_{2,1}$ and $\mu_{2,2}$ (hence, $U_2=2$); and, let Chapter κ_3 include four Units $\mu_{3,1}$, $\mu_{3,2}$, $\mu_{3,3}$ and $\mu_{3,4}$ (hence, $U_3=4$). Finally, let each Unit include (the same number of) four Items (hence, $I_{c,u}=4$, where $c \in \{1,2,3\}$ and $u \in \{1, \dots, U_c\}$). Then,

1) The single index of the second Unit $\mu_{3,2}$ of the third Chapter κ_3 equals

$$m = 2 + U_1 + U_2 = 7, \text{ and}$$

2) The single index of the first Item $\tau_{3,2,1}$ in second Unit $\mu_{3,2}$ of the third Chapter κ_3 equals

$$n = 1 + (I_{3,1}) + (I_{1,1} + I_{1,2} + I_{1,3} + I_{2,1} + I_{2,2}) = 25. \quad \blacksquare$$

We point out that a single index is preferable because it allows the development of simpler expressions for computing useful parameters below.

3.1. Types of student assessment

In the context of this work we have assumed two types of student assessment, namely *exam-assessment* and *self-assessment*. First, *exam-assessment* regards a scheduled exam whose objective is to give marks to students. Here, a *focus-of-attention* vector is defined by an instructor and it remains constant during an exam, the same for all participating students. Hence, in the context of this work, exam-assessment is always *non-adaptive*. Second, *self-assessment* regards an individual student's self-motivated study towards self-improvement. Here, a *focus-of-attention* vector is updated, after each student response, towards improving student competence. Hence, in the context of this work, self-assessment is always *adaptive*. The utility of a *focus-of-attention* vector is explained next.

A *focus-of-attention* vector $fAt = (p(\mu_1), \dots, p(\mu_{M_e}))$ is used to randomly draw Items from the Item Bank as follows. Let an Item Bank include seven Units and let a *focus-of-attention* vector be $fAt = (0.2, 0.18, 0.15, 0.25, 0.22, 0, 0)$ – Note that $0.2 + 0.18 + 0.15 + 0.25 + 0.22 + 0 + 0 = 1$. Then, an Item is selected from Unit 1 with probability 0.2, from Unit 2 with probability 0.18, from Unit 3 with probability 0.15, from Unit 4 with probability 0.25, from Unit 5 with probability 0.22, whereas no Item can be selected from Units 6 and 7. Within a Unit, an Item is selected randomly according to the *uniform* probability distribution.

3.2. Exam-assessment

The major advantage of using a *focus-of-attention* vector for selecting Items during an exam is that different tests of a similar level of difficulty can be drawn. Therefore, it becomes feasible to alleviate “fairly” the problem of plagiarism regarding students who sit in front of

computer monitors tightly next to one another. An exam-assessment lasts a number of minutes, which (number) is calculated based on Item duration $UID(\tau_m)$ (see in Appendix A.2).

We point out that we have used PARES only for midterm exam-assessment(s). In other words, we have carried out the final exam in the traditional manner using paper-and pencil.

3.3. Self-assessment based on Bayesian cybernetics

Consider the block-diagram in Fig.4. An instructor-defined “reference score” R is compared with the “measured score” $g_e(i)$ of a student- i . For nonpositive error $R-g_e(i) \leq 0 \Leftrightarrow R \leq g_e(i)$ the closed-loop opens; hence, self-assessment stops because the objective of increasing a student score to level R (or above) was met. Otherwise, for positive error $R-g_e(i) > 0 \Leftrightarrow R > g_e(i)$, the corresponding student’s *focus-of-attention* vector is updated as detailed in Appendix B.2. Based on the latter vector, an Item (question) is drawn randomly from the Item Bank and is posed to the student. The student response is employed to compute the new student *competence state* vector. The latter is used to compute the new score $g_e(i)$. The previous cycle repeats. Fig. 5 details in a flow-chart the corresponding algorithm.

The major advantage of using a *focus-of-attention* vector for generating the next Item during self-assessment is that we can direct student-learning towards a “passing” student score R (in this work we used $R = 5$) by concentrating student attention on where a student mostly needs to be improved on.

Useful parameters including *likelihoods* $p(a_n|L_j)$ and *Unit Item Delays* $UID(\mu_m)$, $m=1, \dots, M_e$ are initialized as detailed in Appendix A. Moreover, a student *competence state* vector $s_e(i) = (p(L_1|\mathbf{a}), \dots, p(L_J|\mathbf{a}))$ can be computed as detailed in Appendix B.1.

Each time a student responds to an Item then 1) the student *competence state* vector $s_e(i)$ is recomputed by Eq. (B1) using all of the student’s previous answers, and 2) “personalized” student *evidence* (probabilities) $p_p(a_n)$ are computed by the following version of Eq. (A1).

$$p_p(a_n) = \sum_{j=1}^J p(a_n | L_j) p(L_j | \mathbf{a}), \quad (4)$$

where $a_n \in \{R, W, B\}$.

We remark that the likelihoods $p(a_n|L_j)$ in Eq. (4) are computed in Appendix A.1, whereas the *posteriors* $p(L_j|\mathbf{a})$ are computed by Eq. (B1) and are used in Eq. (4) as “personalized” student *priors*. In conclusion, evidence $p_p(a_n)$ is different from evidence $p(a_n)$ of Eq. (A1) in that $p_p(a_n)$ regards an individual student, whereas $p(a_n)$ regards the whole body of students in a course. In conclusion, evidence $p_p(a_n)$ is used to compute the *focus-of-attention* vector $fAt(i) = (p(\mu_1), \dots, p(\mu_{M_e}))$ for a student $i \in \{1, \dots, S_e\}$ by Eq. (B5).

4. The PARES software platform

This section outlines the architecture of software *Platform for Adaptive and Reliable Evaluation of Students* (or, *PARES* for short), which enables an implementation of the assessment techniques above. Here we summarize information useful for reproducing our techniques. For further details see in (Kaburlasos et al., 2003, 2004; Marinagi et al., 2004).

4.1. PARES modules

PARES is an interactive software including three modules, namely *Administrator Module*, *Instructor Module*, and *Module for Adaptive Assessment of Students* (or, *MAAS* for short) used by administrators, course instructors, and students, respectively. In the following we summarize the functions of PARES modules with emphasis on the novel MAAS.

An administrator uses the *Administrator Module* to keep track of useful “logistics” in the *system DB (database)* as well as to assign “access privileges” to instructors/students.

Course instructors use the *Instructor Module*, first, to update an Item Bank (IB) in a course and, second, to compose tests. An instructor may also define here useful “test parameters”.

Students use (*Module*) *MAAS* either for exam-assessment or for self-assessment. Pull-down menus enable students to: connect to the server after entering their username and password, choose the course to be examined in, and choose between exam-assessment or self-assessment. When questions appear on the screen (Fig. 2) students may supply answers to individual Items (questions), submit a test and, finally, print out a test report. Note that MAAS differs from other tools for adaptive assessment in that it applies Bayesian decision-making. A number of additional (minor) novelties include: MAAS considers not only right/wrong answers but also blank answers; students may be assigned different competence levels $L_j, j=1, \dots, J$; moreover, MAAS can combine adaptive Item selection with uniform (random) Item selection. Finally, MAAS uses a repository of Items organized hierarchically.

4.2. Application details

A system administrator had installed both the *system DB* and the *Administrator Module* on an Internet-accessed server. More specifically, the *Administrator Module* was installed on the server of a Computer Lab, whereas the modules used by instructors and students were installed either on a client machine at the Computer Lab or on a workstation with network access to the server. Since PARES is a client/server application, the aforementioned three modules can connect simultaneously to the *system DB*. Note that a large number of different students can access PARES simultaneously as well as “asynchronously” either for exam-assessment or for self-assessment. Username/password can secure user authentication.

For each course, an instructor may store a number of exams, however only one exam is *active* at a time. Exam duration is calculated automatically by summing up the corresponding *Unit Item Delays* (see in Appendix A.2) – We point out that *Unit Item Delays* remain constant during successive years of application. At the beginning of a test the “on screen” timer starts counting down. If time expires before deliberate student submission then student answers are submitted automatically. After all students have submitted their answers, the instructor prints an *evaluation report* including a list of participating students together with their scores.

It is possible to assign a different exam to each student thus to contain plagiarism during an exam. Moreover, the (random) Item selection techniques of PARES based on Unit-contents guarantee that all students will receive exams of similar difficulty.

Exam-assessment here is always *non-adaptive*, whereas self-assessment is always *adaptive*. Students may install MAAS on their home PCs and use a client application to connect to the laboratory server. Self-assessment may be customized by course instructors who initialize *Unit-contents* (see in Appendix A.3). An alternative is to allow students choose chapters for their self-assessment. However, in the latter case students cannot define *Unit-contents*.

During self-assessment, useful parameters are initialized based on massive student responses (see in Appendix A). Then, testing becomes adaptive based on (continually) updated, after each student response, *competence state* vectors (see in Appendix B.1). Self-assessment continues until certain termination criteria are met (Fig.5, step 3.5). Each time a student answers an Item, the student is informed of his/her personal score “on line” in order to encourage self-improvement.

4.3. Implementation details

PARES is a client/server software application developed in the Java programming language, which has the advantage of easy installation to various environments. The database of the application was developed on the DBMS Oracle Database Server 9i. Hence, the application of PARES does not depend on a particular operating system platform. The Oracle Database Server and the client/server application can be installed on Windows environment as well as on either Unix or Linux environments. Moreover, the environment where the PARES client/server application is installed is independent of the environment where the Oracle Database Server is installed.

5. Pilot application and results

We carried out experiments in classroom environment at the Department of Industrial Informatics of the Technological Educational Institution of Kavala, Greece for four consecutive academic years from 2002/2003 to 2005/2006 as follows.

5.1. Conditions

We employed PARES in two courses including, first, the 3rd semester course (*Introduction to Software Engineering*), which emphasizes methodologies for analysis, design, coding, testing, and maintenance of software and, second, the 5th semester course *Intelligent Systems*, which emphasizes analysis and design of expert and neural/fuzzy systems.

We minimized the effects of confounding variables as follows. Each year included, on the average, students of similar social and economic status. To avoid the effect of different teachers, the same researcher/author of this work taught one course during the four academic years from 2002/2003 to 2005/2006. The time given to a student to answer a specific Item (question) was constant as explained in section 5.2. Furthermore, extensive pretesting confirmed that two different years included, at the beginning of an experiment, students of similar capacity. Therefore, we can be reasonably confident that the statistical tests below are detecting the effect of our treatment and not some other significant difference between students of two different years.

The Item Bank (IB) of a course included around 200 Items. In the first academic year 2002/2003 we carried out only one final exam per course; whereas, in the following years we carried out, in addition, two midterm exam-assessments per course. A midterm exam, including 15 Items on the average, was meant for *formative* assessment (Cowie & Bell, 1999); that is, the results of a midterm exam were used by a course instructor for corrective (teaching) actions towards improving average student score in the next exam. The final exam was always carried out in a traditional manner using paper-and-pencil.

A student was trained, during a semester, to use PARES software fluently. Self-assessment was carried out regularly once a week in scheduled sessions. The last time for self-assessment, before an exam, was one week. Both the content and the degree of difficulty of the questions in a midterm exam were similar to the corresponding ones in the final exam.

5.2. Academic year 2002/2003

The first version of PARES was available in the Spring of 2003. Therefore, PARES was employed during academic year 2002/2003 only for computing initial estimates of useful

parameter values as explained in Appendix A. In particular, values for *Unit Difficulties* $UD(\mu_m)$, hence values for *Unit Item Delays* $UID(\mu_m)$, $m=1, \dots, M_e$ were computed during academic year 2002/2003 and kept constant thereafter.

A total number of 17 out of 26 students passed the *Software Engineering* course, therefore the corresponding student throughput was $\sim 65\%$ (Table 1, first line); whereas, 24 out of 71 students passed the *Intelligent Systems* course, therefore the corresponding student throughput was $\sim 34\%$ (Table 2, first line). The first line of Tables 1 and 2 also displays the corresponding final exam score statistics.

5.3. Academic year 2003/2004

The first version of PARES was available in the Fall of 2003 including self-assessment with a uniform (random) selection of Items from an Item Bank (IB). However, for comparison reasons, self-assessment was not used during academic year 2003/2004. In particular, during academic year 2003/2004, module MAAS was used only in two midterm exam-assessments.

Student throughput was $\sim 67\%$ in the *Software Engineering* course (Table 1, second line), whereas it was $\sim 37\%$ in the *Intelligent Systems* course (Table 2, second line). The second line of Tables 1 and 2 summarizes the corresponding midterm/final exam score statistics.

5.4. Academic year 2004/2005

The first version of PARES was used during academic year 2004/2005 including module MAAS for self-assessment with a uniform (random) selection of Items from an IB.

Student throughput was $\sim 73\%$ in the *Software Engineering* course (Table 1, third line), whereas it was $\sim 56\%$ in the *Intelligent Systems* course (Table 2, third line). The third line of Tables 1 and 2 summarizes the corresponding midterm/final exam score statistics.

Regarding the *Software Engineering* course, in particular, we remark that even though student throughput increased from academic year 2003/2004 ($\sim 67\%$) to academic year 2004/2005 ($\sim 73\%$), nevertheless the corresponding average student score in the final exam decreased from 5.54 to 5.20, respectively. We attributed the latter to the fact that several *Software Engineering* classes were missed during academic year 2004/2005; in conclusion, a larger number of students passed the corresponding course but with a smaller average score. Hence, we call the latter academic year “singular”, regarding the *Software Engineering* course only.

5.5. Academic year 2005/2006

The second version of PARES was used during academic year 2005/2006 including module MAAS for self-assessment with adaptive Bayesian selection of Items from an IB.

Fig. 6, in the foreground and the background, respectively, displays the percentage-wise distribution of 64 students over the range (0-10) of scores in steps of 0.5 in the 2nd midterm exam and the final exam of the *Software Engineering* course. Likewise, Fig. 7 displays the corresponding distributions of 133 students in the *Intelligent Systems* course. Note that student throughput was ~75% in the *Software Engineering* course (Table 1, last line), whereas it was ~62% in the *Intelligent Systems* course (Table 2, last line). The last line of Tables 1 and 2 summarizes the corresponding midterm/final exam score statistics.

5.6. Tests of statistical significance and discussion

Tables 1 and 2 suggest an increase of average student score in both the 2nd midterm exam and the final exam. In the following we substantiate, rigorously, any significance of our proposed treatment (i.e. student improvement techniques) by statistical hypothesis testing.

We tested statistically various combinations of two student groups, namely *treatment group* and *control group*. Each group regarded student scores in an exam in a course. The treatment group corresponded to a “later” year than the control group; for instance, had a control group been chosen in academic year 2003/2004 then a treatment group ought to be in a “later” academic year including 2004/2005 and 2005/2006. Quantile plots, reasonably linear for both groups, confirmed normality. In conclusion, the employment of a *t*-test was justified. More specifically, we used a *t*-test for two population means with variances both unknown and unequal (Kanji, 2006). The corresponding test statistic was $t = (\bar{x}_t - \bar{x}_c) / \sqrt{\frac{s_t^2}{n_t} + \frac{s_c^2}{n_c}}$, which

may be compared with Student’s *t*-distribution $t(df)$ with degrees of freedom given by $df =$

$$\left(\frac{s_t^2}{n_t} + \frac{s_c^2}{n_c} \right)^2 / \left[\frac{1}{n_t - 1} \left(\frac{s_t^2}{n_t} \right)^2 + \frac{1}{n_c - 1} \left(\frac{s_c^2}{n_c} \right)^2 \right],$$

where two independent random samples of size n_t and n_c were taken (i.e. one sample per student group) with sample means \bar{x}_t and \bar{x}_c , and variances s_t^2 and s_c^2 regarding the treatment group and the control group, respectively. Since we hope to show that the treatment group (with mean μ_t) scored better than the control group (with mean μ_c) we tested statistically the following hypotheses

$$\text{Null hypothesis } H_0: \mu_t = \mu_c$$

$$\text{Alternative hypothesis } H_a: \mu_t > \mu_c$$

The (1- α)-confidence interval in the one-tailed test of significance above regarding the mean

amount of improvement was computed by $(\bar{x}_t - \bar{x}_c) \pm t^* \sqrt{\frac{s_t^2}{n_t} + \frac{s_c^2}{n_c}}$, where t^* is the upper (1- α)

critical value for the $t(df)$ distribution.

Table 3 displays statistical hypothesis testing results regarding Final Exam Scores, whereas Table 4 displays the corresponding results regarding 2nd Midterm Exam Scores. A cell in either Table 3 or Table 4 shows the values of statistic t , degrees of freedom (df), and 95%/99% confidence intervals. All aforementioned values were computed by the formulas above based on the entries of Tables 1 and 2. Moreover, confidence% (in the alternative hypothesis $H_a: \mu_t > \mu_c$) values in a cell of either Table 3 or Table 4 were computed by a statistical computer package.

Table 3 shows that, first, any increase of student score in the Final Exam of the *Software Engineering* course is not statistically significant at level $\alpha=5\%$. Therefore, the null hypothesis H_0 cannot be rejected. In other words, our proposed treatment (i.e. student improvement techniques) does not appear to improve student performance in the *Software Engineering* course. Second, an increase of student score in the Final Exam of the *Intelligent Systems* course is statistically significant (usually) at level $\alpha=1\%$. Therefore, the null hypothesis H_0 cannot be accepted. In other words, our proposed treatment (techniques) appears to improve student performance in the *Intelligent Systems* course.

We point out that even though the null hypothesis H_0 cannot be rejected for the *Software Engineering* course, nevertheless the confidence% (in the alternative hypothesis $H_a: \mu_t > \mu_c$) in Table 3 typically increases from academic year 2003/2004 to academic year 2005/2006 (with the exception of the “singular” academic year 2004/2005). The aforementioned increase can be interpreted as “weak” statistical evidence for the effectiveness of our proposed treatment (techniques).

Table 4 shows that an increase of student score in the 2nd Midterm Exam of both courses *Software Engineering* and *Intelligent Systems* is statistically significant at level $\alpha=1\%$. Hence, the null hypothesis H_0 cannot be accepted. In other words, our proposed treatment (techniques) appears to improve student performance in a midterm exam in both courses.

In conclusion, Tables 3 and 4 present strong statistical evidence for the effectiveness of our proposed treatment, i.e. student improvement techniques. In particular, the incremental (functional) enhancements of PARES module MAAS from academic year 2002/2003 to academic year 2005/2006 described from section 5.2 to section 5.5, respectively, have resulted in a sustained improvement in 1) final exam student scores in the *Intelligent Systems* course (Table 3), and 2) midterm exam student scores in both courses *Software Engineering* and *Intelligent Systems* (Table 4). Note that a “95%/99% confidence interval” in either Table 3 or Table 4 should be interpreted as follows. For example, consider the confidence interval “ $0.86 \pm 0.43/0.61$ ” in the down-right corner of Table 3 regarding the *Intelligent Systems* course. Then, the corresponding “95% confidence interval” is $[0.86 - 0.43, 0.86 + 0.43] = [0.43,$

1.29]; furthermore, the corresponding “99% confidence interval” is $[0.86-0.61, 0.86+0.61] = [0.25, 1.47]$. Nevertheless, an improvement in final exam student scores of the *Software Engineering* course was not statistically significant at level $\alpha=5\%$ (Table 3). Finally, note that student throughput as well as the largest student score in an exam improved incrementally as shown in both Tables 1 and 2.

A comparison of the application of PARES module MAAS to the 3rd semester *Software Engineering* course, on one hand, with application to the 5th semester *Intelligent Systems* course, on the other hand, showed statistically significant improvements regarding midterm exam student scores in both courses (Table 4). The aforementioned improvements were attributed to self-assessment since the latter (self-assessment) was the only significant difference between a control student group and a treatment student group. More specifically, students in a treatment group can focus their attention on where they mostly need to be improved on. Furthermore, the statistical significance regarding improvements in the final exam student scores was “weak” for the *Software Engineering* course, whereas it was “strong” for the *Intelligent Systems* course (Table 3). The latter discrepancy, regarding final exam student scores in the two courses, was attributed to the specific contents of a course as well as to the much larger margin for potential improvement in the *Intelligent Systems* course. Finally, we also remark that the majority of students appeared to enjoy using PARES.

6. Conclusion and future work

This work has presented easily-transferable, innovative feedback techniques (based on Bayesian statistics) for drawing questions from an Item Bank towards personalized multi-student improvement. The proposed techniques were implemented by a modular software tool, namely *Platform for Adaptive and Reliable Evaluation of Students* (or, PARES for short) for distance assessment. PARES includes three software modules, which were briefly described above. The emphasis here was on student self-assessment by software *Module for Adaptive Assessment of Students* (or, MAAS for short). A pilot application in two Computer Science courses during four consecutive academic years has demonstrated statistically significant incremental improvements from year to year.

The benefits of automating a quicker delivery of University quality education to a large body of students can be substantial. For instance, a quicker delivery of University quality education can result in substantial financial savings in the supporting funds for education; moreover, the sooner graduating students can contribute sooner to production. Potential future work is discussed in the following.

Plagiarism during the exams is a considerable problem in the Greek higher education. A potential side-advantage of PARES is its capacity to contain plagiarism by presenting a different exam, of similar difficulty level as well as of similar content, to each student. Moreover, a PARES-based midterm exam can be used in order to “filter” a limited number of students to the final exam. Currently, PARES is employed “locally” for assessment in a supervised lab environment. Future plans include an extension of PARES for Web-based assessment. Furthermore, a course Item Bank (IB), which currently includes around 200 Items is expected to increase in the future. Future work might also consider designing analytically the “controller” block, in the closed (feedback) loop in Fig. 4, using techniques from automatic control systems in engineering (Houpis et al., 2005).

Acknowledgements

This work has been supported, in part, from 2003 to 2008 by the third European framework programme “Operational Programme in Education and Initial Vocational Training II” under project Informatics Studies Support contract no. 03-3-005.

Appendix A

This Appendix describes initialization of useful parameters. The first step is to administer the Items of an Item Bank IB_e , regarding a specific *educational subject* (i.e. *course*) e , to a random group of students at the end of a semester. Then, the student responses are massively recorded in order to initialize useful parameters for the students in the following semester. We point out that an initialization needs to be carried out as described previously in order to ensure that students have been fully exposed to the teaching material required for being able to answer to all Items in IB_e . Moreover, in order to compute “dependable” initial estimates, we have pursued a random selection of each Item $\tau_n, n \in \{1, \dots, N_e\}$ in IB_e at least ten times.

A.1. Initializing priors, likelihoods, and evidences

Given the scores of the abovementioned group of students, one can compute the proportion $p(L_j)$ of students in each level $L_j, j=1, \dots, J$ of student competence. Hence, *priors* $p(L_1), \dots, p(L_J)$ are initialized.

Given the scores of the abovementioned group of students, we can compute the proportion of students that responded correctly (i.e. $a_n=R$), wrongly (i.e. $a_n=W$), or did not respond at all (i.e. $a_n=B$) to an Item $\tau_n, n=1, \dots, N_e$, given student’s level of performance. Hence, for each Item τ_n the *likelihoods* $p(a_n|L_1), \dots, p(a_n|L_J)$, where $a_n \in \{R, W, B\}$, are initialized.

Based on the *priors* and *likelihoods* above, the *evidences* $p(a_n)$, $a_n \in \{R, W, B\}$ are computed as

$$p(a_n) = \sum_{j=1}^J p(a_n | L_j) p(L_j), \quad n=1, \dots, N_e \quad (A1)$$

A.2. Initializing Unit Difficulty (UD) and Unit Item Delay (UID)

A partition of Item Bank IB_e in Units is proposed initially by a course instructor such that the Items in a Unit are both thematically similar and have similar (in the instructor's judgement) difficulty. Further splitting of certain Units might occur based on evidence as detailed next.

Let μ_m , $m \in \{1, \dots, M_e\}$ be a Unit in a course e including I_m items. We compute the *average probability* $av(\mu_m)$ of correct responses in unit μ_m as follows: $av(\mu_m) = \frac{1}{I_m} \sum_{k=1}^{I_m} p(a_{n_k} = R)$,

where $\tau_{n_k} \in \mu_m$ – Note that the *evidence* probabilities $p(a_n=R)$, $n \in \{1, \dots, N_e\}$ are already known from Appendix A.1. We define the corresponding *Unit Difficulty* $UD(\mu_m)$ as

$$UD(\mu_m) = (1 - av(\mu_m)) \cdot 100 \quad (A2)$$

Note that in case the standard deviation of the series $p(a_{n_k} = R)$, $k=1, \dots, I_m$ is larger than an instructor-defined threshold σ_T then the instructor manually splits Unit μ_m to “uniform” (sub-)Units such that the standard deviation in each Unit is less than σ_T . In conclusion, the Item Bank IB_e is partitioned in sets of Items (i.e. Units) characterized by the same difficulty value $UD(\mu_m)$. Then, $UD(\mu_m)$ is fixed for all subsequent semesters.

In order to compute the expected test duration, we have estimated the time a student needs to respond to a single item of a Unit μ_m , $m \in \{1, \dots, M_e\}$, namely *Unit Item Delay* or $UID(\mu_m)$ for short, as follows. A course instructor measures the time a student needs to respond to the *easiest* and the *most difficult* Item in Item Bank IB_e , respectively, t_{\min} and t_{\max} . In conclusion, we define the expected time $UID(\mu_m)$ a student needs to respond to an Item of Unit μ_m as

$$UID(\mu_m) = t_{\min} + (t_{\max} - t_{\min}) \cdot UD(\mu_m) / 100 \quad (A3)$$

Since $UD(\mu_m)$ is fixed, as explained above, it follows that $UID(\mu_m)$ is, likewise, fixed.

A.3. Initializing Unit-contents

The Unit-contents (percentages) $\pi(u_i)$ of Definition 9, where $i=1, \dots, M_e$, are initialized by a course instructor/advisor according to teaching priorities.

Appendix B

This Appendix describes the computation of both a *competence state* vector and a *focus-of-attention* vector $fAt(i)$ for a student $i \in \{1, \dots, S_e\}$ in an *educational subject* (i.e. *course*) e .

B.1. Computation of a student competence state vector

Consider a specific student $i \in \{1, \dots, S_e\}$. In order to initialize the corresponding student *competence state* vector $s_e(i)$, during a self-assessment session, a series $\tau_{n_1}, \dots, \tau_{n_K}$ of a number “ K ” of Items is drawn randomly from the Item Bank IB_e . Let, respectively, the student supply a series $\mathbf{a} = a_{n_1}, \dots, a_{n_K}$ of responses, where $a_{n_k} \in \{R, W, B\}$ for $k=1, \dots, K$. We apply Bayes formula to compute the *posteriors* (probabilities) $p(L_j|\mathbf{a}), j \in \{1, \dots, J\}$ next.

$$p(L_j|\mathbf{a}) = \frac{p(\mathbf{a} | L_j)p(L_j)}{\sum_{i=1}^J p(\mathbf{a} | L_i)p(L_i)} = \frac{p(L_j) \prod_{k=1}^K p(a_{n_k} | L_j)}{\sum_{i=1}^J p(L_i) \prod_{k=1}^K p(a_{n_k} | L_i)}, j \in \{1, \dots, J\}, \quad (B1)$$

where all *priors* $p(L_j)$ and *likelihoods* $p(a_{n_k}|L_j)$, $n \in \{1, \dots, N_e\}$, $j \in \{1, \dots, J\}$ are known from Appendix A.1.

An estimate of a student *competence state* vector $s_e(i)$ is given by $s_e(i) = (p(L_1|\mathbf{a}), \dots, p(L_J|\mathbf{a}))$.

B.2. Computation of a student focus-of-attention vector

A *focus-of-attention* vector $fAt(i)$ equals $fAt(i) = (p(\mu_1), \dots, p(\mu_{M_e}))$, where a vector entry $p(\mu_m)$, $m=1, \dots, M_e$ is the probability of selecting an Item from Unit μ_m – Note that, within a Unit, an Item is selected randomly (uniformly). An entry probability $p(\mu_m)$ is calculated as

$$p(\mu_m) = \frac{Cost(\mu_m)\pi(\mu_m)}{\sum_{i=1}^{M_e} Cost(\mu_i)\pi(\mu_i)}, m=1, \dots, M_e, \quad (B2)$$

where percentage $\pi(\mu_m)$ was initialized by a course instructor as described in Appendix A.3. The objective of function $Cost(\mu_m)$ is to modify the instructor-defined percentages $\pi(\mu_m)$, $m=1, \dots, M_e$ based on individual student competence. More specifically, Eq. (B2) changes an instructor-defined percentage $\pi(\mu_m)$ in “proportion to” the corresponding $Cost(\mu_m)$. The following equations Eq. (B3)-(B5) were meant to produce larger costs for Units not characterized by right student answers. Hence, the *focus-of-attention* vector tends to direct student attention to the aforementioned Units by “more often” drawing (randomly) Items from those Units. Apparently, different functions $Cost(\mu_m)$ can be proposed.

In the interest of reducing computations, since a Unit in IB_e is “uniform” as explained in Appendix A.2, we considered a function $r: \{1, \dots, M_e\} \rightarrow \{1, \dots, N_e\}$ in order to represent a Unit μ_m by a randomly selected Item $\tau_{r(m)} \in \mu_m, m=1, \dots, M_e$. Let $c(i, j)$ be the cost/error of assigning competence level L_i to a student whose true competence level is L_j . Let $a_{r(m)} \in \{R, W, B\}$ be the student response to Item $\tau_{r(m)}$. The corresponding cost $C(a_{r(m)})$ was calculated as follows

$$C(a_{r(m)}) = \sum_{i=1}^J \sum_{j=1}^J c(i, j) p(L_i | a_{r(m)}) p(L_j | a_{r(m)}), \quad (B3)$$

where, $c(i, j) = |i-j|$ (Rudner, 2002); furthermore, *posterior* $p(L_j | a_{r(m)})$ was calculated as follows

$$p(L_j | a_{r(m)}) = \frac{p(a_{r(m)} | L_j) p(L_j | \mathbf{a})}{\sum_{k=1}^J p(a_{r(m)} | L_k) p(L_k | \mathbf{a})}, \quad j \in \{1, \dots, J\}, \quad (B4)$$

where the *likelihoods* $p(a_{r(m)} | L_j)$ are known from Appendix A.1, whereas *posteriors* $p(L_j | \mathbf{a})$ are computed by Eq. (B1) and are used as “personalized” student *priors*. In conclusion,

$$Cost(\mu_m) = \sum_{a_{r(m)} \in \{R, W, B\}} p_p(a_{r(m)}) C(a_{r(m)}) \quad (B5)$$

where the “personalized” student *evidence* probabilities $p_p(a_{r(m)})$ are computed by Eq. (4).

We remark that the above calculations are significantly reduced in practice because an advisor/instructor typically defines $\pi(u_i) = 0$ for many Unit-contents.

References

- Assessment Systems Corporation (2001). *The FastTEST Professional Testing System*. St. Paul, MN.
- Brown, S., Race, P., & Bull J. (Eds.) (1999). *Computer-Assisted Assessment in Higher Education*. London, UK: Kogan Page.
- Chen, C.-M., Lee, H.-M., & Chen, Y.-H. (2004). Personalized e-learning system using Item-Response Theory, *Computers and Education*, 44(3), 237-255.
- Cowie, B., & Bell, B. (1999). A model of formative assessment in science education. *Assessment in Education*, 6, 101-116.
- Daly, C., & Horgan J.M. (2004). An automated learning system for Java programming. *IEEE Transactions on Education*, 47(1), 10-17.

- Davidovic, A., Warren, J., & Trichina, E. (2003). Learning benefits of structural example-based adaptive tutoring systems. *IEEE Transactions on Education*, 46(2), 241-251.
- Davies, P. (1999). Learning through assessment: OLAL On Line Assessment and Learning. In *Proceedings of the Third Annual Computer Assisted Assessment Conference*, Loughborough, England.
- Ferguson, R. (1969). *Computer-Assisted Criterion-Referenced Measurement*. University of Pittsburgh Learning Research and Development Center, Pittsburgh.
- García, P., Amandi, A., Schiaffino, S., & Campo, M. (2007). Evaluating Bayesian networks' precision for detecting students' learning styles. *Computers and Education* 49(3), 794-808.
- Guzmán, E., Conejo, R., & Pérez-de-la-Cruz, J.-L. (2007) Improving student performance using self-assessment tests. *IEEE Intelligent Systems*, 22(4), 46-52.
- Harvey, J., & Moge, N. (1999). Pragmatic issues when integrating technology into the assessment of students. S. Brown, P. Race, & J. Bull (Eds.), *Computer-Assisted Assessment in Higher Education*, London: Kogan Page.
- Houppis, C.H, Rasmussen, S.J., & Garcia-Sanz, M. (2005) *Quantitative Feedback Theory: Fundamentals and Applications*, 2nd ed. CRC/Taylor & Francis.
- Huang, C.-J., Liu, M.-C., Chu, S.-S., & Cheng, C.-L. (2007). An intelligent learning diagnosis system for Web-based thematic learning platform. *Computers and Education* 48(4), 658-679.
- Kaburlasos, V.G., Marinagi, C.C., & Tsoukalas, V.T. (2004). PARES: A software tool for computer-based testing and evaluation used in the Greek higher education system. In *Proceedings of the Fourth IEEE International Conference on Advanced Learning Technologies* (pp. 771-773), Joensuu, Finland.
- Kaburlasos, V.G., Moussiades, L., Tsoukalas, V.T., Iliopoulou, A., & Alevizos, T. (2003). Adaptive technological education delivery and student examination based on machine learning tools. In *Supplementary Proceedings of International Conference on Artificial Neural Networks & International Conference on Neural Information Processing* (pp. 478-481), Istanbul, Turkey.
- Kanji, G.K. (2006). *100 Statistical Tests*, 3rd ed. London, UK: SAGE Publications.
- Kingsbury, G.G, & Weiss, D.J. (1983). A comparison of IRT-based adaptive mastery testing and a sequential mastery testing procedure. In: D.J. Weiss (Ed.), *New horizons in testing: Latent trait theory and computerized adaptive testing*. (pp. 257-283). New York: Academic Press.

- Kingsbury, G.G., & Zara, A.R. (1989). Procedures for selecting items for computerized adaptive tests. *Applied Measurement in Education*, 2, 359-375.
- Leung, C.K., Chang, H.H., & Hau, K.T. (2000). Content balancing in stratified computerized adaptive testing designs. *Paper presented at the Annual Meeting of the American Educational Research Association*, New Orleans, LA.
- Leung, C.K., Chang, H.H., & Hau, K.T. (2003). Computerized adaptive testing: A comparison of three content balancing methods. *The Journal of Technology, Learning and Assessment*, 2(5).
- Lewis, C., & Sheehan, K. (1990). Using Bayesian decision theory to design a computerized mastery test. *Applied Psychological Measurement*, 14, 367-386.
- Lilley, M., Barker, T., & Britton, C. (2004). The development and evaluation of a software prototype for computer-adaptive testing. *Computers and Education*, 43(1-2), 109-123.
- Linacre, J.M. (2000). Computer-Adaptive Testing: A Methodology Whose Time Has Come. *Development of Computerized Middle School Achievement Test [in Korean]*, S. Chae, U. Kang, E. Jeon, J.M. Linacre, Seoul, South Korea, Komesa Press.
- Lord, F.M. (1980). *Applications of Item Response Theory to practical testing problems*. Hillsdale, NJ: Lawrence Erlbaum.
- Marinagi, C.C., Tsoukalas, V.T., & Kaburlasos, V.G. (2004). Development and use of a software tool for improving the average student performance in the Greek higher education system. In *Proceedings of the 34th ASEE/IEEE Frontiers in Education Conference*, Savannah, Georgia.
- McDonald, A. (2002). The impact of individual difference on the equivalence of computer-based and paper-and-pencil educational assessments. *Computers and Education*, 39, 299-312.
- McKinley, R., & Kingston, N. (1987) Exploring the use of IRT equating for the GRE subject test in mathematics. Educational Testing Service, Princeton, NJ, Research Report 87-21.
- Millán, E., & Pérez-de-la-Cruz, J.L. (2002). A Bayesian diagnostic algorithm for student modeling and its evaluation. *User Modeling and User-Adapted Interaction*, 12(2-3), 281-330.
- Owen, R.J. (1975). A Bayesian sequential procedure for quantal response in the context of adaptive mental testing. *Journal of the American Statistical Association*, 70, 351-356.
- Papanikolaou, K.A., Grigoriadou, M., Kornilakis, H., & Magoulas, G.D. (2002). INSPIRE: an Intelligent System for Personalized Instruction in a Remote Environment. In S. Reich, M. Tzagarakis, P.M.E. De Bra, *Hypermedia: Openness, Structural Awareness, and Adaptivity*.

Lecture Notes in Computer Science, 2266, (pp. 215-225). Berlin, Heidelberg: Springer-Verlag.

Pérez, M.S., Herrero, P., & Sánchez, F.M. (2005). Are Web self-assessment tools useful for training?. *IEEE Transactions on Education*, 48(4), 757-763.

Reckase, M.D. (1979). Some decision procedures for use with tailored tests. In D.J. Weiss (Ed.), *Proceedings of the 1979 Computerized Adaptive Testing Conference*, (pp. 79-100) Minneapolis: University of Minnesota, Department of Psychology, Computerized Adaptive Testing Laboratory.

Reckase, M.D. (1983). A procedure for decision making using tailored testing. In D.J. Weiss (Ed.), *New horizons in testing: Latent trait test theory and computerized adaptive testing* (pp. 237-255). New York: Academic Press.

Rudner, L.M. (2002). An examination of decision-theory adaptive testing procedures. In *Proceedings of the Annual meeting of the American Educational Research Association*, (pp. 437-446), New Orleans, LA.

Rudner, L.M. (2003). The classification accuracy of measurement decision theory. In *Proceedings of the Annual meeting of the National Council on Measurement in Education*, Chicago.

Russell, M., & Haney, W. (2000). Bridging the gap between testing and technology in Schools. *Education Policy Analysis Archives*, 8(19). Available: <http://epaa.asu.edu/epaa/v8n19.html>.

Sampson, D., Karagiannidis, C., & Cardinali, F. (2002). An architecture for Web-based e-learning promoting re-usable adaptive educational e-content. *Educational Technology & Society of International Forum of Educational Technology & Society and IEEE Computer Society Learning Technology Task Force*, ISSN 1436-4522, *Special Issue on Innovations in Learning Technologies*, 5(4).

Slater, N., & Howie, K. (2003). User requirements of the “ultimate” online assessment engine. *Computers and Education*, 40, 285-306.

Smailes, J. (2003). Strategies for engaging students in computer based assessment-stage 1, taking stock. In *Proceedings of the Seventh International Computer Assisted Assessment Conference*, University of Loughborough, England.

Thelwall, M. (2000). Computer-based assessment: a versatile educational tool. *Computers and Education* 34, 37-49.

Valenti, S. (2003). Information technology for assessing student learning: Introduction to the special series. *Journal of Information Technology Education*, 2, 181-184.

Vos, H.J. (2000). A Bayesian procedure in the context of sequential mastery testing. *Psicologica*, 21, 191-211.

Weber, G., & Brusilovsky, P. (2001). ELM-ART: An adaptive versatile system for Web-based instruction. *International Journal of Artificial Intelligence in Education*, 12, 351-384.

Wiener, N. (1965). *Cybernetics, or Control and Communication in the Animal and the Machine*, 2nd ed. Cambridge, MA: MIT Press.

Xenos, M. (2004). Prediction and assessment of student behaviour in open and distance education in computers using Bayesian networks. *Computers and Education* 43(4), 345-359.

Vitae

Vassilis George Kaburlasos received the Diploma degree from the National Technical University of Athens, Greece in 1986, and the M.Sc. and Ph.D. degrees from the University of Nevada, Reno in 1989 and 1992, respectively, all in electrical engineering. He has made contributions to 16 research projects, funded either publicly or privately, as researcher, principal investigator, or as a project leader in the U.S.A. and in the European Union. He has authored or coauthored more than 80 scientific research papers in conferences, refereed journals, and books. He is the author of a research monograph book entitled *Towards a Unified Modeling and Knowledge-Representation Based on Lattice Theory – Computational Intelligence and Soft Computing Applications*, series: Studies in Computational Intelligence, vol. 27 (Heidelberg, Germany: Springer-Verlag 2006). Also, he is co-editor (with Gerhard X. Ritter) of a book entitled *Computational Intelligence Based on Lattice Theory*, series: Studies in Computational Intelligence, vol. 67 (Heidelberg, Germany: Springer-Verlag 2007). Dr. Kaburlasos is a Professor with the Department of Industrial Informatics, Technological Educational Institution of Kavala, Greece. His research interests include intelligent machine modeling applications. He is a member of several professional, scientific, and honor societies around the world including the Technical Chamber of Greece, Sigma Xi, Phi Kappa Phi, Tau Beta Pi, Eta Kappa Nu.

Catherine C. Marinagi received the Diploma degree in Mathematics from the University of Patras, Greece in 1987 and the Ph.D. in Artificial Intelligence from the University of Athens in 1999. She has made contributions to 5 research projects, as a researcher, and to 1 development project, as a scientific leader. She has (co-)authored 19 scientific research papers in conferences, refereed journals, and books. She is currently serving as an Associate Professor at the Department of Logistics, Technological Educational Institution of Halkida, Greece teaching Software Engineering, Databases, and Information Systems. Her research interests include AI Planning and Scheduling, Intelligent Agents, e-Learning and Assessment.

Vassilis Themistocles Tsoukalas received a Bachelor degree from the Technological Educational Institution of Athens, Greece in 2000 in Informatics. He has made contributions to 3 research projects as programmer/developer. He has coauthored 7 scientific research papers in conferences and refereed journals. Currently he is Professor Assistant and Network/System Administrator with the Department of Industrial Informatics, Technological Educational Institution of Kavala, Greece. His research interests include client/server/web/n-tier/enterprise architecture applications.

Figure Legends

- Fig. 1. An Item, as it is stored in the Item Bank (IB) regarding a course in *Software Engineering*.
- Fig. 2. The Item (QUESTION 1) of Fig. 1, as it is displayed on a student's monitor. Line "ANSWERS (correct : 1)", underneath the block-diagram, informs a student that only one answer is correct.
- Fig. 3. Feedback displayed on a student monitor when a student supplies a wrong answer to the question in Fig. 2: The correct answer, the (wrong) student answer, and a comment regarding the student's wrong answer are displayed.
- Fig. 4. Improvement of the score of student- i in a course e during self-assessment based on closed-loop (feedback) techniques. In this work we have used a positive/nonpositive error driven controller for computing a *focus-of-attention* vector; the latter is used to randomly select an Item (question) from the Item Bank IB_e . More specifically, the controller either computes deterministically a *focus-of-attention* vector (for a *positive* error) or it opens the loop for a *nonpositive* error $R-g_e(i) \leq 0 \Leftrightarrow R \leq g_e(i)$. In the latter case self-assessment stops.
- Fig. 5. The algorithms for both student *exam-assessment* and student *self-assessment*. For computational details see in the Appendices A and B.
- Fig. 6. Foreground: Percentage-wise distribution of 64 students over the range (0-10) of scores in the 2nd midterm exam of the *Software Engineering* course during academic year 2005/2006. Background: The corresponding final exam distribution.
- Fig. 7. Foreground: Percentage-wise distribution of 133 students over the range (0-10) of scores in the 2nd midterm exam of the *Intelligent Systems* course during academic year 2005/2006. Background: The corresponding final exam distribution.

Table Legends

Table 1

A summary of statistics regarding the *Software Engineering* course

Table 2

A summary of statistics regarding the *Intelligent Systems* course

Table 3

Statistical significance test values for various (academic year) combinations of control/treatment groups regarding **Final Exam Scores** in two courses

Table 4

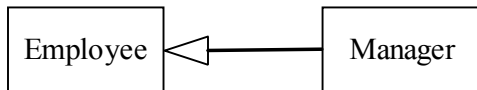
Statistical significance test values for various (academic year) combinations of control/treatment groups regarding **2nd Midterm Exam Scores** in two courses

Item Type: Multiple-choice

Item Difficulty: 30

Question:

What kind of relationship between classes is represented by the following UML notation?



Answers:

1) Aggregation

Correct: No

Comment: In UML notation the aggregation is drawn as solid lines from the aggregates (parts) to the aggregator (whole) with a diamond arrowhead on the aggregator's part.

See also: http://iiu.teikav.gr/courses/soft_eng/ch7

2) Association

Correct: No

Comment: In UML notation, association is drawn as a solid line between two classes.

Symbols indicating multiplicity are drawn at both ends of a line. In bidirectional association no arrows appear, while in unidirectional association a black arrow indicates the target class. Optionally, the association name is placed next to the middle of the line and a role name is attached to either or both ends of a line.

See also: http://iiu.teikav.gr/courses/soft_eng/ch7

3) Generalization/specialization

Correct: Yes

Comment:

See also: http://iiu.teikav.gr/courses/soft_eng/ch7

4) Composition

Correct: No

Comment: In UML notation, composition is drawn as solid lines from the components to the whole with a solid (filled-in) diamond arrowhead on the whole.

See also: http://iiu.teikav.gr/courses/soft_eng/ch7

Fig. 1.

The screenshot shows a web browser window titled "STUDENT ASSESSMENT INFORMATICS SYSTEM: SELF-ASSESSMENT -ningewrg". The interface includes a menu bar with "File" and "Exam", and a toolbar with a "Submit" button. The main content area displays "QUESTION 1" with the text: "What kind of relationship between classes is represented by the following UML notation?". Below the text is a UML class diagram showing a class "Employee" on the left and a class "Manager" on the right. A solid line with an open arrowhead points from "Manager" to "Employee", representing a generalization/specialization relationship. Below the diagram, the text "ANSWERS (correct : 1)" is followed by four radio button options: "Aggregation", "Association", "Generalization/specialization", and "Composition". At the bottom of the window, a status bar indicates "Remaining time: 59 minutes".

Fig. 2.

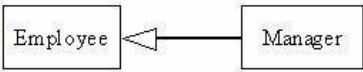
STUDENT ASSESSMENT INFORMATICS SYSTEM: SELF-ASSESSMENT -

File Exam

Submit

QUESTION 1

What kind of relationship between classes is represented by the following UML notation?



```
classDiagram
    Employee <|-- Manager
```

CORRECT ANSWERS

Generalization/specialization

USER ANSWERS

Aggregation

COMMENTS

In UML notation the aggregation is drawn as solid lines from the aggregates (parts) to the aggregator (whole) with a diamond arrowhead on the aggregator's part.

Fig. 3.

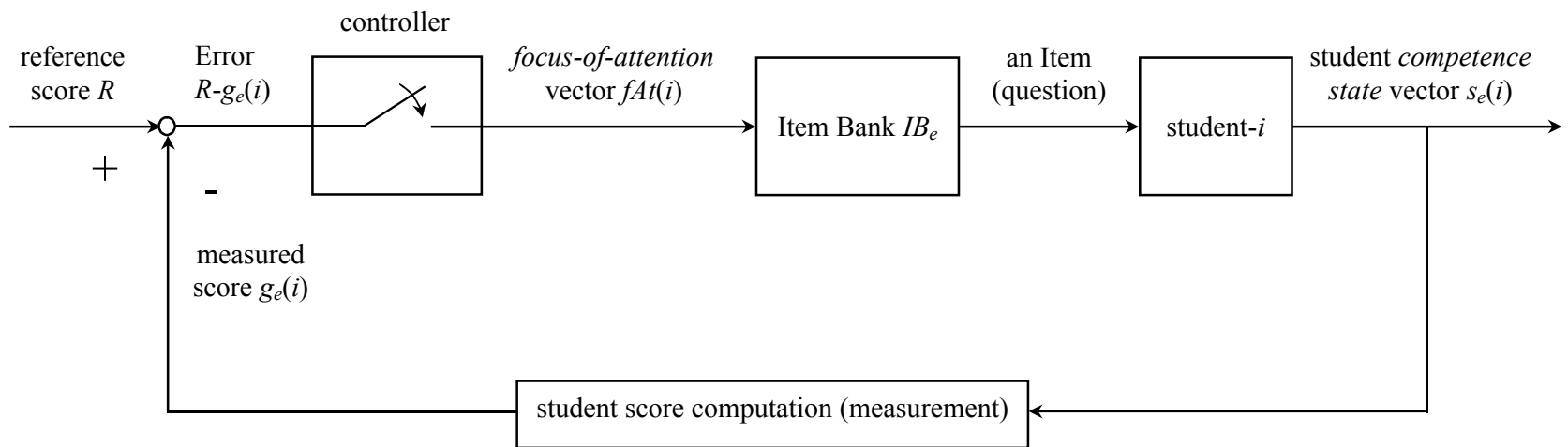


Fig. 4.

1. An instructor compiles a tree-structured Item Bank IB_e in a specific *educational subject* (i.e. *course*) e by defining a set of N_e Items τ_n , $n \in \{1, \dots, N_e\}$ partitioned in M_e Units μ_m , $m \in \{1, \dots, M_e\}$ such that each Unit includes Items of similar difficulty.
2. Option #1: **Exam-assessment**.
 - 2.1. The instructor defines a *Unit-content* percentage $\pi(\mu_m)$ for each Unit μ_m .
 - 2.2. Students sit in front of computer monitors in the lab.
 - 2.3. The *focus-of-attention* vector $fAt = (\pi(\mu_1), \dots, \pi(\mu_{M_e}))$ is used to randomly select an instructor-defined number “Q” of Items (questions) for each student.
 - 2.4. A student is required to respond within a defined time to all questions.
3. Option #2: **Self-assessment**.
 - 3.0. Process the responses, of a randomly selected group of students, to all Items τ_n in the Item Bank IB_e in order to initialize *priors* $p(L_j)$, *likelihoods* $p(a_n|L_j)$, *evidences* $p(a_n)$, *Unit Difficulties* $UD(\mu_m)$, and *Unit Item Delays* $UID(\mu_m)$, where $j \in \{1, \dots, J\}$.
 - 3.1. An advisor proposes a *unit-content* percentage $\pi(\mu_m)$ for each Unit μ_m for student- i .
 - 3.2. Student $i \in \{1, \dots, S_e\}$ may sit in front of a computer monitor, any time.
 - 3.3. Initialize student *competence state* vector $s_e(i) = (p(L_1|\mathbf{a}), \dots, p(L_J|\mathbf{a}))$ based on a series \mathbf{a} of student- i responses to a number of Items drawn randomly from IB_e .
 - 3.4. A student advisor defines a reference score R . Initially, assume $g_e(i) = 0$.
 - 3.5. If $R \leq g_e(i)$ then stop self-assessment; else, compute a student *focus-of-attention* vector $fAt(i) = (p(\mu_1), \dots, p(\mu_{M_e}))$.
 - 3.6. Use the *focus-of-attention* vector $fAt(i)$ to randomly select an Item (question) from the Item Bank IB_e .
 - 3.7. Consider the response of student- i to the last Item (question) in order to compute a new *competence state* vector $s_e(i) = (p(L_1|\mathbf{a}), \dots, p(L_J|\mathbf{a}))$.
 - 3.8. Compute the new student score $g_e(i)$.
 - 3.9. Go to step 3.5.

Fig. 5.

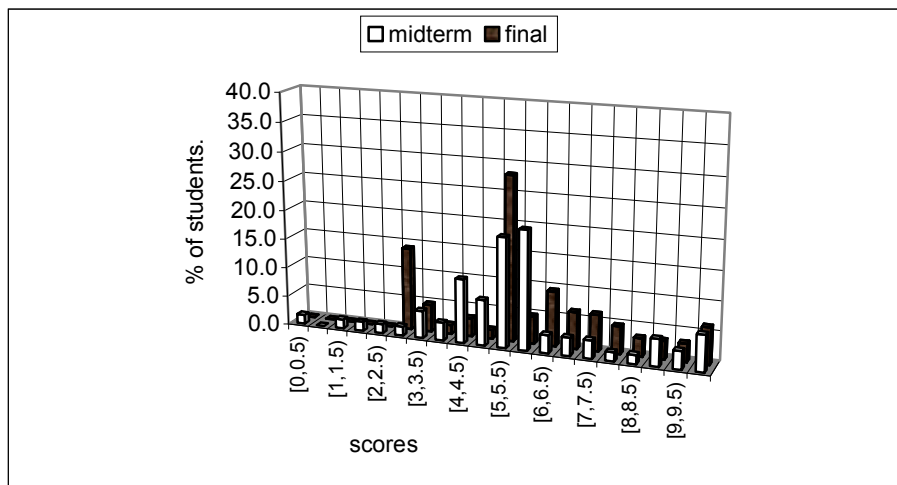


Fig. 6.

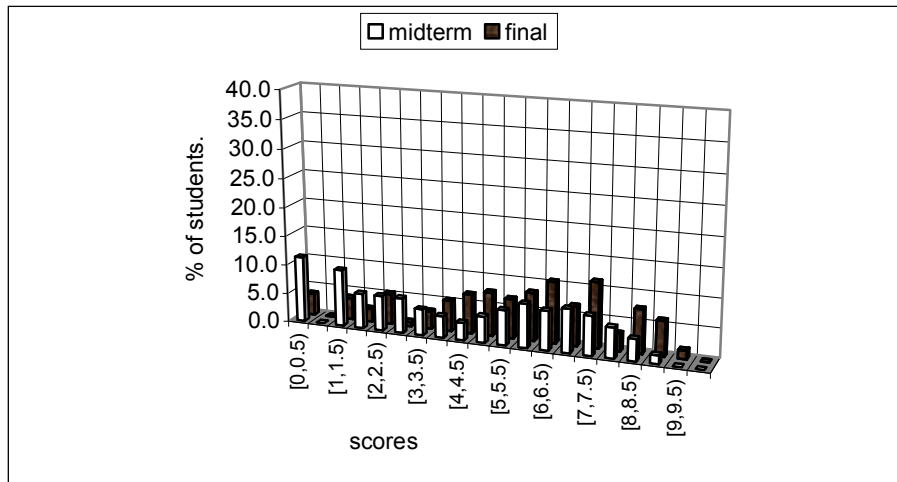


Fig. 7.

Table 1

A summary of statistics regarding the *Software Engineering* course

academic year	no. of students	2 nd midterm exam score statistics			final exam score statistics			student throughput (approx.)
		range	average	standard deviation	range	average	standard deviation	
2002/03	26	—	—	—	[0.7, 9]	5.13	2.14	~65%
2003/04	54	[0, 6.5]	2.44	1.57	[1.8, 8.6]	5.44	1.44	~67%
2004/05	59	[0, 9]	3.85	2.12	[1.0, 9.5]	5.20	2.18	~73%
2005/06	64	[0, 10]	5.48	2.07	[2.5, 10]	5.65	2.07	~75%

Table 2

A summary of statistics regarding the *Intelligent Systems* course

academic year	no. of students	2 nd midterm exam score statistics			final exam score statistics			student throughput (approx.)
		range	average	standard deviation	range	average	standard deviation	
2002/03	71	—	—	—	[0, 6.5]	3.26	1.78	~34%
2003/04	81	[0, 5.6]	1.97	1.74	[0.1, 7.2]	3.89	1.61	~37%
2004/05	88	[0, 7.5]	2.88	2.24	[0, 8]	4.48	1.60	~56%
2005/06	133	[0, 8.8]	4.09	2.61	[0, 9]	5.34	2.30	~62%

Table 3

Statistical significance test values for various (academic year) combinations of control/treatment groups regarding **Final Exam Scores** in two courses

Statistic t / degrees of freedom df / confidence% (in the alternative hypothesis $H_a: \mu_t > \mu_c$) (the 95%/99% confidence interval)			
<i>Course: Software Engineering</i>			
Treatment	2003-04	2004-05	2005-06
Control Group			
2002-03	0.669 / 36.27 / 74.62% (0.31 ± 0.78/1.12)	0.138 / 48.70 / 55.46% (0.07 ± 0.84/1.21)	1.054 / 45.03 / 85.14% (0.52 ± 0.82/1.18)
2003-04	–	-0.695 / 101.29 / 24.40% (-0.24 ± 0.57/0.81)	0.646 / 112.14 / 74.052% (0.21 ± 0.53/0.76)
2004-05	–	–	1.171 / 118.87 / 87.816% (0.45 ± 0.63/0.90)
<i>Course: Intelligent Systems</i>			
Treatment	2003-04	2004-05	2005-06
Control Group			
2002-03	2.275 / 142.34 / 98.78% (0.63 ± 0.45/0.65)	4.493 / 142.34 / 99.99% (1.22 ± 0.44/0.63)	7.159 / 176.17 / 99.99% (2.08 ± 0.48/0.68)
2003-04	–	2.387 / 165.66 / 99.09% (0.59 ± 0.40/0.58)	5.412 / 207.85 / 99.99% (1.45 ± 0.44/0.62)
2004-05	–	–	3.277 / 218.42 / 99.93% (0.86 ± 0.43/0.61)

Table 4

Statistical significance test values for various (academic year) combinations of control/treatment groups regarding **2nd Midterm Exam Scores** in two courses

Statistic t / degrees of freedom df / confidence% (in the alternative hypothesis $H_a: \mu_t > \mu_c$) (the 95%/99% confidence interval)		
<i>Course: Software Engineering</i>		
Treatment	2004-05	2005-06
Control Group		
2003-04	4.039 / 106.49 / 99.99% (1.41 ± 0.57/0.82)	9.059 / 114.77 / 99.99% (3.04 ± 0.55/0.79)
2004-05	–	4.308 / 119.65 / 99.99% (1.63 ± 0.62/0.89)
<i>Course: Intelligent Systems</i>		
Treatment	2004-05	2005-06
Control Group		
2003-04	2.961 / 162.50 / 99.82% (0.91 ± 0.50/0.72)	7.122 / 210.22 / 99.99% (2.12 ± 0.49/0.69)
2004-05	–	3.677 / 204.65 / 99.98% (1.21 ± 0.54/0.77)