Modeling the learner's cognitive state transitions using fuzzy logic techniques for adaptive learning

Ph.D. Thesis of Konstantina Ch. Chrysafiadi

Supervisor professor: Professor Maria Virvou

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

UNIVERSITY OF PIRAEUS

Department of Informatics

February 2014

<u>THREE-MEMBER ADVISORY COMMITTEE</u>

1. **Maria Virvou**, Professor of the department of Informatics of the University of Piraeus (**supervisor professor**).

2. **Nikolaos Alexandris**, Emeritus Professor of the department of Informatics of the University of Piraeus.

3. **Evangelos Fountas**, Emeritus Professor of the department of Informatics of the University of Piraeus.

This Ph. D. thesis was presented to the department of Informatics of the University of Piraeus, on 28 February 2014 and was approved by the following seven-member examining committee:

1. **Maria Virvou**, Professor of the department of Informatics of the University of Piraeus (**supervisor professor**).

2. **Nikolaos Alexandris**, Emeritus Professor of the department of Informatics of the University of Piraeus.

3. **Evangelos Fountas**, Emeritus Professor of the department of Informatics of the University of Piraeus.

4. **Emmanouil Giakoumakis,** Professor of the department of Informatics of Athens University of Economics and Business.

5. **Nikolaos Malevris**, Professor of the department of Informatics of Athens University of Economics and Business.

6. **Katerina Kabassi,** Professor of applications of Technological Educational Institute of Ionian Islands.

7. **Efthimios Alepis,** Lecturer of the department of Informatics of the University of Piraeus.

Η παρούσα διδακτορική διατριβή παρουσιάστηκε στο τμήμα Πληροφορικής του Πανεπιστημίου Πειραιώς, στις 28 Φεβρουαρίου 2014 και εγκρίθηκε από την ακόλουθη επταμελή εξεταστική επιτροπή:

1. **Μαρία Βίρβου**, Καθηγήτρια του τμήματος Πληροφορικής του Πανεπιστημίου Πειραιώς (**επιβλέπουσα καθηγήτρια**).

2. **Νικόλαος Αλεξανδρής**, Ομότιμος Καθηγητής του τμήματος Πληροφορικής του Πανεπιστημίου Πειραιώς.

3. **Ευάγγελος Φούντας**, Ομότιμος Καθηγητής του τμήματος Πληροφορικής του Πανεπιστημίου Πειραιώς.

4. **Εμμανουήλ Γιακουμάκης,** Καθηγητής του τμήματος Πληροφορικής του Οικονομικού Πανεπιστημίου Αθηνών.

5. **Νικόλαος Μαλεύρης**, Καθηγητής του τμήματος Πληροφορικής του Οικονομικού Πανεπιστημίου Αθηνών.

6. **Κατερίνα Καμπάση,** Καθηγήτρια Εφαρμογών Τ.Ε.Ι. Ιόνιων Νήσων

7. **Ευθύμιος Αλέπης,** Λέκτορας του τμήματος Πληροφορικής του Πανεπιστημίου Πειραιώς.

Dedicated to

my parents Christodoulos & Georgia

Acknowledgments

The elaboration of this Ph.D. thesis was to me a beautiful, creative and constructive experience. Its completion needed hard work and personal struggle. However, it would not be possible without the support and the direct or indirect contribution of many people to whom I would like to express my sincere thanks.

Initially, I would like to thank the supervisor of my dissertation, the Professor of the department of Informatics of the University of Piraeus, Maria Virvou, for the opportunity that she gave me to work on such an interesting and original topic, and for the scientific support and mentoring that she gave me during the elaboration of the Ph.D. thesis. Her advice was valuable and the excellent cooperation that we had were important succorers for the completion of the thesis.

Then, I would like to thank the Emeritus Professors of the department of Informatics of the University of Piraeus, Nikolaos Alexandris and Evangelos Fountas, for their support and contribution. I would also like to thank the other members of the seven-members examining committee, the Professors of the department of Informatics of Athens University of Economics and Business, Emmanouil Giakoumakis and Nikolaos Malevris, the Professor of applications of Technological Educational Institute of Ionian Islands Katerina Kabassi, and the Lecturer of the department of Informatics of the University of Piraeus, Efthimios Alepis, for the time that they spent and for the honor that they made me to participate in the examination of my thesis.

In addition, I thank the department of Informatics of University of Piraeus for the incentives that gave to me to deal with research and material and technical support that it provided me. I, also, thank the Research Center of the University of Piraeus for the financial support that provided me during my participation in international conferences. I owe special thanks, also, to the students of the postgraduate course in Computer Science of the department of Informatics of University of Piraeus, of years: 2008-2009; 2009-2010; 2010-2011; 2011-2012; 2012-2013, for their participation in the evaluation of my research's implementation and application, and for their accurate observations and constructive advice.

Additionally, I am very grateful to my family for their support and tolerance that they showed all the years of the elaboration of my Ph.D. thesis. I thank my parents, Christodoulos and Georgia, for their unconditional love, their faith in me and the financial and psychological support that they have given to me. Also, I thank my husband Thanos for the understanding, the constant encouragement and his apparent faith in my abilities. Finally, I thank my brother Vassilis for the support and for his major contribution to the illustration of my Ph.D. thesis.

Abstract

In this Ph.D. thesis a novel approach of web-based education that performs individualized instruction, adapting the delivery of the knowledge domain to the individual learner's learning needs and pace, is presented. It includes fuzzy logic techniques to represent the learner's knowledge and cognitive state. The presented Intelligent Tutoring System includes a rule-based fuzzy logic mechanism for providing personalized tutoring to each learner and an innovative module, which is responsible for tracking cognitive state transitions of learners with respect to their progress or non-progress.

The presented approach models either how learning progresses or how the student's knowledge can be decreased. In particular, it performs user modeling by dynamically identifying and updating the student's knowledge level for all the concepts of the domain knowledge. Its operation is based on Fuzzy Cognitive Maps (FCMs). They are used to represent the dependencies among the domain concepts. The presented student model uses fuzzy sets to represent the student's knowledge level as a subset of the domain knowledge. Thus, it combines fuzzy theory with the overlay model. Moreover, it employs a novel inference mechanism that dynamically updates user stereotypes using fuzzy sets. This mechanism is triggered after any change of the student's knowledge level of a domain concept. Then, it updates the student's knowledge level of the concepts, which are related with the concept that the student has learnt or forgotten. The transition of a learner from one stereotype to another reveals her/his learning state each time. In particular, it reveals if a student learns or not, if s/he forgets and reasons these states.

The presented novel approach was fully implemented and evaluated. Particularly, an original integrated environment for personalized e-training in programming and the programming language C, which is called ELaC, was developed. The specific knowledge domain was chosen due the fact that in the domain of computer programming there are many different programming languages and learners have a variety of different backgrounds and characteristics. Therefore, it is suitable for the implementation and evaluation of the thesis' issue. ELaC incorporates the presented fuzzy student modeling approach. Thereby, recognizes when a new domain concept is completely

unknown to the learner, or when it is partly known due to the learner having previous related knowledge. Furthermore, it recognizes when a previously known domain concept has been completely or partly forgotten by the learner. This system was used by the students of a postgraduate program in the field of Informatics in the University of Piraeus, Greece, in order to learn how to program in the programming language C.

For the evaluation of the fuzzy student model approach, two well-known and common-used evaluation methods were chosen: the Kirkpatrick’s model and the layered evaluation method. The results of the evaluation were very encouraging. They demonstrated that the system models the student’s cognitive state and adapts dynamically to her/his individual needs by scheduling the sequence of lessons instantly, allowing her/him to complete the e-training course at her/his own pace and according to her/his ability.

It has to be noted that the presented novel combination of overlay model and stereotypes with fuzzy sets is significant as the students’ level of knowledge is represented in a more realistic way by automatically modeling the learning or forgetting process of a student with respect to the FCMs. The application of this approach is not limited to adaptive instruction, but it can also be used in other systems with changeable user states, such as e-shops, where consumers’ preferences change over the time and affect one another. Therefore, the particular approach constitutes a novel generic fuzzy tool, which offers dynamic adaptation to users’ needs and preferences of adaptive systems.

Περίληψη

Στη παρούσα διδακτορική διατριβή παρουσιάζεται μια καινοτόμος προσέγγιση της εξ’ αποστάσεως εκπαίδευσης, που επιτελεί εξατομικευμένη διδασκαλία, προσαρμόζοντας την παράδοση του γνωστικού αντικειμένου στις ανάγκες και τον ρυθμό μάθησης του κάθε εκπαιδευόμενου. Η προσέγγιση αυτή περιλαμβάνει τεχνικές ασαφούς λογικής για την αναπαράσταση του γνωστικού επιπέδου και της γνωστικής κατάστασης του εκπαιδευόμενου. Το παρουσιαζόμενο Ευφυές Διδακτικό Σύστημα περιλαμβάνει έναν μηχανισμό βασισμένο σε κανόνες ασαφούς λογικής για την παροχή εξατομικευμένης διδασκαλίας σε κάθε εκπαιδευόμενο, και μια καινοτόμο λειτουργική μονάδα, η οποία είναι υπεύθυνη για την παρακολούθηση των μεταβάσεων μεταξύ των γνωστικών καταστάσεων ενός εκπαιδευόμενου όσον αφορά στην πρόοδο ή μη του μαθητή.

Η παρουσιαζόμενη προσέγγιση μοντελοποιεί τόσο την εξέλιξη της μάθησης όσο και την μείωση της γνώσης του εκπαιδευόμενου. Συγκεκριμένα, μοντελοποιεί τον εκπαιδευόμενο αναγνωρίζοντας και ενημερώνοντας δυναμικά το επίπεδο γνώσης του εκπαιδευόμενου για όλες τις επιμέρους έννοιες του γνωστικού αντικειμένου. Η λειτουργία της παρουσιαζόμενης προσέγγισης είναι βασισμένη στους Ασαφείς Γνωστικούς Χάρτες (Fuzzy Cognitive Maps – FCMs), οι οποίοι χρησιμοποιούνται για την αναπαράσταση των εξαρτήσεων μεταξύ των εννοιών του γνωστικού αντικειμένου. Το παρουσιαζόμενο μοντέλο μαθητή χρησιμοποιεί ασαφή σύνολα για την αναπαράσταση της γνώσης του εκπαιδευόμενου ως υποσύνολο του γνωστικού αντικειμένου. Επομένως, συνδυάζει την θεωρία της ασαφής λογικής (fuzzy logic) με το μοντέλο επικάλυψης (overlay model). Επίσης, διαθέτει ένα καινοτόμο μηχανισμό εξαγωγής συμπερασμάτων που ενημερώνει δυναμικά τα στερεότυπα των χρηστών χρησιμοποιώντας ασαφή σύνολα. Ο μηχανισμός αυτός ενεργοποιείται μετά από οποιαδήποτε μεταβολή του γνωστικού επιπέδου του εκπαιδευόμενου σε μια έννοια του γνωστικού αντικειμένου. Έπειτα, ενημερώνει το γνωστικό επίπεδο του εκπαιδευόμενου σε όλες τις επιμέρους έννοιες του γνωστικού αντικειμένου που σχετίζονται με την έννοια που ο εκπαιδευόμενος έμαθε ή ξέχασε. Η μετάβαση του εκπαιδευόμενου από το ένα στερεότυπο σε άλλο αποκαλύπτει την γνωστική

του κατάσταση κάθε φορά. Συγκεκριμένα, αποκαλύπτει εάν ο εκπαιδευόμενος μαθαίνει ή όχι, εάν ξεχνάει και αιτιολογεί τις καταστάσεις αυτές.

Η παρουσιαζόμενη καινοτόμος προσέγγιση υλοποιήθηκε, εφαρμόστηκε και αξιολογήθηκε πλήρως. Πιο συγκεκριμένα, αναπτύχθηκε ένα πρωτότυπο ολοκληρωμένο περιβάλλον, με το όνομα ELaC, για την εξατομικευμένη εξ' αποστάσεως εκπαίδευση στον προγραμματισμό υπολογιστών και στην γλώσσα προγραμματισμού 'C'. Επιλέχθηκε το συγκεκριμένο γνωστικό αντικείμενο λόγω του γεγονότος ότι στο πεδίο του προγραμματισμού ηλεκτρονικών υπολογιστών υπάρχουν πολλές διαφορετικές γλώσσες προγραμματισμού και οι εκπαιδευόμενοι έχουν ποικίλα διαφορετικά υπόβαθρα και χαρακτηριστικά. Ως εκ τούτου, το συγκεκριμένο γνωστικό αντικείμενο είναι κατάλληλο για την εφαρμογή και την αξιολόγηση του θέματος που διαπραγματεύεται η συγκεκριμένη διατριβή. Το ELaC ενσωματώνει την παρουσιαζόμενη προσέγγιση της ασαφής μοντελοποίησης του εκπαιδευομένου. Με αυτόν τον τρόπο, αναγνωρίζει πότε μια νέα έννοια του γνωστικού αντικειμένου είναι εντελώς άγνωστη στον εκπαιδευόμενο, ή πότε είναι εν μέρει γνωστή λόγω του γεγονότος ότι ο εκπαιδευόμενος έχει προηγούμενη σχετική γνώση. Επιπλέον, αναγνωρίζει πότε ο εκπαιδευόμενος ξεχνάει πλήρως ή εν μέρει μία έννοια που ήταν προηγουμένως γνωστή. Το σύστημα ELaC χρησιμοποιήθηκε από τους φοιτητές ενός μεταπτυχιακού προγράμματος Πληροφορικής του Πανεπιστημίου Πειραιά, προκειμένου να μάθουν πως να προγραμματίζουν στη γλώσσα προγραμματισμού 'C'.

Για την αξιολόγηση την προσέγγιση της ασαφής μοντελοποίησης του εκπαιδευομένου, επιλέχθηκαν δύο γνωστές και ευρέως χρησιμοποιημένες μέθοδοι αξιολόγησης: το μοντέλο του Kirkpatrick και η πολυεπίπεδη μέθοδος αξιολόγησης (layered evaluation model). Τα αποτελέσματα της αξιολόγησης ήταν πολύ ενθαρρυντικά. Απέδειξαν ότι το σύστημα μοντελοποιεί τη γνωστική κατάσταση του μαθητή και προσαρμόζεται δυναμικά στις ατομικές ανάγκες του εκπαιδευόμενου προγραμματίζοντας τη σειρά των μαθημάτων στη στιγμή, επιτρέποντάς του να ολοκληρώσει την πορεία της ηλεκτρονικής κατάρτισης στο δικό του ρυθμό και σύμφωνα με τις δικές του ικανότητες.

Πρέπει να σημειωθεί ότι παρουσιαζόμενος καινοτόμος συνδυασμός του μοντέλου επικάλυψης (overlay model) και των στερεοτύπων με τα ασαφή σύνολα είναι σημαντικός, καθώς το επίπεδο γνώσης των εκπαιδευόμενων αναπαρίσταται με ένα πιο ρεαλιστικό τρόπο, μοντελοποιώντας αυτόματα την διαδικασία της μάθησης ή της λήθης ενός εκπαιδευόμενου σε σχέση με τους ασαφείς Γνωστικούς Χάρτες (FCMs). Η εφαρμογή αυτής της προσέγγισης δεν περιορίζεται στη προσαρμοσμένη εκπαίδευση, αλλά μπορεί επίσης να χρησιμοποιηθεί σε άλλα συστήματα με μεταβλητές καταστάσεις χρηστών, όπως είναι τα ηλεκτρονικά καταστήματα, όπου οι προτιμήσεις των καταναλωτών αλλάζουν με την πάροδο του χρόνου και επηρεάζουν η μία προτίμηση την άλλη. Συνεπώς, η συγκεκριμένη προσέγγιση αποτελεί ένα καινοτόμο γενικό εργαλείο ασαφής λογικής, το οποίο παρέχει δυναμική προσαρμογή στις ανάγκες και τις προτιμήσεις των χρηστών των προσαρμοστικών συστημάτων.

Table of contents

Introduction

In the past decade there has been an enormous growth of the field of computer-based learning that includes e-learning, mobile learning, educational games and standalone educational applications. The rapid development of computer and Internet technologies, and the accessibility of e-learning applications by a large and heterogeneous group of learners at any time and place, has led to a rapid and significant growth of Web-based learning environments. However, traditional web-based and standalone educational systems still have several shortcomings when compared to real-life classroom teaching, such as lack of contextual and adaptive support, lack of flexible support of the presentation and feedback, lack of the collaborative support between students and systems (Xu, Wang & Su, 2002). That is the reason why researches in the field of e-learning have expanded their interests on adaptive e-learning, which is suitable for teaching heterogeneous student populations (Schiaffino, Garcia & Amandi, 2008).

Web-based educational systems offer easy access to knowledge domains and learning processes from everywhere for everybody at any time. As a result, users of web-based educational systems are of varying backgrounds. They have heterogeneous needs, different levels of knowledge and abilities. As Graf and Kinshuk (2010) have stated, student's individual differences play a central role in web-based learning. Therefore, successful teaching through the web demands user-friendly interface, multimedia and hypermedia techniques, adaptivity to individual students' needs and reasoning abilities. Consequently, the challenge is to develop Web-based educational systems that adapt dynamically to each individual student for effective delivery of knowledge domain.

Creating an adaptive learning system that meets students' requirements can be challenging since students learn with not only different needs, but also different learning characteristics (Lo, Chan & Yen, 2012). An adaptive system must be capable of managing learning paths adapted to each user, monitoring user activities, interpreting those using specific models, inferring user needs and preferences and exploiting user and knowledge domain to dynamically facilitate the learning process (Boticario, Santos & van Rosmalen, 2005). In other words, an adaptive educational system has to provide personalization to the specific needs, knowledge and background of each individual student.

This problem often occurs in the domain of computer programming, since there are many different programming languages and learners have different backgrounds and characteristics. Programming language learners can vary from novice programmers, to more experienced programmers who know programming languages other than that being taught. Consequently, each learner has her/his own learning pace, and educational environments have to adapt to this. In fact, provision of the same instructional conditions to all students can be pedagogically ineffective (Akbulut, and Cardak, 2012). Each learner should not be told to read the same material in the same order; rather, their learning material should be delivered with respect to their knowledge level and personal needs.

Obviously, in learning a new programming language a novice programmer has to learn many more domain concepts than does a more experienced programmer, who already knows the principles and the basic structures of computer programming. Furthermore, if a learner already knows an algorithm (e.g., calculating the sum of integers in a 'for' loop), there is no need to learn another similar algorithm (e.g., counting in a 'for' loop). Similarly, if a learner knows a programming structure (e.g., one-dimensional arrays), it is easier to understand another programming structure (e.g., multidimensional arrays), so this new structure should not be considered as being completely unknown to the learner. Similarly, if a learner’s performance on a domain concept is poor, this suggests that s/he has forgot another relevant domain concept. For example, if a learner has difficulties in calculating a sum in a ‘while’ loop, her/his knowledge of the previous domain concept of “calculating a sum in a ‘for’ loop” has eroded. In fact, the learner's knowledge of a domain concept is subject to change. Hence web-based educational systems have to recognize the learner's knowledge level and how this changes, and then they provide effective instruction, adapting the delivery of the knowledge domain to the learner's learning needs and pace.

A solution to the above problem is adaptive navigation support technology, known for its ability to help students acquire knowledge faster, improve learning outcomes and reduce navigational overhead (Brusilovsky, Hsiao and Folajimi, 2011). As Brusilovsky et al. (Brusilovsky, Sosnovsky and Yudelson, 2006) discovered affective navigation support has the ability to increase the amount of student work with non-mandatory educational content. Many Intelligent Tutoring Systems (ITSs) adopt this technology, which supports user navigation in hyperspace by adapting to the goals, preferences and knowledge of the individual user (Brusilovsky, 2007). ITSs are computerized learning environments that incorporate computational models from the cognitive sciences, learning sciences, computational linguistics, artificial intelligence, mathematics, and other fields (Graesser, Conley, and Olney, 2012). They belong to an advanced generation of computer-based instruction systems that provide students with a highly personalized learning experience by adapting content and presentation to the student's needs and preferences (Jeremic, J. Jovanovic, and D. Gaševic, 2012). ITSs' ability to adapt content dynamically renders them adaptive learning environments.

When applying adaptive navigation support technology, an important aspect that has to be specified is the knowledge domain representation, which is a description of expertise in the subject-matter domain of the ITS. The common used techniques for knowledge domain representation are: hierarchies, network of concepts; linkage graphs and concept maps. In this way, either the order in which each domain concept has to be taught or the knowledge dependencies that exist between the domain concepts of the learning material have to be represented. However, the representation of the relations between concepts is, mainly, restricted to “part-of”, “is-a” and prerequisite relations. There is the need to represent how the student's knowledge level of a domain concept is affected by her/his knowledge level of other related domain concepts. The representation of this kind of relations of the learning material's domain concepts is performed by Fuzzy Cognitive Maps (FCMs) (Chrysafiadi and M. Virvou, 2012). FCM is a cognitive map within which the relations between the elements of a "mental landscape" (like concepts, events, and project resources) can be used to compute the "strength of impact" of these elements. Therefore, the knowledge dependencies between the domain concepts of the learning material and the “strength of impact” of each concept on others can be represented through FCMs. However, the knowledge domain representation has to be combined with a well-designed student model, which is responsible for how the system uses the knowledge domain module to make the right decisions to offer personalized instruction and support to the learner.

The ability of an ITS to provide adaptivity is based, mainly, on the technology of student modeling (Devedzic, 2006) . Student modeling has been introduced in ITSs, but its use has been extended to most current educational software applications that aim to be adaptive and personalized. Student modeling is one of the key factors that affects automated tutoring systems in making instructional decisions (Li et al., 2011), since it is the process of gathering relevant information in order to infer the current cognitive state of the student, and to represent it so as to be accessible and useful to the ITS for offering adaptation (Brusilovsky and Millán, 2007). A student model allows understanding and identification of student needs. By keeping a model for every user, a system can successfully personalize its content and utilize available resources accordingly (Kyriacou, 2008). For example, in an adaptive educational application, a student model can be used to achieve accurate student diagnosis and predict a student’s needs. In return, it offers individualized courses (Gaudioso, Montero and Hernandez-del-Olmo, 2010), adaptive navigation support (Castillo, Gama and Breda, 2009), help and feedback to students (Tsiriga and Virvou, 2003a; Chrysafiadi and Virvou, 2008), allowing them to learn in their own pace (Chrysafiadi and Virvou, 2013).

There are a variety of techniques of student modeling. The most widely known techniques are overlay models and stereotypes. Other techniques of student modeling are: perturbation; machine learning techniques; cognitive theories; constraint-based model; fuzzy logic techniques; Bayesian networks; ontologies. According to Yang, Kinshuk and Graf (2010), in order to carry out the personalization efficiently, the student model needs to consider both domain dependent (e.g. knowledge level, error, misconceptions) and domain independent characteristics (e.g. age, learning styles). That is achieved by using a compound student model bringing together features of more than one student models.

Student modeling in many cases deals with uncertainty. Learning is a complicated process. It cannot be accurately said that a learner knows or does not know a domain concept. For example, a new domain concept may be completely unknown to the learner but in other circumstances it may be partly known due to previous related knowledge of the learner. On the other hand, domain concepts which were previously known by the learner may be completely or partly forgotten. Hence, currently they may be partly known or completely unknown. In this sense, the level of knowing cannot be accurately represented. Finally, the teaching process itself changes the status of knowledge of a user. This is happened due to the fact that a learner accepts new concepts while being taught. In view of the above, the representation of the learner's knowledge is a moving target and a learning control system is a nonlinear system. A solution to this problem is the use of fuzzy logic. In literature, fuzzy logic has been used to control nonlinear systems (Tong and Li, 2012; 2013). Fuzzy logic was introduced by Zadeh, as a methodology for computing with words, which cannot be done equally well with other methods (Zadeh, 1996). Integrating fuzzy logic into the student model of an ITS is a good idea, since the fuzzy logic based methods are more consistent with the human-being decision-making processes (Shakouri & Tavassoli, 2012).

Many adaptive tutoring systems for programming languages like ELM-ART (Weber and Brusilovsky, 2001), JavaGuide (Hsiao, Sosnovsky and Brusilovsky, 2010) and Protus (Klašnja-Milićevića et al., 2011) have implemented a student model that identifies the student's needs and dynamically adapts the educational process to meet them. However, they do not take into account how the level of a learner's knowledge of a concept can affect the level of her/his knowledge of another related domain concept. To achieve this, the knowledge dependencies that exist between the learning materials' domain concepts have to be represented. Frequently, Bayesian networks have been used to model dependencies between different domain concepts. However these dependencies are restricted to prerequisite and granularity relationships between the domain concepts (Millán, Loboda and Pérez-de-la-Cruz, 2010); some examples of this are Andes (Conati, Gertner and Vanlehn, 2002), AdaptErrEx (Goguadze et al., 2011a; 2011b) and INQPRO (Ting and Phon-Amnuaisuk, 2012). However, these systems do not determine how the knowledge level of a domain concept is affected by changes in the knowledge level of another related concept. Kavčič (2004b) has modeled how a student's knowledge level of a concept is improved when s/he subsequently applies it as prerequisite knowledge when learning a following concept. However, Kavčič's research only deals with how learning progresses; it deals neither with the possible decrease of knowledge due to the student forgetting some previously-learned concepts, nor the fact that the pace at which a student can learn the subsequent concepts depends upon how well s/he knows prerequisite concepts.

In view of the above, this work presents a novel ITS, which includes fuzzy logic techniques. Particularly, it includes a rule-based fuzzy logic mechanism for providing personalized tutoring to each learner and an innovative module, which is responsible for tracking cognitive state transitions of learners with respect to their progress or non-progress. In order to represent the student's knowledge level, a subset of the domain model is used as the student's knowledge model. So, an overlay student model is used. The rule-based fuzzy logic mechanism updates the overlay model by defining the student's knowledge level of all the concepts of the knowledge domain, each time. It uses fuzzy sets to represent the student's knowledge level and a mechanism of rules over the fuzzy sets. This mechanism is triggered after any change of the student's knowledge level of a domain concept. Then, it updates the student's knowledge level of the concepts that are related with the concept that the student has learned or forgot. The operation of the fuzzy mechanism is based on the FCMs that are used to represent the dependencies among the domain concepts. The new defined knowledge level of a student is used to activate the first dimension of the system's stereotype model that concerns the knowledge level of the student. The system's stereotype model has two more dimensions: the second dimension concerns the type of errors and the third concerns previous knowledge of the student on related with the learning material issues. The transition of a learner from one stereotype to another reveals her/his learning state. In particular, it reveals if a student learns or not, if s/he forgets and reasons these states. So, the system adapts its responses to each individual student dynamically providing individualized instruction, examination and advice.

The presented novel system was fully implemented and evaluated. In particular, a fully implemented and evaluated integrated environment for personalized e-training in programming and the language C, which is called ELaC, is presented. Software development relies on many different programming languages and tools ranging from procedural to object oriented and query languages and it is often the case that people have to learn a new language while they may or may not already know other languages. Given, the variety of backgrounds of prospective learners of programming, it is not easy to develop learning environments for all of them. In view of these problems, ELaC incorporates a student model responsible for identifying and updating the student's knowledge level, taking the different pace of learning of each individual learner into account. In particular, ELaC retains static information about each student, such as her/his previous experience on computer programming and the programming languages that s/he already knows. It also retains dynamic information, such as errors, misconceptions and progress, which is inferred by ELaC during the learner’s interaction with the system. In each learning session, ELaC recognizes the learner's knowledge level and the changes that occur in the state of her/his knowledge of a domain concept; it then updates the student's overall knowledge level according to the knowledge dependencies between the learning material’s domain concepts and the learner's progress. ELaC recognizes when a new domain concept is completely unknown to the learner, or when it is partly known due to the learner having previous related knowledge. Furthermore, it recognizes when a previously-known domain concept has been completely or partly forgotten by the learner. Thus it models either the possible increase or decrease of the learner’s knowledge. Furthermore, each time it checks if the learner's errors were due to possible confusion with features of another previously known programming language. In this case, ELaC responds accordingly by adapting instantly the sequence of learning lessons. The personalization achieved, allows every learner to complete the e-training course on their own pace and ability. ELaC was fully implemented in a postgraduate program in the field of informatics at the University of Piraeus, Greece, to teach students how to program with “C”. Its evaluation showed very encouraging results. The system can adapt dynamically to each individual learner’s needs by scheduling the sequence of lessons instantly. This personalization allows each learner to complete the e-training course at their own pace and according to their ability.

The remainder of this work is organized in three parts. The first part concerns the relevant work that has been done in the fields of Intelligent Tutoring Systems, knowledge domain representation, student modeling, programming tutoring systems, and fuzzy logic theory. The second part describes a novel approach of adaptive tutoring that includes fuzzy logic techniques. In particular, in this part a novel ITS architecture, which includes a tracking cognitive state transitions module and a fuzzy knowledge definer module, is presented. Then, a technique for knowledge domain representation that includes hierarchies and Fuzzy Cognitive Maps is described. In addition, a compound student model that combines stereotypes and overlay model with fuzzy logic techniques is presented. Furthermore, a novel rule-based fuzzy logic module for modeling automatically the learning or forgetting process of a student is described. Finally, in the second part, it is described the contribution of constructing state-chart diagrams to track the cognitive state transitions of learners. The third part describes a fully implemented and evaluated integrated environment for personalized e-training in programming and the language ‘C’, which is called ELaC. The implementation of ELaC was done according to the novel approach, which is described in the second part. Therefore, in the third part, the system's architecture, its knowledge domain representation and its hybrid student model are presented. Also, rule-based fuzzy logic module and the cognitive state transitions module of ELaC are described. Then, examples of operation of ELaC and related screen shots of the application are presented. The evaluation of the system follows. Finally, the conclusions drawn from this work are presented.

PART A:
Related work

1. Intelligent Tutoring Systems

"Intelligent tutoring system" (ITS) is a broad term, encompassing any computer program that contains some intelligence and can be used in learning. The technology of intelligent tutoring systems is an outgrowth of the earlier computer-aided instruction or CAI model. Just as human tutors arguably provide the most effective form of human instruction (Bloom 1984), intelligent tutoring systems (ITS) arguably provide the most effective form of e-learning (Dodds and Fletcher 2004; Wisher and Fletcher 2004). Research in ITS has been investigating how to make computer-based tutors more flexible, autonomous and adaptive to the needs of each student by endowing them with explicit knowledge of the relevant components of the teaching process and with reasoning capabilities to turn this knowledge into intelligent behavior (Conati, 2009). The design and development of such tutors lie at the intersection of computer science, cognitive psychology and educational research; this intersecting area is normally referred to as cognitive science (see Fig. 1) (Nwana, 1990).

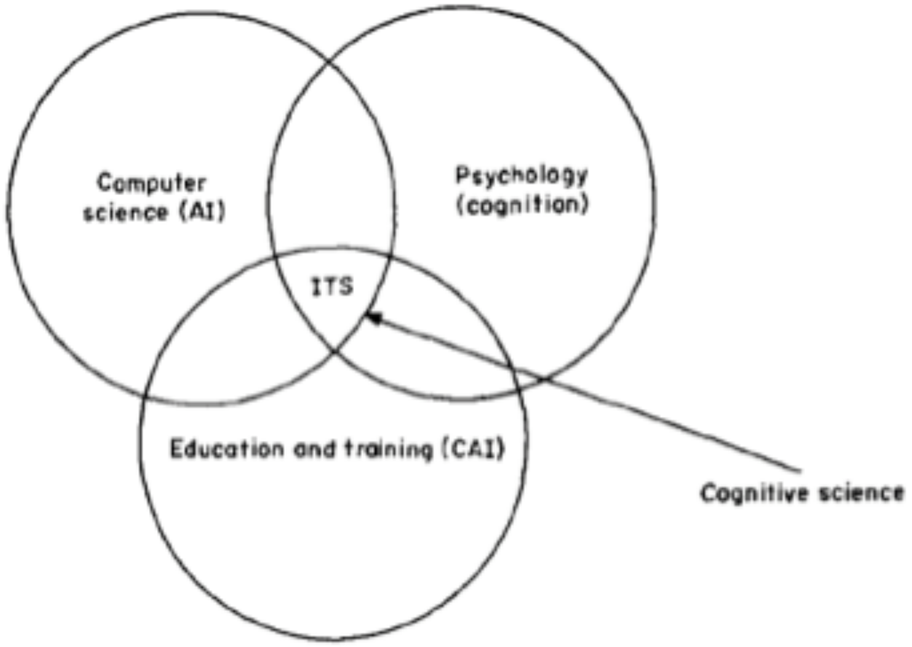


Figure 1: ITS domains (Nwana, 1990)

Intelligent Tutoring Systems are educational software systems that use artificial intelligence techniques to adapt the instruction to the individual student. They could provide an excellent one-on-one support to improve students' conceptual understanding. They can be accessed by teachers and students through the Web or as a stand-alone application. ITSs have merged as effective tools to promote active knowledge construction by capitalizing on the merits of one-on-one human tutoring in an automated fashion (Graesser, Conley, and Olney, 2012; Psotka, Massey & Mutter, 1988; Sleeman & Brown,1982; Woolf, 2009). According to Badaracco and Martinez (2013), an ITS provides direct customized instruction or feedback to students in their learning processes by means of Artificial Intelligence (AI) techniques, being mainly applied to knowledge representation, managing an instruction strategy as an expert both in the teaching and pedagogical issues in order to diagnose properly the student learning status at any time.

According to Conde et al. (2009) a ITS will have to cope with several features:

- To allow tutoring every task of people with disabilities, giving more autonomy in working environments.
- To have a multimodal Task Management System for data integration from different sources (speech, images, videos, and text) associated with each personalized profile.
- To be integrated into a mobile platform, i. e. a mobile telephone or PDA.
- To contain a multimedia interface that has to be friendly, reliable, flexible, and ergonomically adapted.
- To integrate a human emotional predictive management in order to prevent risk, emergency and blockage situations that can damage these people and interfere with their integration into working and social environments.
- To be entirely configurable by stakeholders without technological knowledge in order to enable an easy and flexible access.
- To show the capability of exporting the system to other collectives, i. e. the elderly.

To fulfill its objective, an ITS is organized by an architecture (Fig. 2) composed by (Shute & Psotka, 1996; de Souza & Ferreira, 2002):

- A domain model that stores the learning material needed for students' tutoring (what is taught?).
- A student model that stores information on the current user's knowledge to provide system adaptation (who is taught?).
- A tutoring component, which controls the tutoring process and make appropriate decisions based on the information provided by the other components of the ITS. It includes diagnosis of the student (Wenger, 1987) and instructional model (how is it taught?) (Bourdeau & Grandbastien, 2010).
- A Graphical User Interface (GUI) that allows the system to interact with the user (Nkambou, 2010; Wenger, 1987).

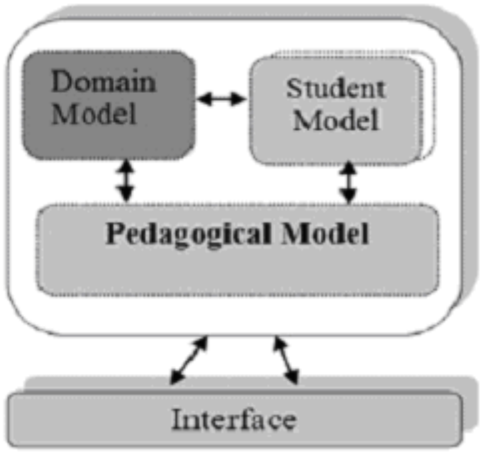


Figure 2: Generic architecture of ITS (Shute & Psotka, 1996)

ITSs have proven their effectiveness not only in controlled lab studies, but also in real classrooms (Anderson et al., 1995; Mitrovic, Martin & Suraweera, 2007; Mitrovic et al., 2004; VanLehn et al., 2005; Koedinger et al., 1997). These computer-based tutors achieve significant improvements in comparison to classroom learning due to their fine-grained knowledge of the instructional domain, their ability to employ various pedagogical strategies, and their student modelling capabilities which enables individualized instruction. According to

D'Mello et al. (2012) the ITSs that have been successfully implemented and tested, in United States, have produced learning gains with average effect sizes ranging from .79 to 1 sigma[1] (Corbett, 2001; Koedinger et al.,1997; VanLehn, 2011; VanLehn et al.,2007). When compared to classroom instruction and other naturalistic controls, the 1.0 effect sizes obtained by ITSs are superior to the .39 sigma effect for computer-based training, the.50 sigma effect for multimedia, and the .40 sigma effect obtained by novice human tutors (Cohen, Kulik & Kulik, 1982; Corbett, 2001; Dodds and Fletcher, 2004; Wisher and Fletcher, 2004).

Several researchers have developed several projects that utilized ITS in many areas. For example, Aleven, McLaren and Sewall (2009) have worked on improving students' understanding of mathematics in order to enhance students' achievements in high-stakes standards-based tests (K-12). Butz, Duarte and Miller (2006) have built Interactive Multimedia Intelligent Tutoring System (IMITS), an ITS that situated students in a virtual environment as if they have already graduated, hired, and required to solve real life problems, in the field of the electrical engineering. Graesser et al. (2005) have constructed Auto Tutor, which handled teaching driving, tackling a practical, real life, problem. Arnau et al. (2013) have built Hypergraph Based Problem Solver (HBPS), an Intelligent Tutoring Systems (ITS) that aims at supervising the resolution of word problems. The system allows the user to follow any resolution path, without imposing restrictions other than the definition of quantities before they are used. Myneni and Narayanan (2012) have designed and evaluated Virtual Physics System (ViPS), an ITS for exploring and learning physics concepts within the context of a particular class of simple machines. It was constructed during the investigation of the teaching and learning of physics concepts in middle schools. Schramm, et al. (2012) have designed an ITS that can teach UML skills to novice programmers.

[1] An effect-size measures the strength of a relationship between two variables. Cohen's d (see below) is a common measure of effect size in standard deviation units between two samples with means M1 and M2 and standard deviations s1 and s2. According to Cohen (1992), effect sizes approximately equal to .3, .5, and .8 represent small, medium, and large effects, respectively. $d = \frac{(M_1 - M_2)}{\sqrt{\frac{s_1^2 + s_2^2}{2}}}$. In learning contexts, an effect size of 1.0 sigma is roughly equivalent to an improvement of one letter grade.

Melis and Siekmann (2004) have built ActiveMath, a web-based intelligent tutoring system for mathematics. Muñoz-Merino et al. (2012) have constructed ISCARE (Information System for Competition based on pRoblem solving in Education), which is an innovative ITS that allows the competition among students for improving their learning process in a course. The tool takes some ideas from the Swiss-system widely used in chess and adapts them to the educational area. D’Mello et al. (2012) have developed an intelligent tutoring system (ITS) that aims to promote engagement and learning by dynamically detecting and responding to students’ boredom and disengagement. Mitrovic et al. (2011) have designed Thermo-Tutor, an ITS that teaches thermodynamic cycles in closed systems. Latham et al. (2010) have built Oscar, a conversational intelligent tutoring system, which dynamically predicts and adapts to a student’s learning style throughout the tutoring conversation. Oscar aims to mimic a human tutor to improve the effectiveness of the learning experience by leading a natural language tutorial and modifying the tutoring style to suit an individual’s learning style. AutoTutor is an intelligent tutoring system that simulates a human tutor’s natural language interaction with a student, so a student learns by having a conversation with AutoTutor (Graesser et al. 2005; VahnLehn et al., 2007). It constituted an inspiration for the design of GnuTutor, an open source intelligent tutoring system, whose goal is to create a freely available, open source ITS platform that can be used by schools and researchers alike (Olney, 2009). Conde et al. (2009) has developed LAGUNTXO, an ITS that facilitates the integration of people with intellectual disabilities into their working and social environments. Crowley and Medvedeva (2006) have developed a general intelligent tutoring system for teaching visual classification problem solving.

Furthermore, other architectures of ITSs have been suggested, some largely similar, but others radically different to the general architecture that is depicted in Fig. 2. For example, Anderson's Advanced Computer Tutoring (ACT) ITS architecture (Anderson, Boyle & Reiser, 1985) (Fig. 3) has the following four components:

[1] **The domain expert**: this module contains all the correct rules used for solving problems in the domain.

[2] **The bug catalogue**: this is an extensive library of common misconceptions and errors for the domain.

[3] **The teaching knowledge** (tutoring module).

[4] **The user interface**.

This architecture has been used by the Geometry tutor (Anderson, Boyle & Yost, 1985) and the Lisp tutor (Anderson & Reiser, 1985).

O'Shea et al. (1984) have present a five-ring model (Fig. 4). Its components include: student history, student model, teaching strategy, teaching generation and teaching administration.

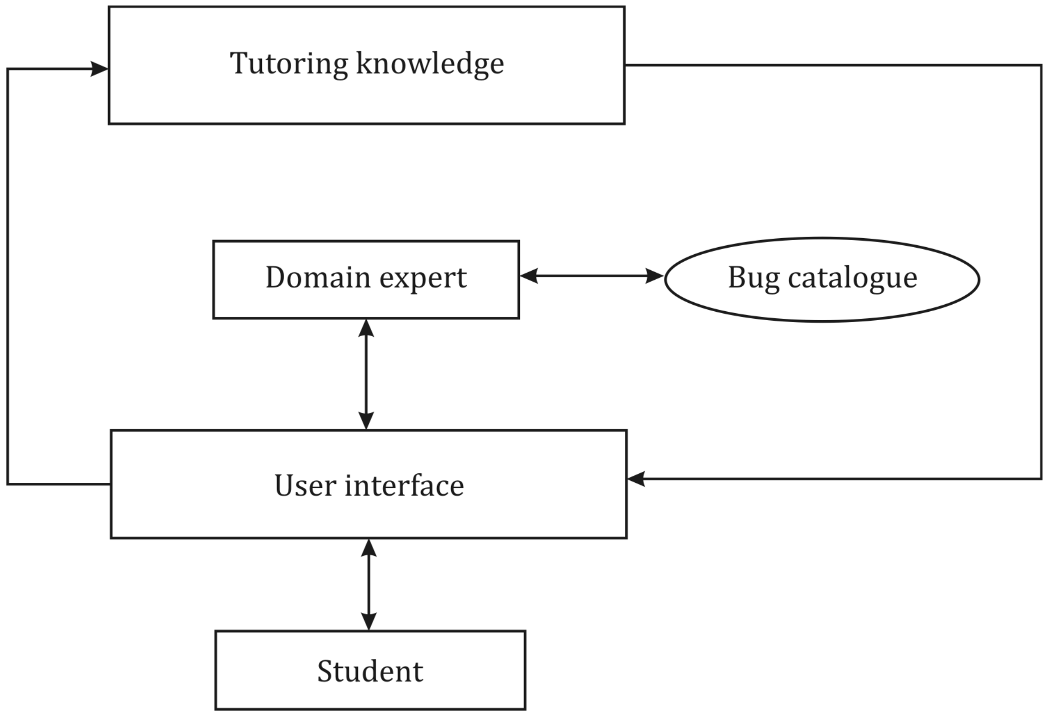


Figure 3: ACT's architecture (Anderson, Boyle & Reiser, 1985)

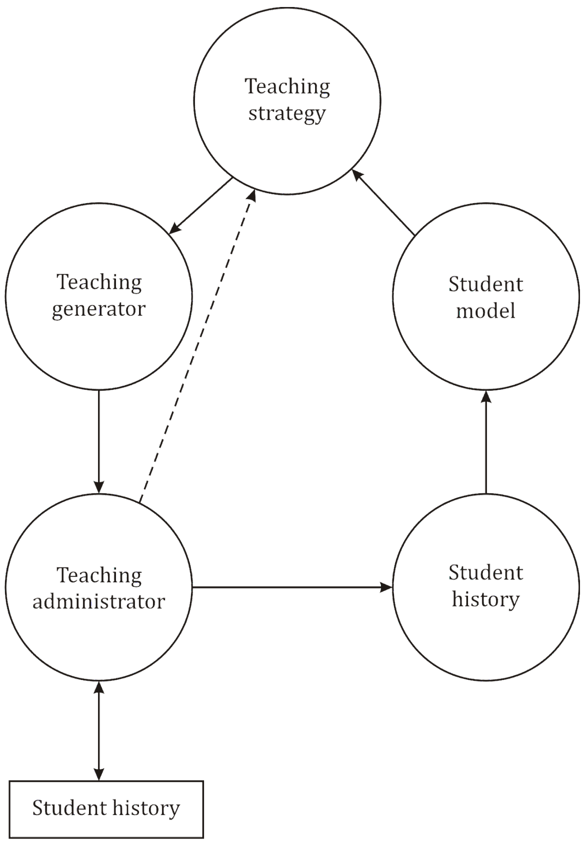


Figure 4: O'Shea et al's ITS architecture (O'Shea et al, 1984)

Badaracco and Martinez (2011) have been designed a novel architecture that extends the components of the general architecture of an ITS (Fig. 5). The proposed architecture is based on competency education (ITS-C). It establishes a link between the use of ITS and the pedagogical model based on competences. In ITS-C the diagnosis process estimates, updates and stores, in the student's model, the knowledge achieved by the student during the learning process. In figure 5 DMCo implies a domain model of competency (it is represented by a semantic network whose nodes are competence units, competence elements, descriptors and their relations), CuDM implies a curriculum domain model (it deploys the DMCo according to a teaching strategy that defines the competences associated to a professional profile to perform a training proposal in different situations), SMC implies the student model of competence (it stores student's information, whose data are updated through a diagnosis process).

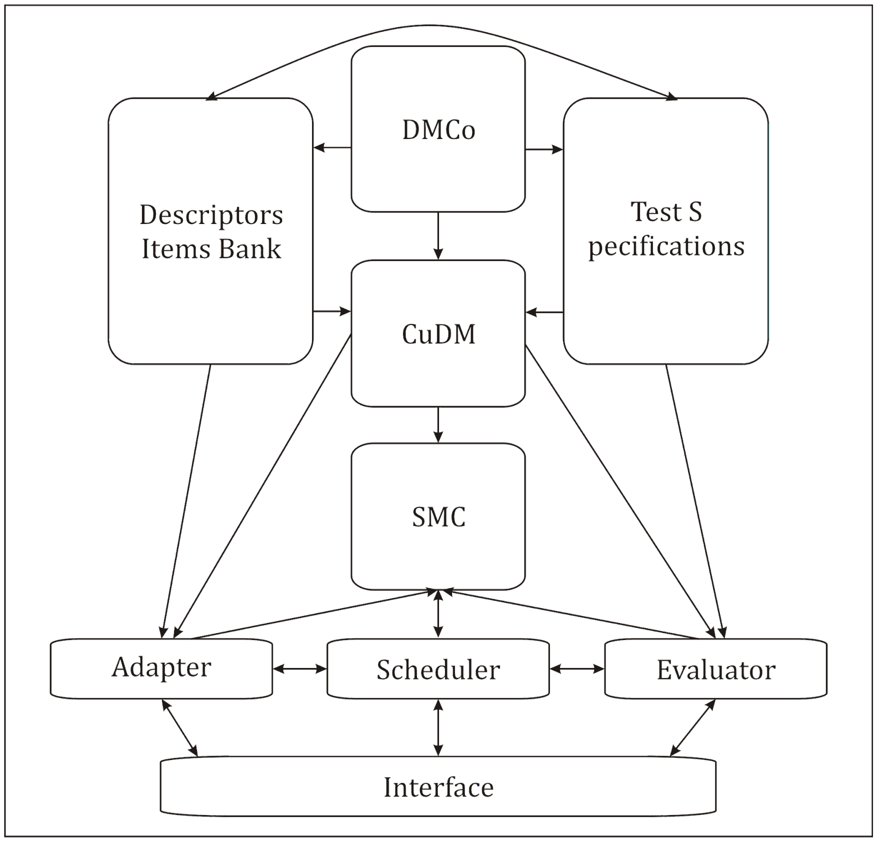


Figure 5: An ITS-C architecture (Badaracco & Martinez, 2011)

Meshref and Mohamed (2012) have modified the traditional structure of an ITS to improve system performance. They have added the Knowledge Manipulation Module and the Reporting Module (Fig. 6). Using the knowledge manipulation module the instructor can add, modify, delete, and edit any quiz question or lecture content. The reporting module was created to facilitate briefing each student's learning status to different instructors, who can see the result of their pedagogical strategies as the system assesses and tutors each student.

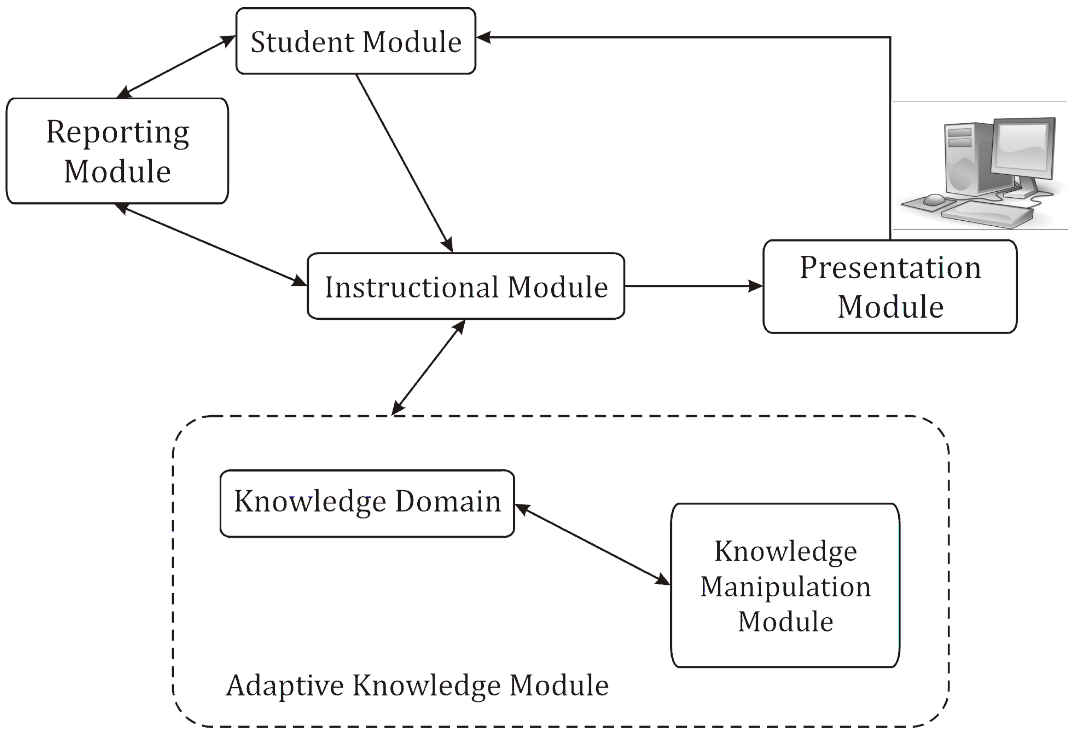


Figure 6: A modified structure of an ITS (Meshref & Mohamed, 2012)

Keles et al. (2009) have developed ZOSMAT, a student-centered intelligent tutoring system for tutoring of mathematics. It can be used for the purpose of either individual learning or real classroom environment with the guidance of a human tutor during a formal education process. This characteristic of ZOSMAT distinguishes itself from other intelligent tutoring systems. ZOSMAT follows a student in each stage of the learning process and guides her/him about what s/he will need to do. The main architecture of ZOSMAT consists of six components: ZOSMAT manager, question bank, student model, content structure, expert model, and user interface. In summary, the manager coordinates the different ZOSMAT components so they can work together smoothly. Conceptually, the relationship among the six components: ZOSMAT manager, content structure, student model, question bank, expert model and user interface can be viewed as in figure 7.

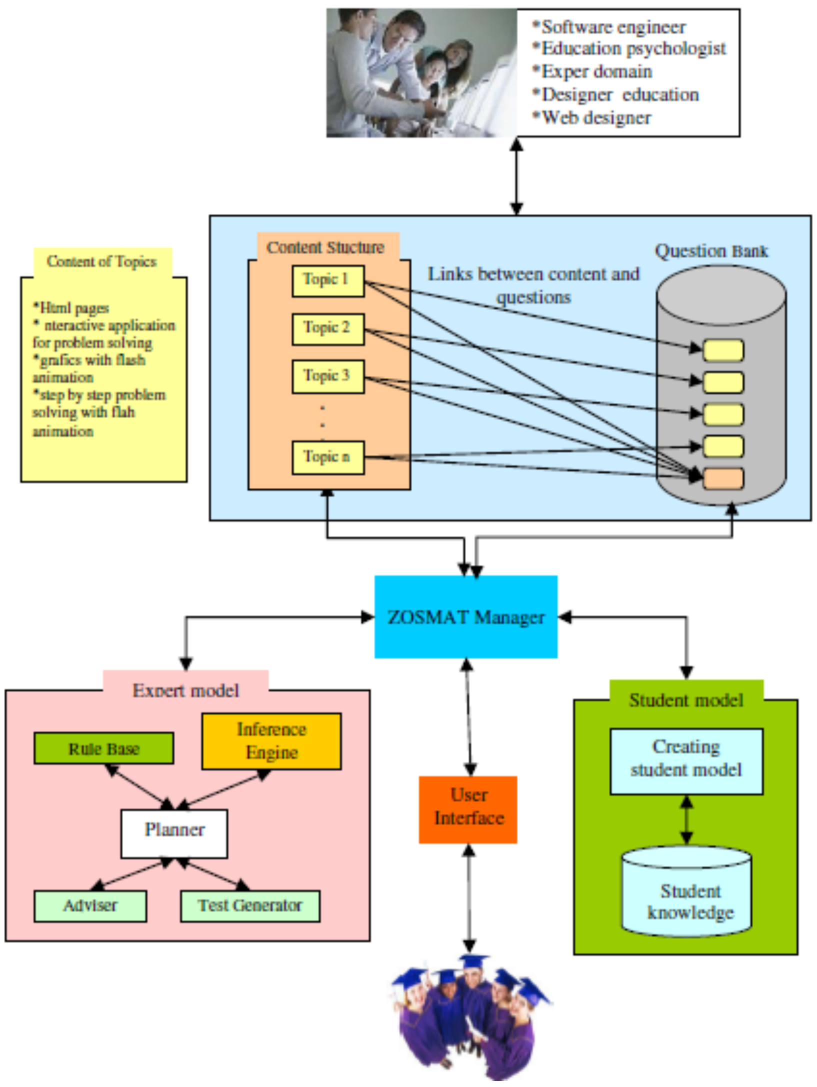


Figure 7: The architecture of ZOSMAT (Keles et al., 2009)

Moreover, Kaklauskas, Zavadskas and Ditkevičius (2006) have developed an ITS for Construction and Real Estate Management Master Degree Studies (ITS-CREM) in order to increase the efficiency and quality of e-learning studies at Vilnius Gediminas Technical University in Lithuania. ITS-CREM is used for the whole master degree study program, while traditional ITSs are used only for a single subject or module. ITS-CREM consists of six subsystems: Domain Model (it contains information and knowledge that the tutor is teaching), Student Model (it stores data that is specific to each individual student), Tutor and Testing Model (it provides a model of the teaching process and supports transition to a new knowledge state), Database of Computer Learning Systems (enables the use of the following Web- based computer learning systems: construction, real estate, facilities management, international trade, ethics, innovation, sustainable development and building refurbishment, etc.) , Decision Support Subsystem (is used in mostly all components of the ITS-CREM by giving different levels of intelligence for these components. It aids and strengthens some kinds of decision processes) and Graphic Interface.

Also, Kazi, Haddawy and Suebnukarn (2012) have proposed an alternative to the traditional ITS architecture by using a hint generation strategy that leverages domain ontology to provide effective feedback. This strategy has been incorporated in METEOR, a tutoring system for medical Problem-based learning

2. Knowledge Domain Representation

The knowledge domain module is one of the most major modules of an Intelligent Tutoring System. It contains a description of the knowledge that represents expertise in the subject-matter domain the ITS is teaching. The particular module has been introduced in Intelligent Tutoring Systems, but its use has been extended to most current educational software applications that aim to be adaptive and/or personalized. The knowledge domain module is responsible for the representation of the subject matter taking into account the course modules, which involve domain concepts. Either the order in which each domain concept has to be taught or the knowledge dependencies that exist between the domain concepts of the learning material have to be represented.

The knowledge domain representation is the base for the representation of the learner's knowledge, which is usually performed as a subset of the knowledge domain. Therefore, the knowledge domain representation in an adaptive and/or personalized tutoring system is an important factor for providing adaptivity. As Peylo et al. (2000) have pointed out, to enable communication between system and learner at content level, the domain model of the system has to be adequate with respect to inferences and relations of domain entities with the mental domain of a human expert. The appropriate approach for knowledge representation makes easier the selection of the appropriate educational material (learning objects) so as to support the teaching of the specific domain of knowledge.

The most common used techniques of knowledge domain representation in adaptive tutoring systems are hierarchies and networks of concepts. A hierarchical knowledge representation is usually used in order to specify the order in which the domain concepts of the learning material have to be taught (Chen and Shen, 2011; Siddara and Manjunath 2007; Vasandani and Govindury 1995), and can be implemented through trees (Kumar 2005; Geng et al. 2011). For example, in INMA, which is a knowledge-based authoring tool for music education, the knowledge domain is described in terms of hierarchies (Virvou, Lampropoulos & Tsihrintzis, 2006). Also, Siddappa, Manjunath & Kurian (2009) have developed a multilevel hierarchical model for the representation of knowledge domain of an intelligent tutoring system for numerical method (ITNM). This multilevel hierarchical model was based on various aptitude levels of students. Examples of hierarchical representation are depicted in figures 8 and 9. However, hierarchies do not give information about the dependency relations that exist between the domain concepts. This kind of information is given by the network representation. Many adaptive tutoring systems, such as Web-PTV (Tsiriga and Virvou 2003a; 2003b), DEPTHS (Jeremić, Jovanović & Gasĕvić, 2009) and IDEAL (Khamis 2011) use a network of concepts for representing the knowledge domain. In a network of concepts, nodes represent concepts and arcs represent relations between concepts, such as “part-of”, “is-a” and prerequisite relations. Examples of knowledge domain representation with network of concepts are depicted in figures 10 and 11.

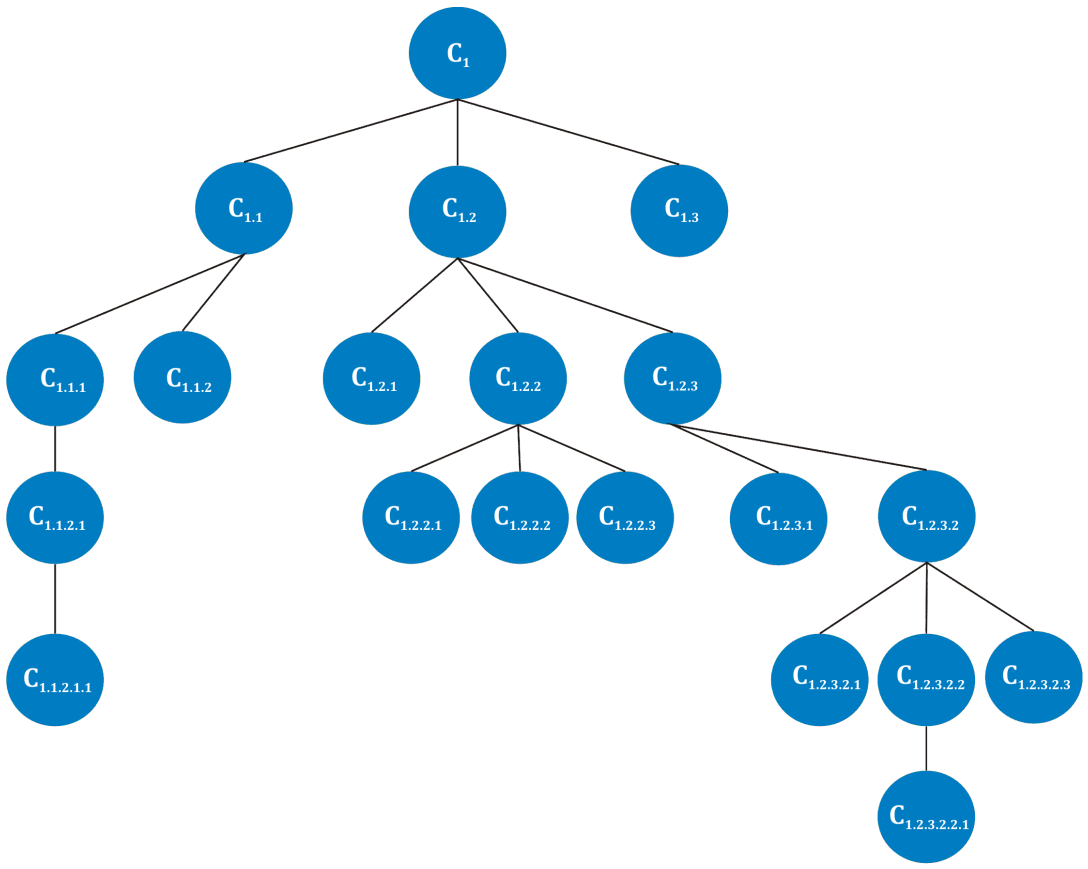


Figure 8: A hierarchical tree of domain concepts

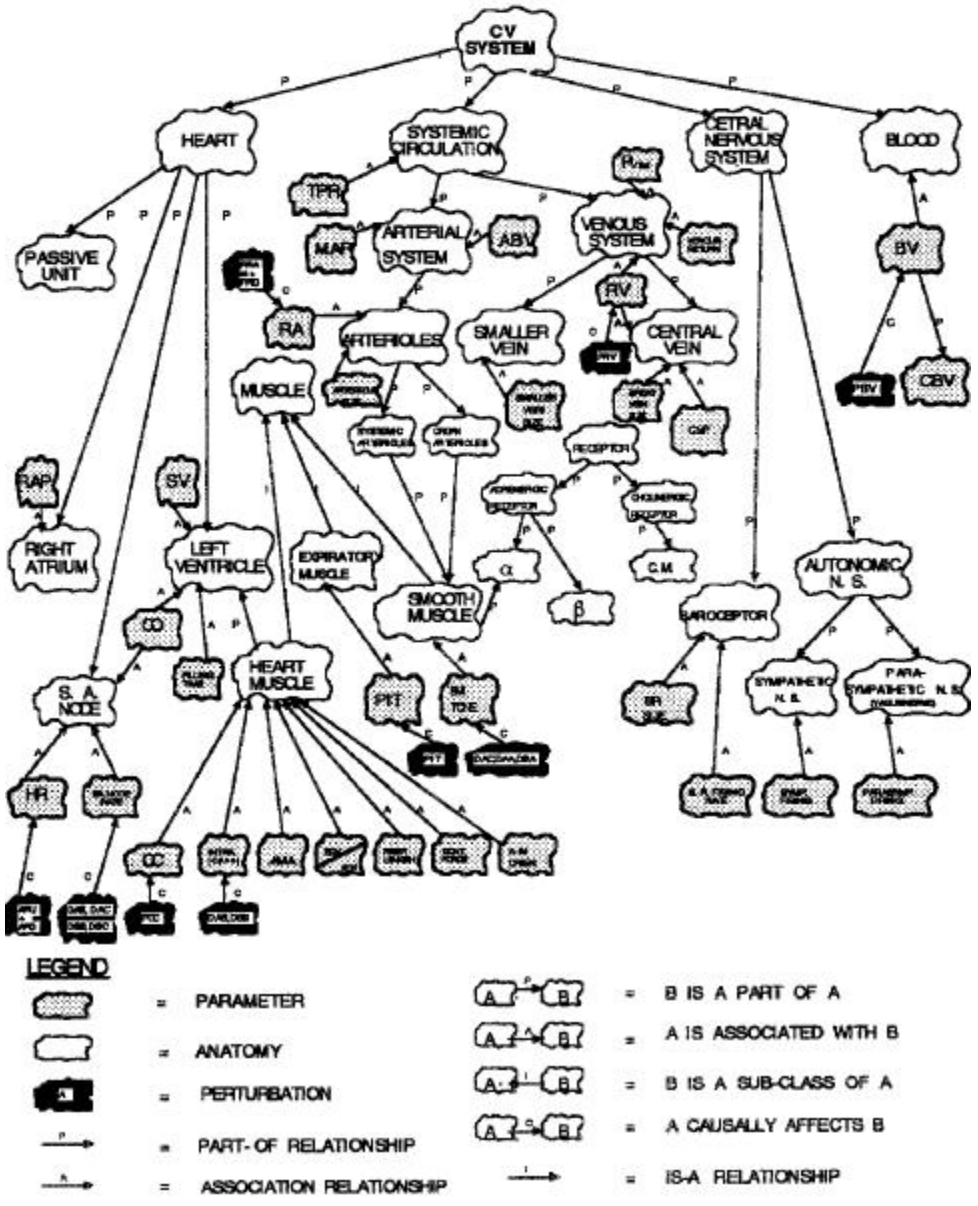


Figure 9: The hierarchical knowledge representation of CIRCSIM-TUTOR (Khuwaja et al., 1992)

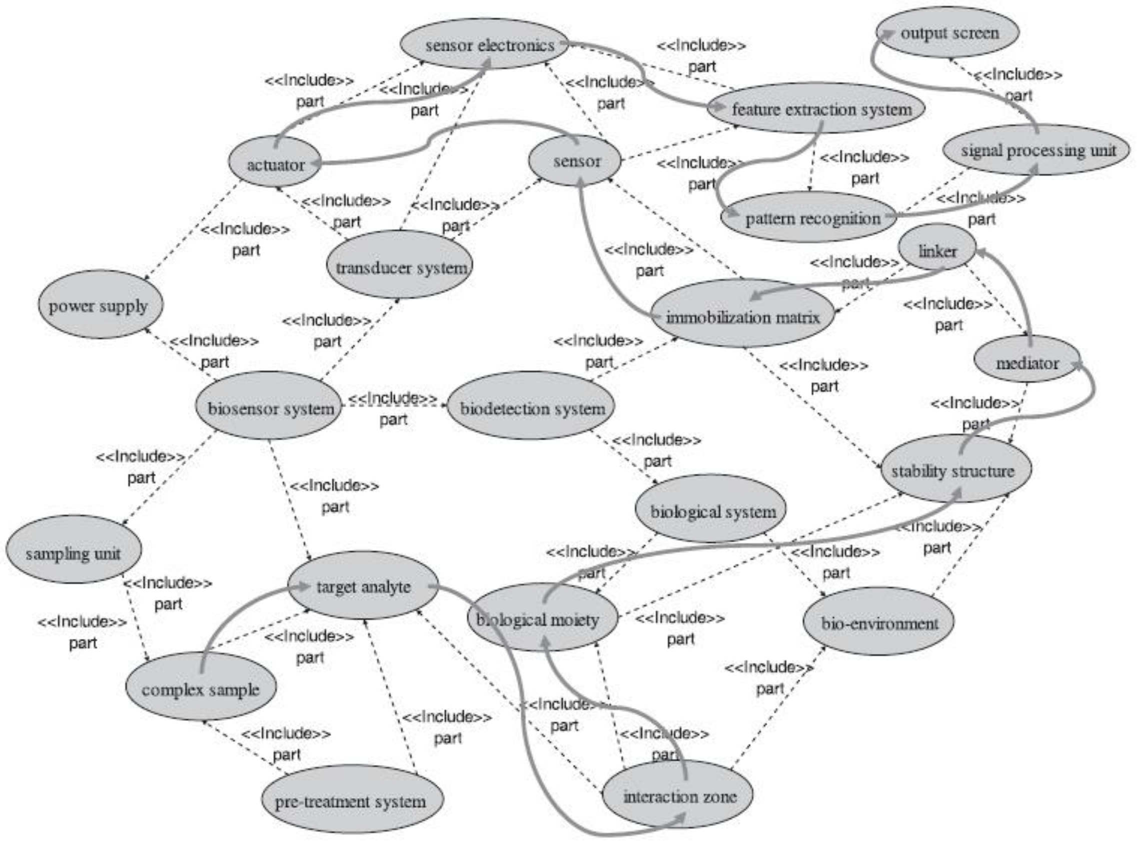


Figure 10: A network of concepts (Batzias & Siontorou, 2012)

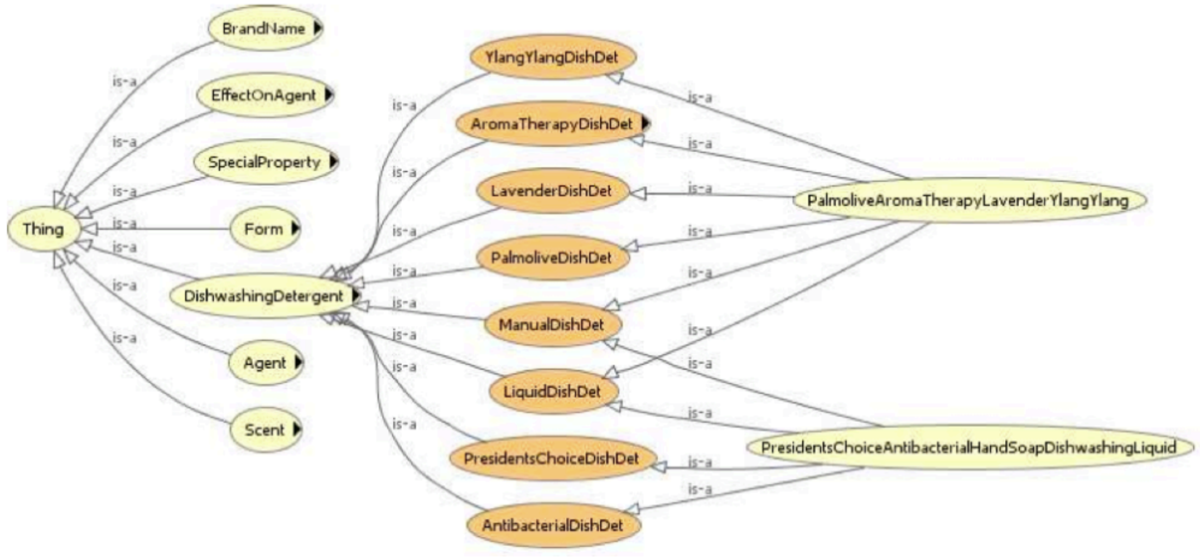


Figure 11: A network of concept for the semantic web (Rodriguez-Castr0, Glaser & Carr, 2010)

Another method for knowledge representation, especially in the case of educational systems, is ontologies (Breuker & Muntjewerff, 1999). Ontologies are a widely-used technique for facilitating understanding, communication and interoperation between human and software agents, by permitting the clear definition and explicit specification of all the basic concepts and terms of a concrete field (Panagiotopoulos, Seremeti & Kameas, 2010). Ontology-based knowledge representation includes machine-readable definitions of basic concepts in the domain and relations among them, while permits sharing common understanding of the structure of information among people and software agents and enables reuse of knowledge domain (Malik, Prakash & Rizvi, 2011).

Panagiotopoulos et al. (2012) have proposed a methodology that relies on the notion of ontology so as to represent the knowledge domain of the first year's course module “PLI10 - Introduction to Informatics” of the master program of the information technology of the Hellenic Open University (Fig. 12). This course module is specialized in the subject area of Programming Languages (C Programming).

Yu and Zhiping (2009) have used a three-level ontology-based knowledge representation structure in order to improve the sharing and reusing of multimedia teaching materials.

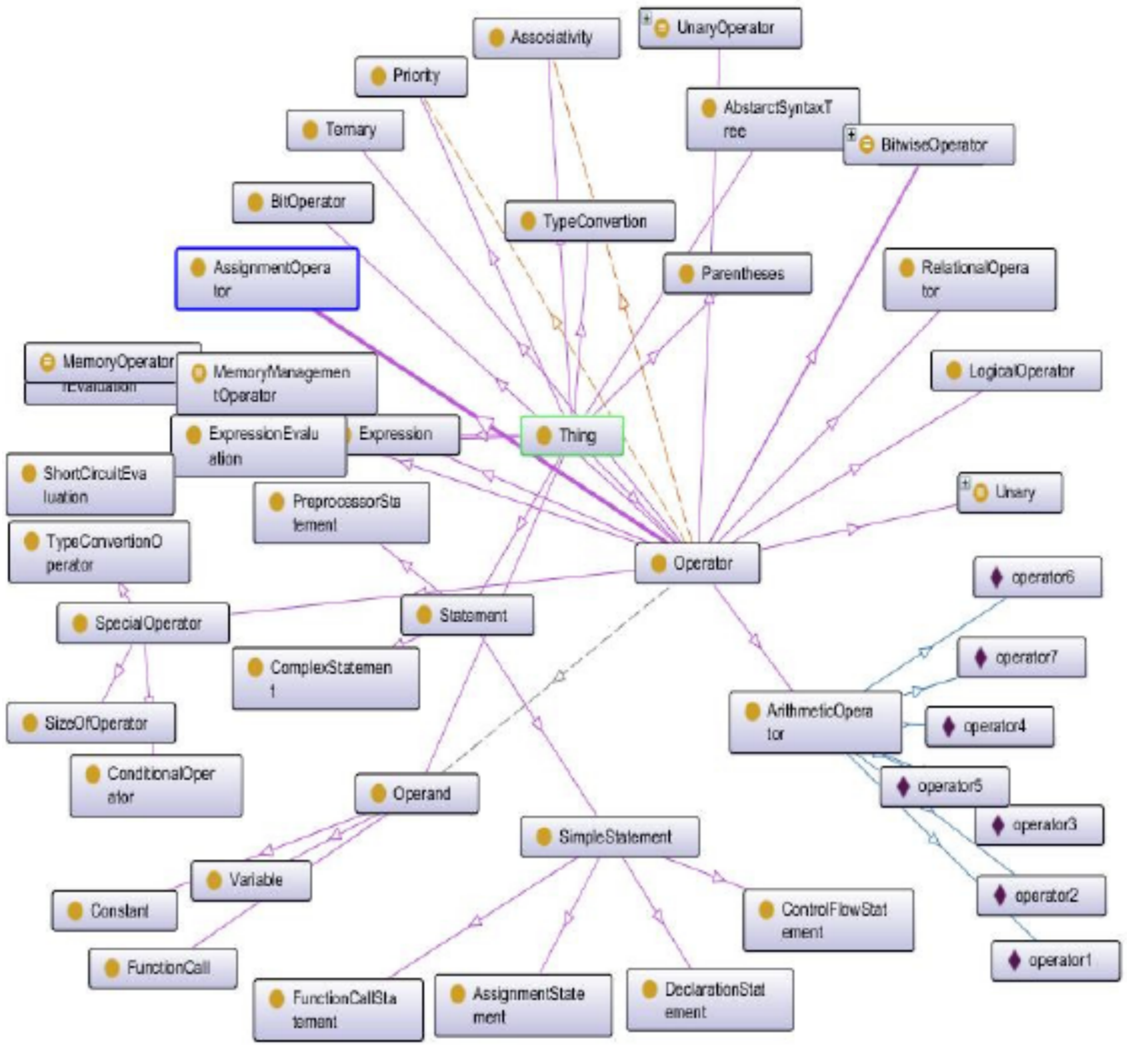


Figure 12: ontology model for C programming (Panagiotopoulos et al., 2012)

In addition to the above knowledge representation techniques, there is the method of concept maps (Fig. 13). Concept maps were developed in 1972 in the course of Novak's research program at Cornell where he sought to follow and understand changes in children's knowledge of science (Novak & Musonda, 1991). They are graphical tools for organizing and representing knowledge (Novak, 1990). According to Novak and Cañas (2008), some of the important characteristics of a concept map are: (i) the concepts are represented in a hierarchical fashion with the most inclusive, most general concepts at the top of the map and the more specific, less general concepts arranged hierarchically below; (ii) the inclusion of cross-links that are relationships or links between concepts in different segments or domains of the concept map; (iii) the feature of specific examples of events or objects that can be added to a concept map and help to clarify the meaning of a given concept. Concept maps is a very useful tool for analyzing and explaining concepts, but they are not so useful for the representation of the structure of the knowledge domain and the dependence relationships between its concepts.

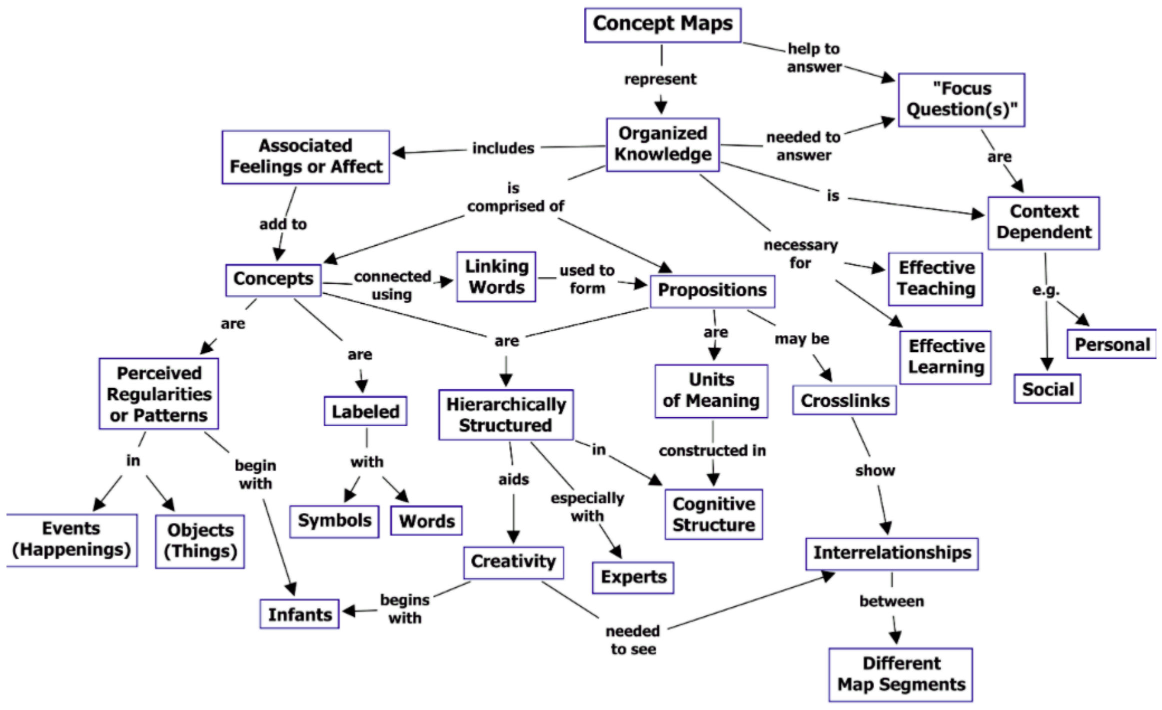


Figure 13: A concept map showing the key features of concept maps (Novak and Cañas (2008)

Cognitive maps represent the dependence and causal relationships between the domain concepts of a learning material. A concept map is a diagram that depicts suggested relationships between concepts. It is a graphical tool that designers, engineers, technical writers, and others use to organize and structure knowledge. It is a type of mental representation which serves an individual to acquire, code, store, recall, and decode information about the relative locations and attributes of phenomena in their everyday or metaphorical spatial environment. Cognitive maps have been studied in various fields, such as psychology, education, archaeology, planning, geography, cartography, architecture, landscape architecture, urban planning, management and history

(Knight, 2002).

A concept map typically represents ideas and information as boxes or circles, which it connects with labeled arrows in a downward-branching hierarchical structure. The relationship between concepts can be articulated in linking phrases such as causes, requires, or contributes to (Novak & Cañas, 2006). For example the cognitive map of figure 14 depicts the sequence of the week's days (www.timestructures.com/cognitive_maps_of_time.htm). Also, figure 15 (www.ochoadeaspuru.com/fuzcogmap/offeranddemand.php) depicts the positive and/or negative causal relationships between the concepts of the offer and demand process. Finally, the cognitive map that is depicted in figure 16 represents the results of bad weather driving.

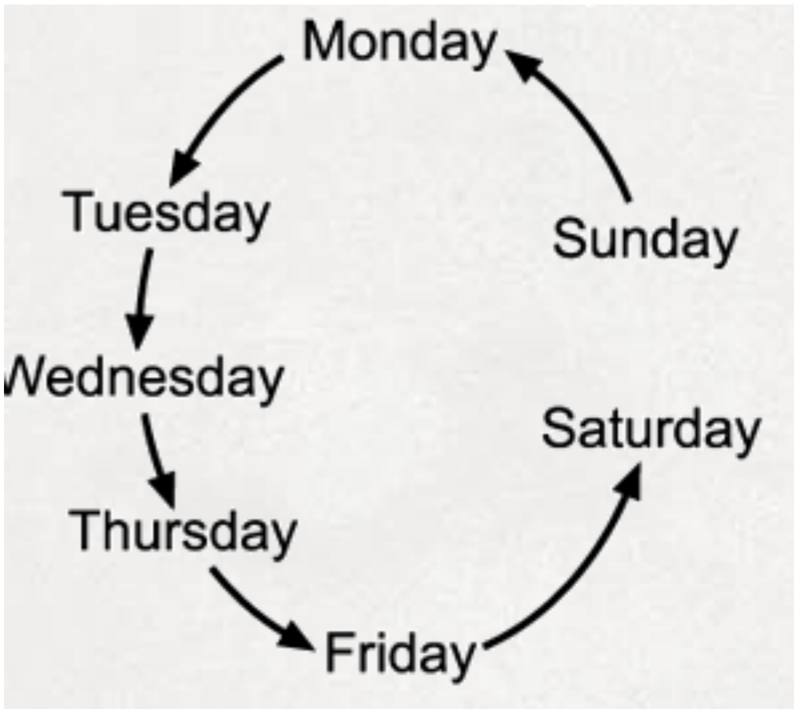


Figure 14: The sequence of week's days (www.timestructures.com/cognitive_maps_of_time.htm)

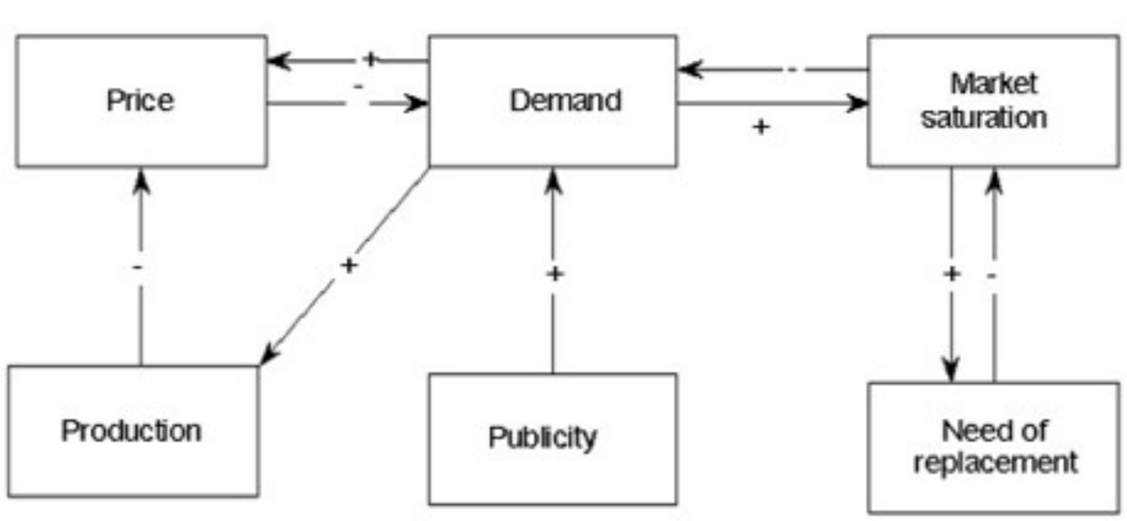


Figure 15: offer and demand causal relationships (www.ochoadeaspuru.com/fuzcogmap/offeranddemand.php)

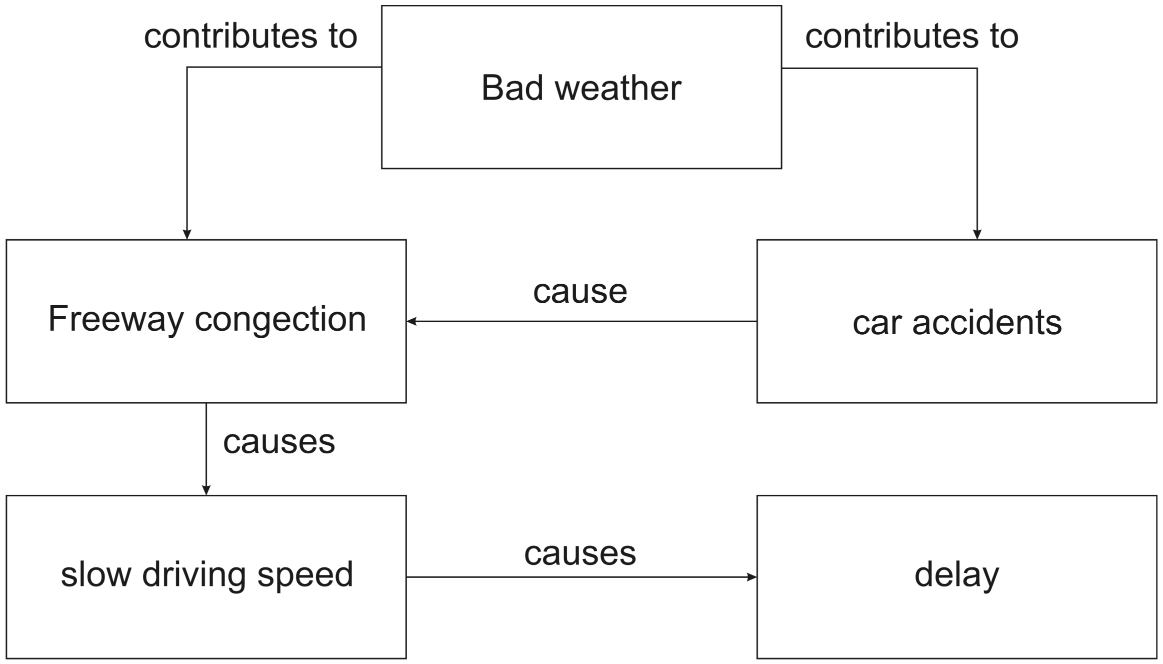


Figure 16: Results of bad weather driving

A crucial factor for a successful intelligent and/or personalized intelligent tutoring system is the knowledge domain representation. There are a variety of methods for the knowledge representation of the learning material: trees, network of concepts, cognitive maps, ontologies. The developer has to choose each time the most appropriate knowledge representation method or the most appropriate combination of knowledge representation methods, accordingly to the needs of the tutoring system. The elements (modules) of the knowledge representation schema, which represent the domain concepts of the learning material, are performed in a variety of formats: images, doc files, pdf files, ptt files, xml files, html files.

3. Student Modeling

A student model is the base for personalization in computer-based educational applications. It is a core component in any intelligent or adaptive tutoring system that represents many of the student’s features such as knowledge and individual traits (Brusilovsky & Millan, 2007). Self (1990) has pointed out that student modeling is a process devoted to represent several cognitive issues such as analyzing the student’s performance, isolating the underlying misconceptions, representing students’ goals and plans, identifying prior and acquired knowledge, maintaining an episodic memory, and describing personality characteristics. Therefore, by keeping a model for every user, a system can successfully personalize its content and utilize available resources accordingly (Kyriacou, 2008).

Imagine the student model as an ‘avatar’ of a real student in the virtual world, the dimensions of the student model correspond to the aspects of the physical student and the properties of the student model represent the characteristics of the real student (Yang, Kinshuk, & Graf, 2010). Student modeling is one of the key factors that affects automated tutoring systems in making instructional decisions (Li et al., 2011), since a student model enables understanding and identification of students’ needs (Sucar & Noguez, 2008). Student modeling can be defined as the process of gathering relevant information in order to infer the current cognitive state of the student, and to represent it so as to be accessible and useful to the tutoring system for offering adaptation (Thomson & Mitrovic, 2009).

As a consequence, a crucial factor for designing an adaptive educational system is the construction of an effective student model. In order to construct a student model, it has to be considered what information and data about a student should be gathered, how it will update in order to keep it up-to-date, and how it will be used in order to provide adaptation (Millán, Loboda and Pérez-de-la-Cruz,, 2010; Nguyen & Do, 2009). In fact, when a student model is constructed, the following three questions have to be answered: i) “What are the characteristics of the user we want to model?, ii) “How we model them?”, iii)

“How we use the user model?”.

In a recent review, Self (1988) identified twenty different uses that had been found for student models in existing ITSs. From analyzing this list, he notes that the functions of student models could be generally classified into six types.

(1) **Corrective**: to help eradicate bugs in the student's knowledge.

(2) **Elaborative**: to help correct 'incomplete' student knowledge.

(3) **Strategic**: to help initiate significant changes in the tutorial strategy other than the tactical decisions of 1 and 2 above.

(4) **Diagnostic**: to help diagnose bugs in the student's knowledge.

(5) **Predictive**: to help determine the student's likely response to tutorial actions.

(6) **Evaluative**: to help assess the student or the ITS.

3.1 Features and uses of student models

3.1.1 Student’s characteristics to model

A significant initial stage of constructing a student model is the selection of appropriate students’ characteristics that should be considered. The question: “what aspects of the student should we model in a specific intelligent tutoring system?” has to be answered when a new student model is built (Gonzalez, Burguillo & Llamas, 2006). According to Yang, Kinshuk & Graf (2010) in order to carry out the personalization efficiently, the student model needs to consider both domain dependent and domain independent characteristics. Also, some of these characteristics are static while others are dynamic. According to Jeremić, Jovanović & Gašević (2012) static features, such as email, age, native language etc., are set before the learning process takes place, in most cases using questionnaires. They usually remain unchanged throughout the learning session, although some of these data can be changed directly by a student through available options menu. Furthermore, according to the particular researchers, dynamic features come directly from the student’s interactions with the system and are those that the system constantly updates during learning sessions based on the collected data.

Therefore, the challenge is to define the dynamic student's characteristics that constitute the base for the system's adaptation to each individual student's needs. These characteristics include knowledge and skills, errors and misconceptions, learning styles and preferences, affective and cognitive factors, meta-cognitive factors. Knowledge refers to the prior knowledge of a student on the knowledge domain as well as her/his current knowledge level. This is usually measured through questionnaires and tests that the student has to complete during the learning process. Furthermore, through these tests as well as observing student's actions, the system can identify the misconceptions of students. Learning style refers to individual skills and preferences that affect how a student perceives, gathers and processes learning materials (Jonassen & Grabowski, 1993). According to Popescu (2009) some learners prefer graphical representations, others prefer audio materials and others prefer text representation of the learning material, some students prefer to work in groups and others learn better alone. Adapting courses to the learning preferences of the students has a positive effect on the learning process, leading to an increased efficiency, effectiveness and/or learner satisfaction (Popescu, Badica & Moraret, 2010). A proposal for modeling learning styles, which are adopted by many ITSs, is the Felder –Silverman learning style (FSLSM). FSLSM classifies students in four dimensions: active/reflective, sensing/intuitive, visual/verbal, and sequential/global (Felder & Silverman, 1988; Felder & Soloman, 2003). Another method for modeling learning styles is the Myers-Briggs Type Indicator (MBTI) (Bishop & Wheeler, 1994), which identifies the following eight categories of learning styles: extrovert, introvert, sensing, intuitive, thinking, feeling, judging, perceiving.

Given that there is evidence that experienced human tutors monitor and react to the emotional state of the students in order to motivate them and to improve their learning process (Johnson, Rickel & Lester, 2000; Lehman, Matthews, D'Mello & Person, 2008), an intelligent tutoring system should interpret the emotional state of students and adapt its behavior to their needs, giving an appropriate response for those emotions (Katsionis & Virvou, 2004). Therefore, affective factors are student characteristics that should be considered to build a student model. The affective states can be the following: happy, sad, angry, interested, frustrated, bored, distracted, focused, confused (Balakrishman, 2011). Rodrigo et al. (2007) have found that some of these emotions, like boredom or frustration, lead students to an off-task behavior. Off-task behavior means that students' attention becomes lost and they engage in activities that neither have anything to do with the tutoring system nor include any learning aim (Cetintas et al., 2010). Among typical off-task behavior examples are surfing the web, devoting time to off-topic readings, talking with order students without any learning aims (Baker et al., 2004). These behaviors are associated with deep motivational problems (Baker, 2007), and consequently, modeling affective factors can be a base for modeling students' motivation.

One of the most common categories of student characteristics that are described in a student model are student's cognitive features. These features refer to aspects such as attention, knowledge, ability to learn and understand, memory, perception, concentration, collaborative skills, abilities to solve problems and making decisions, analyzing abilities, critical thinking. However, students need not only to have cognitive abilities, but they also need to be able to critically assess their knowledge in order to decide what they need to study (Mitrovic & Martin, 2006). Thereby, adaptive and/or personalized tutoring systems must consider students' meta-cognitive skills. According to Flavell (1976) meta-cognition concerns to the active monitoring, regulation and orchestration of information processes in relation to cognitive objects on which they bear. In other words, the notion of meta-cognition deals with students' ability to be aware of and control their own thinking, for example, how they select their learning goals, use prior knowledge or intentionally choose problem-solving strategies (Barak, 2010). Some meta-cognitive skills are reflection, self-awareness, self-monitoring, self-regulation, self-explanation, self-assessment, and self-management (Pẽna & Kayashima, 2011).

3.1.2 How a student model is used

It has been said that a well-designed tutoring system actively undertakes two tasks: that of the diagnostician, discovering the nature and extent of the student's knowledge, and that of the strategist, planning a response using its findings about the learner (Glaser, Lesgold & Lajoie, 1987; Michaud & McCoy, 2004; Spada, 1993). This is the main role of student model, which is the base for personalization in Intelligent Tutoring Systems (Devedzic, 2006). The information of a student model is used by the system in order to adapt its responses to each individual student dynamically providing personalized instruction, help and feedback.

The student model is used for accurate student diagnosis in order to predict students' needs and adapt the learning material and process to each individual student's learning pace. It is used to produce highly accurate estimations of the student's knowledge level and cognitive state in order to deliver to them the most appropriate learning material. Furthermore, an adaptive and/or personalized tutoring system can consult the student model in order to recognize the learning style and preferences of a student and make a decision about the learning strategy that is likely to be the most effective for her/him. Moreover, an adaptive and/or personalized educational system can select appropriate learning methods in order to increase the effectiveness of tutorial interactions and improve the learning and motivation by predicting of student affective state. In addition, a student model can be used for identifying the student's strength and weaknesses in order to provide her/him individualized advice and feedback. Also, the system can provide the learner with more complicated tasks and proper learning methods in order to enhance deep learning and help her/him to become a better learner, by identifying her/his meta-cognitive skills.

3.2 Student Modeling Approaches

3.2.1 Overlay

One of the most popular and common used student models is the overlay model. It was invented by Stansfield, Carr and Coldstein (1976) and has been used in many systems ever since. The main assumption underlying the overlay model is that a student may have incomplete but correct knowledge of the domain. Therefore, according to the overlay modeling, the student model is a subset of the domain model (Martins et al., 2008; Vélez et al., 2008), which reflects the expert-level knowledge of the subject (Brusilovsky & Millán, 2007; Liu & Wang, 2007). The differences between the student's and the expert's set of knowledge are believed to be the student's lack of skills and knowledge, and the instructional objective is to eliminate these differences as much as possible (Bontcheva & Wilks, 2005; Michaud & McCoy, 2004; Staff, 2001). Consequently, the domain is decomposed into a set of elements and the overlay model is simply a set of masteries over those elements (Nguyen & Do, 2008). The pure overlay model assigns a Boolean value, yes or no, to each element, indicated whether the student knows or does not know this element, while in its modern form, an overlay model represents the degree to which the user knows such a domain element by using a qualitative measure (good-average-poor) or a qualitative measure such as the probability that the student knows the concept (Brusilovsky & Millán, 2007). It is obvious that the overlay model technique requires that the domain model of the adaptive and/or personalized tutoring system represents individual topics and concepts. Thereby, its complexity depends on the granularity of the domain model structure and on the estimate of the student knowledge (Martins et al., 2008). So, the overlay model can represent the user knowledge for each concept independently and this is the reason for its extensive use.

Kassim, Kazi, and Ranganath (2004) used an overlay student model in a web-based intelligent learning environment for digital systems (WILEDS) in order to represent dynamically the emerging knowledge and skills of each student. Also, in MEDEA (Carmona & Conejo, 2004) student modeling is performed by an overlay model in which for each domain concept an estimation of the student knowledge level on this concept is stored. InfoMap (Lu et al., 2005; Lu, Ong & Hsu, 2007), which is designed to facilitate both human browsing and computer processing of the domain ontology in a system, uses an overlay student model in combination with a buggy model for identification of the deficient knowledge. Another adaptive tutoring system that performs student modeling through an overlay student model is ICICLE (Michaud & McCoy, 2004). ICICLE's student model attempts to capture the user's mastery of various grammatical units and thus can be used to predict the grammar rules s/he is most likely using when producing language. Kumar (2006a; 2006b) has used overlay technique for student modeling in programming tutors. Glushkova (2008) applied a qualitative overlay student model for modeling learners' knowledge level to DeLC system. However, because she wanted to model, also, learners' manner of access to training resources, their preferences, habits and behaviors during the learning process, she combined the overlay model with stereotype modeling (stereotypes are referred below). Furthermore, LS-Plan (Limongelli et al., 2009), which is a framework for personalization and adaptation in e-learning, uses a qualitative overlay model. IWT (Albano, 2011) models competence in mathematics in an e-learning environment through an overlay model, which applies an ontology-based representation of the knowledge domain. Also, Mahnane, Laskri and Trigano (2012) build an adaptive hypermedia system that integrates thinking style (AHS-TS) by applying an overlay model. Finally, PDinamet (Gaudioso, Montero & Hermandez-del-Olmo, 2012) is a web-based adaptive learning system for the teaching of physics in secondary education, which uses an overlay model in order to provide effective and personalized selection of the appropriate learning resources.

It is obvious from the applications of the overlay model that it does not allow representing neither the incorrect knowledge that the student acquired or might have acquired, nor the different cognitive needs, preferences and learners' behavior and personality. As Rivers (1989) point out, overlay models are inadequate for sophisticated models 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. That is the reason why many adaptive and/or personalized tutoring systems perform student modeling, combing overlay model with other student modeling approaches like stereotypes, perturbation and fuzzy techniques.

3.2.2 Stereotypes

Another common used approach of student modeling is stereotyping. Stereotypes were introduced to user modeling by Rich (1979) in the system called GRUNDY. The main idea of stereotyping is to cluster all possible users of an adaptive system into several groups according to certain characteristics that they are typically shared. Such groups are called stereotypes. More specifically, a stereotype normally contains the common knowledge about a group of users. A new user will be assigned into a related stereotype if some of his/her characteristics match the ones contained in the stereotype. According to Kay (2000) the stereotype M has a set of trigger conditions {tMi}, where each {tMi} is a Boolean expression based upon components {c_i} of the student model, and a set of retraction conditions, {rMi}. The primary action of a stereotype is illustrated by Eq (1) and a stereotype is deactivated when any of the retraction conditions becomes true (Eq (2)).

$$if\ \exists i,\ tM_i = true \rightarrow active(M) \qquad \textbf{(1)}$$

$$if\ \exists j,\ rM_i = true \rightarrow not\ activate(M) \qquad \textbf{(2)}$$

The stereotype is a particularly important form of reasoning about users and also student modeling with stereotypes is often a solution for the problem of initializing the student model by assigning the student to a certain group of students (Tsiriga & Virvou, 2002).

Stereotypes have been used for student modeling in many adaptive and/or personalized tutoring systems and often they are combined with other methods of user modeling. INSPIRE (Grigoriadou et al., 2002; Papanikolaou et al., 2003) is an intelligent system for personalized instruction in a remote environment, which classifies knowledge on a topic to one of the four levels of proficiency (insufficient, rather insufficient, rather sufficient, sufficient). Except of stereotypes, it uses, also, a fuzzy approach to deal with the uncertainty of student diagnosis, as well as an overlay model. Web-PTV (Tsiriga & Virvou, 2003a; 2003b), which teaches the domain of the passive voice of the English language, provides individualized tutoring and advice through the student model that is based on stereotypes and a machine learning technique. Carmona, Castillo and Millán (2008) used a stereotype-like approach, which classifies students in four dimensions according to their learning styles, in combination with Bayesian networks in order to design a system that is able to select the more adequate objects for each student. A stereotype-like approach was used, also, in WELSA for adapting the courses to the learning preferences of each student (Popescu, Badica & Moraret, 2009). AUTO-COLLEAGUE (Tourtoglou & Virvou, 2008; Tourtoglou & Virvou, 2012) that is an adaptive and collaborative learning environment for UML performs student modeling through a hybrid student model based on perturbation and the stereotype-based modeling technique. The stereotypes of AUTO-COLLEAGUE are related to three aspects of the performance of the trainee (the level of expertise, the performance type and the personality). Also, Chrysafiadi and Virvou (2008) developed a three-dimensional stereotype approach (1st dimension: knowledge level, 2nd dimension: type of programming errors, 3rd dimension: previous knowledge) for a web-based educational application that teaches the programming language Pascal (Web_Tutor_Pas), in order to adapt its responses to each individual student dynamically. Moreover, Kofod-Petersen et al. (2008) adopted the stereotyping approach for modeling the learners of their intelligent learning environment. CLT (Durrani & Durrani, 2010), which is a C++ tutor, is another adaptive tutoring system that uses stereotypes in order to provide an individualized learning environment. Finally, a stereotype-like approach of student modeling is used in Wayang Outpost, which is a software tutor that helps students learn to solve standardized-test type of questions, in particular for a math test called Scholastic Aptitude Test, and other state-based exams taken at the end of high school in the USA, in order to discern factors that affect student behavior beyond cognition (Arroyo, Meheranian & Woolf, 2010).

The advantages of using the stereotype technique are that the knowledge about a particular user will be inferred from the related stereotype(s) as much as possible, without explicitly going through the knowledge elicitation process with each individual user and the information about user groups/stereotypes can be maintained with low redundancy (Zhang & Han, 2005). Furthermore, Kay (2000) reports that an appealing property of the stereotype is that it should enable a system to get started quickly on its customized interaction with the user.

However, stereotypes deal with problems as well. Stereotype approach is quite inflexible due to the fact that stereotypes are constructed in a hand-crafted way before real users have interacted with the system and they are not updated until a human does so explicitly (Tsiriga & Virvou, 2002). Moreover, Kass (1991) argues that stereotypes suffer from two problems. First, in order to use them, the set of system users must be divisible into classes; however, such classes may not exist. Second, even if it is possible to identify classes of system users, the system designer must build the stereotypes; this is a process that is both time-consuming and error-prone.

3.2.3 Perturbation

A perturbation student model is an extension of the overlay model that represents the student's knowledge as including possible misconceptions as well as a subset of the expert's knowledge (Mayo, 2001). It represents learners as the subset of expert's knowledge, like the overlay model, plus their mal-knowledge (Nguyen & Do, 2008). This extension allows for better remediation of student mistakes, since the fact that a student believes something that is incorrect is pedagogically significant (Surjono & Maltby, 2003). The perturbation student model is useful for diagnostic reasoning. According to Martins et al. (2008), the perturbation student model is obtained by replacing the correct rules with the wrong rules, which when applied they lead to the answers of the student. Since there can be several reasons for a student wrong answer (several wrong values that lead to the student answer) the system proceeds to generate discriminating problems and presents them to the student to know exactly the wrong rules that this student has.

The collection of mistakes included in a perturbation model is usually called bug library and can be built either by empirical analysis of mistakes (enumeration) or by generating mistakes from a set of common misconceptions (generative technique). For enumerative modeling, the system developers analyze the model and determine possible errors students can make or are prone to make (Smith, 1998). The theory used on most existing systems is the enumerative bug theory, which is also known as the “buggy” model because it was first used in the intelligent tutoring system called BUGGY (Brown & Burton, 1975). However, enumerative modeling techniques suffer from costly computational requirements. Furthermore, as Clancey (1988) points out, not all students may fit the program’s pre-enumerated set of bugs and he adds that a more capable program would attempt to generate a description of bugs from patterns in a particular student’s behavior and a model of how bugs come about. This is called generative modeling, in which the system uses a cognitive model to detect students’ errors. Errors are regarded as failed extrapolation of the concepts learned, and if the general form of the extrapolation errors can be found, then the majority of the errors can be explained (Mayo, 2001).

The past decade many adaptive and/or personalized tutoring systems have embedded a perturbation student model for reasoning the students’ behavior. Surjono and Maltby (2003) used a perturbation student model to perform a better remediation of student mistakes. LeCo-EAD (Faraco, Rosatelli & Gauthier, 2004) modeled students’ knowledge and misconceptions through an enumerative perturbation student model, which included both correct and incorrect knowledge propositions, in order to provide personalized feedback and support to the distant students in real time. An enumerative perturbation student model was also applied by Lu et al. (2005) in an intelligent tutoring system that taught basic arithmetic to children (InfoMap). Their perturbation student model, which involved 31 types of addition errors and 51 types of subtraction errors, allowed the reasoning of students’ errors and helped the system to expand the explanation during the feedback to the students. Finally, Baschera and Gross (2010) used a perturbation student model for spelling tarining, which represented student’s strength and weaknesses, in order to allow for appropriate remediation actions to adapt to students' needs.

3.2.4 Machine learning techniques

Student modeling involves a process of making inferences about the student's behavior taking into account her/his knowledge level, cognitive abilities, preferences, skills, aptitudes e.t.c. The processes of observation of student's action and behavior in an adaptive and/or personalized tutoring system, and of induction, should be made automated by the system. A solution for this is machine learning, which is concerned with the formation of models from observations and has been extensively studied for automated induction (Webb, 1998). Observations of the user's behavior can provide training examples that a machine learning system can use to induce a model designed to predict future actions (Webb, Pazzani & Billsus, 2001).

According to Sison and Shimura (1998), machine learning or machine-like techniques have so far been used in two areas of student modeling research:

- To induce a single, consistent student model from multiple observed student behaviors.
- For the purpose of automatically extending or constructing from scratch the bug library of student modelers.

Hence, the use of machine learning techniques in student modeling has become increasingly popular. Web-EasyMath (Tsiriga & Virvou, 2002; 2003c) uses a combination of stereotypes with the machine learning technique of the distance weighted k-nearest neighbor algorithm, in order to initialize the model of a new student. The student is first assigned to a stereotype category concerning her/his knowledge level and then the system initializes all aspects of the student model using the distance weighted k-nearest neighbor algorithm among the students that belong to the same stereotype category with the new student. Baker et al. (2004) applied a machine learning model in order to identify if a student is gaming the intelligent tutoring system in a way that leads to poor learning. Furthermore, Baker (2007) constructed a machine learning model that can automatically detect when a student using an intelligent tutoring system is off-task, i.e. engaged in behavior which does not involve the system or a learning task. In ADAPTAPlan (Jurado et al., 2008) fuzzy logic was used to evaluate students' assignments and to update the student model with their preferences by means of machine learning techniques. A machine learning technique was used, also, in the adaptive hypermedia educational system GIAS (Castillo, Gama & Breda, 2009), in combination with stereotypes. The adaptation techniques of GIAS are focused on the appropriate selection of the course's topics and learning resources, based on the student's goals, knowledge level, learning style. Wang, Yang and Wen (2009) adopted support vector machines, a machine learning method based on statistical learning theory, in order to provide personalized learning resource recommendation. POOLE III (Inventado et al., 2010) used a combination of Bayesian networks and machine learning technique in order to observe students' reactions while using an intelligent tutoring system and adjust feedback automatically to each individual learner. Similarly, Baker, Goldstein and Heffernan (2010) applied a student model based on a combination of Bayesian networks and machine learning technique. The machine learning constitutes the student model able to assess the probability that a student learned skill at a specific problem step and thus the system can predict the student knowledge. In addition, Al-Hmouz et al. (2010; 2011) combined two machine-learning techniques in order to model the learner and all possible contexts related to her/his current situation in an extensible way so that they can be used for personalization. Cetintas et al. (2010) used machine-learning techniques for the performing of the automatic detection of off-task behaviors in intelligent tutoring systems. SimStudent (Li et al., 2011) used a machine learning technique in order to construct student models automatically and improve the accuracy of prediction of real students learning performance. Finally, Balakrishnan (2011) build a student model upon ontology of machine learning strategies in order to model the effect of affect on learning and recognize for any learning task, what learning strategy, or combination thereof, is likely to be the most effective.

3.2.5 Cognitive Theories

Many researchers (e.g. Salomon, 1990; Welch & Brownell, 2000) point out that technology is effective when developers thoughtfully consider the merit and limitations of a particular application while employing effective pedagogical practices to achieve a specific objective. That is the reason that many researchers adopt cognitive theories in student models. A cognitive theory attempts to explain human behavior during the learning process by understanding human's processes of thinking and understanding. The Human Plausible Reasoning (HPR) theory, the Ortony, Clore and Collins (1988) (OCC) theory and the Control-Value theory are some cognitive theories that have been used in student modeling.

Particularly, the Human Plausible Reasoning (HPR) theory (Collins & Michalski, 1989) is a domain-independent theory originally based on a corpus of people's answers to everyday questions, which categorizes plausible inferences in terms of a set of frequently recurring inference patterns and a set of transformations on those patterns (Burstein & Collins, 1988; Burstein, Collins & Baker, 1991). It serves the purpose of adding more “human” reasoning to the computer and this is the reason that it has been used in F-SMILE (Virvou & Kabassi, 2002). The student model in F-SMILE uses a novel combination of HPR with a stereotype-based mechanism, in order to generate default assumptions about learners until it acquires sufficient information about each individual learner.

In addition, the OCC cognitive theory of emotions (Ortony, Clore & Collins, 1988), which allows modeling possible emotional states of students, has been used by Conati and Zhou (2002) for recognizing user emotions for their educational game prime climb. VIRGE is another ITS-game which has adopted OCC theory in order to provide important evidence about students' emotions while they learn (Katsionis & Virvou, 2004; Virvou, Katsionis & Manos, 2005). Moreover, Hernández, Sucar and Arroyo-Figueroa (2010) applied an affective student model combining the OCC theory with Bayesian Networks. The OCC theory has also used in a Mobile Medical Tutor (MMT) for modeling possible states that a tutoring agent may use for educational purposes (Alepis & Virvou, 2011). Finally, the emotional student model of PlayPhysics (Muñoz et al., 2011) uses, except of Bayesian Networks, the Control-Value theory (Pekrun et al., 2007), which is an integrative framework that employs diverse factors, eg. cognitive, motivational and psychological, to determine the existence of achievement emotions.

3.2.6 Constraint-Based Model

The Constraint-Based Model (CBM) has been proposed by Ohlsson (1994). It is based on Ohlsson's theory of learning from errors (Ohlsson, 1996), which proposes that a learner often makes mistakes when performing a task, even when s/he has been taught the correct way to to it. In CBM constraints are used to represent both domain and student knowledge. According to Martin (1999) a constraint is characterized by a relevance clause and a satisfaction clause: the relevance clause is a condition that must be true before the constraint is relevant to the current solution, and once it has been met, the satisfaction clause must be true for the solution to be correct. In other words, in CBM each constraint represents a bug in the form “when 'satisfaction clause' fails to hold true for 'relevance condition', the learner has introduced a bug”. Consequently, in CBM the knowledge domain is represented as set of constraints and the student model is the set of constraints that have been violated.

Therefore, the CBM approach provides a theoretically sound and practical solution to the intractable problem of student modeling (Ohlsson & Mitrovic, 2006). According to Mitrovic, Mayo, Suraweera and Martin (2001), the most important advantages of CBM are: its computational simplicity, the fact that it does not require a runnable expert module, and the fact that it does not require extensive studies of student bugs as in enumerative modeling. These advantages have lead researchers to apply the CBM approach to tutoring systems in a variety of domains. One such system is the SQLT-Web, which is a web-enabled intelligent tutoring system that teaches the SQL database language (Mitrovic, 2003). It observes students' actions and adapts to their knowledge and learning abilities, modeling the student with the CBM approach. KERMIT, which is an ITS that teaches conceptual database design, maintains two kind of student models: short-term, which is implemented as CBM, and long-term ones that is implemented as an overlay model (Suraweera & Mitrovic, 2004). Another system that uses CMB for student modeling is COLLECT-UML, which is an ITS that teaches object-oriented design using Unified Modeling Language (Baghaei, Mitrovic & Irwin, 2005). Furthermore, the CBM approach has been used in J-LATTE, which is an ITS that teaches a subset of the Java programming language (Holland, Mitrovic & Martin, 2009). INCOM is another tutoring system in the field of programming, which has used the CBM approach to diagnose the student's errors (Le & Menzel, 2009). Finally, Weerasinghe and Mitrovic (2011) have adopted CBM, in order to model the student's knowledge in EER-Tutor, which is another ITS that teaches conceptual database design and the early standalone version of which was KERMIT.

3.2.7 Fuzzy student modeling

Learning is not a “black and white” paper, but it is a complex process. Determining a student’s knowledge is not a straightforward task, since it often depends on and is reflected through things that cannot be directly observed and measured (Jeremić, Jovanović & Gasěvić, 2012). Especially in an intelligent tutoring system, where there is no direct interaction between the teacher and the student and technical difficulties, such as network congestion, cause difficulties and problems in gathering information about students’ mental state and behavior, the presence of uncertainty in student diagnosis is increased (Grigoriadou et al., 2002). One possible approach to encounter this uncertainty is fuzzy logic that was introduced by Zaheh (1965) as a methodology for computing with words, which cannot be done equally well with other methods (Zadeh, 1996), since the fuzzy logic based methods are more consistent with the human-being decision-making process (Shakouri & Tavassoli, 2012). Fuzzy logic is able to handle uncertainty in everyday problems caused by imprecise and incomplete data as well as human subjectivity (Drigas, Argyri & Vrettaros, 2009).

Fuzzy logic techniques can be used to improve the performance of an educational environment, in which decisions about the learning material that should be delivered and the feedback and advices that should be given to each individual learner, have to be taken, since according to Shakouri and Menhaj (2008) an algorithm based on fuzzy decision making helps to select the optimum model considering a set of criteria and model specifications. Indeed as Chrysafiadi and Virvou (2012) showed, the integration of fuzzy logic into the student model of an ITS can increase learners' satisfaction and performance, improve the system's adaptivity and help the system to make more valid and reliable decisions. Therefore, several researchers have incorporated fuzzy logic techniques in student modeling, due to its ability to naturally represent human conceptualizations.

Xu, Wang and Su (2002) used fuzzy models to represent a student profile in order to provide personalized learning materials, quiz and advices to each student. In F-CBR-DHTS (Tsaganoua et al., 2003) the diagnosis of students' cognitive profiles of historical text comprehension was done with fuzzy techniques. Moreover, TADV (Kosba, Dimitrova & Boyle, 2003; 2005) includes a student model, which combines an overlay model with fuzzy techniques, to represent the knowledge of individual students and their communication styles. Kavčič (2004a) succeeded to provide personalization of navigation in the educational content of InterMediActor system through the construction of a navigation graph and the adoption of fuzzy logic into student reasoning. A fuzzy-based student model was applied, also, by Stathacopoulou et al. (2005) to a discovery-learning environment that aimed to help students to construct the concepts of vectors in physics and mathematics. The use of fuzzy techniques allowed the diagnostic model to some extent imitate teachers in diagnostic students' characteristics, and equips the intelligent learning environment with reasoning capabilities that can be further used to drive pedagogical decisions depending on the student learning style. In addition, Salim and Haron (2006) constructed a framework for individualizing the learning material structure in an adaptive learning system, which utilized the learning characteristics and provide a personalized learning environment that exploit pedagogical model and fuzzy logic techniques. Jia et al. (2010) applied fuzzy set theory to the design of an adaptive learning system in order to help learners to memory the content and improve their comprehension. Furthermore, Goel, Lallé and Luengo (2012) used fuzzy logic representation for student modeling. This fuzzy student model facilitated student reasoning based on imprecise information coming from the student-computer interaction and performed the prediction of the degree of error a student makes in the next attempt to a problem. Finally, DEPTHS (Jeremić, Jovanović & Gasěvić, 2012), which is an intelligent tutoring system for learning software design patterns, models the student's mastery and cognitive characteristics through a combination of stereotype and overlay modeling with fuzzy rules that are applied during the learning process to keep student model update.

3.2.8 Bayesian Networks

Another well-established tool for representing and reasoning about uncertainty in student models is Bayesian Networks (Conati, Gertner & Vanlehn, 2002). A Bayesian Network (BN) is a directed acyclic graph in which nodes represent variables and arcs represent probabilistic dependence or causal relationships among variables (Pearl, 1988). The causal information encoded in BN facilitates the analysis of action sequences, observations, consequences and expected utility (Pearl, 1996). In student modeling nodes of a BN can represent the different components/dimensions of a student such as knowledge, misconceptions, emotions, learning styles, motivation, goals e.t.c.. Mayo and Mitrovic (2001) has classified Bayesian student modeling approaches into three types, according to how the structure of the network and prior, conditional probabilities are elicited. These types of Bayesian student models are expert-centric models, which use experts to specify the structure of the network and its corresponding initial prior and conditional probabilities, efficiency-centric models that restrict the structure of the network in order to maximize efficiency, and data-centric models, which use data from previous experiment and/or pre-tests to generate the network and its probabilities.

Bayesian networks have attracted a lot of attention from theorists and system developers due to their sound mathematical foundations and also for a natural way of representing uncertainty using probabilities (Jameson, 1996; Liu, 2008). The attractiveness of Bayesian models comes from their high representative power and the fact that they lend themselves to an intuitive graphical representation, as well as the fact that they offer a well defined formalism that lends itself to sound probability computations of unobserved nodes from evidence of observed nodes (Desmarais & Baker, 2012). Furthermore, the presence of capable and robust Bayesian libraries (eg. SMILE), which can be easily integrated into the existing or new student modeling applications, facilitates the adoption of BNs in student modeling (Millán, Loboda & Pérez-de-la-Cruz, 2010).

In Andes, which is a tutoring system for Newtonian physics, students' reasoning is performed by a constrained-based model (Shapiro, 2005). In this system, also, a probabilistic student model with Bayesian networks is used in order to manage uncertainty (Conati, Gertner & Vanlehn, 2002). Millán and Pérez-de-la-cruz (2002) improved the accuracy and efficiency of the diagnosis process through a student model, which applied Bayessian networks and Adaptive Testing Theory. Also, Bunt and Conati (2003) used Bayesian Networks to model the students of an intelligent exploratory learning environment for the domain of mathematic functions, which was named Adaptive Coach for Exploration (ACE). They built a student model capable of detecting when the learner is having difficulty exploring and of providing the types of assessments that the environment needs to guide and improve the learner's exploration of the available material. A Bayesian student model was applied in an assessment-based learning environment for English grammar, which is called English ABLE and was used by pedagogical agents to provide adaptive feedback and adaptive sequencing of tasks (Zapata-Rivera, 2007). AMPLIA, which is an intelligent learning environment employed as a resource in medical students' training, supports the development of probabilistic diagnostic reasoning and modeling of diagnostic hypotheses through a hybrid student model that combines Bayesian networks with cognitive theories (Viccari et al., 2008). Moreover, eTeacher (Schiaffino, Garcia & Amande, 2008) used Bayesian networks in order to detect a student's learning style automatically from the student's actions. Similarly, modeling of student's learning style was performed using Bayesian networks in DesignFirstITS, which is an ITS that helps novices to learn object oriented design by creating UML class diagrams (Parvez & Blank, 2008). Furthermore, Conati and Mclaren (2009) developed a probabilistic model of user affect, which recognizes a variety of user emotions by combining information on both the causes and effects of emotional reactions within a Dynamic Bayesian Network. A similar significant attempt to recognize and convey emotions in order to enhance students' learning and engagement was done by Muñoz et al. (2010) in PlayPhysics, an emotional game-based learning environment for teaching physics. Furthermore, Millán, Loboda and Pérez-de-la-Cruz (2010) used Bayesian networks as a practical tool for student modeling in order to provide personalized instruction to the domain of engineering. Similarly, in TELEOS a Bayesian network based student model was used in order to explicitly diagnose the student's knowledge state and cognitive behavior (Chieu et al., 2010). Hernández, Sucar and Arroyo-Figueroa (2010) developed an affective model (ABM) for intelligent tutoring systems, which was based on an affective student model that was represented as a dynamic Bayesian network. In addition, a Bayesian student model was used in Crystal Island, which is a game-based learning environment, in order to predict student affect and improve learning and motivation (Sabourin, Mott & Lester, 2011). Another Bayesian student model was applied to AdaptErrEx in order to model learners' skills and misconceptions (Goguadze et al., 2011a; 2011b). Finally, INQPRO system predicts the acquisition of scientific inquiry skills, which are composed of implicit skills as hypothesis formulation, variable identification, data comparison and drawing conclusion, by modeling students' characteristics with Bayesian networks (Ting & Phon-Amnuaisuk, 2012).

3.2.9 Ontology-based student modeling

Recently a lot of research has been done on the crossroad of user modeling and web ontologies, since both disciplines attempt to model real world phenomena qualitatively and due to the fact that the majority of user modeling projects have been deployed on the web and web ontologies are becoming defacto standard for web-based knowledge representation (Sosnovsky & Dicheva, 2010). The fact that an ontology supports the representation of abstract enough concepts and properties so as to be easily reused and, if necessary, extended in different application contexts, enables the reasoning on the information represented in the ontology (Clemente, Ramírez & de Antonio, 2011). Therefore, ontologies seem to can help to student reasoning. Winter, Brooks and Greer (2005) analyzed the ways ontologies can help student modeling. Among the advantages of ontology-based models they named “formal semantics, easy reuse, easy probability, availability of effective design tools, and automatic serialization into a format compatible with popular logical inference engines”. Moreover, according to Peña and Sossa (2010) meta-data and ontologies facilitate building a large-scale web of machine-readable and machine-understandable knowledge, and therefore they facilitate the reuse and the integration of resources and services, so that web-based educational systems and student models can provide better applications.

Several student models have been built based on ontologies. The Personal Reader uses semantic web technologies and ontologies in order to represent information about learners, which is needed to recommend appropriate learning resources relevant to user interests, learner performance in different courses within one domain or different domains, user goals and preferences (Dolog et al., 2004). Pramitasari et al. (2009) developed a student model ontology based on student performance as representation of prior knowledge and learning style, in order to create personalization for e-learning system. Also, OPAL is an ontology-based framework, which provides content presentation, and navigation assistance depending on the requirements of individual users and it adapts specifically to a learner’s knowledge and interests of the subject (Cheung, Wan &

Cheng, 2010). Peña and Sossa (2010) adopted a semantic representation and management of student models with ontologies in order to represent learners' knowledge, personality, learning preferences and content, and to deliver the appropriate option of lecture to students. MAEVIF makes sensible tutoring decisions and provide the most suitable feedback to the student in each moment through a student model, which is based on ontologies and diagnosis rules (Clemente, Ramirez & de Antonio, 2011). Finally, Nguyen et al. (2011) introduced an ontology-based student model used in a Social Network for Information Technology Students (SoNITS) to help the organization of knowledge and the reasoning on skill relationships.

3.3 Comparative discussion

The aim of this section is to discover the tendencies that there are on student modeling the past years. We are interested in finding the student modeling techniques that have been used by researchers as well as the student's characteristics that they have chosen to model. For this purpose, a list of adaptive and/or personalized tutoring systems that have been developed in the past ten years is presented. The main criterion for the selection of these adaptive and/or personalized systems was the validity of the search engine that was used to find them. Mostly, the listing systems are results of Scopus. Scopus is the world's largest abstract and citation database of peer-reviewed literature and it is considered as one of the most valid search engine for research papers. In particular, 91.30% of the adaptive and/or personalized systems that are listed below are results of the Scopus search engine. Furthermore, a respectable number of these systems have been evaluated. In particular, the percentage of the evaluated systems is 73.24%. The rest systems, which have not been evaluated, have been found, however, in Scopus. Therefore, they are valid. They are trends, which have not been established yet.

The results of our findings are presented in the following tables. To be more specific, table 1 presents the student modeling approaches that have been used in a variety of adaptive and/or personalized tutoring systems from 2002 up to

2007, table 2 presents the student modeling approaches that have been used in the development of adaptive and/or personalized tutoring systems from 2008 up to now, table 3 presents the student's characteristics that have been modeled by the student model of several adaptive and/or personalized tutoring systems from 2002 up to 2007, and table 4 presents the student's characteristics that have attracted the interest of many researchers the past five years concerning the student model. The data in these tables are presented with chronological order.

First, we want to make a comparative discussion about the student modeling techniques. As we notice in table 1 and table 2, the most common used student modeling techniques are the overlay and stereotype modeling. Indeed, the years 2002 up to 2007 the overlay student model was used more usually. Furthermore, the use of the perturbation student model was most common until 2007, while the use of machine learning techniques and the integration of cognitive theories seem to be stable during the past ten years. Moreover, the years 2002 up to 2007 researchers preferred to integrate fuzzy logic techniques into the student model in order to deal with the uncertainty of student's diagnosis. However, the past five years another probabilistic model has been added to researchers' preferences for dealing with the uncertainty. This probabilistic model is Bayesian student model, which is based on Bayesian networks. Also, an interest has started to arise, mainly the past two years, about the ontology-based student model.

Table 1: 2002-2007 (student modeling tendencies)

	Overlay	Stereotypes	Perturbation	Machine Learning	Cognitive Theories	Fuzzy	Bayesian Networks	Ontology-Based
Web-EasyMath		X		X				
Andes							X	
Millán & Pérez-de-la-cruz, 2002					X		X	
Xu, Wang & Su, 2002						X		
Conati & Zhou, 2002					X			
F-SMILE		X			X			
INSPIRE	X	*				X		
Web-PTV		X		X				
Surjono & Maltby, 2003	X	X	X					
ACE							X	
F-CBR-DHTC		*				X		
TADV	X					X		
LeCo-EAD			X					
WILEDS	X							
MEDEA	X							
InterMediActor	X					X		
The Personal Reader	X							X
Baker, Corbett & Koedinger, 2004				X				
ICICLE	X	X						
Web-IT		X			X			
VIRGE					X			
Stathakopoulou, Magoulas, Grigoriadou & Samaracou, 2005						X		
InfoMap	X		X					
Kumar, 2006	X							
Salim & Haron, 2006		*				X		
English ABLE							X	
Baker, 2007				X				

*: stereotype-like

Table 2: 2008-2012 (student modeling tendencies)

	Overlay	Stereotypes	Perturbation	Machine Learning	Cognitive Theories	Fuzzy	Bayesian Networks	Ontology-Based
DeLC	X	X						
AUTO-COLLEAGUE		X						
Petersen, Petersen, Bye & Kolas, 2008		X						
Web_Tutor_Pas		X						
Carmona, Castillo & Millan, 2008		*					X	
Alepis, Virvou & Kabassi, 2008					X			
DesignFirstITS							X	
ADAPTAPlan				X		X		
e-Teacher							X	
GIAS		X		X				
LS-Plan	X							
Pramitasari et al., 2009								X
Wang, Yang & Wen, 2009				X				
Conati & Maclaren, 2009							X	
WELSA		*						
CLT		X						
ABM					X		X	
Baschera & Gross, 2010			X					
Cetintas, Si, Xin & Hord, 2010				X				
POOLE III				X			X	
Baker, Goldstein & Heffernan, 2010				X			X	
Al_Hmouz et al., 2010		*		X				
Jia et al., 2010						X		
Millán, Loboda & Pérez-de-la-Gruz, 2010							X	
PlayPhysics					X		X	
Peña & Sossa, 2010								X
OPAL	X							X
Wayang Outpost		*						
SimStudent				X				
Balakrishnan, 2011				X				X
Crystal Island							X	
AdaptErrEx							X	
IWT	X							X
MAEVIF								X
SoNITS								X
DEPTHS	X	X				X		
MMT					X			
Tourtoglou, Virvou, 2012		X	X					
CoStaT	X	X	*					
FuzKSD	X	X				X		
AHS-TS	X	*						
PDinamet	X							
Goel, Lallé & Luengo, 2012						X		
INQPRO							X	

*: approach-like (stereotype-like or perturbation-like)

Table 3: 2002-2007 (tendencies on students modeling characteristics)

	knowledge	errors/ misconceptions	learning styles & preferences	other cognitive aspects	affective features	moti-vation	meta-cognitive features
Web-EasyMath	X						
Andes	X	X		X			X
Millán & Pérez-de-la-cruz, 2002	X	X					
Xu, Wang & Su, 2002	X			X			
Conati & Zhou, 2002					X		
F-SMILE	X	X					
INSPIRE	X		X				
Web-PTV	X			X			
Surjono & Maltby, 2003	X	X	X				
ACE	X		X				
F-CBR-DHTC	X			X			
TADV	X		X				
LeCo-EAD	X	X					
WILEDS	X						
MEDEA	X						
InterMediActor	X						
The Personal Reader	X		X				
Baker, Corbett & Koedinger, 2004	X	X				X	
ICICLE	X						
Web-IT	X			X			
VIRGE	X				X		
Stathakopoulou, Magoulas, Grigoriadou & Samaracou, 2005	X		X	X		X	
InfoMap	X	X					
Kumar, 2006	X						
Salim & Haron, 2006			X				
English ABLE	X						
Baker, 2007					X	X	

Table 4: 2008-2012 (tendencies on student modeling characteristics)

	knowledge	errors/ misconceptions	learning styles & preferences	other cognitive aspects	affective features	moti-vation	meta-cognitive features
DeLC	X		X	X			
AUTO-COLLEAGUE	X			X	X		
Petersen, Petersen, Bye & Kolas, 2008				X	X		
Web_Tutor_Pas	X	X					
Carmona, Castillo & Millan, 2008			X				
Alepis, Virvou & Kabassi, 2008					X		
DesignFirstITS	X		X				
ADAPTAPlan	X		X	X			
e-Teacher	X		X				
GIAS	X		X	X			
LS-Plan	X		X				
Pramitasari et al., 2009	X		X				
Wang, Yang & Wen, 2009	X		X	X			
Conati & Maclaren, 2009					X		
WELSA			X				
CLT	X			X			
ABM	X		X	X	X		
Baschera & Gross, 2010	X	X					
Cetintas, Si, Xin & Hord, 2010	X					X	
POOLE III	X				X		
Baker, Goldstein & Heffernan, 2010	X						
Al_Hmouz et al., 2010	X		X	X			
Jia et al., 2010	X		X	X			
Millán, Loboda & Pérez-de-la-Gruz, 2010	X	X		X	X	X	X
PlayPhysics	X				X	X	
Peña & Sossa, 2010	X		X	X			
OPAL	X		X			X	
Wayang Outpost	X				X		X
SimStudent	X						
Balakrishnan, 2011	X	X			X		
Crystal Island					X		
AdaptErrEx	X	X					
IWT	X		X				X
MAEVIF	X		X				
SoNITS	X						
DEPTHS	X		X	X			
MMT					X	X	
Tourtoglou, Virvou, 2012	X	X		X			
CoStaT	X	X		X			
FuzKSD	X						
AHS-TS	X		X	X			
PDinamet	X						
Goel, Lallé & Luengo, 2012	X						
INQPRO	X						

Furthermore, many researchers have used a hybrid student model, which brings together various features of different techniques of student modeling, in order to combine various aspects of student’s characteristics. So, there are hybrid student models that combine overlay with stereotype modeling techniques, or stereotypes with machine learning techniques, or an overlay student model with fuzzy logic techniques, or Bayesian networks with machine learning algorithms. The above combinations of student modeling techniques are just some examples. In table 5 we present the possible combinations that have been applied to a variety of adaptive and/or personalized tutoring systems from 2002 up to now. To be more specific, in this table we present for each combination of student modeling approaches the percentage of the adaptive and/or personalized tutoring systems that have used them. To be informed about the tendency that there is in student modeling techniques’ combinations, we should read each row of the table 5. For example, by reading the first row, we are informed about the fact that the most common used combination of an overlay student model is with stereotypes (38.89%), while no one adaptive and/or personalized tutoring system has used a compound student model which brings together an overlay model with machine learning algorithms or Bayesian networks. Moreover, a frequent combination of an overlay student model is with fuzzy logic techniques. Also, some researchers have attempted to combine overlay with perturbation model or ontologies. Therefore, an overlay student model usually is combined with stereotypes or fuzzy logic techniques; stereotypes are blended, mainly, with overlay but they are also combined with machine learning or fuzzy logic techniques; perturbation student model is combined only with overlay and stereotypes; machine learning techniques are used mostly to support stereotype modeling but there is also an interest to combine them with Bayesian networks; cognitive theories are applied with stereotypes and Bayesian Networks; fuzzy logic usually is used with overlay or stereotype student models; Bayesian networks are blended, mainly, with machine learning techniques and cognitive theories, but also may researchers used them in combination with stereotypes; and finally, ontologies are primarily combined with overlay student modeling. Attempts to construct a hybrid student model that combines more than two student modeling techniques, have also be made, but they are few.

The need for blended student models arises from the need to model a variety of student's characteristics. It has been proven that some student model approaches are ideal for representing some particular aspects of the student's characteristics. For example, the overlay student model is useful for the presentation of the student's mastery on the knowledge domain; stereotypes are ideal to represent student's learning styles; the perturbation student model specializes in detecting the student's misconceptions; cognitive theories, such as OCC theory, seem to have been established as a standard method for emotion recognition; and probabilistic student modeling approaches are ideal for representing more abstract and subjective aspects of the student's characteristics such as affective, cognitive and meta-cognitive features. In order to detect the tendency of student modeling techniques in relation to student modeling characteristics we have constructed tables 3 and 4 that are presented above. As we notice, the years 2002 up to 2007, the researchers were focused, primarily, on modeling the knowledge level, the errors and misconceptions, and the learning styles of students. That is, maybe, the reason for the increasing use of overlay, perturbation and stereotype student modeling approaches these years. The following years the interest of modeling the student's learning styles and preferences, and other cognitive aspects, continued to grow. Also, the interest on modeling motivation, affective aspects and meta-cognitive features of students has arisen. That is why researchers have turned their interest to new student modeling approaches, such as Bayesian networks and ontology-based techniques.

In order to make a comparative discussion about the student modeling techniques in relation to student modeling characteristics, we have constructed table 6. This table presents the percentage of student modeling approaches that have been used the past decade, in order to model particular student's characteristics. As we notice, the most preferred technique for representing the student's mastery on the knowledge domain is the overlay student model. Furthermore, it seems that a perturbation student model represents better the student's errors and misconceptions. Stereotyping seems to be ideal for modeling student's learning styles, preferences and other cognitive factors such as memory, attention and perception. Due to the uncertainty that characterizes students' cognitive aspects; many researchers chose to integrate fuzzy logic techniques into the student model. In addition, Bayesian networks are preferred for modeling affective aspects of student's characteristics, such as emotions and feelings, and meta-cognitive aspects, such as self-regulation, self-explanation and self-assessment, while motivation seems to be modeled better by machine learning techniques. Another approach that seems to be ideal for performing affective student modeling is the utility of cognitive theories. The ontology-based student model does not seem to prevail in the modeling of a particular student's characteristic, This happens, maybe, because it is a new student modeling approach and there is no an adequate number of researches and applications on this domain.

Table 5: Tendencies on Blended Student Models

	Overlay	Stereotypes	Perturbation	Machine Learning	Cognitive Theories	Fuzzy	Bayesian Networks	Ontology-Based
Overlay		38.89%	16.67%	0%	0%	27.78%	0%	16.67%
Stereotypes	34.78%		13.04%	17.39%	8.70%	21.74%	4.35%	0%
Perturbation	50%	50%		0%	0%	0%	0%	0%
Machine Learning	0%	50%	0%		0%	12.5%	25%	12.5%
Cognitive Theories	0%	40%	0%	0%		0%	60%	0%
Fuzzy	45.46%	45.46%	0%	9.09%	0%		0%	0%
Bayesian Networks	0%	16.67%	0%	33.33%	50%	0%		0%
Ontology-Based	75%	0%	0%	25%	0%	0%	0%	

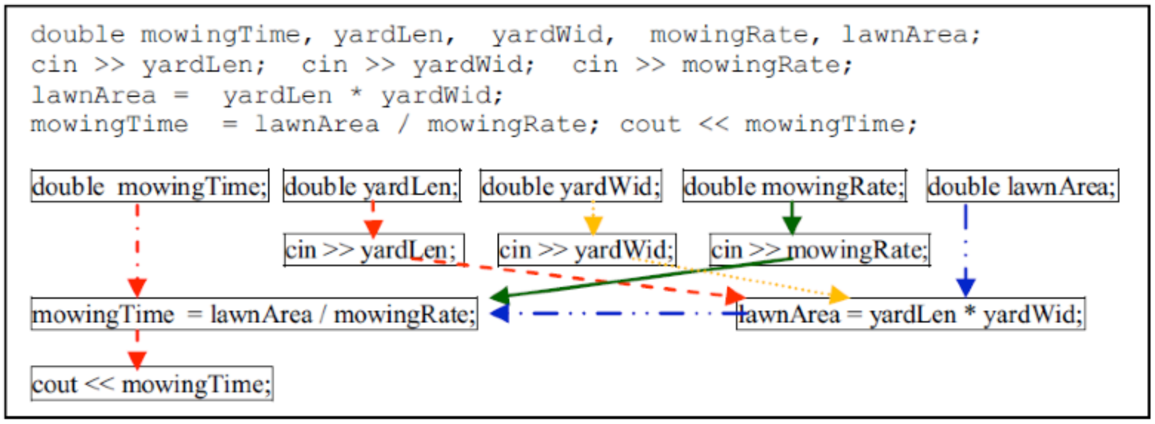
Table 6: Student modeling approaches in relation to student modeling characteristics

	knowledge	errors/ misconceptions	learning styles & preferences	other cognitive aspects	affective features	moti-vation	meta-cognitive features
Overlay	24.36%	0%	18.92%	12.05%	0%	10%	20%
Stereotypes	16.67%	13.33%	27.03%	37.5%	16.67%	0%	0%
Perturbation	2.56%	40%	0%	0%	0%	0%	0%
Machine Learning	15.39%	13.33%	10.81%	15.63%	16.67%	30%	0%
Cognitive Theories	2.56%	6.67%	0%	3.13%	33.33%	20%	0%
Fuzzy	14.10%	0%	18.92%	18.75%	0%	10%	0%
Bayesian Networks	14.10%	20%	8.11%	9.38%	27.78%	20%	60%
Ontology-Based	10.26%	6.67%	16.22%	3.13%	5.36%	10%	20%

4. Programming tutoring systems

Programming tutoring systems are tutoring systems that teach learners algorithms or programming languages, usually without intervention from a human teacher. They aim to provide personalized instruction and feedback to learners. They use a variety of computing technologies and approaches for knowledge representation and student modeling to enable learning in a meaningful and effective manner. Below, examples of programming tutoring systems are presented. The techniques of knowledge representation and student modeling that they use are described.

Jin et al. (2012) have been proposed a new technique for program representation for automatic hint generation for a programming tutor. This technique is based on linkage graphs. A linkage graph for a program is a directed acyclic graph, as shown in figure 17, where nodes represent program statements and directed edges indicate order dependencies. If statements *I* and *J* access the same variable *x*, and *J* is the first statement after *I* that accesses variable *x*, then *J* directly depends on *I* and an edge from node *I* to node *J* with label *x* is added.



```
double mowingTime, yardLen,  yardWid,  mowingRate, lawnArea;
cin >> yardLen;  cin >> yardWid;  cin >> mowingRate;
lawnArea =  yardLen * yardWid;
mowingTime  = lawnArea / mowingRate; cout << mowingTime;
```

Figure 17: The linkage graph for a program to calculate money earned for mowing grass. The colored directed edges identify variable dependency between nodes. (Jin et al., 2012)

Furthermore, they have used a 2-dimensional matrix to represent a linkage graph. Table 7 (left) shows the matrix for the program in figure 17. Variable v_0 shows up in statements 0, 9 and 10, represented as 1's in the corresponding rows, indicating that variable v_0's linkage starts with statement 0 and consists of edges (0,9) and (9,10). Table 7 shows matrices for two programs that differ only in order. Since there are no variable dependencies among statements 0-4 and among 5-7, they are equivalent.

The proposed linkage graph representation is used to record and reuse student work as a domain model. Also, an overlay comparison to compare in-progress work with complete solutions in a twist on the classic approach to hint generation has been used. The above approach uses educational data mining and machine learning techniques to automate the creation of a domain model and hints from student problem-solving data.

Table 7: Linkage Graph Matrixes. The left is for program in Fig. 17; the right is equivalent. (v_0=mowingTime, v_1=yardLen, v_2=yardWidth, v_3=mowingRate, and v_4=lawnArea) (Jin et al., 2012)

	v_0	v_1	v_2	v_3	v_4
0. double v_0;	1				
1. double v_1;		1			
2. double v_2;			1		
3. double v_3;				1	
4. double v_4;					1
5. cin >> v_1;		1			
6. cin >> v_2;			1		
7. cin >> v_3;				1	
8. $v_4 = v_1 * v_2$;		1	1		1
9. $v_0 = v_4 / v_3$;	1			1	1
10. cout << v_0;	1				

	v_0	v_1	v_2	v_3	v_4
0. double v_4;					1
1. double v_3;				1	
2. double v_2;			1		
3. double v_1;		1			
4. double v_0;	1				
5. cin >> v_3;				1	
6. cin >> v_2;			1		
7. cin >> v_1;		1			
8. $v_4 = v_1 * v_2$;		1	1		1
9. $v_0 = v_4 / v_3$;	1			1	1
10. cout << v_0;	1				

Marie des Jardins et al. (2011) have developed PtP ITS that extends The RUR–Python Learning Environment (RUR-PLE), a game-like virtual environment to help students learn to program, provides an interface for students to write their own Python code and visualize the code execution (Roberge 2005). PtP ITS consists of three components: (1) a Bayesian student model that tracks student competence, (2) a diagnosis module that provides tailored feedback to students, and (3) a problem selection module that guides the student's learning process.

Protus (PRogramming TUtoring System) (Vesin et al., 2011) is another adaptive and intelligent web-based tutoring system for programming. One of the most important features of Protus is the adaptation of the presentation and navigation of a course material based on particular learner knowledge. This system aims at automatically guiding the learner’s activities and recommend relevant actions during the learning process. An overlay model in which the current state of a learner’s knowledge level is described as a subset of the overall architecture represents the learner knowledge base in Protus. The learner model includes learner’s personal information, background, goals, and learning style as well as his/her competence levels for each concept node and each unit in the content tree, and an overall subject competence level. The learner modeling is derived from the knowledge contained in the ontology. Various conditions are captured in the body of SWRL (Semantic Web Rule Language) rules. As a result of the firing of rules, updates to learner model are generated and data about learner’s navigation and progress through course is collected. This data can be further used to implement the concept of adapted content and adapted navigation. The architecture of Protus is depicted in figure 18.

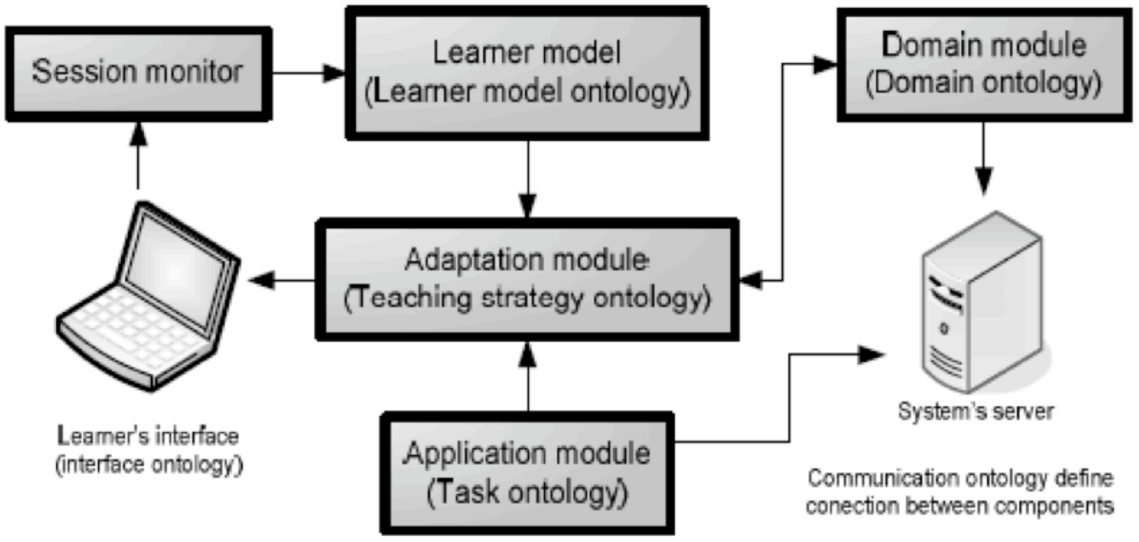

Session monitor
Learner model (Learner model ontology)
Domain module (Domain ontology)
Adaptation module (Teaching strategy ontology)
Learner's interface (interface ontology)
System's server
Application module (Task ontology)
Communication ontology define conection between components

Figure 18: Global architecture of Protus (Vesin et al., 2011)

Another intelligent tutoring system from programming is J-LATTE (Holland, Mitrovic & Martin, 2009). It teaches a subset of the Java programming language. J-LATTE supports two modes: concept mode, in which the student designs the program without having to specify contents of statements, and coding mode, in which the student completes the code. The system represents the knowledge domain in terms of constraints, which describe features of correct solutions (Ohlsson, 1994). A constraint consists of a relevance condition that identifies problem states for which the constraint is important, and a satisfaction condition that specifies the features the solution must have to be correct. Constraints in J-LATTE are categorized as syntax, semantic and style constraints. Syntax constraints check that the student's solution consists of valid Java code, such as "Each assignment must contain a valid expression on the right-hand side." Semantic constraints check whether the student's solution is the correct answer for the given problem. For example, an example semantic constraint is "If the problem requires a function to be applied to a range of values, the solution must contain a loop." Each constraint evaluates one small fragment of the solution; in the case of the previous example, there are other constraints that check that the loop is in the correct place, the counter is initialized and incremented correctly, and the loop condition is correct. Style constraints encourage the student to follow good practice, such as "The return statement should be at the end of a method." Semantic constraints compare the student's solution to the formal specification of the problem, which specifies a list of requirements.

BITS (Butz, Hua & Maguire, 2006) is a web-based intelligent tutoring system for teaching programming (the programming language C++). The decision making process conducted in BITS is guided by a Bayesian network approach to support students in learning computer programming. BITS can help a student navigate through the online course materials, recommend learning goals, and generate appropriate reading sequences. Figure 19 shows the overall architecture of BITS. Four modules contained in BITS are Bayesian networks, the knowledge base, the user interface, and the adaptive guidance module. In BITS the user interface module has been divided into two interface submodules, the input submodule and output submodule. Also, the adaptive guidance module (this module corresponds with the teaching-strategies component in the general framework of an ITS) has been divided into three submodules: Navigation Support, Prerequisite Recommendations, and Generating Learning Sequences.

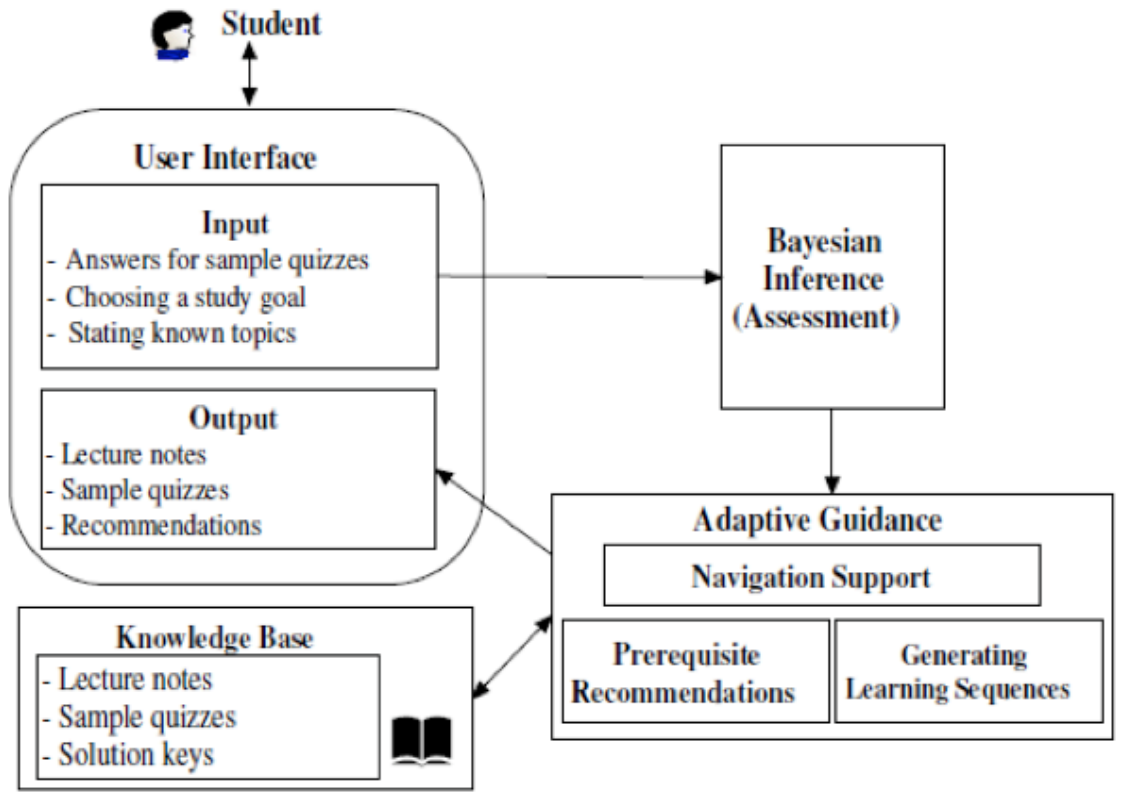


Figure 19: Architecture of BITS (Butz, Hua & Maguire, 2006)

The knowledge domain of BITS is represented by directed acyclic graph. A node in the graph represents each concept. There exist learning dependencies among knowledge concepts, namely, prerequisite relationships. A directed edge from one concept (node) to another is added if knowledge of the former is a prerequisite for understanding the latter. For example, consider the following instance of the “For Loop” construct in C++: for(i=1; i<=10; i++). To understand the “For Loop” construct, one must first understand the concepts of “Variable Assignment” (i=1), “Relational Operators” (i<=10), and “Increment/Decrement Operators” (i++). These prerequisite relationships can be modelled as the directed acyclic graph depicted in Figure 20. The particular figure depicts a small portion of the entire directed acyclic graph implemented in BITS. The entire directed acyclic graph implemented in BITS consists of 29 nodes and 43 edges, which is shown in figure 21.

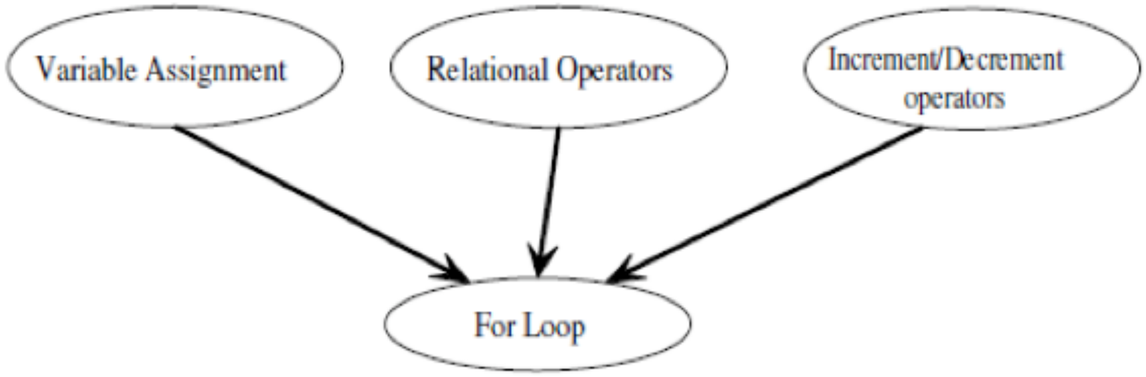


Figure 20: Modeling the prerequisite concepts of the "For Loop" construct (Butz, Hua & Maguire, 2006)

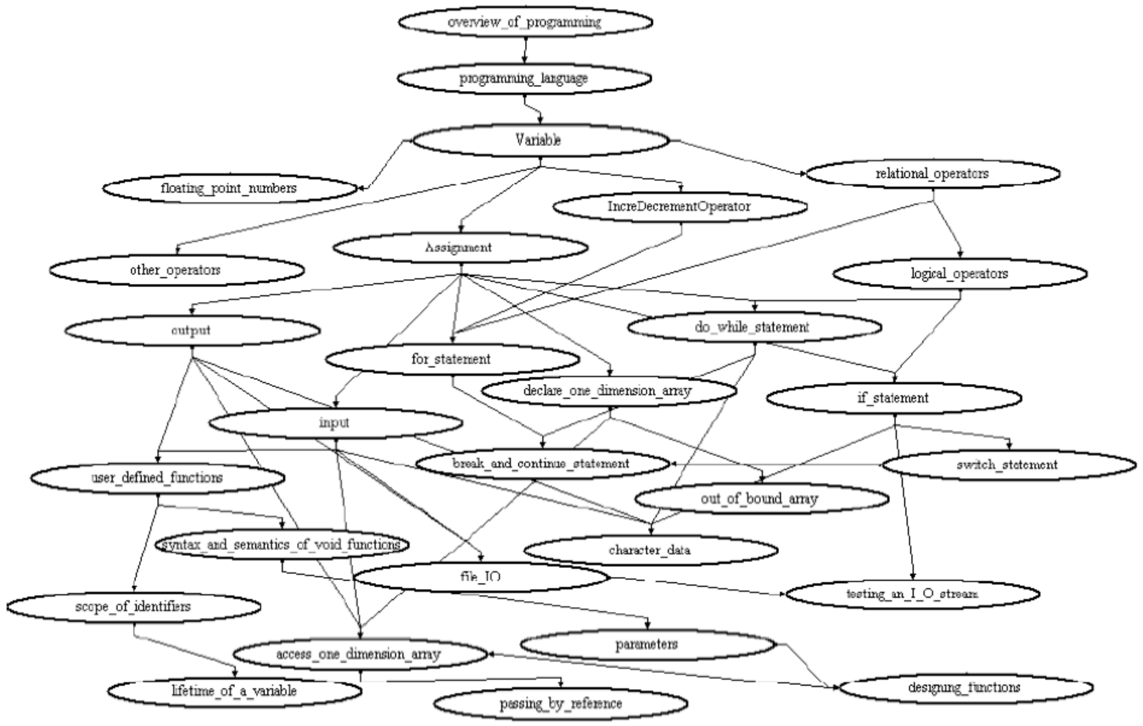


Figure 21: The entire DAG implemented in BITS (Butz, Hua & Maguire, 2006)

INCOM is another tutoring system in the field of programming, which has used the CBM approach to diagnose the student's errors (Le & Menzel, 2009). The system INCOM is intended to help students of the course Logic Programming at the University of Hamburg. It is meant to coach students individually solving their homework assignments to better prepare them for classroom activities. The system informs the student about possible errors occurring in her solution attempts and remedial hints are given to improve the solutions. The system's evaluation indicated that the students using INCOM have improved their programming skills significantly after using the system (Le, Menzel & Pinkwart, 2009).

INCOM uses semantic tables and weighted constraints. The concept of the semantic table comprises two ideas: 1) it describes several solution strategies, and 2) it represents model solutions in a generalized form, thus covering different implementation variants. For the sample problem Salary: "*A salary database is implemented as a list whose odd elements represent names and even elements represent salary in Euro. For example: [meier, 3600, schulze, 5400, mueller, 6300, ..., bauer, 4200]. Please, define a predicate which creates a new salary list according to following rules: 1) Salary less equal 5000 Euro will be raised 3%; 2) Salary over 5000 Euro will be raised 2%. The new salary list would be: [meier, 3708, schulze, 5508, mueller, 6426, ..., bauer, 4226]. Notice: the representation of 3% and 2% corresponds to 0.03 and 0.02 in Prolog, respectively*", the semantic table contains two kinds of information: besides a signature for the predicate to be implemented (Table 8), it mainly consists of a set of generalised sample solutions that describe the semantic requirements of a predicate definition in relational form. In addition, all unification conditions are expressed explicitly and clause heads as well as subgoals are represented in normal form (Table 9). The normal form representation reveals the underlying programming techniques.

Constraint weights represent the principles of a domain (of an area of study or certain problem tasks) (Ohlsson, 1994) and are used to check the student's implementation against the semantic requirements of correct solutions (Mitrovic et al., 2001). In INCOM, they are used to facilitate a comparison of competing error explanations. This allows the system to hypothesize the solution strategy possibly pursued by the student. Furthermore, they can be used to determine the order in which feedback is presented to the student. Particularly, constraint weights are taken from the interval [0; 1] which can be conceived of as a measure of importance with 0 indicating the most important requirements (Table 10). Given the fact that a clause contributes more information to the overall correctness of the solution than an argument or a functor, a constraint, which examines an argument, should be specified as being less important compared to a constraint checking a sub goal. Weighted constraints are used for the following purposes: i) *declaration constraints* (they define requirements for the signature specification, that is the number of argument positions of the predicate, as well as the type and instantiation mode of each argument position), ii) *general constraints* (they express general semantic principles of the programming language), iii) *semantic constraints* (they examine whether a solution fulfills the requirements of the task description), and iv) *pattern constraints* (they are used to model standard solution strategies). Patterns can be organised in a hierarchy (Figure 22 where a sub-pattern can inherit the characteristics of a super-pattern and contain new techniques.

Table 8: A predicate's signature for the task Salary specified in the semantic table (Le & Mendel, 2009)

Name	Argument	Type	Mode	Synonyms	Description
p	A1	list	input	old list, salary database	This argument stands for the old salary list
p	A2	list	output	new list, new salary database	This argument stands for the new salary list

Table 9: Semantic requirements for implementation of the task Salary (Le & Menzel)

Clause	Head	Subgoal	Description
1	p(OL,NL)	OL=[]	Basecase list empty
		NL=[]	New list also empty
2	p(OL,NL)	OL=[N,S|T]	N:Name; S:Salary
		NL=[N,Sn|Tn]	Build a new list
		S=<5000	Salary =< 5000
		Sn is S+S*0.03	Salary is increased
		p(T, Tn)	Recurse old list
3	p(OL,NL)	OL=[N,S|T]	N:Name; S:Salary
		NL=[N,Sn|Tn]	Build a new list
		S>5000	Salary>5000
		Sn is S+S*0.02	Salary is increased
		p(T,Tn)	Recurse old list

Table 10: Proposed constraint weights (Le & Menzel, 2009)

Constraint Weight	Checking Issues
0.01	Clause existence
0.1	Subgoal existence
0.3	Correctness of comparison operators
0.5	Argument existence, position, co-reference
0.7	Subgoal order, factors of a multiplication term
0.8	Correctness of nominator, denominator of a fraction term, anonymous variables

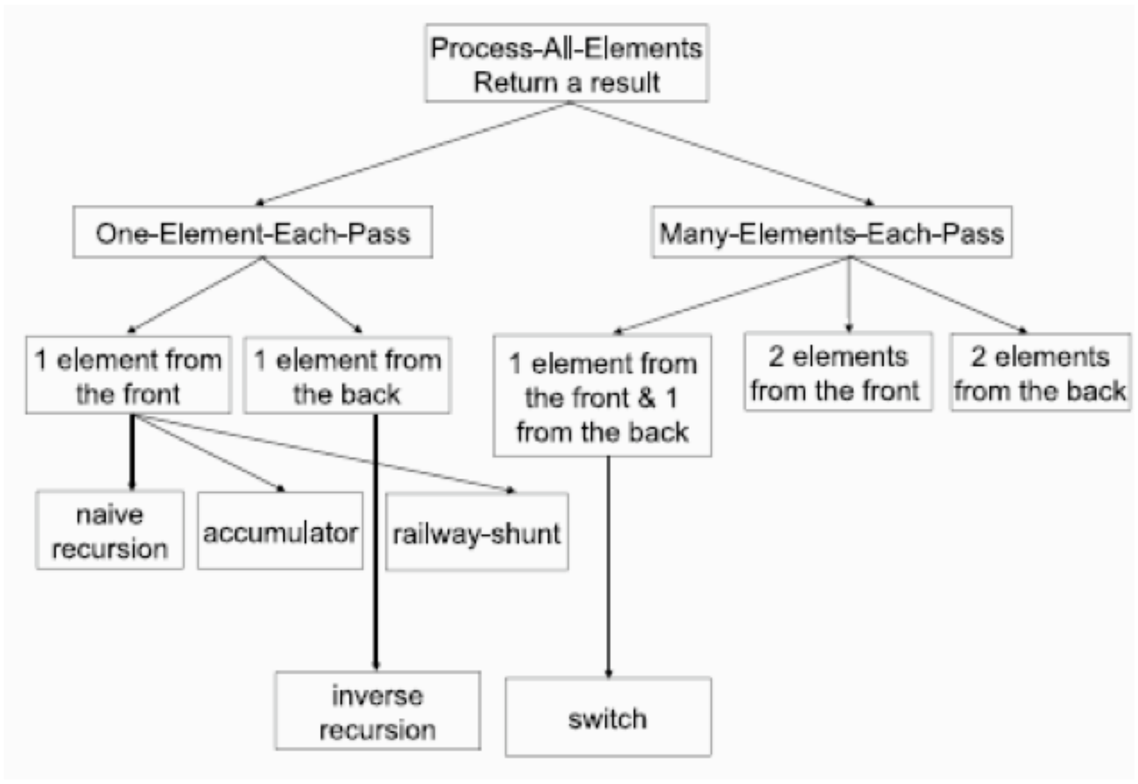


Figure 22: A small hierarchy of Prolog patterns (Le & Menzel, 2009)

SQLT-Web is a web-enabled intelligent tutoring system that teaches the SQL database language (Mitrovic, 2003). The system observes students' actions and adapts to their knowledge and learning abilities. The system has been used by senior computer science students at the University of Canterbury and has been found easy to use, effective and enjoyable (Mitrovic & Ohlsson, 1999). The architecture of SQLT-Web is depicted in figure 23. The interface of SQLT-Web has been designed to be robust, flexible, and easy to use and understand. The knowledge about the domain that SQLT-Web contains is represented as a set of constraints. SQLT-Web currently contains more than 600 constraints, and this number is likely to increase as new problems requiring new situations are added to the system. All constraints are problem-independent; they describe the basic principles of the domain, and do not involve any elements of problems directly. The constraints are modular, and are not related to each other. SQLT-Web analyses the student's solution once when it is submitted, by matching it to the constraints and the ideal solution. SQLT-Web maintains information about a student in his/her student model, which summarizes student's knowledge and the history of the current and previous sessions. Initially, SQLT-Web acquires information about a student through a login screen. Individual student models are stored permanently on the server, and retrieved for each student's session. Students who are inactive for a long period of time are automatically logged off (after 120 minutes) and their models are moved back to long-term storage.

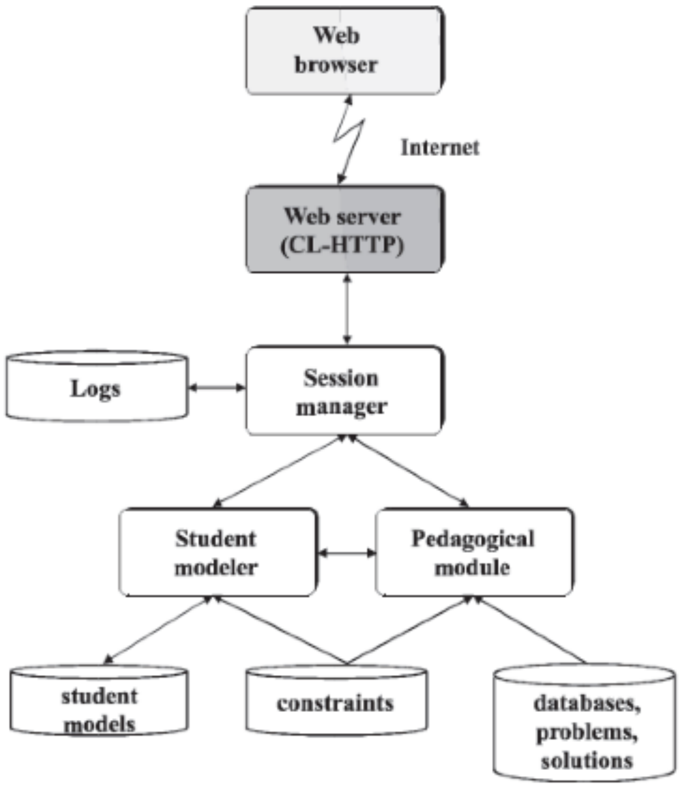


Figure 23: The architecture of SQLT-Web

Kumar (2006a) has used the concept map of the domain, enhanced with pedagogical concepts called learning objectives, as the overlay student model in intelligent tutors for programming. The resulting student model is fine grained, and has several advantages: (1) it facilitates better adaptation during problem generation; (2) it makes it possible for the tutor to automatically vary the level of locality during problem generation to meet the needs of the learner; (3) it clarifies to the learner the relationship among domain concepts when opened to scrutiny; (4) the tutor can estimate the level of understanding of a student in any higher-level concept, not just the concepts for which it presents problems; and (5) two tutors in a domain can affect each other's adaptation of problems.

A prototype of tutoring system, called C++ Loop Tutor (CLT) (Durrani & Durrani, 2010), was developed to deal with cognitive abilities, knowledge and interaction history of student. The overall architecture of CLT is depicted in figure 24. It includes the following components: Instructor model (it is responsible to instruct the student), Student model (it contains complete information about the student), Expert model (it has a complete repository of lectures, quizzes, problems, solutions, and examples), Evaluation module (It evaluates quizzes solved by the students and provides scores to the student model), Feedback module (it gives feedback to student based on interaction with the tutor), User Interface module. The system builds individual student model by assessing cognitive abilities and knowledge domain of the student using some diagnostic tests. Based on these assessments the system assigns the student to one of the stereotypes from the set of stereotypes. There are 27 total stereotypes. Following are some of the stereotypes that were used in the prototype.

Stereotype 1: verbal ability – low, numerical ability –high, spatial ability – medium

Stereotype 2: verbal ability – medium, numerical ability – high, spatial ability – low

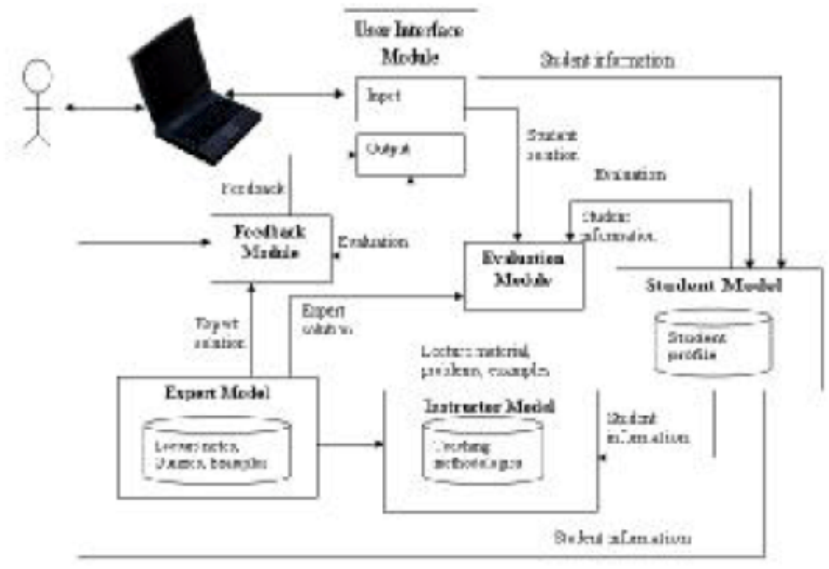


Figure 24: Proposed architecture of CLT (Durrani & Durrani, 2010)

Consequently, many efforts have been done for adaptive learning of programming. A variety of student modeling techniques has been used, like Bayesian networks, machine-learning techniques, overlay model, stereotypes, constraint-based model. Furthermore, the knowledge domain representation techniques that have been preferred mainly are graphs (like trees, network of concepts, hierarchies, concept maps) and constraints. Below, a summary table (table 11), which includes the programming tutoring systems that were discussed and their characteristics, is presented.

Table 11: Programming Tutoring Systems

System	Knowledge domain	Student modeling	Aim of the system
Jin et al. (2012)	Linkage graph	Machine learning	Hint generation
PtP ITS	Concept map	Bayesian Network	Personalized feedback and problem selection
Protus	Content tree & ontologies	Overlay model & rule-based reasoning & ontologies	Adaptation of the presentation and navigation
J-LATTE	Terms of constraints	Constraint-based model	Personalized feedback
BITS	Directed graphs	Bayesian Network	Adaptive guidance
INCOM	Semantic tables & weighted constrains & hierarchies	Constraint-based model	Diagnose students' errors
SQLT-Web	Set of constraints	Constraint-based model	Adaptive learning
Kumar (2006)	Concept maps	Overlay model	Adaptive learning
CLT	--------------	Stereotypes	Adaptivity according to students' cognitive abilities

5. Fuzzy Logic Theory

5.1 An overview

Fuzzy logic was introduced by Zadeh (1965) to encounter imprecision and uncertainty. It deals with reasoning that is approximate rather than fixed and exact. Fuzzy logic variables may have a truth value that ranges in degree between 0 and 1. Fuzzy logic is able to handle uncertainty in everyday problems caused by imprecise and incomplete data as well as human subjectivity (Drigas, Argyri & Vrettaros, 2009). According to them, a fuzzy set is defined as an ordered set (x, $u_A(x)$), where $x \in X$ and $u_A(x) \in [0,1]$, equipped with a membership function $\mu_A(x)$: X→ [0,1], where

$$\mu_A(x) = \begin{cases} 1, & x \quad \textit{absolutely} \quad \textit{in} \quad A \\ 0, & x \quad \textit{absolutely} \quad \textit{not} \quad \textit{in} \quad A \\ (0,1), & x \quad \textit{partially} \quad \textit{in} \quad A \end{cases}$$

Value $u_A(x)$ is called degree of membership or membership value. For example x can be an age (eg. 23) and A can be a characterization, like young or old.

Basically, fuzzy logic is a precise logic of imprecision and approximate reasoning (Zadeh, 1975; 1979). More specifically, as Zadeh (2008) has stated, fuzzy logic may be viewed as an attempt at formalization/mechanization of two remarkable human capabilities: (i) the capability to converse, reason and make rational decisions in an environment of imprecision, uncertainty, incompleteness of information, conflicting information, partiality of truth and partiality of possibility, and (ii) the capability to perform a wide variety of physical and mental tasks without any measurements and any computations (Zadeh, 2001). One of the principal contributions of fuzzy logic is its high power of precisiation of what is imprecise (Zadeh, 2008). This capability of fuzzy logic suggests that it may find important applications in the realms of economics, linguistics, law and other human-centric fields.

According to Zadeh (2008), fuzzy logic FL has many facets (Fig. 25):

➢**The logical facet (FLl):** it is fuzzy logic in its narrow sense. FLl may be viewed as a generalization of multivalued logic. The agenda of FLl is similar in spirit to the agenda of classical logic (Godo et al., 1991; Esteva & Godo, 2006; Hajek, 1998; Novak, Perfilieva & Mockor, 1999; Novak, 2006; Reghis & Roventa, 1998).

➢**The fuzzy-set-theoretic facet (FLs)**: it is focused on fuzzy sets, that is, on classes whose boundaries are unsharp. The theory of fuzzy sets is central to fuzzy logic. Historically, the theory of fuzzy sets preceded fuzzy logic in its wide sense.

➢**The epistemic facet (Fle)**: it is concerned with knowledge representation, semantics of natural languages and information analysis. In FLe, a natural language is viewed as a system for describing perceptions.

➢**The relational facet (FLr)**: it is focused on fuzzy relations and, more generally, on fuzzy dependencies.

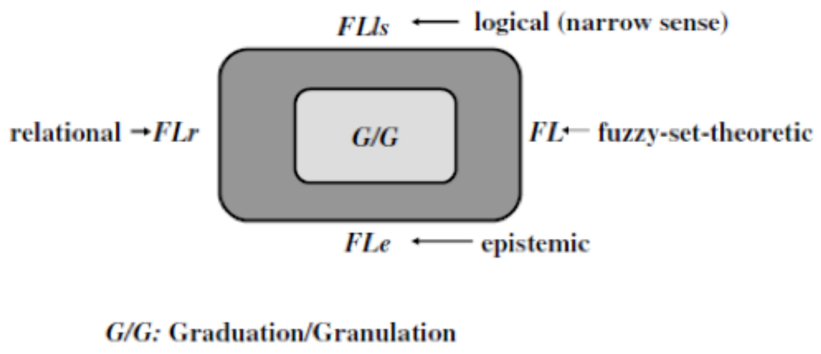


Figure 25: Principal facets of fuzzy logic (FL). The core of FL is graduation/granulation, G/G (Zadeh, 2008)

The basic concepts of graduation and granulation form the core of Fuzzy Logic and are the principal distinguishing features of fuzzy logic (Zadeh, 2008). More specifically, in fuzzy logic everything is or is allowed to be graduated, and everything is or is allowed to be granulated. Granulation may be viewed as a form of information compression of variables and input/output relations. An instance of granulation is the concept of a linguistic variable (Zadeh, 1973) that is used in almost all applications of fuzzy logic. A simple example of a linguistic variable is shown in figure 26 (Zadeh, 2008).

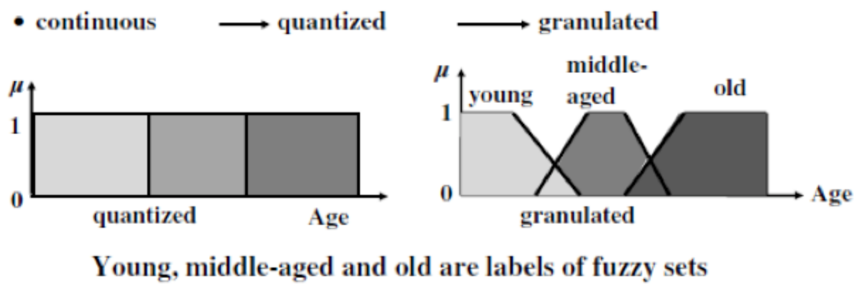


Figure 26: Granulation of Age; young, middle-aged and old are linguistic (granular) values og age (Zadeh, 2008)

5.2 Contributions of fuzzy logic

In this section, the contributions of fuzzy logic are described, according to Zadeh (2008). The most visible, the best understood and the most widely used contribution of fuzzy logic is the concept of a linguistic variable and the associated machinery of fuzzy if–then rules. A key idea, which underlies the machinery of linguistic variables and fuzzy if–then rules is centered on the use of information compression. In fuzzy logic, information compression is achieved through the use of fuzzy granulation. Furthermore, one of the important features of fuzzy logic is its high power of cointensive precisiation. What this implies is that better models of reality can be achieved through the use of fuzzy logic. In addition, fuzzy logic is ideal for computing with words, since natural languages are intrinsically imprecise. Fuzzy logic has, also, contributed to the computational theory of perceptions. In particular, humans have a remarkable capability to perform a wide variety of physical and mental tasks without any measurements and any computations. In performing such tasks humans employ perceptions. To endow machines with this capability what is needed is a formalism in which perceptions can play the role of objects of computation. The fuzzy-logic-based computational theory of perceptions serves this purpose. Moreover, fuzzy logic addresses the problem of computation with imprecise probabilities. Finally, fuzzy logic is used as a modeling language.

5.3 Applications of fuzzy logic

Fuzzy logic has application in many fields, where there are imprecision and/or uncertainty or computing with words issues. Mendel (2007) has categorizes the applications of fuzzy logic in: approximation; clustering; control; databases; decision making; embedded agents; health care; hidden Markov models; neural networks; noise cancellation; pattern classification; quality control; spatial query; wireless communications. Some of the applications of fuzzy logic that are appeared in the literature and belong to the above categories are the following:

- **Approximation**: TSK/steel strip temperature (Mendez & Castillo, 2005).
- **Clustering**: C spherical shells algorithm (Hwang & Rhee, 2004), fuzzy C-means (Rhee & Hwang, 2001).
- **Control**: marine and traction diesel engines (Lynch, Hagras & Callaghan, 2005), integrated development platform (Sepulveda et al., 2005), evolutionary computing/NL dynamic plants (Castillo, Huesca & Valdez, 2005), buck DC–DC converters (Lin, Hsu & Lee, 2005), tracking mobile objects/robotic soccer games (Figueroa et al., 2005), liquid-level (Wu & Tan, 2004), proportional control (Tan & Lai, 2004), autonomou mobile robots (Hagras, 2004a), autonomous mobile robots/hierarchical (Hagras, 2004b), adaptive control of nonlinear plants (Melin & Castillo, 2003a).
- **Databases**: summarization (Niewiadomski et al, 2006).
- **Decision making**: variation in human decision making (Ozen, Garibaldi & Musikasuwan, 2004).
- **Embedded agents**: ambient intelligent environments (Doctor, Hagras & Callaghan, 2004; 2005).
- **Health care**: clinical diagnosis (John & Innocent, 2005), differential diagnosis (Di Lascio, Gisolfi & Nappi, 2005), nursing assessment (Wills, John & Lake, 2004).
- **Hidden Markov models**: phoneme recognition (Zeng &Liu, 2004).

- **Neural networks**: fuzzy perceptron (Rhee & Hwang, 2002a).
- **Noise cancellation**: adaptive noise cancellation (Castillo & Melin, 2004).
- **Pattern classification**: fuzzy k-nearest neighbor (Rhee & Hwang, 2002b).
- **Quality Control**: sound speakers (Melin & Castillo, 2003b).
- **Spatial query**: spatial objects (Rahimi et al., 2003).
- **Wireless communications**: wireless sensors/power on-off control (Liang & Wang, 2005), wireless sensor network lifetime analysis (Shu & Liang, 2005).

In addition, fuzzy set theory has been applied in educational systems. Some of these applications that are appeared in the literature are the following: Alves, Amaral, & Pires, 2008; Bai & Chen, 2006a; 2006b; 2008; Biswas, 1995; Chang & Sun, 1993; Cheng & Yang, 1998; Chen & Lee, 1999; Chiang & Lin, 1994; Echauz & Vachtsevanos, 1995; Jili et al., 2009; Jurado et al.,2008; Kosba, Dimitrova & Boyle , 2003; Law, 1996; Ma & Zhou, 2000; Nykänen, 2006; Severav, 2006; Suarez-Cansino & Hernandez-Gomez, 2008; Wang & Chen, 2006a, 2006b; Weon & Kim, 2001; Wilson, Karr, & Freeman, 1998; Wu, 2003. Nevertheless, its application in educational field is restricted mainly in assessment and educational grading systems.

5.4 Type-1 Fuzzy Sets

In 1965, Lotfi Zadeh published the first paper about fuzzy sets. According to him, unlike a crisp set whose membership function takes on only two values 0 or 1, the membership function of a fuzzy set can range between 0 and 1. For example, if A is a fuzzy set and $\mu_A(x)$ its membership function, then $0 \leq \mu_A(x) \leq 1$.

Fuzzy sets theory has been extensively used for computing with words, attaching word label to the fuzzy set. For example, the words “young”, “middle aged” and “old” are used to describe the fuzzy sets that represent the human's age (Fig. 27).

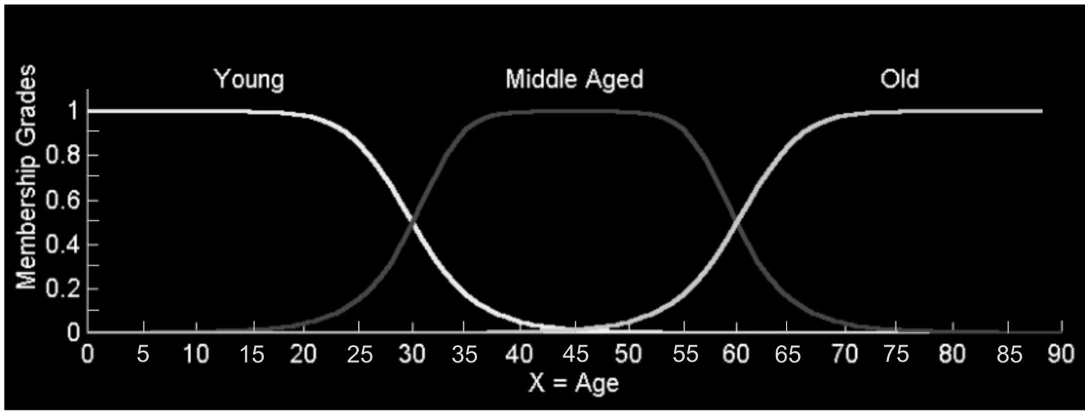


Figure 27: Fuzzy sets of age

This first approach of fuzzy sets theory is called type-1 fuzzy sets. Two common examples of a membership function of type-1 fuzzy sets are depicted in figure 28.

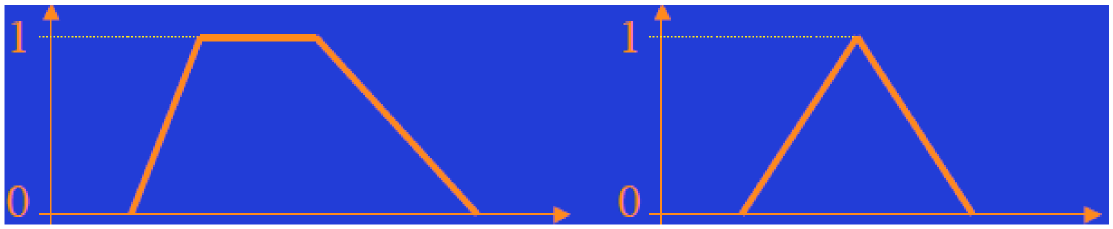


Figure 28: Membership functions of type-1 fuzzy sets (http://ewh.ieee.org/cmte/cis/mtsc/ieeecis/Mendel.pdf)

From the very beginning of fuzzy sets, criticism was made about the fact that the membership function of a type-1 fuzzy set has no uncertainty associated with it, something that seems to contradict the word fuzzy (wikipedia). When something is uncertain, like a measurement, it is difficult to determine its exact value, and of course type-1 fuzzy sets make more sense than using sets (Zadeh, 1975). However, it is not reasonable to use an accurate membership function for something uncertain. Type-1 fuzzy sets used in conventional fuzzy systems cannot fully handle the uncertainties that are present in intelligent systems (Castillo & Melin, 2008). To handle these uncertainties, Lotfi Zadeh (1975) proposed a more sophisticated kind of fuzzy sets theory that is called type-2 fuzzy sets (Mizumoto & Tanaka, 1976; Mendel, 2001).

5.5 Interval Type-2 Fuzzy Sets

Using type-2 fuzzy logic can reduce the amount of uncertainty in a system. This is happened due to the fact that type-2 fuzzy logic offers better capabilities to handle linguistic uncertainties by modeling vagueness and unreliability of information (Liang & Mendel, 2000). The concept of a type-2 fuzzy set was introduced first by Zadeh (1975) as an extension of the concept of an ordinary fuzzy set, i.e. a type-1 fuzzy set. The membership function of a general type-2 fuzzy set is three-dimensional (Fig. 29). In particular, type-2 fuzzy sets have grades of membership that are themselves fuzzy. At each value of the primary variable (e.g., pressure, temperature), the membership is a function (and not just a point value) –the secondary MF -, whose domain –the primary membership - is in the interval [0,1], and whose range - secondary grades – may also be in [0,1] (Mendel, 2003). In other words, the third dimension is the value of the membership function at each point on its two-dimensional domain that is called its footprint of uncertainty (FOU). Such sets are useful in circumstances where it is difficult to determine the exact MF for a FS, as in modeling a word by a FS.

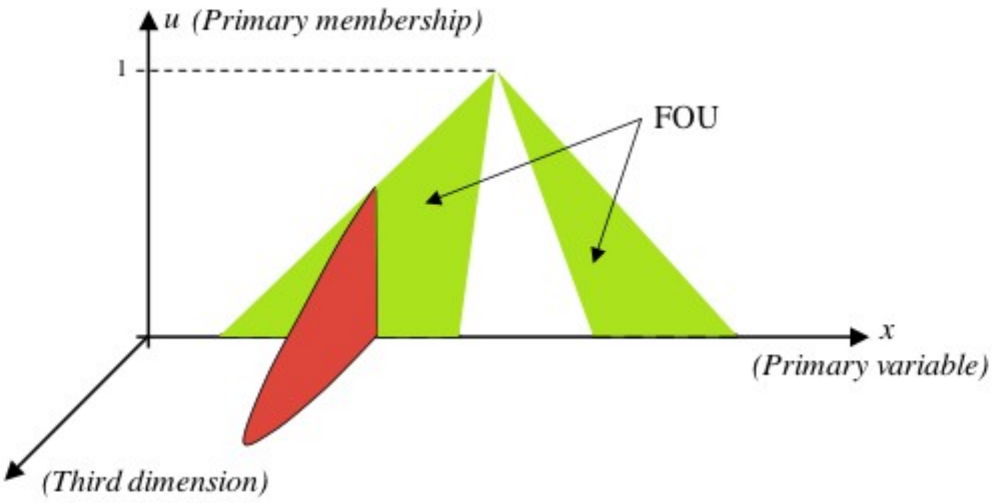


Figure 29: Membership function of a type-2 fuzzy set (wikipedia)

An example of type-2 fuzzy sets is the following one, which have been referred by Mendel (2003): “Suppose the variable of interest is eye contact, which we denote as x. Let's put eye contact on a scale of values 0-10. One of the terms that might characterize the amount of perceived eye contact (e.g., during flirtation) is "some eye contact." Suppose that we surveyed 100 men and women, and asked them to locate the ends of an interval for some eye contact on the scale 0-10. Surely, we will not get the same results from all of them, because words mean different things to different people. One approach to using the 100 sets of two end-points is to average the end-point data and to use the average values for the interval associated with some eye contact. We could then construct a triangular (other shapes could be used) MF, MF(x), whose base end-points (on the x-axis) are at the two average values and whose apex is midway between the two end-points. This type-1 triangular MF can be displayed in two-dimensions. Unfortunately, it has completely ignored the uncertainties associated with the two end-points. A second approach is to make use of the average values and the standard deviations for the two end-points. By doing this we are blurring the location of the two end-points along the x-axis. Now locate triangles so that their base end-points can be anywhere in the intervals along the x-axis associated with the blurred average end-points. Doing this leads to a continuum of triangular MFs sitting on the x-axis, e.g. picture a whole bunch of triangles all having the same apex point but different base points, as in figure 30. For purposes of this discussion, suppose there are exactly 100 (N) such triangles. Then at each value of x, there can be up to N MF values, $MF_1(x)$, $MF_2(x)$,..., $MF_N(x)$. Let's assign a weight to each of the possible MF values, say w_{x1}, w_{x2},..., w_{xN} (see Fig.30). We can think of these weights as the possibilities associated with each triangle at this value of x. At each x, the MF is itself a function -the secondary MF- $(MF_i(x), w_{xi})$, where i=1, ..., N. Consequently, the resulting type-2 MF is three-dimensional."

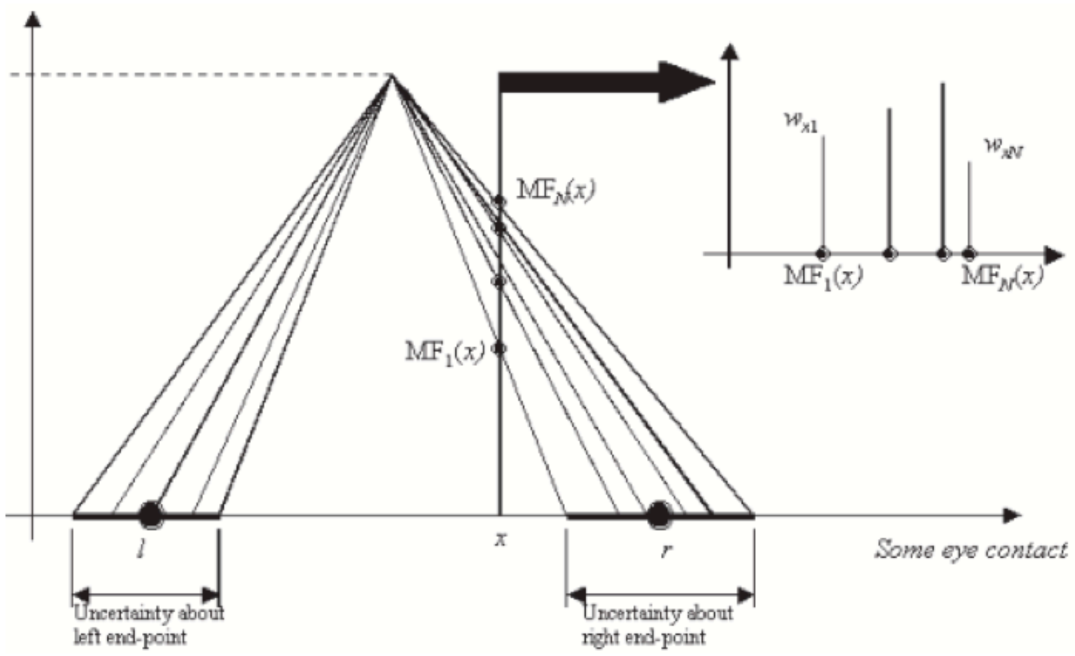


Figure 30: Triangular MFs when base end-points (l and r) have uncertainty intervals associated with them. This is not a unique construction (Mendel, 2003)

For an interval type-2 fuzzy set that third-dimension value is the same (e.g., 1) everywhere, which means that no new information is contained in the third dimension of an interval type-2 fuzzy set. So, for such a set, the third dimension is ignored, and only the FOU is used to describe it. The more (less) area in the FOU the more (less) is the uncertainty (Mendel, 2001). The FOU represents the blurring of a type-1 membership function, and is completely described by its two bounding functions (Fig. 31), a lower membership function (LMF) and an upper membership function (UMF), both of which are type-1 fuzzy sets.

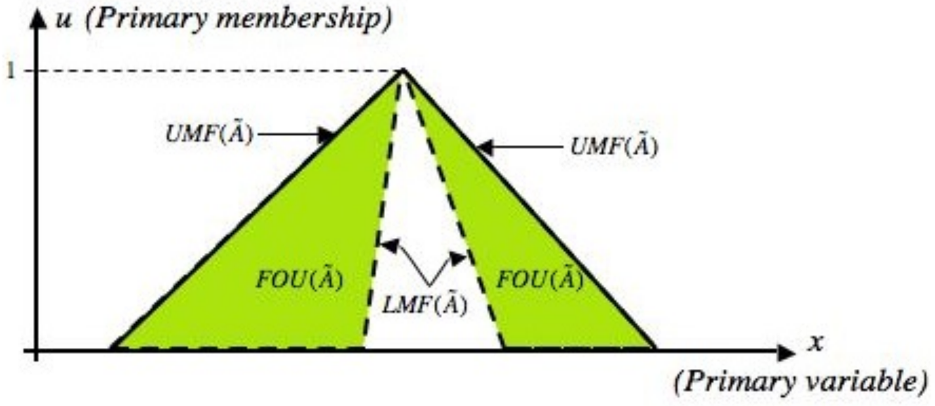


Figure 31: The FOU (wikipedia)

5.6 Rule-based fuzzy logic system

Type-2 fuzzy sets are finding very wide applicability in rule-based fuzzy logic systems (FLSs). Rules are the heart of a FLS. They are provided by experts or are extracted from numerical data. In either case, the rules are expressed as a collection of IF-THEN statements (e.g., IF temperature is moderate and pressure is high, then rotate the valve a bit to the right (wikipedia)). Fuzzy sets are associated with the terms that appear in the antecedents (IF-part) or consequents (THEN-part) of rules. Membership functions are used to describe these fuzzy sets.

As Karnik, Mendel, and Liang (1999) have stated, quite often, the knowledge used to construct rules in a FLS is uncertain. According to them, this uncertainty leads to rules having uncertain antecedents and/or consequents, which in turn translates into uncertain antecedent and/or consequent membership functions. They have referred the following example: when rules are collected by surveying experts, if we first query the experts about the locations and spreads of the fuzzy sets associated with antecedent and consequent terms, it is very likely that we will get different answers from each expert. This leads to statistical uncertainties about locations and spreads of antecedent and consequent fuzzy sets. Such uncertainties can be incorporated into the descriptions of these sets using type-2 membership functions. In addition, experts often give different answers to the same rule-question; this results in rules that have the same antecedents but different consequents. In such a case, it is also possible to represent the output of the FLS built from these rules as a fuzzy set rather than a crisp number. This can also be achieved within the type-2 framework.

A type-2 FLS is depicted in figure 32. As it is represented in this figure, two steps are required to go from an interval type-2 fuzzy set to a number. The first step is called type-reduction. In this step an interval type-2 fuzzy set is reduced to an interval-valued type-1 fuzzy set. There are a comparable number of type-reduction methods (Mendel, 2001). A well-known algorithm that is used for typr-reduction is the KM Algorithm is used for type-reduction (Karnik & Mendel, 2001). Although this algorithm is iterative, it is very fast. The second step is called defuzzification. It occurs after type-reduction. Because a type-reduced set of an interval type-2 fuzzy set is always a finite interval of numbers, the defuzzified value is just the average of the two end-points of this interval. Defuzzification maps the type-1 FS that came of the type-reduction step into a crisp number, by finding the centroid of the type-reduced set.

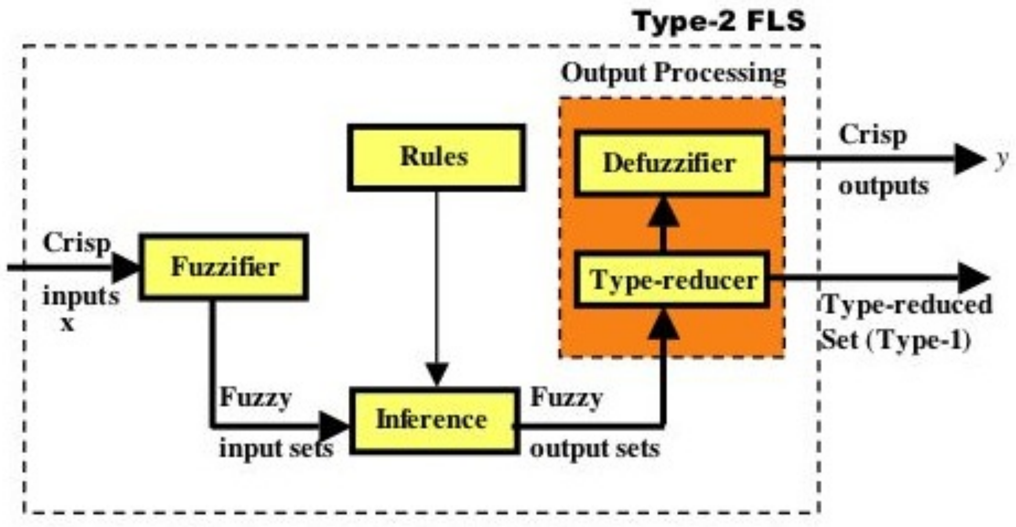


Figure 32: A type-2 FLS (wikipedia)

5.7 Fuzzy Cognitive Maps

Fuzzy Cognitive Maps are inference networks, using cyclic directed graphs, for knowledge representation and reasoning. A Fuzzy Cognitive Map is a cognitive map within which the relations between the elements (e.g. concepts, events, project resources) of a "mental landscape" can be used to compute the "strength of impact" of these elements. Fuzzy Cognitive Mapping is a combination of fuzzy logic and cognitive mapping, and it is a way to represent knowledge of systems which are characterized of uncertainty and complex processes. FCMs were introduced by Kosko (1986; 1992) and since then they have gradually emerged as a powerful paradigm for knowledge representation (Song et al., 2012). They provide a more flexible and natural mechanism for knowledge representation and reasoning, which are essential to intelligent systems (Miao et al., 2010). They constitute a way to represent real-world dynamic systems; in a form that corresponds closely to the way humans perceive it (Papageorgiou, 2011a).

A FCM illustrates the whole system as a combination of concepts and the various relations that exist between its concepts (Azadeh, Ziaei & Moghaddam, 2012; Song et al., 2011; Stula, Stipanicev & Bodrozic, 2010) (Fig. 33). A FCM consists of nodes (N_1, N_2, … N_n), which represent the important elements of the mapped system, and directed arcs (e_{ij}), which represent the causal relationships between two nodes (N_i, N_j). The directed arcs are labeled with fuzzy values in the interval [-1, 1] that show the “strength of impact” between the factors. A positive value indicates a positive causality between two factors, while a negative value indicates a negative causality between two factors. In particular, lets f_1 and f_2 be two related factors in a FCM. The positive value on the directed arc that connect f_1 with f_2, means that the increase of the value of f1 leads to the increase of the value of f_2, or the decrease of the value of f1leads to the decrease of the value of f_2. If the value is negative, it means that the increase of the value of f_1 leads to the decrease of the value of f_2, or the decrease of the value of f_1 leads to the increase of the value of f_2.

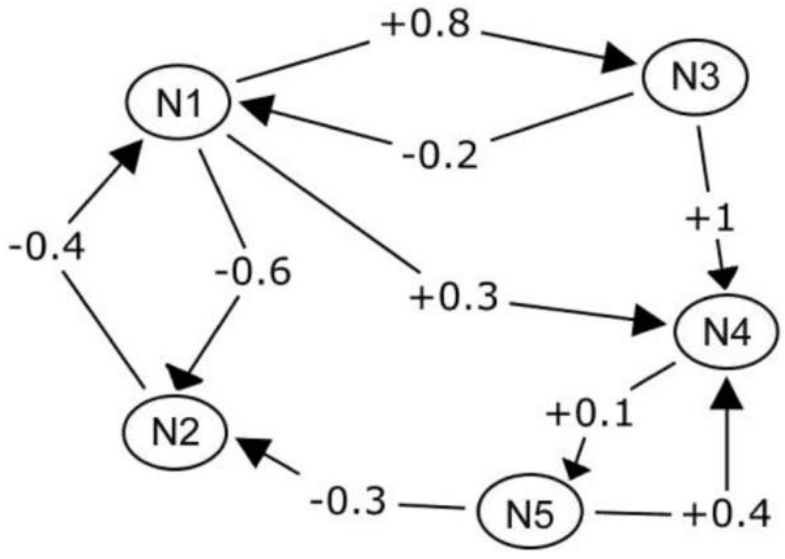


Figure 33: A Fuzzy Cognitive Map

The mathematical formulation of a FCM, according to Stach et al. (2005), is the following: A FCM is a 4-tuple (N, E, C, f), where:

1. $N=\{N_1, N_2, \dots N_n\}$ is the set of n concepts forming the nodes of a graph

2. E: $(N_i, N_j) \rightarrow e_{ij}$ is a function of N x N to K associating e_{ij} to a pair of concepts (N_i, N_j), with e_{ij} denoting a weight of directed edge from N_i to N_j, if $i \neq j$ and e_{ij} equal to zero if i = j. Thus, $E(NxN) = (e_{ij}) \in K^{nxn}$ is a connection matrix. For example, the connection matrix of the FCM that is depicted in figure 33 is the following (table 12):

Table 12: The connection matrix of the FCM of Fig. 33

	N1	**N2**	**N3**	**N4**	**N5**
N1	0	-0.6	0.8	0.3	0
N2	-0.4	0	0	0	0
N3	-0.2	0	0	1	0
N4	0	0	0	0	0.1
N5	0	-0.3	0	0.4	0

3. C: $N_i \rightarrow C_i$ is a function that at each concept N_i associates the sequence of its activation degrees such as for t є N, $C_i(t)$ є L given its activation degree at the moment t. C(0) є L^n indicates the initial vector and specifies initial values of all concept nodes and C(t) є L^n is a state vector at certain iteration L.

4. f:R→L is a transformation function, which includes recurring relationship on t≥0 between C(t+1) and C(t).

$$\forall i \in \{1,2....,n\}, C_i(t+1) = f(\sum_{\substack{i=1 \\ j\neq i}}^{n} e_{ij}C_j(t))$$

The transformation function is used to confine the weighted sum to a certain range, which is usually set to [0, 1].

According to Codara (1998), FCMs can be used for various purposes, including:

- to reconstruct the premises behind the behavior of a given agent, to understand the reasons for their decisions and for the actions the take, highlighting any distortions and limits in their representation of the situation (explanatory function),

- ➢ to predict future decisions and actions, or the reasons that a given agent will use to justify any new occurrences (prediction function),
- ➢ to help decision-makers ponder over their representation of a given situation in order to ascertain its adequacy and possibly prompt the introduction of any necessary changes (reflective function),
- ➢ to generate a more accurate description of a difficult situation (strategic function).

The main reasons for using the FCM approach are (van Vliet, Kok & Veldkamp, 2010): easy of use, easy to construct and parameterize, flexibility in representation, low time performing, easily, understandable/transparent to non-experts and lay people (Rodriguez-Repiso, Setchi & Salmeron, 2007), handle with complex issues related to knowledge elicitation and management. A collection of papers with applications of FCMs in various disciplines is presented in Glykas (2010). Over the past two decades, FCMs have attracted wide varieties of researchers in terms of representing knowledge and artificial intelligence in engineering applications (Aguilar 2005). They have been extensively used in a wide range of applications (Craiger et al., 1996; Kosko, 1999; Miao & Liu, 2000; Rodriguez-Repiso, Setchi & Salmeron, 2007; Stylios & Groumpos, 2004). Furthermore, according to Papageorgiou (2011a), in the past decade, FCMs have gained considerable research interest and are widely used to analyze causal systems such as system control, decision-making, management, risk analysis, text categorization, prediction etc. Paradigms of typical problems solved by FCMs are depicted in table 13, while ten journal papers on FCM applications and modeling in different scientific fields, which were published the first two months of year 2011, are depicted in table 14. However, the contribution of FCMs to the knowledge representation of an adaptive tutoring system has not been discussed before.

Table 13: Paradigms of typical problems solved by FCMs (Papageorgiou, 2011)

Paradigm	Typical problems solved by FCMs
	Description
Control	Prediction, interpreting, monitoring
Business	Planning, management, decision making, inference
Medicine	Decision support, modeling, prediction, classification
Robotics	Navigation, learning, prediction
Environment	Knowledge representation, reasoning, stakeholders' analysis, policy making
Information technology	Modeling, analysis

Table 14: FCM applications at the first two months of 2011 (Papageorgiou, 2011a)

Research works published in 2011	FCM applications at the first two months of 2011	
	Application area	**Problem solving**
Kannappan, Tamilarasi & Papageorgiou (2011)	Medicine	Classification, prediction
Papageorgiou (2011b)	Medicine	Knowledge representation, decision making
Beena & Ganguli (2011)	Structural damage detection	Learning
Song et al. (2011)	Business	Classification and prediction
Hanafizadeh & Aliehyaei (2011)	Soft system	Modeling, analysis
Iakovidis & Papageorgiou (2011)	Medicine	Decision making, reasoning
Lee, Bae & Koo (2011)	Sales assessment	Reasoning
Chytas, Glykas & Valiris (2011)	Business	Planning, analysis
Jetter & Schweinfort (2011)	Solar energy	Modeling, policy scenarios
Baykasoglu, Durmusoglu & Kaplanoglu (2011)	Industrial process control	Learning, control

6. Conclusions

A web-based tutoring or learning application is used by a very heterogeneous audience, which consists of learners with different characteristics and needs. Also, the teacher is absent during the instruction process. So, the delivery of the appropriate material to each individual learner is confronted with difficulties. These facts indicate the need for high adaptivity. The system has to recognize the needs, abilities, and preferences of each individual student and guide them to study the appropriate learning material.

Many adaptive tutoring systems for programming languages have implemented a student model that identifies the student's needs and dynamically adapts the educational process to meet them. However, they do not take into account how the level of a learner’s knowledge of a concept can affect the level of her/his knowledge of another related domain concept. Furthermore, up to date programming tutoring systems do not take into account the pace of learning of learners and cognitive states, such as forgetting previously learned material.

Learning is not a “black or white” process. It is a complicated process. It cannot be accurately said that a learner knows or does not know a domain concept. For example, a new domain concept may be completely unknown to the learner but in other circumstances it may be partly known due to previous related knowledge of the learner. On the other hand, domain concepts, which were previously known by the learner, may be completely or partly forgotten. Hence, currently they may be partly known or completely unknown. In this sense, the level of knowing cannot be accurately represented. Finally, the teaching process itself changes the status of knowledge of a user. This is happened due to the fact that a learner accepts new concepts while being taught.

To achieve this, the knowledge dependencies that exist between the learning materials' domain concepts have to be represented. Frequently, Bayesian networks have been used to model dependencies between different domain concepts. However these dependencies are restricted to prerequisite and granularity relationships between the domain concepts (Millán, Loboda & Pérez-de-la-Cruz, 2010); some examples of this are Andes (Conati, Gertner & Vanlehn, 2002), AdaptErrEx (Goguadze et al., 2011a; 2011b) and INQPRO (Ting & Phon-Amnuaisuk, 2012). However, these systems do not determine how the knowledge level of a domain concept is affected by changes in the knowledge level of another related concept. That is not determined in the programming tutoring systems, like Protus; J-LATTE; BITS; INCOM and SQLT-Web, which have been presented in chapter 4 neither. A solution to deal with this is Fuzzy Cognitive Maps (FCMs). The knowledge domain is represented in a more realistic way using FCMs, due to their ability to compute the "strength of impact" of their nodes (elements).

Kavčič (2004b) has modeled how a student's knowledge level of a concept is improved when s/he subsequently applies it as prerequisite knowledge when learning a following concept. However, this research only deals with how learning progresses; it deals neither with the possible decrease of knowledge due to the student forgetting some previously-learned concepts, nor the fact that the pace at which a student can learn the subsequent concepts depends upon how well s/he knows prerequisite concepts. Nevertheless, the learner's knowledge is a moving target. The knowledge level of a domain concept is increased when the student's performance is improved. Alternatively, it is decreased when the student forgets. Improvement of the knowledge level of a domain concept should lead to the increase of the knowledge level of all the related concepts (prerequisite and following), with his concept. Similarly, poor performance on a domain concept should lead to decrease of the knowledge level of all the related concepts with this concept.

In view of the above, an effective adaptive tutoring system has to be responsible for tracking cognitive state transitions of learners with respect to their progress or non-progress. The alterations on the state of student's knowledge level are not linear. They deal with uncertainty. Thus, a solution to represent these is fuzzy logic. Therefore, the target of this research is to develop a rule-based fuzzy logic student model, which models the cognitive state transitions of learners, such as forgetting, learning or assimilating. The student model has to monitor the learner's performance in a domain concept, update the learner's overlay model and make dynamic decisions on how the teaching syllabus is presented to the learner to fit his/her personal needs.

PART B:
A novel approach combining fuzzy techniques

1. The targets of the research

The research's target is to develop an intelligent tutoring system for programming language C, which:

- identifies and updates the student's knowledge level;
- provides individualized adaptive advice;
- reasons for the learner's errors;
- provides adaptation of the instructional material, taking into account the individuality of learners in terms of background, skills and pace of learning;
- allows each individual learner to complete the e-learning course at their own pace, taking decisions about which concepts should be delivered, which concepts need revision and which concepts are known and do not need rereading;
- helps learners to save time and effort during the learning process;
- automatically models the learning or forgetting process of a student;
- tracks cognitive state transitions of learners with respect to their progress or non-progress;
- represents the knowledge dependencies between the domain concepts of the knowledge domain;
- computes the “strength of impact” of each domain concept of the knowledge domain on another domain concept.

2. An ITS architecture including a tracking cognitive state transitions module and a fuzzy knowledge definer module

An ITS is any computer system that contains some intelligence and can be used in learning (Feedman, 2000), providing direct customized instruction or feedback to students, i.e. without the intervention of human beings, whilst performing a task (Psotka, Massey & Mutter, 1988). A typical ITS consists of four different subsystems or modules: the interface module which provides the means for the student to interact with the ITS, the domain module which contains a description of the knowledge or behaviors that represent expertise in the subject-matter domain the ITS is teaching, the student module which uses a student model containing descriptions of student knowledge or behaviors, including his or her misconceptions and knowledge gaps, and the tutor module which receives any mismatch between a student's behavior or knowledge and subsequently takes corrective action, such as providing feedback or remedial instruction (The technology of ITS have been described in details in part A section 1).

The mainly target of an ITS is to offer adaptivity trying to provide the support, processes and tasks that a teacher can offer. Therefore, it has to be able to recognize the students' needs and define her/his knowledge level considering her/his overall learning process and abilities. It has to be able to model automatically the learning and forgetting process of a student. This is achieved satisfactory through fuzzy rules, which deal with the uncertainty that there is in learning process and student diagnosis. Therefore, an ITS has to include a rule-based fuzzy module, which is responsible for identifying and updating the overall student's knowledge level, after a change has occurred on her/his knowledge level of a domain concept.

Furthermore, the system has to track the cognitive state transitions of learners with respect to their progress or non-progress. In such a way, it offers more effective adaptive instruction and learning support. Thus, a module that constructs automatically state-chart diagrams to keep track of cognitive state transitions is required.

The novel ITS architecture that includes a cognitive state transitions module and a fuzzy knowledge definer module is depicted in figure 34.

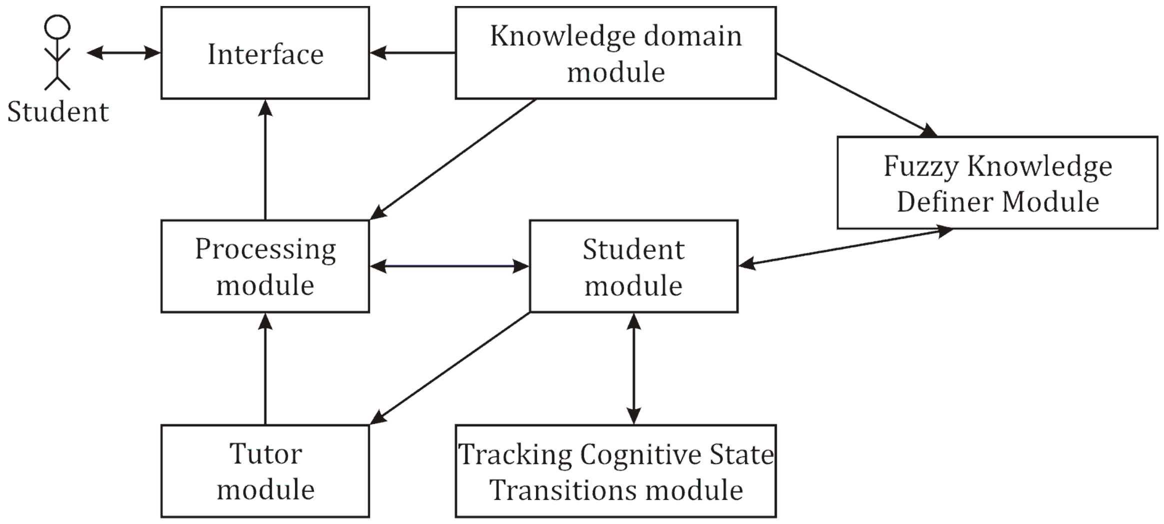


Figure 34: System's architecture

3. Domain representation using hierarchies and Fuzzy Cognitive Maps

The knowledge domain module is one of the most major modules of an ITS. It contains a description of the knowledge or behaviors that represent expertise in the subject-matter domain the ITS is teaching. The particular module has been introduced in ITS but its use has been extended to most current educational software applications that aim to be adaptive and/or personalized. The knowledge domain module is responsible for the representation of the subject matter taking into account the course modules, which involve domain concepts. Either the order in which each domain concept has to be taught or the knowledge dependencies that exist between the domain concepts of the learning material have to be represented.

A hierarchical knowledge representation is usually used in order to specify the order in which the domain concepts of the learning material have to be taught (Chen and Shen, 2011; Siddara and Manjunath, 2007; Vasandani and Govindury, 1995), and can be implemented through trees (Kumar 2005; Geng et al. 2011). Thus, a hierarchical tree, in which the intermediate nodes represent the domain concepts of the learning material and the leaf nodes represent the learning objectives, is essential for the knowledge domain representation (figure 35). It represents the order in which each domain concept has to be taught.

However, hierarchies do not give information about the dependency relations that exist between the domain concepts. The network of concepts representation gives this kind of information. However, in a network of concepts the relations between concepts are restricted to “part-of”, “is-a” and prerequisite relations. They do not give answers to the questions “If a student learn the concept A, which is her/his knowledge level of the depended domain concept B?”, or “If the student's knowledge of concepts A, B and C improves, how is her/his knowledge of the depended concept D affected?”, or “If the student has misconceptions on the domain concept A, how is her/his knowledge level of the depended concepts B, C and D affected?”. In other words, they do not represent how the knowledge of a domain concept of the teaching material, may be affected by the knowledge of another domain concept.

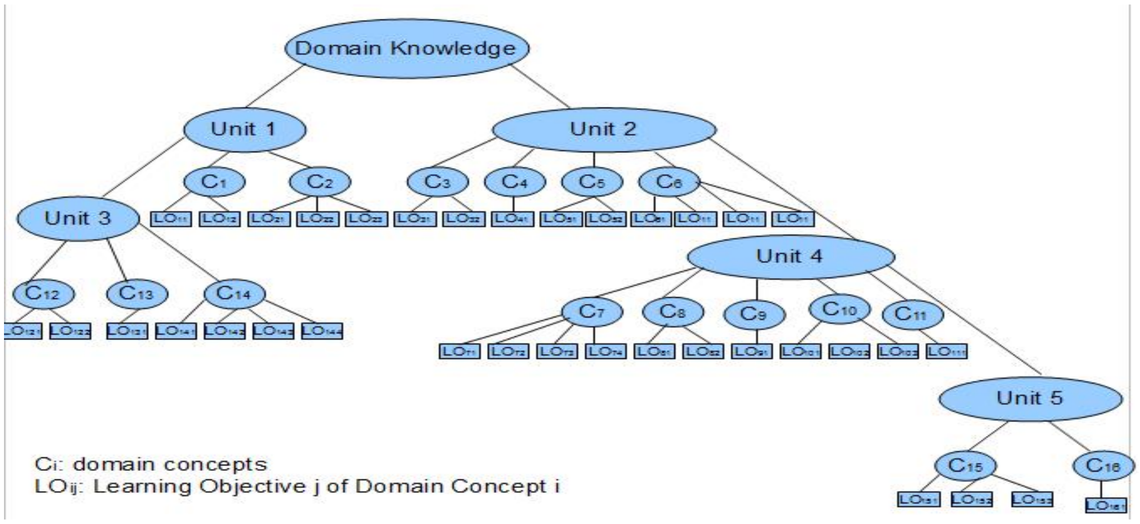


Figure 35: Hierarchical tree

However, the domain concepts that constitute the learning material are not independent from each other. The student's knowledge level of a domain concept usually is affected by her/his knowledge level of other related domain concepts. For example, a new domain concept may be completely unknown to the learner but in other circumstances it may be partly known due to previous related knowledge of the learner. On the other hand, domain concepts, which were previously known by the learner, may be completely or partly forgotten. Hence, currently they may be partly known or completely unknown. Therefore, the knowledge representation approach has to allow the system to recognize either the domain concepts that are already partly or completely known for a learner, or the domain concepts that s/he has forgot, taking into account the learner's knowledge level of the related concepts. For this reason, the knowledge dependencies that exist between the domain concepts of the learning material, as well as their "strength of impact" on each other have to be represented. A solution to this is the theory of Fuzzy Cognitive Maps (FCMs), which is used to model the behavior of complex systems (Leon et al 2011). A FCM is a cognitive map within which the relations between the elements (e.g. concepts, events, project resources) of a "mental landscape" can be used to compute the "strength of impact" of these elements.

A FCM illustrates the whole system as a combination of concepts and the various causal relations that exist between their concepts (Azadeh, Ziaei & Moghaddam, 2012), (Song et al., 2011). They are used to represent the dependencies that exist among the knowledge level of the domain concepts of the knowledge domain. Particularly, they represent the fact that the knowledge level of a domain concept is increased when the knowledge level of a related topic improves, as well as the fact that the knowledge level of a domain concept is decreased when the knowledge level of a depended topic is not satisfactory. The nodes of the FCM depict the domain concepts of the learning material, the directed arcs which connect the related domain concepts represent the causal relationships and the values which are labeled on arcs depict the degree at which the knowledge level of a domain concept is affected regarding the knowledge level of its related domain concepts (Fig. 36). The values which are labeled on arcs of the FCM are only positive, since the increase of the knowledge level of a domain concept leads to the increase of the knowledge level of a depended domain concept, and the decrease of the knowledge level of a domain concept leads to the decrease of the knowledge level of a depended domain concept. Therefore, the values of the arcs belong to the interval (0, 1].

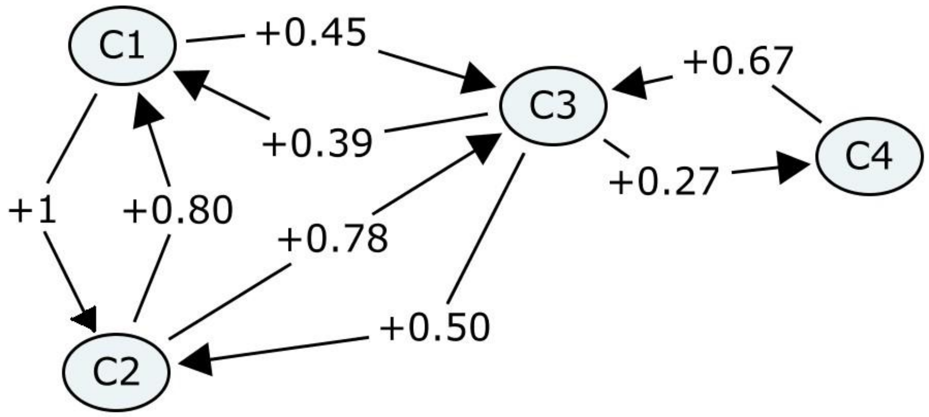


Figure 36: Knowledge domain representation using FCM

The arcs in the FCM, which represent the domain concepts' dependencies of the knowledge domain, are bidirectional. Furthermore, the value of the arc $C_i \rightarrow C_j$ is not essentially equal to the value of the arc $C_j \rightarrow C_i$. This is happened due to the fact that changes on the knowledge level of C_i may affect the knowledge level of C_j in a different degree than changes on the knowledge level of C_j affect the knowledge level of C_i. It has to be clear that the value 1 on the directed arc that connects two dependent domain concepts does not mean that the two dependent concepts are the same. It implies that if a learner knows a domain concept of a section, s/he may know a related concept of another section at the same degree. The percentage of increase or decrease of the knowledge level of a domain concept that occurs due to changes on the knowledge level of another concept related with this domain concept is defined by experts of the knowledge domain.

Thus, in a correspondence with its theoretical definition (Stach et al., 2005), a FCM that is used to represent the knowledge domain of the learning material is a 4-tuple (C, W, KL, f), where:

- $C=\{C_1, C_2, \ldots C_n\}$ is the set of concepts of the knowledge domain.
- W: $(C_i, C_j) \rightarrow w_{ij}$ is a connection matrix, where w_{ij} is a weight of the directed arc from C_i to C_j, which denotes that the knowledge level of the concept C_i affects that of concept C_j.
- KL is a function that at each concept C_i associates the sequence of its activation degree. In other worlds, $KL_i(t)$ indicates the value of a concept's knowledge level at the moment t.
- f is a transformation function. For the definition of the transformation function the following limitation has to be taken into account. Only the knowledge level of the most recently read concept affects the knowledge level of a domain concept, each time. The reason for this is the fact that the learner's knowledge level is affected either by the new knowledge that s/he has obtained, or by the knowledge that s/he has forgot, each time. Consequently, the KL value of a concept is affected only by the KL value of the most recently read concept, regarding the weight of the directed arc

that connects them. Therefore, the transformation function for a FCM, which is used to represent the knowledge domain of the learning material, is defined as: $KL_i(t+1)=f(KL_i(t)\pm w_{ji}*p_j*KL_i(t)/100)$, where p_j is the percentage of the difference on the value of the knowledge level of the most recently read concept C_j, with $p_j=(KL_j(t+1)-KL_j(t))*100/KL_j(t)$. Also, the + is used in case of increase and the – is used in case of decrease.

It must be referred that the initial values KL(0) for concepts are zero. The reason for this is the fact that a learner is considered as novice in the beginning of the learning process. The changes on these values indicate the progress or no-progress of the learner.

For a better computation of the knowledge dependencies, the connection matrix (w-matrix) of the FCM is used. This matrix depicts the “strength of impact” between the concepts of the learning material. It represents the weight of the directed arcs of FCM in a more simple and explicit way. For example, table 15 is the w-matrix of the FCM that is depicted in figure 36. The number of rows of this matrix is equal to the number of columns, and it is also equal to the number of the nodes (domain concepts) that are depicted to the FCM. The values of the directed arcs of the FCM are written into the cells of the matrix. The matrix is completed row by row. The value of the “strength of impact” of the domain concept that corresponds to the matrix’s row i on the domain concept that corresponds to the matrix’s column j is written into the matrix's cell (i, j). For example, the value of the “strength of impact” of the domain concept C_1 on the domain concept C_3, which are depicted in figure 36, is written in the corresponding matrix's cell (1, 3) (table 15). The values of the matrix’s main diagonal are zero, since changes on the knowledge of a domain concept cannot affect the domain concept itself. This confronts to the FCM definition (Part A, Section 5.7), which denotes that $e_{ij}=0$ if $i=j$.

Table 15: W-matrix with knowledge dependencies of Fig. 36

	C1	C2	C3	C4
C1	0	1	0.45	0
C2	0.80	0	0.78	0
C3	0.39	0.50	0	0.27
C4	0	0	0.67	0

The knowledge domain representation has to be combined with a well-designed student model, which will be is responsible for how the system will utilize the information which is included in the knowledge domain module, in order to make the right decisions for offering personalized instruction and support. The student model uses the knowledge dependencies between the domain concepts of the learning material, as well as the “strength of impact” on each other in order to recognize the possible increase or decrease of the learner's knowledge. This approach of the knowledge domain representation allows the tutoring system to deliver the learning material to each individual learner dynamically taking into account her/his learning different learning pace. Consequently, the ability of the presented knowledge representation approach to depict the possible increase or decrease of the learner’s knowledge constitutes the particular approach as a novel driver for the adaptive and/or personalized tutoring systems for providing personalized presentation of the learning material.

4. Combining stereotypes and overlay model with fuzzy logic techniques

A compound student model, which brings together various features of different techniques of user modeling, is the solution for offering a more adaptive learning system. The reason for this is the fact that the student model needs to combines various aspects of student’s characteristics that is both domain dependent and domain independent in order to carry out the personalization efficiently (Yang, Kinshuk & Graf, 2010). This way, the model not only can exhibit unique individual characteristics and preferences of each learner by monitoring and tracing the changes of their knowledge, skills, interests, but also classify the learners according to their performance, individual learning behaviors and activities (Guangbing, Kinshuk & Graf, 2010). That is the reason for the development of a novel compound student, which combined overlay technique and stereotypes with fuzzy logic. In particular, the novel student model consists of two layers (Fig. 37). The first layer includes a qualitative uncertainty-based weighted overlay model, which represents the learner's knowledge. The learner's knowledge level of each domain concept of the learning material, which is the base for the construction of the overlay model, is defined by a set of fuzzy logic rules. The second layer includes a three-dimensional stereotype model. The first dimension consists of stereotypes that represent the learner’s knowledge level and vary from novices to experts; the second dimension consists of two stereotypes and concerns the type of errors that a learner can make and the third dimension concerns prior knowledge of the student on related knowledge domain fields.

The qualitative uncertainty-based weighted overlay model is used to model the variations of the learner's knowledge level. Particularly, it is used to inform the system which domain concepts are learned, which domain concepts are partly known and which domain concepts are completely unknown. The idea of overlay knowledge modeling is to represent an individual user’s knowledge as a subset of the domain model that resembles expert knowledge of the subject (Nguyen &

Do, 2008). A qualitative weighted overlay model is an extension of the pure overlay model that can distinguish several levels of student's knowledge about each concept representing user knowledge of a concept as a qualitative value (Brusilovsky & Anderson, 1998), (Papanikolaou et al., 2003). The uncertainty-based models represent the learner's knowledge as probability that the learner knows the concept (Henze & Nejdl, 1999; Specht & Klemke, 2001). In the presented novel compound student model, the overlay model uses a qualitative value ('unknown', 'insufficiently known', 'known', 'learned') combined with a percentage from 0 to 100% that points the weight of a qualitative value for a concept. For example, '85% Known C_1', means that the concept C_1 is 85% known. Figure 38 depicts an example of the system's overlay model. The concepts, which are colored blue or are colored pink and are circle with blue line (i.e. they are '100% known'), belong to the subset of the domain model that the learner knows or has assimilated. The knowledge state of the intermediate nodes, which depict the sections of the learning material, is determined by the knowledge state of the corresponding nodes-leafs that depict the domain concepts, in which each section is analyzed.

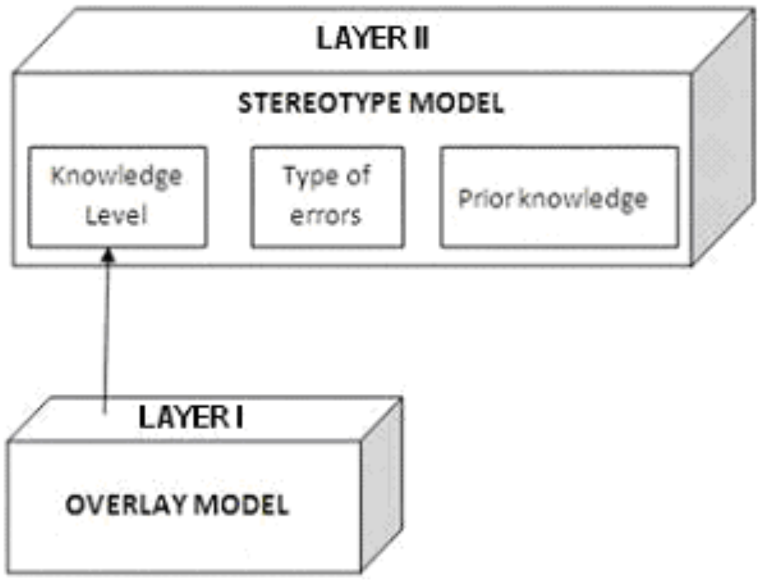


Figure 37: The novel student model

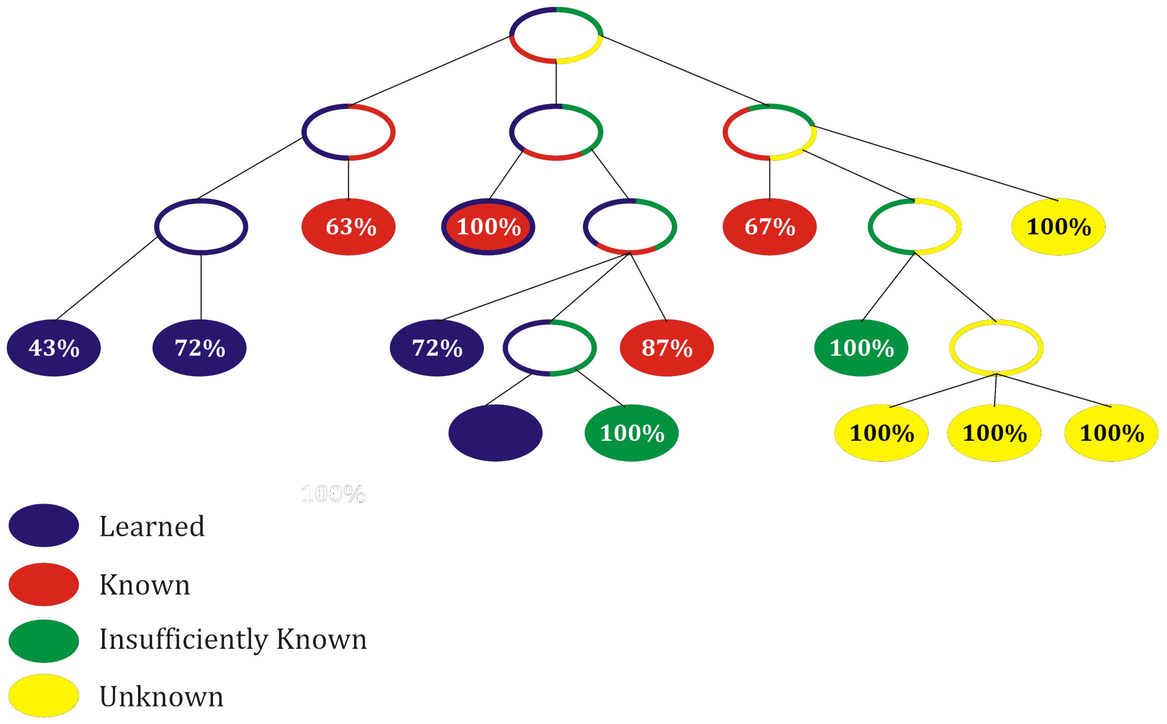


Figure 38: Qualitative uncertainty-based weighed overlay model

A learner is classified to a knowledge level (KL) stereotype category according to which domain concepts the learner knows and how well. The overlay model of the first layer of the student model provides this information. The value of KL varies from "novice" to "expert". The KL stereotype category to which the learner has been classified, gives information about the learning material that should be delivered to the learner.

However, defining the learner's knowledge level is not adequate in order to model individual students' needs and abilities. A learner has misconceptions and the system should detect and reason them. That is the reason for the definition of the 2nd-dimension of the stereotype model. Furthermore, the system should known if a learner has prior knowledge on other related fields with the teaching knowledge domain. In this way, the tutoring system is able to distinguish if an error occurs due to non-learning or due to affecting by prior knowledge. This kind of information is derived by the 3rd-dimension of the stereotype model.

The student model is updated each time new information about the learner is required. New information about the learner is obtained each time s/he interacts with the system. More concretely, each time the learner interacts with the system, s/he takes a test, the results of which determine the learner's knowledge and update her/his overlay model. The second layer or the student model receives information from the overlay model and determines the stereotype category of knowledge level (KL) to which the learner should be classified. The stereotype categories of the second and third dimension of our student model, to which the learner should be classified, are not affected by the information that is received by the overlay model. The stereotype category of the second dimension to which the learner belongs each time, is determined by the type of errors that s/he does during the test. Also, the third dimension of the student model is determined by the learner during her/his registration to the system.

In particular, each time the learner completes a section, s/he is asked to complete a test in order to check her/his knowledge and progress. If s/he succeeds in the test, then s/he is transited to a next knowledge level and the value of KL is increased. Otherwise, if s/he fails, then s/he remains to the same knowledge level or s/he is returned to a previous knowledge level, according to her/his errors. In particular, if the learner's poor performance does not affect the knowledge level of other related domain concepts, which belong to a previous section, then the value of KL remains the same; otherwise the value of KL is decreased. If the learner makes errors that correspond to concepts of previous section, then the system infers that s/he has forgot something from previous sections of the learning material. In particular, if the learner makes errors, which consider concepts of previous knowledge level, then the system checks the value of the stereotype that corresponds to her/his prior knowledge on related fields. If the system decides that the errors were made due to confusion with prior knowledge, then it does not classify the learner to a previous knowledge level, but it points out the error. Otherwise, it classifies the learner to a previous knowledge level reducing the value of KL.

5. A novel rule-based fuzzy logic module for modeling automatically the learning or forgetting process of a student

It is not a straightforward task to define for each learner which concepts are unknown, known or assimilated and at what degree. The overlay model of the presented system obtains this information. The information which is provided by the overlay model have the following format: “The concept ‘If statement’ is 72% Known for the student A”, “The concept ‘calculating a sum in a for loop’ is 60% insufficiently known for the student X”, “The concept ‘assignment statement’ is 100% learned for the student Z”, “The concept sorting algorithms is 100% unknown for student Z”. This information of the overlay model is imprecise. One possible approach to deal with this is fuzzy set techniques, with their ability to naturally represent human conceptualization. That is the reason for the integration of fuzzy logic techniques into the hybrid student model, developing a novel rule-based fuzzy logic module. The particular module leads the system to make inferences about the changes of the user’s state and make useful adaptation decisions, offering dynamic adaptation to users’ needs.

The presented rule-based fuzzy logic module is responsible for identifying and updating the student’s knowledge level of all the concepts of the knowledge domain. Its operation is based on the Fuzzy Cognitive Maps that are used to represent the dependencies among the domain concepts. It uses fuzzy sets to represent the student’s knowledge level and a mechanism of rules over the fuzzy sets, which is triggered after a change has occurred on the student’s knowledge level of a domain concept. This mechanism updates the student’s knowledge level of all related with this concept, concepts. The presented fuzzy module updates the first layer of the system’s two-layer student model, which represents the knowledge level of the student. With this approach the alterations on the state of student’s knowledge level, such as forgetting or learning are represented.

The presented rule-based fuzzy logic module includes the following steps:

- <u>**Step 1**</u>**: Defining the fuzzy sets:**

  The fuzzy sets for representing students' knowledge level of a domain concept, as well as their membership functions must be defined. Fuzzy sets are used to characterize the changeable user's state. For example, ("Unknown", "Known", "Learned"} or ("Unknown", "Insufficiently Known", "Known", "Learned", "Assimilated"} are the fuzzy sets of educational adaptive systems. Therefore, FS_1, FS_2, …, FS_n are the defined, for the adaptive systems, fuzzy sets. Then, their membership functions are the following (x indicates the value of the criterion, which determines the fuzzy sets that are active each time (like degree of success). Also, a, b, c, d and e indicate particular degrees of success like 0, 50, 100. They are thresholds and it is a<b<c<d<e):

$$\mu_{FS_1}(x)=\left\{\begin{array}{cc} 1 & ,x\le a \\ 1-\dfrac{x-a}{b-a} & ,a<x<b \\ 0 & ,x\ge b \end{array}\right\}$$

$$\mu_{FS_n}(x)=\left\{\begin{array}{ll} \dfrac{x-e}{f-e} & ,e<x<f \\ 1 & ,f\le x\le g \\ 0 & ,x\le e \end{array}\right\}$$

$$if \quad i\neq 1 \quad and\, i\neq n, \mu_{FS_i}(x)=\left\{\begin{array}{ll} \dfrac{x-a}{b-a} & ,a<x<b \\ 1 & ,b\le x\le c \\ 1-\dfrac{x-c}{d-c} & ,c<x<d \\ 0 & ,x\le a \quad or\, x\ge d \end{array}\right\}$$

$$\mu_{FS_{i+1}}(x) = \left\{ \begin{array}{ll} \dfrac{x-c}{d-c} & ,c < x < d \\ 1 & ,d \le x \le e \\ 1-\dfrac{x-e}{f-e} & ,e < x < f \\ 0 & ,x \le c \quad or\, x \ge f \end{array} \right\}$$

The above used membership functions are symmetric. The knowledge level of a domain concept changes in a continuous way. Meaning that the knowledge level of a domain concept usually passes gradually from the unknown state to the learned and assimilated state. In the presented rule-based fuzzy logic module there is the limitation that the different knowledge levels of a domain concept can be only of "neighbor" fuzzy sets. Furthermore, membership values correspond to percentages of the offered knowledge in a way that they cover 100% of it, at any time. This gives a more natural and understandable way of representation. For example, it would be non-intuitive to say that domain concept "A" is 0.5 (50%) Insufficiently Known and 0.6 (60%) Known for a student, given that 0.5 plus 0.6 gives 1.1 (110%). So, the sum of the concept's percentage of different knowledge levels has to be 100%, or 1 if the membership value of a concept to a knowledge level category is from 0 to 1. So, the following expression stands:

$$\mu_{\mathrm{FS}_1} + \mu_{\mathrm{FS}_2} + \mu_{\mathrm{FS}_3} + \ldots + \mu_{\mathrm{FS}_n} = 1$$

Therefore, a set (μ_{FS1}, μ_{FS2}, μ_{FS3},... ,μ_{FSn}) is used to express the student knowledge of a domain concept.

- **<u>Step 2:</u> Defining and applying the fuzzy rules:**

  When there is a dependency between two domain concepts, then the knowledge level of the one domain concept can affect the knowledge level of the other domain concept. More specifically, the following are taken into account:

  - Considering the knowledge level of C_i, the knowledge level of its following domain concept C_j is increased or decreased.
  - Considering the knowledge level of C_j, the knowledge level of its prerequisite domain concept C_i is increased or decreased.

Consequently, the student model expands when a change on the knowledge level of a domain concept causes increase on the knowledge level of the related concepts, or it is minimized when a change on the knowledge level of a domain concept causes decrease on the knowledge level of the related concepts with this concept.

In this document, D is defined to represent the knowledge dependency between two domain concepts. The symbolism $\mu_D(C_i, C_j)$ is used to represent the "strength of impact" of C_j on C_i and the symbolism $\mu_D(C_j, C_i)$ is used to represent the "strength of impact" of C_i on C_j. The values of $\mu_D(C_i, C_j)$ and $\mu_D(C_i, C_j)$ are the values of the arcs in the FCMs of the knowledge domain (Fig. 36 in section 3).

Concerning two domain concepts C_i and C_j where C_i is taught before C_j, the knowledge level of the concepts can change according to the following rules. These rules depict how the changes on the knowledge level of the domain concepts of the learning material for a student occur, revealing her/his learning state. In particular, they reveal if s/he learns or not or if s/he forgets. If the knowledge level of a concept is decreased, then the system infers that the student does not learn. If the knowledge level of a previously taught concept is decreased, then the system infers that the student forgets. If the knowledge level of a concept is increased, then the system infers that the student learns, and if the knowledge level of all the related concepts is improved continuously, then the system infers that the student assimilates the learning material.

The rules are based on Kavčič's (2004b) work. That work models mainly how the student's knowledge level of the prerequisites concepts that the student had read previously, is improved when s/he performs better in following concepts. In this way Kavčič's work deals only with how learning progresses. In her work there are no rules that imply the possible decrease of knowledge via the student's forgetting of some previously learned concepts. Moreover, another important problem that is not dealt with in Kavčič's work is the fact that in static educational systems, students are often required to repeat previously known concepts thought the following chapters. However, this practice is quite generic and does not take into account individual features of a student such as how fast they learn or how well they remember previously taught concepts. As such, educational systems do not adapt their pace on individual students. In view of the above, in the presented rule-based fuzzy module, Kavčič's rules have been expanded to deal with the above problems and the rules with these novelties that lead to the dynamic personalization of teaching are the following (where FS_x, FS_y are fuzzy sets that represent knowledge levels with $FS_x<FS_y$, and KL( ) denotes the "Knowledge Level of"):

- ***Based on updates of the KL(C_i), the KL(C_j) is improved according to:***

R1: If KL(C_j)= FS_x and KL(C_i)=FS_x, then KL(C_j)= FS_x with

$$\mu_{\mathrm{FS_x}}(C_j) = \max[\mu_{\mathrm{FS_x}}(C_j), \mu_{\mathrm{FS_x}}(C_i) * \mu_{\mathrm{D}}(C_i, C_j)]$$

R2: If KL(C_j)= FS_x and KL(C_i)=FS_y, then KL(C_j)= FS_y with

$$\mu_{\mathrm{FS_y}}(C_j) = \mu_{\mathrm{FS_y}}(C_i) * \mu_{\mathrm{D}}(C_i, C_j)]$$

❖ ***Based on updates of the KL(Ci), the KL(Cj) is deteriorated according to:***

R3: If KL(Cj) = 100% FS_n, then it does not change.

R4: If KL(C_j)= FS_y and KL(C_i)=FS_x, then KL(C_j)= FS_x with

$$\mu_{FS_x}(C_j) = \mu_{FS_x}(C_i) * \mu_D(C_i, C_j)$$

❖ ***Based on updates of the KL(Cj), the KL(Ci) is improved according to:***

R5: If KL(C_i)= FS_x and KL(C_j)=FS_x, then KL(C_i)= FS_x with

$$\mu_{FS_x}(C_i) = \max[\mu_{FS_x}(C_i), \mu_{FS_x}(C_j) * \mu_D(C_j, C_i)]$$

R6: If KL(C_i)= FS_x and KL(C_j)=FS_y, then KL(C_i)= FS_y with

$$\mu_{FS_y}(C_i) = \mu_{FS_y}(C_j) * \mu_D(C_j, C_i)]$$

❖ ***Based on updates of the KL(Cj), the KL(Ci) is deteriorated according to:***

***R7:** If KL(Ci) = 100% FS_n, then it does not change.*

R8: The formula $x_i = \min[x_i, (1 - \mu_D(C_i, C_j)) * x_i + x_j]$, where x_i and x_j are the values of the criterion, which determines the fuzzy sets that are active each time for C_i and C_j respectively, is used. Then, using the new x_i, the KL(C_i) is determined, calculating the membership functions.

6. Construct statechart diagrams to track the cognitive state transitions of learners

Taking into account all the above, it can be concluded that a domain concept passes through several states during the interaction of the user with the system. These states are expressed through the fuzzy sets that have been defined at the first step of the process of the rule-based fuzzy module, which was described above. In addition, a learner passes through several cognitive states during the learning process. S/he can learn or not, forget, assimilate or not e.t.c.. These states determine the progress of the user each time. They are revealed by the transition from one stereotype of the student model to another. Thus, the system has to decide which stereotypes have to be activated and which stereotypes have to be deactivated, at each interaction of a learner with educational application.

As Tretiakov et al. (2005) have been stated, a state-chart diagram can been used to show the sequence of a student's mind. Consequently, the educational system has to construct statechart diagrams to track the cognitive state transitions of learners, In such a way, it is able to be informed about the progress or non-progress of each individual student. Table 16 depicts the relation between a learner's cognitive state and the transitions among the stereotypes of the second layer of the hybrid student model, which was presented at the section 4. Figure 39 depicts a state-chart diagram that shows the sequence of the changes of the knowledge level of a domain concept of the learning material. Finally, figure 40 depicts the sequence of the changes of a learner's cognitive state in relation with her/his progress or no-progress.

Table 16: The relationship between cognitive state and KL stereotype transitions

COGNITIVE STATE		STEREOTYPE TRANSITIONS
S/he does not learn	s/he doesn't read	No transition to other stereotype
	s/he reads but not learn	
	s/he reads but s/he has difficulty in understanding	
S/he learns		Transition to the next stereotype
S/he forgets		Transition to a previous stereotype
S/he reaches target knowledge		Continuous transition to the advanced stereotypes

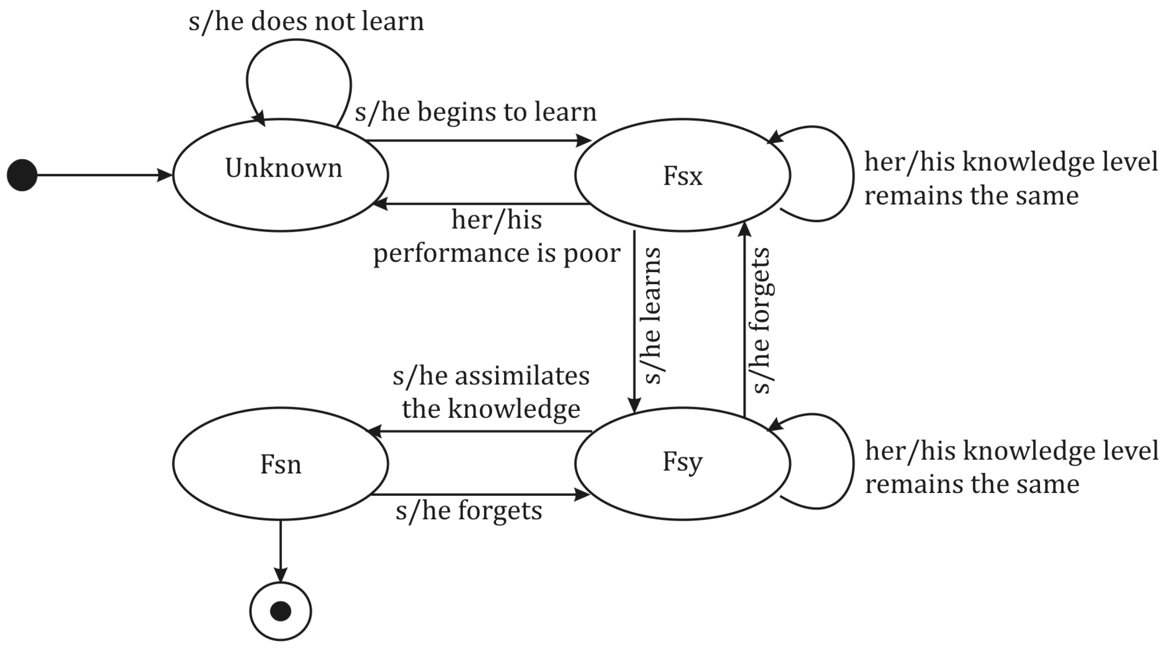

s/he does not learn
s/he begins to learn
Unknown
Fsx
her/his knowledge level remains the same
her/his performance is poor
s/he learns
s/he forgets
s/he assimilates the knowledge
Fsn
Fsy
her/his knowledge level remains the same
s/he forgets

Figure 39: Sequence of the changes of the knowledge level of a domain concept (FSx, FSy and FSn are fuzzy sets that describes the knowledge level)

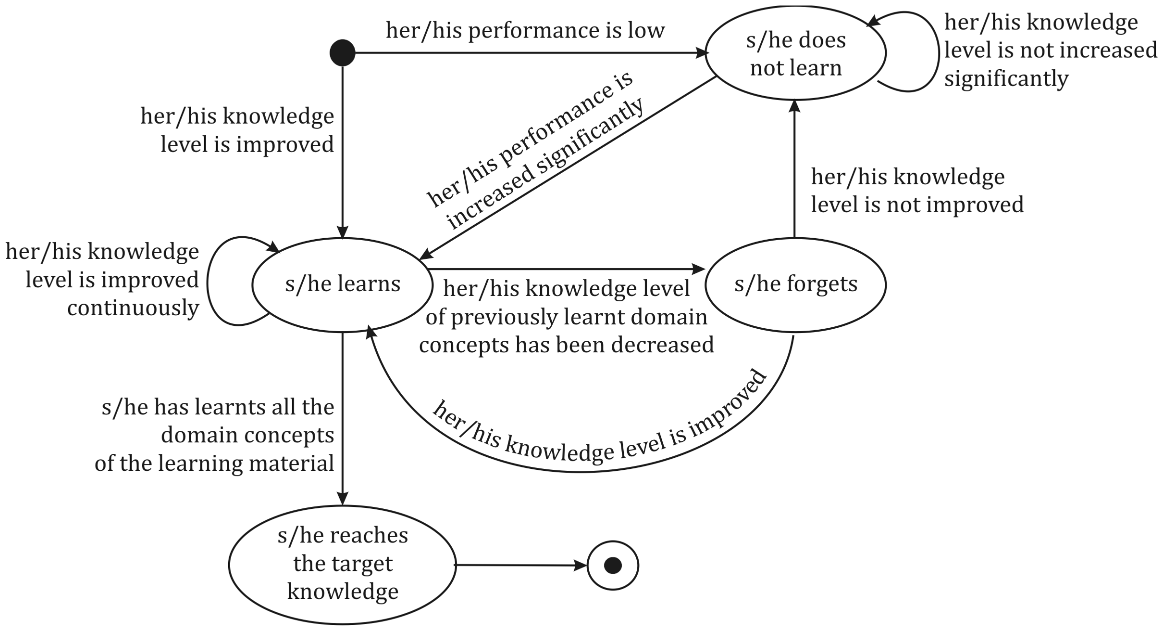

her/his performance is low
s/he does not learn
her/his knowledge level is not increased significantly
her/his knowledge level is improved
her/his performance is increased significantly
her/his knowledge level is not improved
her/his knowledge level is improved continuously
s/he learns
her/his knowledge level of previously learnt domain concepts has been decreased
s/he forgets
s/he has learnts all the domain concepts of the learning material
her/his knowledge level is improved
s/he reaches the target knowledge

Figure 40: Sequence of the changes of the learner's cognitive state

PART C: Application, implementation and evaluation ELaC – A novel system fully operational

1. The system’s architecture

In the presented research a full implementation of the novel approach of adaptive educational system combining fuzzy logic techniques with overlay model and stereotypes, which is described in part B, was done. In particular, an innovative integrated e-learning environment for computer programming and the language C (here in “C”), which is called ELaC, have been developed. The technology of Information Tutoring Systems (ITSs) was consulted for the system's design and development. The architecture model of ELaC is depicted in figure 41. It consists of the following six components:

- **Interface module:** It is responsible for the communication between a student and other modules of the system. It has to be user-friendly and adapt to each individual student dynamically according to her/his need. To achieve it, the interface module has to be informed of the student’s need and characteristics, of the student’s requests and of the corrective actions that the system has to provide. Furthermore, the user interface is responsible for displaying the knowledge domain.

- **Knowledge domain module**: It is responsible for the representation of the subject matter taking into account the course modules, which involve domain topics and learning objectives and the relations among them. One of the most important components of an adaptive educational application is the domain representation. The hierarchy of the domain concepts hast to be represented. Also, the knowledge dependencies between the domain concepts of the learning material and their “strength of impact” have to be represented. Thus, the knowledge domain of the system is organized into a hierarchical tree, in order to represent the difficulty level of the domain topics and the order in which each topic has to be taught. Furthermore, FCMs are used to represent the knowledge dependencies between the domain concepts of the learning material.

- **Student model**: It is responsible for representing the knowledge, skills and needs of each individual student. So, the system has to retain static information about each students (age, years of experience on programming, programming languages that s/he knows, e.t.c.), and dynamic information (errors, misconceptions, progress, e.t.c.), which is obtained during the interaction with the system. According to Nguyen & Do (2009) and Millán, Loboda, & Pérez-de-la-Gruz (2010) the problem of user modeling is described by the following three questions: i) "What are the characteristics of the user we want to model?", ii) "How we model them?", iii) "How we use the user model?". In this research the target is to model the cognitive states of each learner. In other words, the system has to recognize when a learner learns or not, forgets or assimilates concepts of the knowledge domain. Also, the system has to be able to reason for no learning a domain concept. This is happening due to the fact that the learner did not read the concept or because s/he read it but s/he did not understand it? In order to model the student's knowledge we use o qualitative overlay model, which represents the degree to which the learner knows each fragment of the knowledge domain. However, the overlay model cannot represent learner's misconceptions and needs. That's why we use a stereotype model in addition. Moreover, we use error theories in order to explain learner's behavior through her/his errors. Furthermore, it uses fuzzy sets to represent the qualitative values of the overlay model that represent students' knowledge level and a mechanism of rules over the fuzzy sets, which is triggered after a change has occurred on the student's knowledge level of a domain concept. This mechanism identifies and updates the student's knowledge level. Therefore, the system combines an overlay with a stereotype user model and fuzzy logic techniques.

- **CoStaT module**: It is responsible for tracking cognitive state transitions of learners with respect to their progress or non-progress. It constructs automatically statecharts diagrams to keep track of cognitive state transitions. The system consults CoStaT and provides individualized navigation and advising to learners.

- **FuzKSD module**: It is responsible for identifying and updating the student's knowledge level of all the concepts of the knowledge domain. It models automatically the learning or forgetting process of a student. The operation of FuzKSD is based on the FCM that are used to represent the dependencies among the domain concepts. It includes the fuzzy sets that are used to represent the students' knowledge level and the rule-based fuzzy mechanism. This mechanism is triggered after a change has occurred on the student's knowledge level of a domain concept and updates the student's knowledge level of all related concepts with this concept. FuzKSD updates the overlay student model and allows each individual learner to complete the e-learning course at his/her own pace.

- **Tutor module**: It controls the tutoring process and makes appropriate decisions about the instructional model. The decisions are based on the information based on the information provided by the other components of the system. The tutor module has to be informed of the progress and misconceptions of a student and provide individualized instruction, support and advice, following pedagogical strategies, rules and facts.

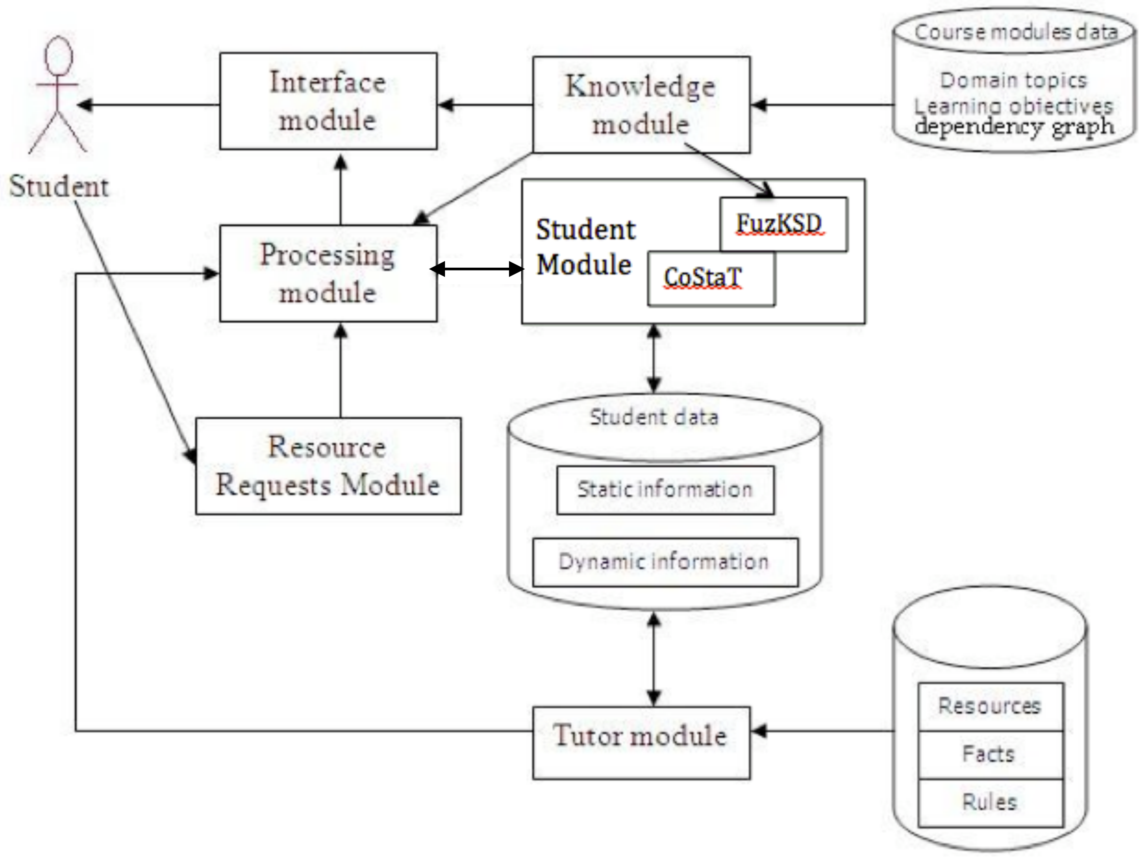


Figure 41: The architecture of ELaC

2. ELaC'S Knowledge domain

The knowledge domain of ELaC is the programming language ‘C’. The content is provided in various formats, mainly in html (including images), word documents and pdf files. The aim of the presented e-training application is not only to teach learners the principles and structures of the programming language ‘C’, but also to teach students the logic of programming as well as algorithms, like calculating sums, averages and maximums or minimums. Thereby, the learning material is decomposed in domain concepts which concern declarations of variables and constants, expressions and operators, input and output expressions, the sequential execution of a program, the if, if-else and if-else if statements, the iteration statements (for loop, while loop, do...while loop), sorting and searching algorithms, arrays, functions. The domain concepts that constitute the learning material are presented in table 17. They are organized into groups, which represent the sections of the learning material. The sections of the learning material are presented to the learner in sequence. In particular, section 1 and 2 are presented first, section 3 follows, then the content of section 4 is delivered e.t.c.

The order in which each section has to be taught is depicted in a hierarchical tree. The hierarchy of this tree depicts, also, the difficulty level of the domain concepts of the learning material. The creation of the hierarchy is based on the knowledge domain, on the logic of programming languages and on the dependencies that exist among the components of a programming language. For example, the teaching of variables and operands proceeds to the teaching of the if statement and the learning of a sorting algorithm presupposes the existence of knowledge of selection and iteration statements. The set of the domain concepts that have to be taught to a student each time depends on the knowledge level of the student. The domain concepts that belong to an upper level have to be taught only if the student has learned all the domain concepts of all the lower levels. For example, the teaching of the ‘for loop’ statement proceeds to the teaching of the arrays and the teaching of a searching algorithm presupposes the existence of knowledge of selection and iteration structures.

In particular, the knowledge domain of ELaC is organized into an hierarchical tree, with domain concepts as intermediate nodes and learning objectives as leaf nodes (Kumar, 2005), following the structure of the tree that was presented at Part B, section 3 (Fig. 35). Each topic has its learning objectives that are classified according to Bloom's taxonomy (1956). Learning objectives determine the concepts that must be understood and learned in each chapter. For example, the learning objectives for the topic of variables are:

- To learn the types of variables and understand their differences and how to use them
- To learn how to declare variables
- To learn how to select the appropriate type of variable each time
- To learn the rules about variables' names
- To learn to use variables saving memory

The hierarchy of the domain concepts of the learning material of ELaC is depicted in figure 42 and figure 43.

Table 17: Learning material

1. Basics	1.1. constants & variables	5. Iteration Structure Unknown no of loops	5.1. while statement
	1.2. assignment statement		5.2. calculating sum in a while loop
	1.3. arithmetic operators		5.3. counting in a while loop
	1.4. comparative operators		5.4. calculating avgr in a while loop
	1.5. logical operators		5.5. calculating max/min in a while loop
	1.6. mathematical functions		5.6. do...while statement
	1.7. input-output statements		
2. Sequence structure	2.1. a simple program structure	6. Arrays	6.1 one-dimensional arrays
			6.2. searching
3.Conditional Structures	3.1. if statement		6.3. sorting
	3.2. if...else if 3.2.1 methodology of finding max/min		6.4. two-dimensional arrays
	3.3. nested if		6.5. processing per row
			6.6. processing per column
4.Iteration Structure Concrete no of loops	4.1. for statement		6.7. processing of diagonals
	4.2. calculating sum in a for loop	7.Sub-programming	7.1. functions
	4.3. counting in a for loop		
	4.4. calculating avgr in a for loop		
	4.5. calculating max/min in a for loop		

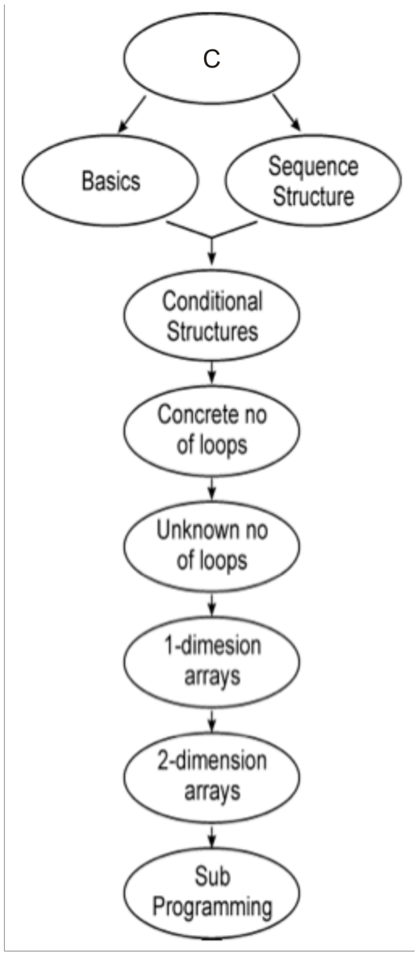


Figure 42: Hierarchical structure of the learning material

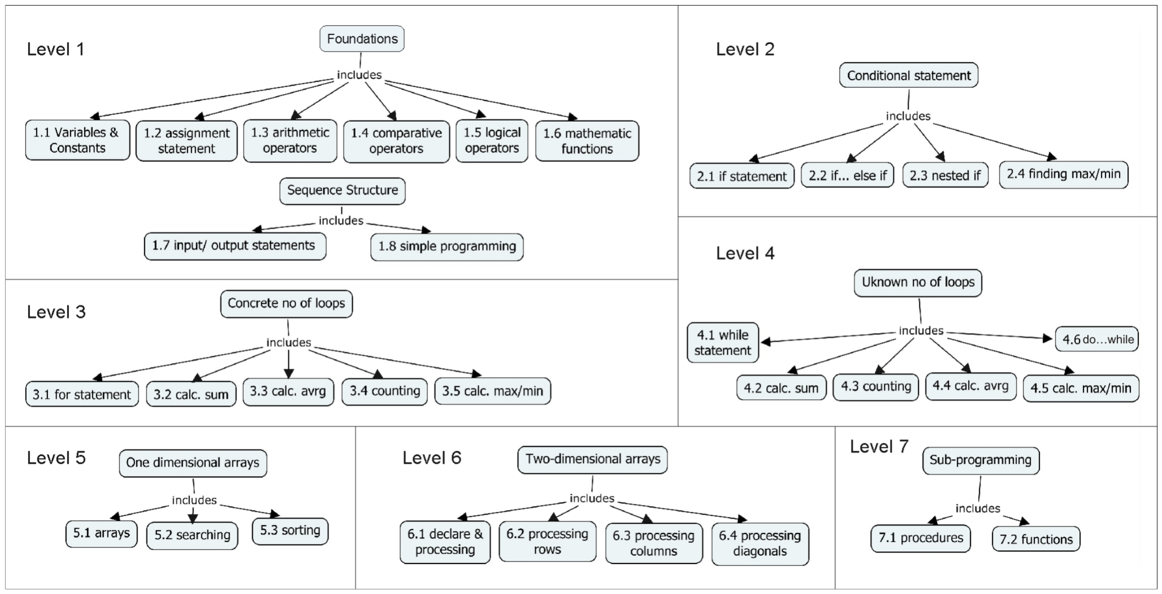


Figure 43: Analysis of the nodes of the hierarchical structure

The hierarchical structure is combined with FCMs. The latter are used to represent the dependencies that exist among the knowledge level of the domain concepts of the knowledge domain. Particularly, they represent the fact that the knowledge level of a domain concept is increased when the knowledge level of a related topic improves, as well as the fact that the knowledge level of a domain concept is decreased when the knowledge level of a depended topic is not satisfactory. For example, if a learner excels at ‘calculating a sum in a “for” loop’, then s/he will have a high knowledge level at ‘calculating a sum in a “while” loop’, too. But, if s/he has a lack of knowledge on how to calculate an average in a “while” loop, then s/he will have, also, poor knowledge on counting and calculating a sum in a “while” loop. The nodes of the FCM depict the domain concepts of the learning material, the directed arcs which connect the related domain concepts represent the causal relationships and the values which are labeled on arcs depict the degree at which the knowledge level of a domain concept is affected regarding the knowledge level of its related domain concepts (Fig. 44).

Changes on the knowledge level of C_i may affect the knowledge level of C_j in a different degree than changes on the knowledge level of C_j affect the knowledge level of C_i. For example, in the field of algorithms, if a learner knows how to calculate an average in an iterative structure with concrete number of loops, it implies that s/he knows also how to calculate a sum in an iterative structure with no concrete number of loops, since calculating an average incorporates calculating the corresponding sum. Similarly, if a student excels at calculating a sum in an iterative structure with no concrete number of loops, it implies that s/he knows partly how to calculate an average in an iterative structure with concrete number of loops, since calculating the sum is the base for calculating an average. However, these two domain concepts do not affect each other in the same degree. The knowledge level of ‘calculating an average in an iterative structure with concrete number of loops’ affects 100% (the value of the corresponding arc is 1) the knowledge level of ‘calculating a sum in an iterative structure with no concrete number of loops’, while the knowledge level of ‘calculating a sum in an iterative structure with no concrete number of loops’ affects 80% (the value of the corresponding arc is 0.8) the knowledge level of ‘calculating an average in an iterative structure with concrete number of loops’.

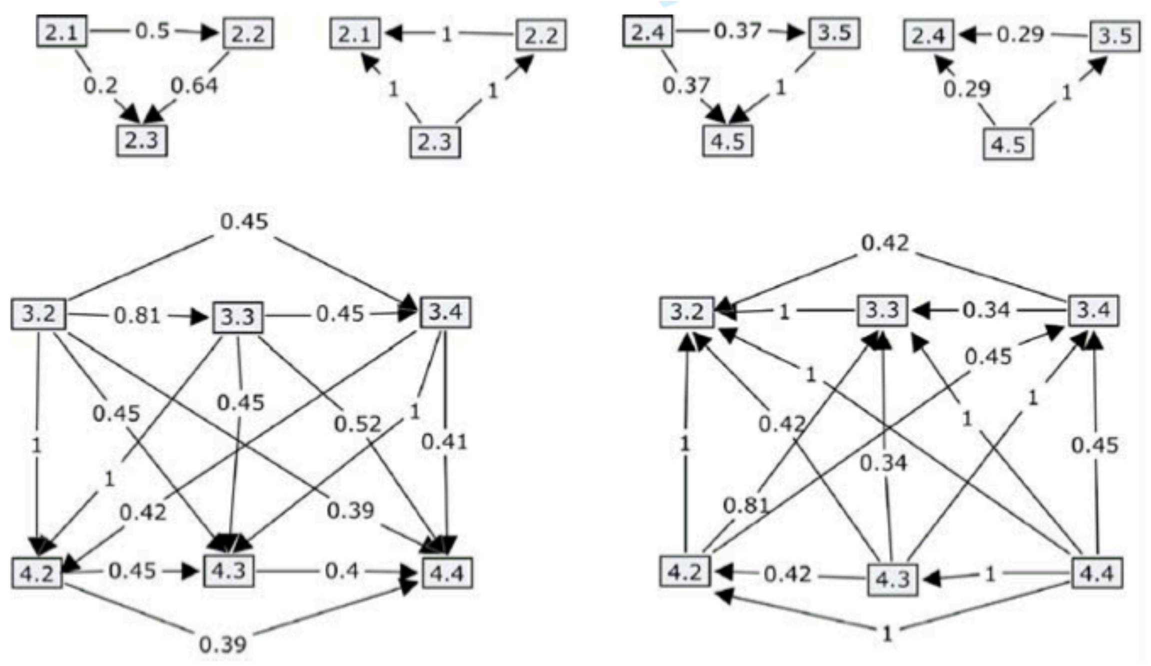


Figure 44: Fuzzy Cognitive Maps of ELaC

The value 1 on the directed arc that connects two dependent domain concepts of the FCMs of ELaC implies that if a learner knows a domain concept of a section, s/he may know a related concept of another section at the same degree. For example, if a learner has been tested and found to have known the “for” loop and the “while” loop and this learner knows how to calculate sum in a “for” loop, s/he will also know how to calculate sum in a “while” loop, since the methodology is the same.

Experts of the knowledge domain have defined the values on the directed arcs of the FCMS of ELaC that represent the degree of dependency between the domain concepts of the learning material. In particular, ten professors of computer programming, whose experience counts twelve years at least, are responsible for the definition and structure of the knowledge domain. They were asked to define, empirically, the dependencies among the domain concepts that are presented in table 17, as well as their "strength of impact" on each other. The professors' experience on teaching computer programming, in combination with the fact that teachers' experience and notions define the educational process and instruction, confirm the accuracy of dependencies among the concepts.

The two-dimensional matrix of ELaC, which is used for a better computation of the knowledge dependencies, is depicted in table 18, which corresponds to the FCM that is depicted in figure 45. The student model uses the knowledge dependencies between the domain concepts of the learning material, as well as the "strength of impact" on each other in order to recognize the possible increase or decrease of the learner's knowledge. This approach of the knowledge domain representation allows to ElaC to deliver the learning material to each individual learner dynamically taking into account her/his learning different learning pace.

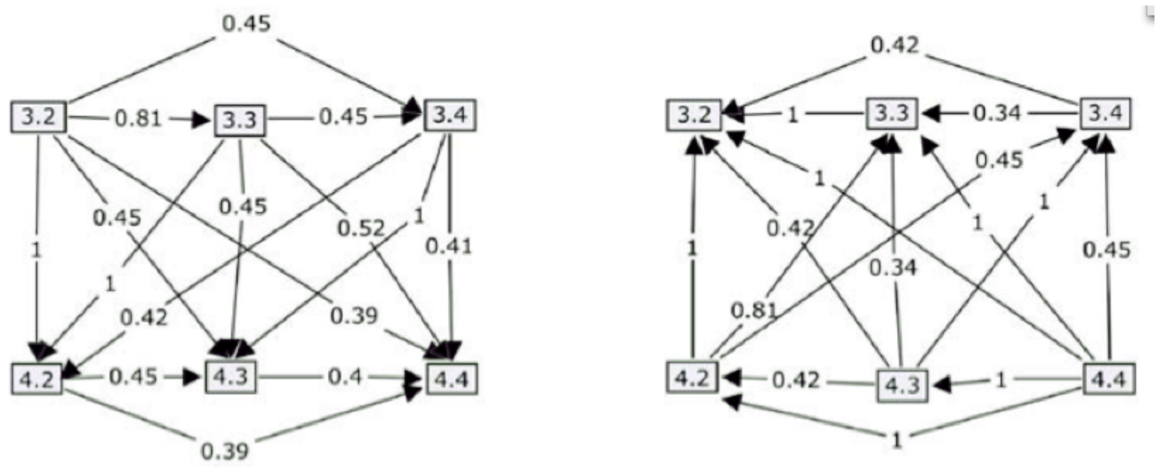


Figure 45: Example of knowledge dependencies between domain concepts of the learning material

Table 18: Matrix with knowledge dependencies between the programming structures

	3.2 Calc. sum in a for loop	3.3 Calc. avrg in a for loop	3.4 Counting in a for loop	4.2 Calc. sum in a while loop	4.4 Calc. avrg in a while loop	4.3 Counting in a while loop
3.2 Calc. sum in a for loop	0	0.81	0.45	1	0.39	0.45
3.3 Calc. avrg in a for loop	1	0	0.45	1	0.52	0.45
3.4 Counting in a for loop	0.42	0.34	0	0.42	0.41	1
4.2 Calc. sum in a while loop	1	0.81	0.45	0	0.39	0.45
4.4 Calc. avrg in a while loop	1	1	0.45	1	0	1
4.3 Counting in a while loop	0.42	0.34	1	0.42	0.41	0

3. ELaC's Student Model

ELaC incorporates a student model responsible for identifying and updating the student's knowledge level, taking the different pace of learning of each individual learner into account. In particular, the student model retains static information about each student, such as her/his previous experience on computer programming and the programming languages that s/he already knows. It also retains dynamic information, such as errors, misconceptions and progress, which is inferred by the system during the learner's interaction with it. In particular, the system uses a two-layered student model (Fig. 46), which combines a qualitative uncertainty-based weighted overlay model with a three-dimensional stereotype model and a rule-based fuzzy mechanism. The overlay model represents the user knowledge, the first dimension of our stereotype model concerns the knowledge level of the student; the second dimension concerns the type of programming errors (semantics or syntactic); and the third concerns previous knowledge of the student on other programming languages.

The overlay technique was chosen to model the learner's knowledge due to the fact that it is the most preferred technique for representing the student's mastery on the knowledge domain. In ELaC's overlay model a qualitative value ('unknown', 'Insufficient known', 'known', 'learned') combined with a percentage from 0 to 100% that points the weight of a qualitative value for a concept is used. For example, '85% Known C_1', means that the concept C_1 is 85% known. Figure 47 depicts an example of the ElaC's overlay model. The concepts, which belong to the subset of the domain model that the learner knows, are cycled with a bold red line.

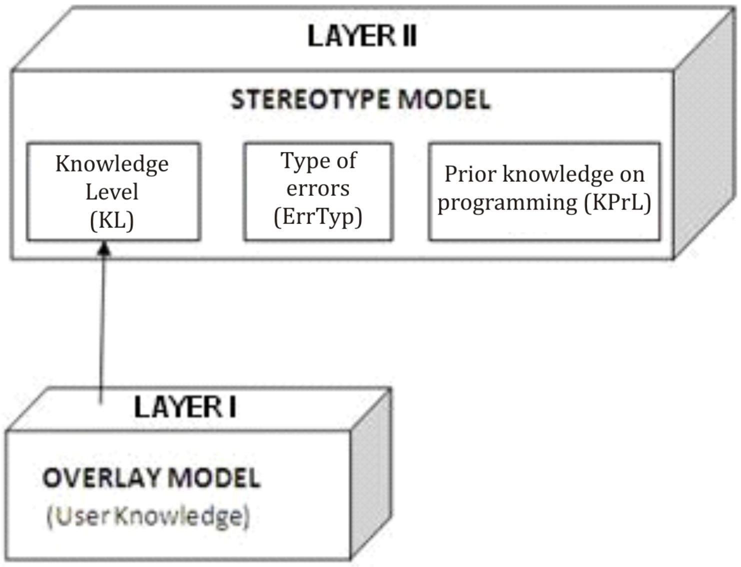


Figure 46: Student model of ELaC

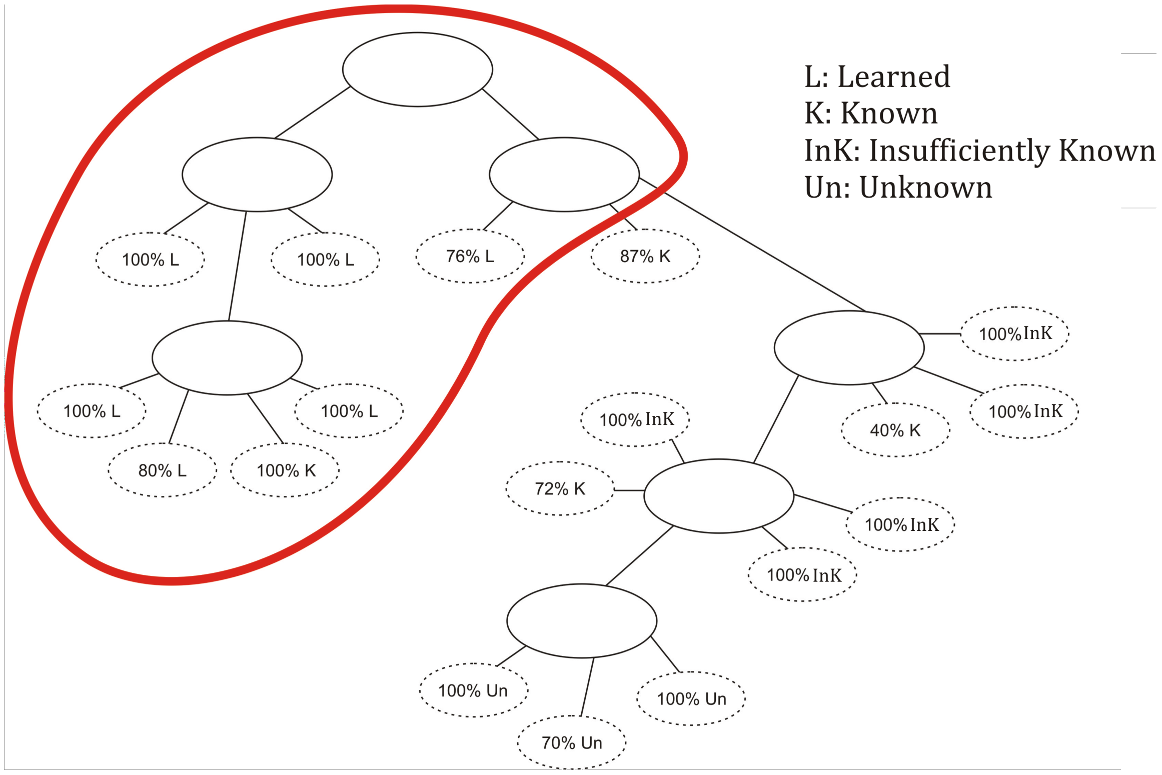


Figure 47: ELaC's qualitative uncertainty-based weighted overlay model

The first dimension of the third-dimensions stereotype model concerns the knowledge level of the learner. It is defined as the KL stereotype and its value represents the expertise of the learner in the algorithms and the programming language ‘C’. The value of the KL stereotype varies from “novice” users, who do not have a structural knowledge of programming and are unable to give an acceptable answer in most cases, to “expert” users, who are able to select, use and combine programming structures creating complex programs.

For the definition of these stereotypes the conceptual framework for analyzing students’ knowledge of programming that was developed by McGill and Volet (1997) and the evaluation method of knowledge of programming that was developed by deRaadt (2007) have been advised. McGill and Violet discern three knowledge types in the view of cognitive psychology: declarative (the basic knowledge of an object), procedural (how to use declarative knowledge for problem solving and decision making), strategic (upper knowledge level), and three knowledge types in the view of educational research: syntactic (basic knowledge), conceptual (be able to combine knowledge, analytical thought) and strategic (integrated knowledge). De Raadt suggests five knowledge levels:

- **No answer:** no knowledge
- **Pre-structural:** substantial lack of knowledge
- **One-structural:** ability to describe a part of code
- **Multi-structural:** ability to describe a program line-line
- **Relational:** ability to describe the whole of a program.

Thus, the eight stereotypes, which are depicted in table 19, were defined for representing knowledge level.

Table 19: Stereotypes of the first dimension (KL stereotypes)

KL Stereotype	Knowledge type	Knowledge level according to de Raadt
Stereotype 1	no knowledge	Level 1
Stereotype 2	declarative – syntactic	Pre-structural
Stereotype 3	declarative – conceptual	One-structural
Stereotype 4	procedural – syntactic	One-structural
Stereotype 5	procedural – syntactic	Multi-structural
Stereotype 6	procedural – conceptual	Multi-structural
Stereotype 7	procedural – conceptual	Relational
Stereotype 8	strategic	Relational

A learner is classified to a KL stereotype category according to which chapters the learner knows and how well. Furthermore, the KL stereotype category to which the learner has been classified, gives information about the learning material that should be delivered to the learner. The value of the KL stereotype is related with the learning material as it is referred bellow:

Stereotype 1: novice learners. The content of section 1 & 2 is delivered to the learner.

Stereotype 2: s/he knows basics of the programming language C, as well as the sequence structure of programming (Section 1 & 2). The content of section 3 is delivered to the learner.

Stereotype 3: s/he knows basics of the programming language C, the sequence structure and the structures of choice (Sections 1, 2 & 3). The content of section 4 is delivered to the learner.

Stereotype 4: s/he knows basics of the programming language C, the sequence structure, the structures of choice and the iteration structure with concrete number of loops (Sections 1, 2, 3 & 4). The content of section 5 is delivered to the learner.

Stereotype 5: s/he knows basics of the programming language C, the sequence structure, the structures of choice and the iteration structures with concrete or unknown number of loops (Sections 1, 2, 3, 4 & 5). The first three concepts of section 6 are delivered to the learner.

Stereotype 6: s/he knows basics of the programming language C, the sequence structure, the structures of choice, the iteration structures and one-dimensional arrays (Sections 1, 2, 3, 4, 5, 6.1, 6.2 & 6.3). The rest content of section 6 is delivered to the learner.

Stereotype 7: s/he knows basics of the programming language C, the sequence structure, the structures of choice, the iteration structures and arrays (Sections 1, 2, 3, 4, 5 & 6). The content of section 7 is delivered to the learner.

Stereotype 8: expert learners. S/he knows all the learning material. No more content is delivered to the learner.

The relation of the eight knowledge levels of students with the learning material have been defined by the ten professors of computer programming, who have defined, also, the knowledge dependencies between the domain concepts of the learning material and their "strength of impact" on each other. A learner is assigned to a knowledge level according to her/his errors. The ten domain experts have also, defined the kind and percentage of errors that determine the learner's knowledge level.

However, defining the learner's knowledge level is not adequate in order to model individual students' needs and abilities. A learner has misconceptions and the system should detect and reason them. That is the reason for the definition of the 2nd-dimension of the stereotype model (ErrTyp). ErrTyp stereotype takes two values: prone to syntax errors and prone to logical errors. Syntax errors are recognized if they belong in one of the following categories: anagrammatism of commands' names, omission of the definition of data, using invalid command names etc. They, usually, indicate that the learner has not read carefully and has not known adequately the chapters that correspond to her /his knowledge level. Logical errors are usually errors of design and occur in case of misconceptions of the program and of the semantics and operation of the commands. They, usually, indicate that the learner has a difficulty in understanding the instructions and their logic.

Furthermore, the system should known if a learner has prior knowledge of another programming language in order to be able to distinguish if an error occurs due to non-learning or due to affecting by another language. This kind of information is derived by the 3rd-dimension of the stereotype model (KPrL). The stereotypes of KPrL are associated with other programming languages that the learner may already know. In particular, KPrL takes the following values: 'none', 'Basic', 'Pascal', 'Java'. If a learner does not know a programming language, KPrL takes the value "none". If a learner knows more than one programming language of the above, KPrL takes two or all these programming languages. So, it takes either the value “none”, or one or more values of the set {Basic, Pascal, Java}. In ELaC the programming languages that the learner already knows are restricted to Basic, Pascal and Java because they seem to be enough for the research scope of this work, but the system can be extended to include other languages.

The student model of ELaC is updated each time new information about the learner is required. New information about the learner is obtained each time s/he interacts with the system. More concretely, each time the learner interacts with the system, s/he completes a test, the results of which determine the learner's knowledge and update her/his overlay model. The second layer or the student model receives information from the overlay model and determines the stereotype category of knowledge level (KL) to which the learner should be classified (Fig. 48). The stereotype categories of the second and third dimension of our student model, to which the learner should be classified, are not affected by the information that is received by the overlay model. The stereotype category of the second dimension to which the learner belongs each time, is determined by the type of errors (logical or syntactic) that s/he does during the test. Also, the third dimension of our user model, which concerns the learner’s prior knowledge of programming, is determined by the learner during her/his registration to the system.

In particular, each time the learner completes a section, s/he is asked to complete a test in order to check her/his knowledge and progress. If s/he succeeds in the test, then s/he is transited to a next knowledge level and the value of KL is increased. Otherwise, if s/he fails, then s/he remains to the same knowledge level or s/he is returned to a previous knowledge level, according to her/his errors. In particular, if the learner's poor performance does not affect the knowledge level of other related domain concepts, which belong to a previous section, then the value of KL remains the same, otherwise the value of KL is reduced. It is remarkable that in the intermediate, advanced and expert levels, the tests become more complex including and combining elements of all the domain concepts that a learner has been already taught. For example, an exercise with a sorting algorithm requires knowledge on variable declaration, selection and iteration statements and arrays. Consequently, if the learner makes errors that correspond to concepts of previous section, then the system infers that s/he has forgot something from previous sections of the learning material. In particular, if the learner makes errors which consider concepts of previous knowledge level, then the system checks the value of the KPrL of the learner's student model to be informed about the programming languages that s/he knows. If it is “none”, then the system classifies the learner to a previous knowledge level reducing the value of KL. Otherwise, ElaC consults the PLS_Lib component, which has a collection of the notations that are used by several programming languages, to infer if the errors were made due to the learner's knowledge on another programming language (for example s/he knows Pascal and writes an assignment statement using “:=” rather than “=”). If this is the cause, then the system does not classify the learner to a previous knowledge level, but it points out the error. A remarkable fact noticed during the use of ElaC, was that a significant percentage of the syntax errors are associated to language particularities in their notations (Java: 3%, Pascal: 23%, Basic: 17%). Some examples of such errors are presented in table 20.

Table 20: Examples of errors associated to language particularities for learners of "C"

	Types of variable	**Assignment operator**	**Arithmetic operators**	**Comparative operators**	**Logical operators**
Java	Use of boolean type				
Pascal	• Use of Boolean type • Use of the keyword Integer for defining the integer variables • Use of the keyword Real for defining the real variables	Use of the :=	• Use of div for the integer result of a division • Use of mod for the remainder of a division	• Use of = for equality • Use of <> for inequality	• Use of and for conjunction • Use of or for disjunction • Use of not for negation
Basic	• Use of Boolean type • Use of the keyword Integer for defining the integer variables • Use of the keyword Decimal for defining the real variables		• Use of ^ for exponentiation • Use of mod for the remainder of a division	• Use of = for equality • Use of <> for inequality	• Use of and for conjunction • Use of or for disjunction • Use of not for negation

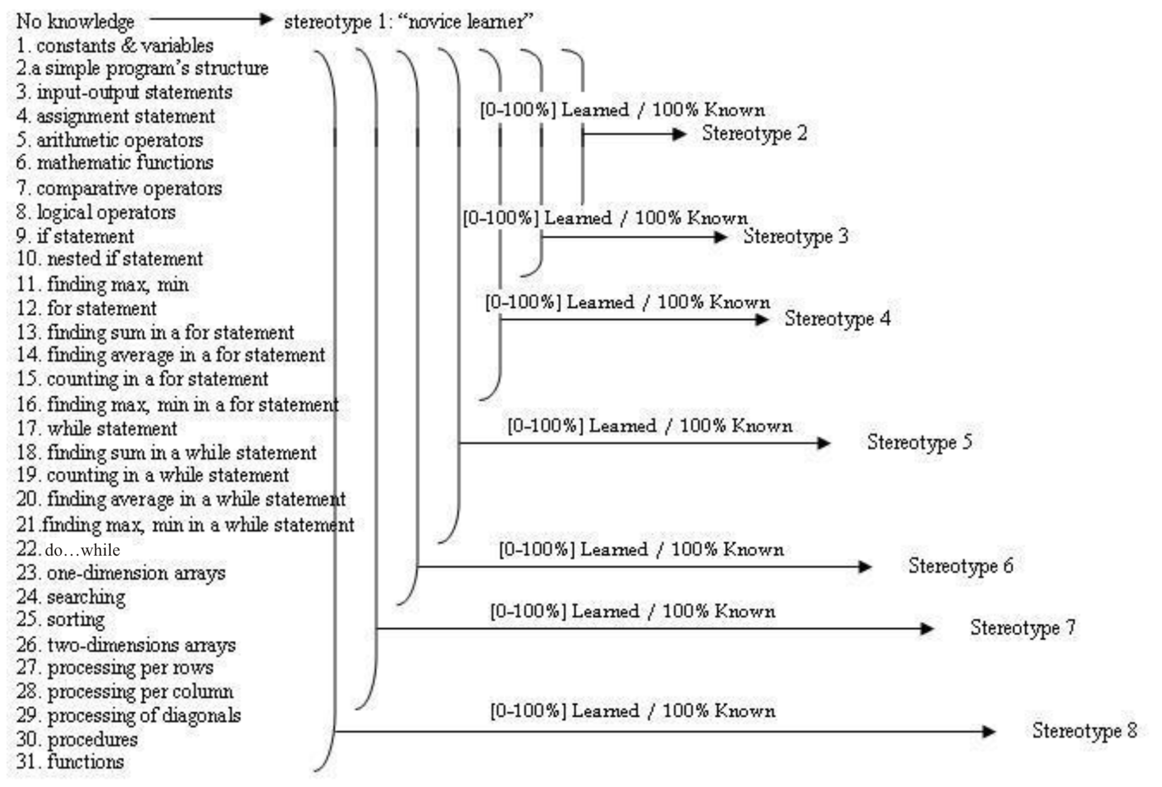

No knowledge
stereotype 1: "novice learner"
1. constants & variables
2.a simple program's structure
3. input-output statements
4. assignment statement
5. arithmetic operators
6. mathematic functions
7. comparative operators
8. logical operators
9. if statement
10. nested if statement
11. finding max, min
12. for statement
13. finding sum in a for statement
14. finding average in a for statement
15. counting in a for statement
16. finding max, min in a for statement
17. while statement
18. finding sum in a while statement
19. counting in a while statement
20. finding average in a while statement
21.finding max, min in a while statement
22. do…while
23. one-dimension arrays
24. searching
25. sorting
26. two-dimensions arrays
27. processing per rows
28. processing per column
29. processing of diagonals
30. procedures
31. functions
[0-100%] Learned / 100% Known
Stereotype 2
[0-100%] Learned / 100% Known
Stereotype 3
[0-100%] Learned / 100% Known
Stereotype 4
[0-100%] Learned / 100% Known
Stereotype 5
[0-100%] Learned / 100% Known
Stereotype 6
[0-100%] Learned / 100% Known
Stereotype 7
[0-100%] Learned / 100% Known
Stereotype 8

Figure 48: Relation between the overlay model and the KL stereotype

3.1 Triggering and Retraction

A system that applies the stereotype-based user modeling needs to define the stereotypes and their triggering and retraction conditions. Triggering conditions leads to the activation of a stereotype, while retraction conditions leads to the deactivation of a stereotype (Kay, 2000). Particularly, the stereotype M has a set of trigger conditions {tMi}, where each {tMi} is a Boolean expression based upon components {ci} of the student model, and a set of retraction conditions, {rMi}. The primary action of a stereotype is illustrated by Eq (1) and a stereotype is deactivated when any of the retraction conditions becomes true (Eq (2)).

$$if\ \exists i,\ tM_i = true \rightarrow active(M) \qquad (1)$$

$$if \exists\, j, rM_i = true \rightarrow not\ activate(M) \qquad (2)$$

In ELaC's student model a set of triggering and retraction conditions represents each learner's cognitive state with respect to her/his progress or no progress, to the kind of programming errors that s/he does and her/his previous knowledge on programming, while s/he interacts with the system. However, in ELaC, a set of trigger and retraction conditions is used, not only for activating and/or deactivating stereotypes, but also for activating and/or deactivating the qualitative value that represents the learner's knowledge of a concept. Therefore, the trigger conditions that activate a concept to be 'Unknown', 'Insufficient Known', 'Known' or 'Learned', as well as the retraction conditions that deactivate a concept to be one qualitative value of these are defined. The set of trigger and retraction conditions of our student model is depicted in the following table (table 21).

Table 21: Trigger and retraction conditions of ELaC

FACTS	
A: S/he knows already another programming language	
B: The state of domain concept C_i for the student is S1*	
C: The state of domain concept C_i for the student is S2*	
EVENTS	
E1: S/he answers exercises of all categories of her/his knowledge level right 80% at least	
E2: S/he does continuous errors in topics which are associated with previous knowledge level	
E3: S/he does above 20% syntactic errors	
E4: S/he does above 80% semantic errors	
E5: Her/his degree of success in the domain concept leads to S1 learning status	
E6: Her/his degree of success in the domain concept leads to S2 learning status	
E7: The learning status of the preceding domain becomes S1	
E8: The learning status of the preceding domain becomes S2	
E9: The learning status of the following domain becomes S1	
E10: The learning status of the following domain becomes S2	
E11: The degree of success in the following domain concept is lower than the dependency degree of it on C_i multiple with the degree of success in C_i.	
FIRST LAYER (overlay model)	
TRIGGER CONDITIONS	**RETRACTION CONDITIONS**
B=true AND [(E8=true OR E10=true)] OR E6=true → activate (S2)	B=true AND [(E8=true OR E10=true)] OR E6=true→ deactivate (S1)
C=true AND [(E7=true OR (E9=true AND E11=true)] OR E5=true →activate (S1)	C=true AND [(E7=true OR (E9=true AND E11=true)] OR E5=true→ deactivate (S2)
SECOND LAYER (3-dimensional stereotype model)	
TRIGGER CONDITIONS	**RETRACTION CONDITIONS**
E1=false→ activate (knowledge level N)	
E1=true→ activate (knowledge level N+1)	E1=true OR (E2=true AND E3=true AND A=false)→ deactivate (knowledge level N)
E2=true AND E3=true AND A=false→ activate (knowledge level N-1)	
E1=false AND E3=true → activate (prone to syntax errors)	E1=true OR (E1=false AND E3=true) → deactivate (prone to logical errors)
E1=false AND E4=true → activate (prone to logical errors)	E1=true OR (E1=false AND E4=true) → deactivate (prone to syntactic errors)
*S1,S2: learning status of a domain concept (Unknown, Insufficiently Known, Known, Learned) with S1<S2	

4. FuzKSD: The rule-based fuzzy logic module of ELaC

For the qualitative values of the qualitative uncertainty-based weighted overlay model, which represent the learner's knowledge level of each domain concept of the learning material, are used fuzzy sets. The following four fuzzy sets for representing students' knowledge level of a domain concept are defined:

- **Unknown (Un)**: the degree of success in the domain concept is from 0% to 60%.
- **Insufficiently Known (InK)**: the degree of success in the domain concept is from 55% to 75%.
- **Known (K)**: the degree of success in the domain concept is from 70% to 90%.
- **Learned (L)**: the degree of success in the domain concept is from 85% to 100%.

The membership functions for the four fuzzy sets are depicted in figure 49 and are the following:

$$\mu_{\mathrm{Un}}(x) = \begin{cases} 1, & x \le 55 \\ 1 - \dfrac{(x-55)}{5}, & 55 < x < 60 \\ 0, & x \ge 60 \end{cases}$$

$$\mu_{\mathrm{InK}} = \begin{cases} \dfrac{x-55}{5}, & 55 < x < 60 \\ 1, & 60 \le x \le 70 \\ 1 - \dfrac{x-70}{5}, & 70 < x < 75 \\ 0, & x \le 55 \quad or \quad x \ge 75 \end{cases}$$

$$\mu_{\mathrm{K}} = \begin{cases} \dfrac{x-70}{5}, & 70 < x < 75 \\ 1, & 75 \le x \le 85 \\ 1 - \dfrac{x-85}{5}, & 85 < x < 90 \\ 0, & x \le 70 \quad or \quad x \ge 90 \end{cases}$$

$$\mu_{\mathrm{L}}(x) = \begin{cases} \dfrac{x-85}{5}, & 85 < x < 90 \\ 1, & 90 \le x \le 100 \\ 0, & x \le 85 \end{cases}$$

where x is the student's degree of success in a domain concept.

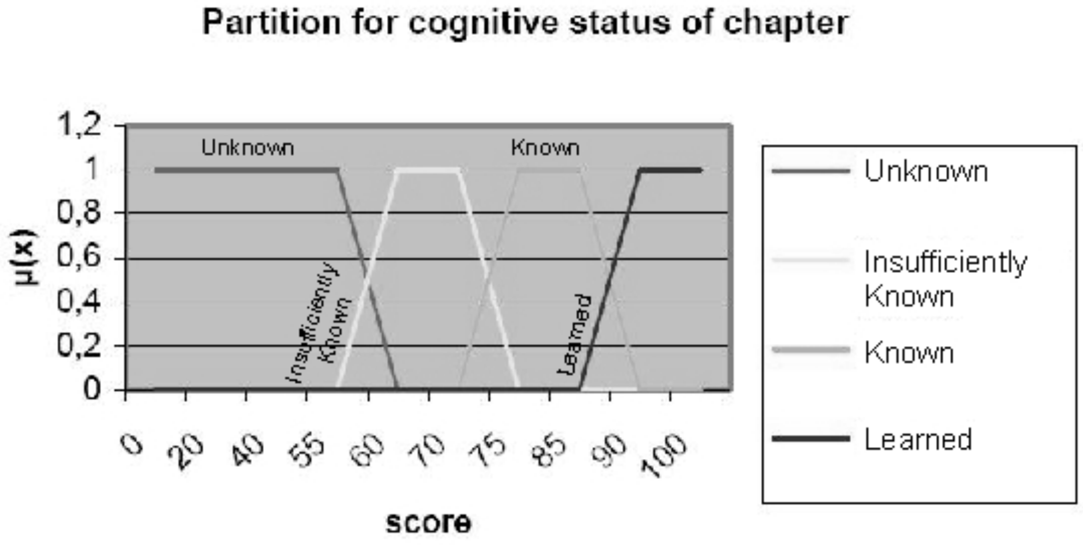


Figure 49: Partition for cognitive status of concept

The knowledge level of a domain concept changes in a continuous way. Meaning that the knowledge level of a domain concept usually passes from the unknown state to the insufficiently known state, from the insufficiently known state to the known state, and from the known state to the learned state.
The following expressions stand:

μ_{Un}, μ_{InK}, μ_K, μ_L € [0,1]

$\mu_{Un}+\mu_{InK}+\mu_K+\mu_L=1$

if $\mu_{Un}>0$, then $\mu_K=\mu_L=0$

if $\mu_{InK}>0$, then $\mu_L=0$

if $\mu_K>0$, then $\mu_{Un}=0$

if $\mu_L>0$, then $\mu_{Un}=\mu_{InK}=0$

To be more specific, lets x=72% is the degree of success of student A in the domain concept "Logical operators". According to the membership functions of the proposed fuzzy logic model, the following values arise: $\mu_{Un}=0$, $\mu_{InK}=0.6$, $\mu_K=0.4$, $\mu_L=0$. It means that the domain concept "Logical operators" is 60% insufficiently known and 40% known by student A. Therefore, a quadruplet (μ_{Un},

μ_{UK}, μ_K, μ_L) is used to express the student knowledge of a domain concept. When there is a dependency between two domain concepts, then the knowledge level of the one domain concept can affect the knowledge level of the other domain concept. More specifically, the following are taken into account:

- Considering the knowledge level of C_i, the knowledge level of its following domain concept C_j is increased or decreased. For example, a student who has been taught how to calculate a sum and how to count, s/he already knows the next concept, which concerns how to calculate an average in a loop. Also, if the system had concerned that the domain concept $C_{3.3}$: "Calculating an average in a 'for' loop" is learned by student A, and s/he is doing errors in its prerequisite domain concept $C_{3.2}$: "Calculating a sum in a 'for' loop", then the knowledge level of $C_{3.3}$ will be decreased.

- Considering the knowledge level of C_j, the knowledge level of its prerequisite domain concept C_i is increased or decreased. For example, if a student knows 70% the chapter "calculating a sum in a 'for' loop" and s/he is examined in the chapter "calculating a sum in a 'while' loop" and succeeds 95%, then the student's knowledge level of the chapter "calculating a sum in a 'for' loop" has to become higher than it is. Also, if the domain concept $C_{2.4}$: "Finding max, min" is considered as known for student A, and s/he is being examined in the domain concept $C_{3.5}$: "Finding max, min in a 'for' loop", and s/he is making 40% errors, then the domain concept $C_{2.4}$ will become insufficiently known.

The values of $\mu_D(C_i, C_j)$ and $\mu_D(C_i, C_j)$, which are the values of the arcs in the FCMs of the knowledge domain, have been determined by experts on programming, as it has been referred at the section 2. Particularly, ten professors of computer programming, whose experience was at least twelve years, were responsible for the definition and structure of the knowledge domain. They were asked to define, empirically, the dependencies among the domain concepts, as well as their "strength of impact" on each other that are presented in table 22.

Table 22: Values of the membership function of dependencies

C_i	Cj	$\mu_D(C_i,C_j)$	$\mu_D(C_j,C_i)$	C_i	Cj	$\mu_D(C_i,C_j)$	$\mu_D(C_j,C_i)$
3.1	**3.2**	0.50	1.00	**4.4**	**5.4**	0.52	1.00
3.1	**3.3**	0.20	1.00	**4.3**	**5.2**	0.42	0.45
3.2	**3.3**	0.64	1.00	**4.3**	**5.3**	1.00	1.00
3.2.1	**4.5**	0.37	0.29	**4.3**	**5.4**	0.41	0.45
3.2.1	**5.5**	0.37	0.29	**4.5**	**5.5**	1.00	1.00
4.2	**4.4**	0.81	1.00	**5.2**	**5.3**	0.45	0.42
4.2	**4.3**	0.45	0.42	**5.2**	**5.4**	0.81	1.00
4.2	**5.2**	1.00	1.00	**5.3**	**5.4**	0.41	1.00
4.2	**5.3**	0.45	0.42	**6.1**	**6.4**	0.43	0.51
4.2	**5.4**	0.39	1.00	**6.4**	**6.5**	0.33	0.99
4.3	**4.4**	0.34	0.45	**6.4**	**6.6**	0.33	0.99
4.4	**5.2**	1.00	0.81	**6.4**	**6.7**	0.27	0.78
4.4	**5.3**	0.45	0.34	**6.5**	**6.6**	0.77	0.77

The rules of the novel rule-based fuzzy module that were described in part B, section 5, are defined for ELaC as it is described below (where KL( ) denotes the "Knowledge Level of"):

❖ **Based on updates of the KL(C_i), the KL(C_j) is improved according to (where FS_X,FS_Y denote knowledge levels with FS_X<FS_Y):**

R1: If KL(C_j)=FS_X and KL(C_i)=FS_X, then KL(C_j)= FS_X with

$$\mu_{\mathrm{FS_X}}(C_j) = \max[\mu_{\mathrm{FS_X}}(C_j), \mu_{\mathrm{FS_X}}(C_i) * \mu_{\mathrm{D}}(C_i, C_j)]$$

R2: If KL(C_j)= FS_X and KL(C_i)=FS_Y, then KL(C_j)=FS_Y with

$$\mu_{\mathrm{FS_y}}(C_j) = \mu_{\mathrm{FS_y}}(C_i) * \mu_{\mathrm{D}}(C_i, C_j)]$$

❖ **Based on updates of the KL(Ci), the KL(Cj) is decreased according to:**

R3: If KL(Cj) = 100% Learned, then it does not change.

R4: If KL(C_j) = FS_x and KL(C_i) = Un, then KL(C_j) = Un with

$$\mu_{\mathrm{Un}}(C_j) = \mu_{\mathrm{Un}}(C_i) * \mu_{\mathrm{D}}(C_i, C_j)$$

FS_x is knowledge level with FS_x>Unknown.

R5: If KL(C_j) = FS_x and KL(C_i) = InK and if $\mu_D(C_i,C_j)$ = 1, then KL(C_j) = InK with $\mu_{InK}(C_j)$=1, else If KL(C_j) = FS_x and KL(C_i) = InK and $\mu_D(C_i,C_j) \neq 1$, then KL(C_j)=K with

$$\mu_{\mathrm{K}}(C_j) = 1 - \mu_{\mathrm{InK}}(C_j) = 1 - \mu_{\mathrm{InK}}(C_i) * \mu_{\mathrm{D}}(C_i, C_j)$$

FS_x is knowledge level with FS_x>Insufficiently Known.

R6: If KL(C_j) = L with $\mu_L(C_j) < 1$ and KL(C_i) = K, then KL(C_j) = K with

$$\mu_{\mathrm{K}}(C_j) = \mu_{\mathrm{K}}(C_i) * \mu_{\mathrm{D}}(C_i, C_j)$$

- **Based on updates of the KL(Cj), the KL(Ci) is improved according to (where FSX,FSY denote knowledge levels with FSX<FSY):**

R7: If KL(C_i)= FS_x and KL(C_j)=FS_x, then KL(C_i)= FS_x with

$$\mu_{\mathrm{FS_x}}(C_i) = \max[\mu_{\mathrm{FS_x}}(C_i), \mu_{\mathrm{FS_x}}(C_j) * \mu_{\mathrm{D}}(C_j, C_i)]$$

R8: If KL(C_i)= FS_x and KL(C_j)=FS_y, then KL(C_i)= FS_y with

$$\mu_{\mathrm{FS_y}}(C_i) = \mu_{\mathrm{FS_y}}(C_j) * \mu_{\mathrm{D}}(C_j, C_i)]$$

- **Based on updates of the KL(Cj), the KL(Ci) is decreased according to:**

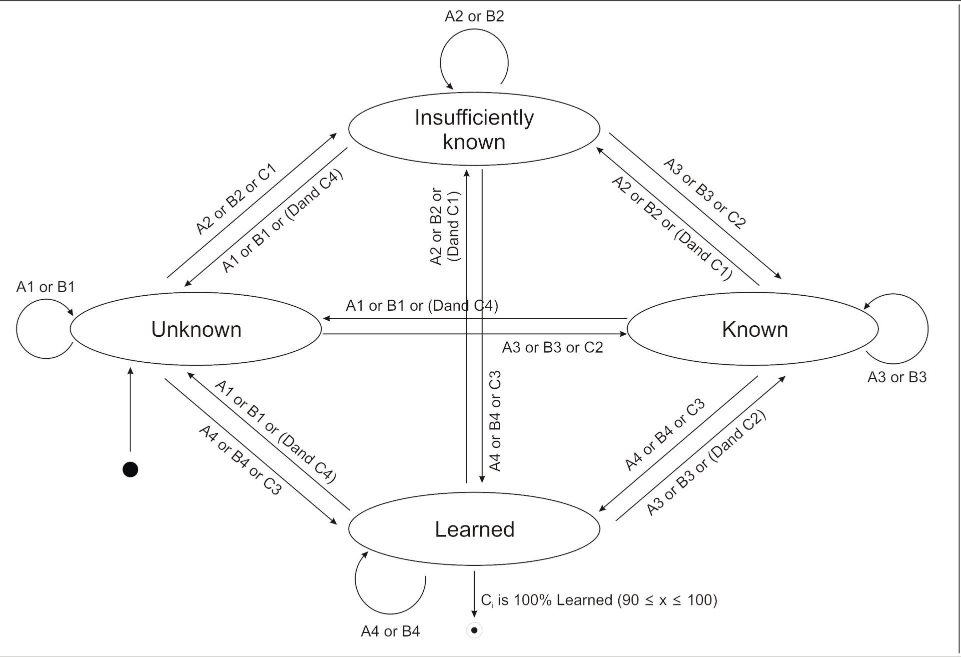
R9: If KL(C_i) = L with $\mu_L(C_i)$=1, then it does not change.

R10: The formula $x_i = \min[x_i, (1 - \mu_{\mathrm{D}}(C_i, C_j)) * x_i + x_j]$
where x_i and x_j are the student's degree of success in C_i and C_j respectively, is used. Then using the new x_i, the new quadruplet (μ_{Un}, μ_{InK}, μ_K, μ_L) for C_i is determined.

5. CoStaT: The cognitive state transitions module of ELaC

Taking into account all the above, it can be concluded that a domain concept passes through four states during the interaction of the user with the system (Fig. 50). Furthermore, a learner passes through several states during the learning process. S/he can learn or not, forget, assimilate or not etc. These states determine the progress of the user each time; they determine the transition from one stereotype to another.

Also, the transition from one stereotype to another can reveal the state of the user. Thus, at each interaction of a learner with educational application, the system has to decide which stereotypes have to be activated and which stereotypes have to be deactivated. A state-chart diagram can been used to show the sequence of a student's mind (Tretiakov et al., 2005). Figure 51 depicts the relation between a learner's cognitive state and the transitions among the stereotypes of the second layer of the ELaC's student model.



Ck Ci Cj

x: the degree of success in CI
y: the degree of success in Cj

A1: $x \le 60$
A2: $60 < x \le 70$
A3: $70 < x \le 85$
A4: $85 < x \le 100$

B1: The knowledge level of Ck becomes unknown
B2: The knowledge level of Ck becomes insufficiently known
B3: The knowledge level of Ck becomes known
B4: The knowledge level of Ck becomes learned

C1: The knowledge level of Cj becomes insufficiently known
C2: The knowledge level of Cj becomes known
C3: The knowledge level of Cj becomes learned
C4: The knowledge level of Cj becomes unknown

D: The degree of success in the Cj is lower than the dependency degree of it on Ci multiple with x

Figure 50: Transition through learning status of a domain concept

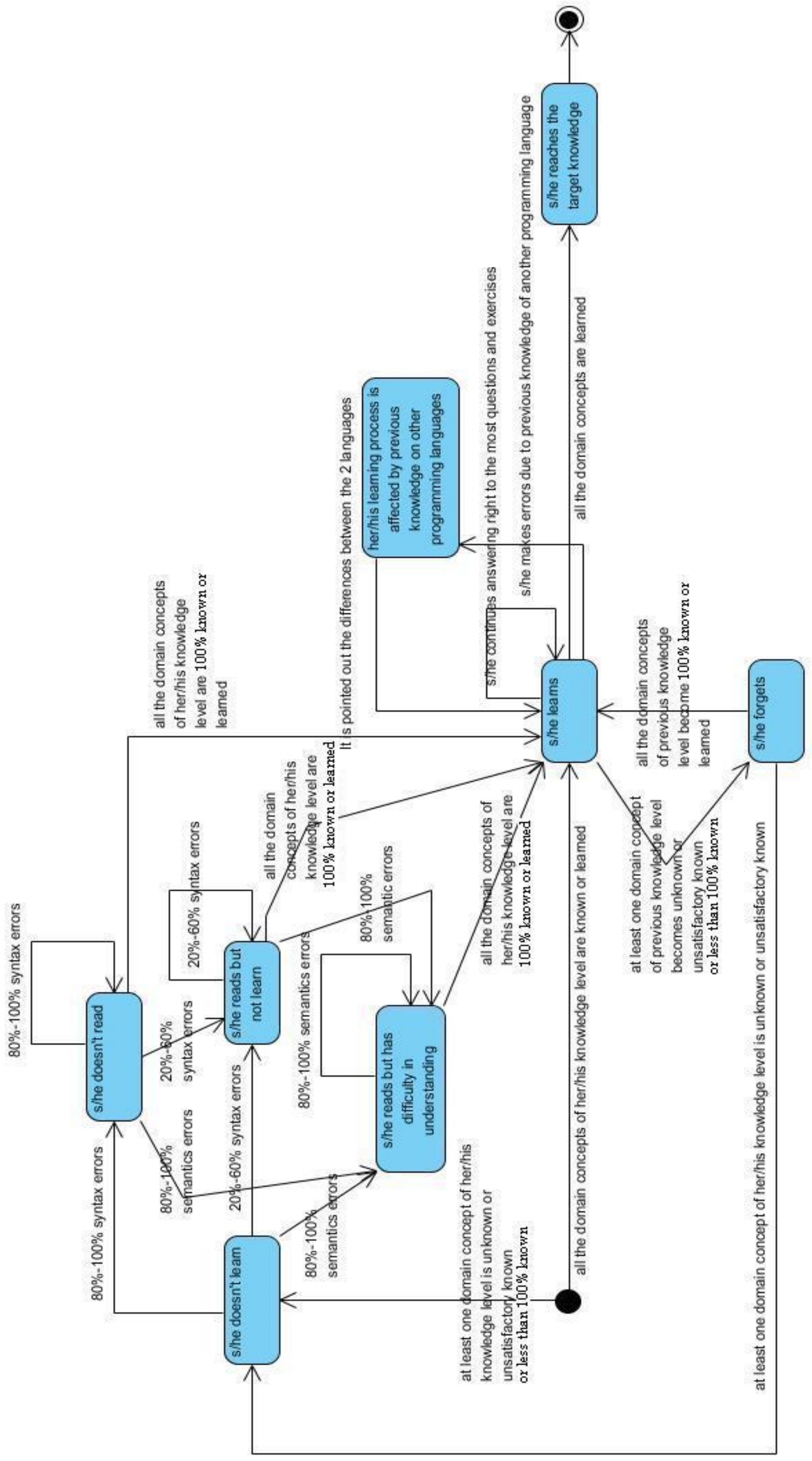

s/he doesn't learn
s/he doesn't read
s/he reads but not learn
s/he reads but has difficulty in understanding
s/he learns
s/he forgets
her/his learning process is affected by previous knowledge on other programming languages
s/he reaches the target knowledge
at least one domain concept of her/his knowledge level is unknown or unsatisfactory known or less than 100% known
all the domain concepts of her/his knowledge level are known or learned
80%-100% syntax errors
20%-60% syntax errors
80%-100% semantics errors
80%-100% semantic errors
all the domain concepts of her/his knowledge level are 100% known or learned
It is pointed out the differences between the 2 languages
s/he continues answering right to the most questions and exercises
s/he makes errors due to previous knowledge of another programming language
all the domain concepts are learned
at least one domain concept of previous knowledge level becomes unknown or unsatisfactory known or less than 100% known
all the domain concepts of previous knowledge level become 100% known or learned
at least one domain concept of her/his knowledge level is unknown or unsatisfactory known

Figure 51: Transition through stereotypes

6. Examples of operation

ELaC was used in a postgraduate program in the field of informatics at the University of Piraeus, in order to offer dynamically personalized e-training in computer programming and the language C. At the beginning, all learners were considered to be novices. At the next interactions, the system delivered to them the appropriate learning material for each individual student's needs by adapting instantly to the learner's individual learning pace. The system's adaptation decisions were based on the values of the student model. The KL value was determined by the results of the tests. There were two kinds of tests: (i) the tests that corresponded to each individual domain concept of the learning material (practice tests), (ii) the final tests that corresponded to the sections of the learning material (they included exercises of a variety of domain concepts). In particular, each time the learner read a domain concept, s/he had to complete a corresponding practice test. When, the learner had completed successfully all the practice tests of the domain concepts of a section (e.g. iterations with concrete number of loops, arrays, sub-programming), then s/he had to complete the final test of the section. If s/he succeeded to the final test, then s/he transited to a next section. Otherwise, s/he had advised to revise some domain concepts. Representative examples of the system's implementation follow.

Example 1:

Mary had learned the sections 1, 2 and 3 and she was taught the domain concepts of the section 4. The errors that she made are usually anagrammatism of commands' names or invalid symbolisms of operands or commands' names. Also, she had no previous knowledge on computer programming. So, the parameters of her student model had the following values: KL=3, **ErrTyp= "prone to syntax errors", KPrL= "none"**. The content of the section 4 was delivered to her and she completed the corresponding tests. She was examined in $C_{4.1}$: "for statement" and succeeded 92%. Then, she was examined in $C_{4.2}$: "calculating sum in a 'for' loop" and succeeded 95%. However, according to the "strength of impact" of the knowledge dependencies that exist between the domain concepts of the learning material (table 22, section 4), $C_{4.2}$ affects 45% the concept $C_{4.3}$, 81% the concept $C_{4.4}$, 45% the concept $C_{5.3}$, 100% the concept $C_{5.2}$, and 39% the concept $C_{5.4}$.

According to the rules over the fuzzy sets that are used by the FuzKSD the following occur:

- According to R_2, the domain $C_{4.3}$ becomes 45% Learned, as it is $\mu_L(C_{4.3}) = \mu_L(C_{4.2}) * \mu_D(C_{4.2}, C_{4.3}) = 1*0.45 = 0.45$. So, the quadruplet of the knowledge state of the concept $C_{4.3}$ is (0, 0, 0.55, 0.45).
- According to R_2, the domain $C_{4.4}$ becomes 81% Learned, as it is $\mu_L(C_{4.4}) = \mu_L(C_{4.2}) * \mu_D(C_{4.2}, C_{4.4}) = 1*0.81 = 0.81$. So, the quadruplet the knowledge state of the concept $C_{4.4}$ is (0, 0, 0.19, 0.81).
- According to R_2, the domain $C_{5.2}$ becomes 100% Learned, as it is $\mu_L(C_{5.2}) = \mu_L(C_{4.2}) * \mu_D(C_{4.2}, C_{5.2}) = 1*1 = 1$. So, the quadruplet of the knowledge state of the concept $C_{4.4}$ is (0, 0, 0, 1).
- According to R_2, the domain $C_{5.3}$ becomes 45% Learned, as it is $\mu_L(C_{5.3}) = \mu_L(C_{4.2}) * \mu_D(C_{4.2}, C_{5.3}) = 1*0.45 = 0.45$. So, the quadruplet of the knowledge state of the concept $C_{4.4}$ is (0, 0, 0.55, 0.45).
- According to R_2, the domain $C_{5.4}$ becomes 39% Learned, as it is $\mu_L(C_{5.4}) = \mu_L(C_{4.2}) * \mu_D(C_{4.2}, C_{5.4}) = 1*0.39 = 0.39$. So, the quadruplet of the knowledge state of the concept $C_{4.4}$ is (0, 0, 0.61, 0.39).

Therefore, the knowledge level of the related concepts changed as it is presented in table 23 (interaction II). As a consequence, the system did not advise Mary to read the domain concepts $C_{4.3}$, $C_{4.4}$, $C_{5.2}$, $C_{5.3}$ and $C_{5.4}$.

Then, she was examined in the domain concept $C_{4.5}$ and succeeded 86%. $C_{4.5}$ affects $C_{5.5}$ in a percentage of 100%. According to R_2, the domain $C_{5.5}$ becomes 20% Learned, as it is $\mu_L(C_{5.5}) = \mu_L(C_{4.5}) * \mu_D(C_{4.5}, C_{5.5}) = 0.2*1 = 0.2$. So, the quadruplet of the knowledge state of the concept $C_{5.5\ is}$ (0, 0, 0.80, 0.20).

When all the domain concepts of the section 4 had become learned, the value of the parameter KL of Mary's student model was increased and became 4. Now, the domain concepts of section 5 should be delivered to Mary. However, the domain concepts $C_{5.2}$, $C_{5.3}$, $C_{5.4}$ and $C_{5.5}$ are considered already learned; therefore Mary has to read only the domain concepts $C_{5.1}$ and $C_{5.6}$.

At last interactions, Mary made errors that are associated with the semantics and operation of the commands. Therefore the value of the ErrTyp stereotype changes and becomes “prone to logical errors”. This value indicates that Mary has a difficulty in understanding the instructions and their logic. As a result, the system decides to deliver her the learning material in a more explanatory way with more examples.

Table 23: Mary's progress

Concepts	**Interaction I**		**Interaction II**		**Interaction III**	
	(μ_{Un}, μ_{UK}, μ_K, μ_L)	**Cognitive state**	**(μ_{Un}, μ_{UK}, μ_K, μ_L)**	**Cognitive state**	**(μ_{Un}, μ_{UK}, μ_K, μ_L)**	**Cognitive state**
$C_{4.1}$	***(0, 0, 0, 1)***	***100% learned***	(0, 0, 0, 1)	100% learned	(0, 0, 0, 1)	100% learned
$C_{4.2}$	(1, 0, 0, 0)	100% unknown	***(0, 0, 0, 1)***	***100% learned***	(0, 0, 0, 1)	100% learned
$C_{4.3}$	(1, 0, 0, 0)	100% unknown	(0, 0, 0.55, 0.45)	45% learned	(0, 0, 0.55, 0.45)	45% learned
$C_{4.4}$	(1, 0, 0, 0)	100% unknown	(0, 0, 0.19, 0.81)	81% learned	(0, 0, 0.19, 0.81)	81% learned
$C_{4.5}$	(1, 0, 0, 0)	100% unknown	(1, 0, 0, 0)	100% unknown	***(0, 0, 0.8, 0.2)***	***20% learned***
$C_{5.1}$	(1, 0, 0, 0)	100% unknown	(1, 0, 0, 0)	100% unknown	(1, 0, 0, 0)	100% unknown
$C_{5.2}$	(1, 0, 0, 0)	100% unknown	(0, 0, 0, 1)	100% learned	(0, 0, 0, 1)	100% learned
$C_{5.3}$	(1, 0, 0, 0)	100% unknown	(0, 0, 0.55, 0.45)	45% learned	(0, 0, 0.55, 0.45)	45% learned
$C_{5.4}$	(1, 0, 0, 0)	100% unknown	(0, 0, 0.61, 0.39)	39% learned	(0, 0, 0.61, 0.39)	39% learned
$C_{5.5}$	(1, 0, 0, 0)	100% unknown	(1, 0, 0, 0)	100% unknown	(0, 0, 0.8, 0.2)	20% learned
$C_{5.6}$	(1, 0, 0, 0)	100% unknown	(1, 0, 0, 0)	100% unknown	(1, 0, 0, 0)	100% unknown

Example 2:

The parameters of Dimitris' student model had the following values: KL=6, **ErrTyp= “prone to syntax errors”, KPrL= “Pascal”**. After having completed a test which involved exercises on two-dimensional arrays, the system discovered that he made more than 40% errors on the assignment statement, and more specifically he used the symbol := rather than the symbol =. The system checked the values of the parameter **KPrL** and was informed that he already knows the programming language Pascal. Thus, it was assumed that Dimitris used the symbol := for assignment due to his previous knowledge on Pascal. So, the system' reaction was to stress the error, but it did not classify him to the knowledge level 1.

Example 3:

The parameters of Alexis' student model had the following values: KL=4, **ErrTyp= "prone to syntax errors", KPrL= "Basic"**. He has successfully finished the practice test of all the domain concepts of his KL stereotype. Then, he completed the final test that corresponded to his knowledge level. The system discovered that he made errors, which concerned the domain concept $C_{4.5}$. Also, ELaC consulted the PLS_Lib component and ascertained that the particular errors did not occur due to Alexis' previous knowledge on the programming language Basic. Particularly, he made 43% errors (the quadruplet of $C_{4.5}$ is (0.6, 0.4, 0, 0). However, $C_{4.5}$ affects at 29% the concept $C_{3.2.1}$ and 100% the concept $C_{5.5}$. According to the rules over the fuzzy sets that are used by the FuzKSD the following occur (table 24):

- According to R_{10} the knowledge state of the concept $C_{3.2.1}$ remains 80% Learned, as it is
  $x_{3.2.1} = \min[x_{3.2.1}, (1\text{-}\mu_D(C_{3.2.1}, C_{4.5}))*x_{3.2.1} + x_{4.5}] = \min[89, (1\text{-}0.37)*89 + 57] = 89$
  So, the quadruplet of the knowledge state of the concept $C_{3.2.1}$ is (0, 0, 0.2, 0.8).
- According to R_4, the domain $C_{5.5}$ becomes 60% Unknown, as it is $\mu_{Un}(C_{5.5}) = \mu_{Un}(C_{4.5})*\mu_D(C_{4.5}, C_{5.5}) = 0.6*1 = 0.6$. So, the quadruplet of the knowledge state of the concept $C_{5.5}$ is (0.6, 0.4, 0, 0).

As a result the system decided that Alexis needs to revise the domain concepts $C_{4.5}$ and $C_{5.5}$ and classified him to a previous knowledge level, reducing the value of KL, which became 3.

Table 24: Alexis' progress

Domain Concepts	**Learner's Knowledge**			
	Before (KL=4)		**After (KL=3)**	
	(μ_{Un}, μ_{UK}, μ_K, μ_L)	**Cognitive state**	**(μ_{Un}, μ_{UK}, μ_K, μ_L)**	**Cognitive state**
1.1 constants & variables	(0, 0, 0, 1)	100% Learned	(0, 0, 0, 1)	100% Learned
1.2 assignment statement	(0,0,0.08, 0.92)	92% Learned	(0,0,0.08, 0.92)	92% Learned
1.3 arithmetic operators	(0, 0, 0, 1)	100% Learned	(0, 0, 0, 1)	100% Learned
1.4 comparative operators	(0,0,0.08, 0.92)	92% Learned	(0,0,0.08, 0.92)	92% Learned
1.5 logical operators	(0, 0, 0, 1)	100% Learned	(0, 0, 0, 1)	100% Learned
1.6. mathematic functions	(0, 0, 0, 1)	100% Learned	(0, 0, 0, 1)	100% Learned
1.7 input-output statements	(0, 0, 0, 1)	100% Learned	(0, 0, 0, 1)	100% Learned
2.1 a simple program's structure	(0, 0, 0, 1)	100% Learned	(0, 0, 0, 1)	100% Learned
3.1 if statement	(0, 0, 0.15, 0.85)	85% Learned	(0, 0, 0.15, 0.85)	85% Learned
3.2 if…else if	(0, 0, 0.15, 0.85)	85% Learned	(0, 0, 0.15, 0.85)	85% Learned
3.2.1 finding max, min	***(0, 0, 0.2, 0.8)***	***80% Learned***	***(0, 0, 0.2, 0.8)***	***80% Learned***
3.3 nested if statement	(0, 0, 0.35, 0.65)	65% Learned	(0, 0, 0.35, 0.65)	65% Learned
4.1 for statement	(0, 0, 0, 1)	100% Learned	(0, 0, 0, 1)	100% Learned
4.2 calc. sum in a for loop	(0, 0, 0.4, 0.6)	60% Learned	(0, 0, 0.4, 0.6)	60% Learned
4.3 counting in a for loop	(0, 0, 0, 1)	100% Learned	(0, 0, 0, 1)	100% Learned
4.4 calc. avrg in a for loop	(0, 0, 0.51, 0.49)	49% Learned	(0, 0, 0.51, 0.49)	49% Learned
4.5 calc. max/min in a for loop	***(0, 0, 0.63, 0.37)***	***37% Learned***	***(0.6, 0.4, 0, 0)***	***60% Unknown***
5.1 while statement	(0, 0, 0.8, 0.2)	20% Learned	(0, 0, 0.8, 0.2)	20% Learned
5.2 calc. sum in a while loop	(0, 0, 0.4, 0.6)	60% Learned	(0, 0, 0.4, 0.6)	60% Learned
5.3 counting in a while loop	(0, 0, 0, 1)	100% Learned	(0, 0, 0, 1)	100% Learned
5.4 calc. avrg in a while loop	(0, 0, 0.59, 0.41)	41% Learned	(0, 0, 0.59, 0.41)	41% Learned
5.5 calc. max/min in a while loop	***(0, 0, 0.63, 0.37)***	***37% Learned***	***(0.6, 0.4, 0, 0)***	***60% Unknown***
5.6 do…until	(1, 0, 0, 0)	100% Unknown	(1, 0, 0, 0)	100% Unknown
6.1 one-dimension arrays	(1, 0, 0, 0)	100% Unknown	(1, 0, 0, 0)	100% Unknown
6.2 searching	(1, 0, 0, 0)	100% Unknown	(1, 0, 0, 0)	100% Unknown
6.3 sorting	(1, 0, 0, 0)	100% Unknown	(1, 0, 0, 0)	100% Unknown
6.4 two-dimensions arrays	(1, 0, 0, 0)	100% Unknown	(1, 0, 0, 0)	100% Unknown
6.5 processing per rows	(1, 0, 0, 0)	100% Unknown	(1, 0, 0, 0)	100% Unknown
6.6 processing per column	(1, 0, 0, 0)	100% Unknown	(1, 0, 0, 0)	100% Unknown
6.7 processing of diagonals	(1, 0, 0, 0)	100% Unknown	(1, 0, 0, 0)	100% Unknown
7.1 functions	(1, 0, 0, 0)	100% Unknown	(1, 0, 0, 0)	100% Unknown

Example 4:

The parameters of Helen's student model had the following values: KL=6, **ErrTyp= "prone to logical errors", KPrL= "Java"**. She is taking the practice test of the concept $C_{6.4}$: "Two dimensional-arrays" and s/he is making 27% errors. So, the degree of success x is 73. Therefore, the quadruplet, which expresses the student knowledge on the domain concept $C_{6.4}$, is (0, 0.4, 0.6, 0). That means that the specific knowledge domain becomes 40% Insufficiently Known and 60% Known for Helen. According to the FCMs, the knowledge level of this concept, affects the knowledge level of the domain concepts $C_{6.1}$: “One-dimension arrays”, $C_{6.5}$: “Working with rows in a two-dimensional array”, $C_{6.6}$: “Working with columns in a two-dimensional array” and $C_{6.7}$: “Working with diagonals in a two-dimensional array”. According to the rules over the fuzzy sets that are used by the FuzKSD the following occur:

- According to R_7, the domain $C_{6.1}$ remains 100% Known. So, the quadruplet of the knowledge state of the concept $C_{6.1}$ is (0, 0, 1, 0).
- According to R_1, the domain $C_{6.5}$ remains 23% Known, as it is $\mu_K(C_{6.5}) = \max[\mu_K(C_{6.5}), \mu_K(C_{6.4}) * \mu_D(C_{6.4}, C_{6.5})] = \max[0.23, 0.6 * 0.33] = 0.23$. So, the quadruplet of the knowledge state of the concept $C_{6.5}$ is (0, 0.77, 0.23, 0).
- According to R_1, the domain $C_{6.6}$ becomes 20% Known, as it is $\mu_K(C_{6.6}) = \max[\mu_K(C_{6.6}), \mu_K(C_{6.4}) * \mu_D(C_{6.4}, C_{6.6})] = \max[0.18, 0.6 * 0.33] = 0.2$. So, the quadruplet of the knowledge state of the concept $C_{6.6}$ is (0, 0.80, 0.20, 0).
- According to R_2, the domain $C_{6.7}$ becomes 16% Known, as it is $\mu_K(C_{6.7}) = \mu_K(C_{6.4}) * \mu_D(C_{6.4}, C_{6.7}) = 0.6 * 0.27 = 0.16$. So, the quadruplet of the knowledge state of the concept $C_{6.7}$ is (0, 0.84, 0.16, 0).

Thus, the system infers that Helen has learn the corresponding domain concepts of the learning material, but she has not assimilated them and it decides that Helen should remain in the same stereotype, which concerns the learner’s knowledge level, to read them again (table 25). Therefore the value of KL remains 6.

Table 25: Helen's progress

Domain Concepts	Learner's Knowledge			
	Before (KL=6)		After (KL=6)	
	(μ_{Un}, μ_{UK}, μ_K, μ_L)	Cognitive state	(μ_{Un}, μ_{UK}, μ_K, μ_L)	Cognitive state
1.1 constants & variables	(0, 0, 0, 1)	100% Learned	(0, 0, 0, 1)	100% Learned
1.2 assignment statement	(0,0,0.08, 0.92)	92% Learned	(0,0,0.08, 0.92)	92% Learned
1.3 arithmetic operators	(0, 0, 0, 1)	100% Learned	(0, 0, 0, 1)	100% Learned
1.4 comparative operators	(0, 0, 0, 1)	100% Learned	(0, 0, 0, 1)	100% Learned
1.5 logical operators	(0, 0, 0, 1)	100% Learned	(0, 0, 0, 1)	100% Learned
1.6. mathematic functions	(0, 0, 0, 1)	100% Learned	(0, 0, 0, 1)	100% Learned
1.7 input-output statements	(0, 0, 0, 1)	100% Learned	(0, 0, 0, 1)	100% Learned
2.1 a simple program's structure	(0, 0, 0, 1)	100% Learned	(0, 0, 0, 1)	100% Learned
3.1 if statement	(0, 0, 0.15, 0.85)	85% Learned	(0, 0, 0.15, 0.85)	85% Learned
3.2 if…else if	(0, 0, 0.15, 0.85)	85% Learned	(0, 0, 0.15, 0.85)	85% Learned
3.2.1 finding max, min	(0, 0, 0, 1)	100% Learned	(0, 0, 0, 1)	100% Learned
3.3 nested if statement	(0, 0, 0.35, 0.65)	65% Learned	(0, 0, 0.35, 0.65)	65% Learned
4.1 for statement	(0, 0, 0, 1)	100% Learned	(0, 0, 0, 1)	100% Learned
4.2 calc. sum in a for loop	(0, 0, 0, 1)	100% Learned	(0, 0, 0, 1)	100% Learned
4.3 counting in a for loop	(0, 0, 0.55, 0.45)	45% Learned	(0, 0, 0.55, 0.45)	45% Learned
4.4 calc. avrg in a for loop	(0, 0, 0.19, 0.81)	81% Learned	(0, 0, 0.19, 0.81)	81% Learned
4.5 calc. max/min in a for loop	(0, 0, 0.63, 0.37)	37% Learned	(0, 0, 0.63, 0.37)	37% Learned
5.1 while statement	(0, 0, 0.80, 0.20)	20% Learned	(0, 0, 0.80, 0.20)	20% Learned
5.2 calc. sum in a while loop	(0, 0, 0, 1)	100% Learned	(0, 0, 0, 1)	100% Learned
5.3 counting in a while loop	(0, 0, 0.61, 0.39)	39% Learned	(0, 0, 0.61, 0.39)	39% Learned
5.4 calc. avrg in a while loop	(0, 0, 0.55, 0.45)	45% Learned	(0, 0, 0.55, 0.45)	45% Learned
5.5 calc. max/min in a while loop	(0, 0, 0.63, 0.37)	37% Learned	(0, 0, 0.63, 0.37)	37% Learned
5.6 do…until	(0, 0, 1, 0)	100% Known	(0, 0, 1, 0)	100% Known
6.1 one-dimension arrays	***(0, 0, 1, 0)***	***100% Known***	***(0, 0, 1, 0)***	***100% Known***
6.2 searching	(0, 0, 0.20, 0.80)	80% Learned	(0, 0, 0.20, 0.80)	80% Learned
6.3 sorting	(0, 0, 0, 1)	100% Learned	(0, 0, 0, 1)	100% Learned
6.4 two-dimensions arrays	***(0, 0.57, 0.43, 0)***	***43% Known***	***(0, 0.40, 0.60, 0)***	***60% Known***
6.5 processing per rows	***(0, 0.77, 0.23, 0)***	***23% Known***	***(0, 0.77, 0.23, 0)***	***23% Known***
6.6 processing per column	***(0, 0.82, 0.18, 0)***	***18% Known***	***(0, 0.80, 0.20, 0)***	***20% Known***
6.7 processing of diagonals	***(1, 0, 0, 0)***	***100% Unknown***	***(0, 0.84, 0.16, 0)***	***16% Known***
7.1 functions	(1, 0, 0, 0)	100% Unknown	(1, 0, 0, 0)	100% Unknown

Example 5:

The parameters of Mary's student model had the following values: KL=3, **ErrTyp= "prone to logical errors", KPrL= "Pascal, Java"**. She is taking the practice test of the concept $C_{4.2}$: "Calculating a sum in a for loop" and she is making 12% errors. So, the degree of success x is 88. According to the membership functions the quadruplet, which expresses the student knowledge on the domain concept $C_{4.2}$, is (0, 0, 0.4, 0.6). That means that the specific knowledge domain becomes 40% Known and 60% Learned for Mary. According to the FCMs, the knowledge level of this concept, affects the knowledge level of the domain concepts $C_{4.3}$: “Counting in a for loop”, $C_{4.4}$: “Calculating average in a for loop”, $C_{5.2}$: “Calculating a sum in a while loop”, $C_{5.3}$: “Counting in a while loop” and C5.4: “Calculating average in a while loop”. According to the rules over the fuzzy sets that are used by the FuzKSD the following occur:

- According to R_2, the domain $C_{4.3}$ becomes 27% Learned, as it is $\mu_L(C_{4.3}) = \mu_L(C_{4.3})*\mu_D(C_{4.2},C_{4.3}) = 0.6*0.45 = 0.27$. So, the quadruplet of the knowledge state of the concept $C_{4.3}$ is (0, 0, 0.73, 0.27).
- According to R_2, the domain $C_{4.4}$ becomes 49% Learned, as it is $\mu_L(C_{4.4}) = \mu_L(C_{4.2})*\mu_D(C_{4.2},C_{4.4}) = 0.6*0.81 = 0.49$. So, the quadruplet of the knowledge state of the concept $C_{4.4}$ is (0, 0, 0.51, 0.49).
- According to R_2, the domain $C_{5.2}$ becomes 60% Learned, as it is $\mu_L(C_{5.2}) = \mu_L(C_{5.2})*\mu_D(C_{4.2},C_{5.2}) = 0.6*1 = 0.6$. So, the quadruplet of the knowledge state of the concept $C_{5.2}$ is (0, 0, 0.4, 0.6).
- According to R_2, the domain $C_{5.3}$ becomes 27% Learned, as it is $\mu_L(C_{5.3}) = \mu_L(C_{5.3})*\mu_D(C_{4.2},C_{5.3}) = 0.6*0.45 = 0.27$. So, the quadruplet of the knowledge state of the concept $C_{5.3}$ is (0, 0, 0.73, 0.27).
- According to R_2, the domain $C_{5.4}$ becomes 23% Learned, as it is $\mu_L(C_{5.4}) = \mu_L(C_{5.4})*\mu_D(C_{4.2},C_{5.4}) = 0.6*0.39 = 0.23$. So, the quadruplet of the knowledge state of the concept $C_{5.4}$ is (0, 0, 0.77, 0.23).

Thus, the system infers that Mary has learn and assimilated the corresponding domain concepts and it is classified her to an upper knowledge level to read the following learning material (table 26). Therefore, the value of KL is increased and becomes 5. The system inferred that Mary does not need to read the learning material of the knowledge level 4.

Table 26: Mary's progress

Domain Concepts	Learner's Knowledge			
	Before (KL=3)		After (KL=5)	
	(μ_{Un}, μ_{UK}, μ_K, μ_L)	Cognitive state	(μ_{Un}, μ_{UK}, μ_K, μ_L)	Cognitive state
1.1 constants & variables	(0, 0, 0, 1)	100% Learned	(0, 0, 0, 1)	100% Learned
1.2 assignment statement	(0,0,0.08, 0.92)	92% Learned	(0,0,0.08, 0.92)	92% Learned
1.3 arithmetic operators	(0, 0, 0, 1)	100% Learned	(0, 0, 0, 1)	100% Learned
1.4 comparative operators	(0, 0, 0, 1)	100% Learned	(0, 0, 0, 1)	100% Learned
1.5 logical operators	(0, 0, 0, 1)	100% Learned	(0, 0, 0, 1)	100% Learned
1.6. mathematic functions	(0, 0, 0, 1)	100% Learned	(0, 0, 0, 1)	100% Learned
1.7 input-output statements	(0, 0, 0, 1)	100% Learned	(0, 0, 0, 1)	100% Learned
2.1 a simple program's structure	(0, 0, 0, 1)	100% Learned	(0, 0, 0, 1)	100% Learned
3.1 if statement	(0, 0, 0.15, 0.85)	85% Learned	(0, 0, 0.15, 0.85)	85% Learned
3.2 if…else if	(0, 0, 0.15, 0.85)	85% Learned	(0, 0, 0.15, 0.85)	85% Learned
3.2.1 finding max, min	(0, 0, 0, 1)	100% Learned	(0, 0, 0, 1)	100% Learned
3.3 nested if statement	(0, 0, 0.35, 0.65)	65% Learned	(0, 0, 0.35, 0.65)	65% Learned
4.1 for statement	(0, 0, 0, 1)	100% Learned	(0, 0, 0, 1)	100% Learned
4.2 calc. sum in a for loop	***(0, 0.60, 0.40, 0)***	***40% Known***	***(0, 0, 0.40, 0.60)***	***60% Learned***
4.3 counting in a for loop	***(0, 0.82, 0.18, 0)***	***18% Known***	***(0, 0, 0.73, 0.27)***	***27% Learned***
4.4 calc. avrg in a for loop	***(0, 0.60, 0.40, 0)***	***32% Known***	***(0, 0, 0.51, 0.49)***	***49% Learned***
4.5 calc. max/min in a for loop	(0, 0, 0.63, 0.37)	37% Learned	(0, 0, 0.63, 0.37)	37% Learned
5.1 while statement	(0, 0, 0.80, 0.20)	20% Learned	(0, 0, 0.80, 0.20)	20% Learned
5.2 calc. sum in a while loop	***(0, 0.60, 0.40, 0)***	***40% Known***	***(0, 0, 0.40, 0.60)***	***60% Learned***
5.3 counting in a while loop	***(0, 0.84, 0.16, 0)***	***16% Known***	***(0, 0, 0.73, 0.27)***	***27% Learned***
5.4 calc. avrg in a while loop	***(0, 0.82, 0.18, 0)***	***18% Known***	***(0, 0, 0.77, 0.23)***	***23% Learned***
5.5 calc. max/min in a while loop	(0, 0, 0.63, 0.37)	37% Learned	(0, 0, 0.63, 0.37)	37% Learned
5.6 do…until	(0, 0, 1, 0)	100% Known	(0, 0, 1, 0)	100% Known
6.1 one-dimension arrays	(1, 0, 0, 0)	100% Unknown	(1, 0, 0, 0)	100% Unknown
6.2 searching	(1, 0, 0, 0)	100% Unknown	(1, 0, 0, 0)	100% Unknown
6.3 sorting	(1, 0, 0, 0)	100% Unknown	(1, 0, 0, 0)	100% Unknown
6.4 two-dimensions arrays	(1, 0, 0, 0)	100% Unknown	(1, 0, 0, 0)	100% Unknown
6.5 processing per rows	(1, 0, 0, 0)	100% Unknown	(1, 0, 0, 0)	100% Unknown
6.6 processing per column	(1, 0, 0, 0)	100% Unknown	(1, 0, 0, 0)	100% Unknown
6.7 processing of diagonals	(1, 0, 0, 0)	100% Unknown	(1, 0, 0, 0)	100% Unknown
7.1 functions	(1, 0, 0, 0)	100% Unknown	(1, 0, 0, 0)	100% Unknown

Example 6:

The parameters of Mike's student model had the following values: KL=4, **ErrTyp= "prone to logical errors", KPrL= "none"**. Mike has successfully finished the practice test of all the concepts of his stereotype. Now, he is taking the final test, which corresponds to his knowledge level and he is making 28% error on the concept $C_{5.2}$: "Calculating a sum in a while loop". So, the degree of success x is 72, and according to the membership functions the quadruplet which expresses the student knowledge on the domain concept $C_{5.2}$, is (0, 0.6, 0.4, 0).

That means that the specific knowledge domain becomes 40% Known and 60% Insufficiently Known for Mike. According to the FCMs, the knowledge level of this concept, affects the knowledge level of the domain concepts $C_{4.2}$: “Calculating a sum in a for loop”, $C_{4.3}$: “Counting in a for loop” and $C_{4.4}$: “Calculating average in a for loop”, which precede to the domain concept $C_{5.2}$. Furthermore, the concept $C_{5.2}$ affects the knowledge level of the domain concepts $C_{5.3}$: “Counting in a while loop” and $C_{5.4}$: “Calculating average in a while loop”, which follow to the domain concept $C_{5.2}$. According to the rules over the fuzzy sets that are used by the FuzKSD the following occur:

- According to R_{10}, the domain $C_{4.2}$ becomes 40% Known, as it is $x_{4.2} = \min[x_{4.2}, (1-\mu_D(C_{4.2}, C_{5.2}))*x_{4.2} + x_{5.2}] = \min[88, (1-1)*88+72] = 72$. So, the quadruplet of the knowledge state of the concept $C_{4.2}$ is (0, 0.6, 0.4, 0).
- According to R_9, the domain $C_{4.3}$ remains 100% Learned.
- According to R_{10}, the domain $C_{4.4}$ becomes 40% Known, as it is $x_{4.4} = \min[x_{4.4}, (1-\mu_D(C_{4.4}, C_{5.2}))*x_{4.4} + x_{5.2}] = \min[87.45, (1-1)*87.45+72] = \min[87.45, 72] = 72$.
  So, the quadruplet of the knowledge state of the concept $C_{4.4}$ is (0, 0.6, 0.4, 0).
- According to R_3, the domain $C_{5.3}$ remains 100% Learned.
- According to R_6, the domain $C_{5.4}$ becomes 32.4% Known, as it is $\mu_K(C_{5.4}) = \mu_K(C_{5.2})*\mu_D(C_{5.2}, C_{5.4}) = 0.4*0.81 = 0.324$. So, the quadruplet of the knowledge state of the concept $C_{5.4}$ is (0, 0.676, 0.324, 0).

Thus, the system infers that Mike has forgot the corresponding concepts and it is classified him to a previous knowledge level to revise them (table 27). Thus, the value of KL is reduced and becomes 3.

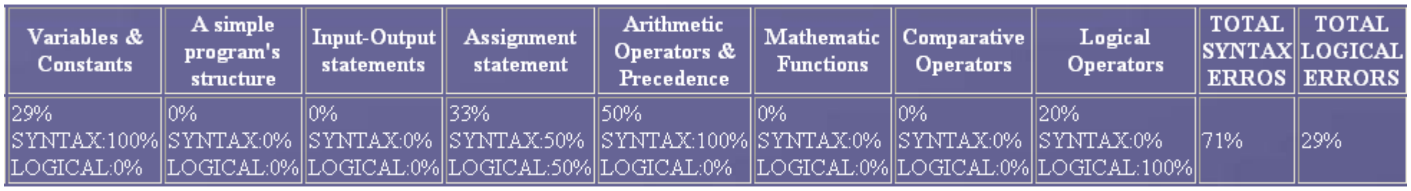
Table 27: Mike's progress

Domain Concepts	Learner's Knowledge			
	Before (KL=4)		After (KL=3)	
	(μ_{Un}, μ_{UK}, μ_K, μ_L)	Cognitive state	(μ_{Un}, μ_{UK}, μ_K, μ_L)	Cognitive state
1.1 constants & variables	(0, 0, 0, 1)	100% Learned	(0, 0, 0, 1)	100% Learned
1.2 assignment statement	(0,0,0.08, 0.92)	92% Learned	(0,0,0.08, 0.92)	92% Learned
1.3 arithmetic operators	(0, 0, 0, 1)	100% Learned	(0, 0, 0, 1)	100% Learned
1.4 comparative operators	(0, 0, 0, 1)	100% Learned	(0, 0, 0, 1)	100% Learned
1.5 logical operators	(0, 0, 0, 1)	100% Learned	(0, 0, 0, 1)	100% Learned
1.6. mathematic functions	(0, 0, 0, 1)	100% Learned	(0, 0, 0, 1)	100% Learned
1.7 input-output statements	(0, 0, 0, 1)	100% Learned	(0, 0, 0, 1)	100% Learned
2.1 a simple program's structure	(0, 0, 0, 1)	100% Learned	(0, 0, 0, 1)	100% Learned
3.1 if statement	(0, 0, 0.15, 0.85)	85% Learned	(0, 0, 0.15, 0.85)	85% Learned
3.2 if…else if	(0, 0, 0.15, 0.85)	85% Learned	(0, 0, 0.15, 0.85)	85% Learned
3.2.1 finding max, min	(0, 0, 0, 1)	100% Learned	(0, 0, 0, 1)	100% Learned
3.3 nested if statement	(0, 0, 0.35, 0.65)	65% Learned	(0, 0, 0.35, 0.65)	65% Learned
4.1 for statement	(0, 0, 0, 1)	100% Learned	(0, 0, 0, 1)	100% Learned
4.2 calc. sum in a for loop	***(0, 0, 0.40, 0.60)***	***60% Learned***	***(0, 0.60, 0.40, 0)***	***40% Known***
4.3 counting in a for loop	***(0, 0, 0, 1)***	***100% Learned***	***(0, 0, 0, 1)***	***100% Learned***
4.4 calc. avrg in a for loop	***(0, 0, 0.51, 0.49)***	***49% Learned***	***(0, 0.60, 0.40, 0)***	***40% Known***
4.5 calc. maxmin in a for loop	(0, 0, 0.63, 0.37)	37% Learned	(0, 0, 0.63, 0.37)	37% Learned
5.1 while statement	(0, 0, 0.80, 0.20)	20% Learned	(0, 0, 0.80, 0.20)	20% Learned
5.2 calc. sum in a while loop	***(0, 0, 0.40, 0.60)***	***60% Learned***	(0, 0.60, 0.40, 0)	***40% Known***
5.3 counting in a while loop	***(0, 0, 0, 1)***	***100% Learned***	***(0, 0, 0, 1)***	***100% Learned***
5.4 calc. avrg in a while loop	***(0, 0, 0.59, 0.41)***	***41% Learned***	***(0, 0.676, 0.324, 0).***	***32.4% Known***
5.5 calc. max/min in a while loop	(0, 0, 0.63, 0.37)	37% Learned	(0, 0, 0.63, 0.37)	37% Learned
5.6 do…until	(0, 0, 1, 0)	100% Known	(0, 0, 1, 0)	100% Known
6.1 one-dimension arrays	(1, 0, 0, 0)	100% Unknown	(1, 0, 0, 0)	100% Unknown
6.2 searching	(1, 0, 0, 0)	100% Unknown	(1, 0, 0, 0)	100% Unknown
6.3 sorting	(1, 0, 0, 0)	100% Unknown	(1, 0, 0, 0)	100% Unknown
6.4 two-dimensions arrays	(1, 0, 0, 0)	100% Unknown	(1, 0, 0, 0)	100% Unknown
6.5 processing per rows	(1, 0, 0, 0)	100% Unknown	(1, 0, 0, 0)	100% Unknown
6.6 processing per column	(1, 0, 0, 0)	100% Unknown	(1, 0, 0, 0)	100% Unknown
6.7 processing of diagonals	(1, 0, 0, 0)	100% Unknown	(1, 0, 0, 0)	100% Unknown
7.1 functions	(1, 0, 0, 0)	100% Unknown	(1, 0, 0, 0)	100% Unknown

Example 7:

George, a novice student, interacted with the system for the first time. He completed the test that involved involved exercises, which concerned the basic concepts of "C". The results are depicted in figure 52. Particularly, he does 29% errors on variables and constants, 33% errors on the assignment statement and 50% errors on the arithmetic operators. Therefore, the_parameters of his student model acquired the following values: KL=1, **ErrTyp= "prone to syntax errors",**

KPrL= “none”. The system infers that George did not read the corresponding domains. So, he is not transmitted to the next stereotype, but he is advised to revise the concepts of his current stereotype.



Variables & Constants	A simple program's structure	Input-Output statements	Assignment statement	Arithmetic Operators & Precedence	Mathematic Functions	Comparative Operators	Logical Operators	TOTAL SYNTAX ERROS	TOTAL LOGICAL ERRORS
29% SYNTAX:100% LOGICAL:0%	0% SYNTAX:0% LOGICAL:0%	0% SYNTAX:0% LOGICAL:0%	33% SYNTAX:50% LOGICAL:50%	50% SYNTAX:100% LOGICAL:0%	0% SYNTAX:0% LOGICAL:0%	0% SYNTAX:0% LOGICAL:0%	20% SYNTAX:0% LOGICAL:100%	71%	29%

Figure 52: Test's results of George

7. Evaluation of ELaC

Although, the adaptation generated by user modeling techniques often tend to improve the user-system interaction, most of the time the exploitation of such techniques makes the system more complex and consequently, it should be evaluated whether the adaptivity really improves the system and whether the user really prefers the adaptive version of it (Gena, 2005). The fact that adding a user model to any software system will most likely make it more complex, less predictable and more buggy, leads us to ask whether or not the user model will actually improve the system (Chin, 2001). Consequently, the evaluation of a student model is a crucial factor and several researchers have attempted to assess the student model of their adaptive system. An assessment of the student model that SQL-Tutor uses is presented in Mitrovic, Marting, and Mayo (2002). Also, Weibelzahl and Weber (2003) performed the evaluation of the accuracy of the student model of an adaptive learning system, called the HTML-Tutor. A more recent attempt to assess the effectiveness and the accuracy of the student model, which was applied in an intelligent tutoring system for learning software design patterns, was done by Jeremić, Jovanović, and Gasěvić (2009).

Even though, there are many evaluation methods available in literature review, as Mulwa et al. (2011) have mentioned, there is no a standard agreed measurement framework for assessing the value and effectiveness of the adaptation yielded by adaptive systems. The most common practice of evaluation is experiments. However, there is not an accurate, clear and agreed framework in which an experiment for the assessing of a student model should be performed. Furthermore, it is important to not only evaluate but also to ensure that the evaluation uses the correct methods, since an incorrect method can lead to wrong conclusions (Gena & Weibelzahl, 2007). As Dempster (2004) has pointed out a well-designed evaluation should provide the evidence if a specific approach has been successful and of potential value to others. That is the reason for choosing to use a combination of two well-known and used techniques in order to assess ELaC's student model. The one evaluation model that we use is the Kirkpatrick's model (1979). It defines four levels of evaluation:

- **Evaluation of reaction**: It is examined what the learners thought and felt about the training. A typical instrument for gathering information regarding students' reactions is questionnaires.
- **Evaluation of learning**: It assesses the extent to which the learners gain knowledge and skills. At this level, each student's learning should be measured by quantitative and objective means.
- **Evaluation of behavior**: It examines what changes in job performance resulted from the learning process.
- **Evaluation of results**: It assesses the effects on the business, organization or environment resulting from the trainee's performance.

The second evaluation model that we selected is a model-based evaluation approach, which is called the layered evaluation framework (Brusilovsky, Karagiannidis & Sampson, 2004). According to this framework the success of adaptation is addressed at two distinct layers:

- **Layer 1:** Evaluation of user modeling. At this layer only the user modeling process is being evaluated. Here the question can be stated as: "are the conclusion drawn by the system concerning the characteristics of the user-computer interaction valid?" or "are the user's characteristics being successfully detected by the system and stored in the user model?"
- **Layer 2:** Evaluation of adaptation decision-making. At this layer only the adaptation decision-making is being evaluated. The question here can be stated as: "are the adaptation decisions valid and meaningful for the given state of the user model?"

Therefore, we use the above evaluation methods in order to construct an evaluation method, which is called PeRSIVA, for assessing the results of ELaC's student model in students' satisfaction, performance, progress, behavior and state, as well as the validity of the conclusions drawn by the student model and the validity of the adaptation decision making.

7.1 Criteria

In order to define an evaluation method, the evaluation criteria have to be defined initially. The proposed criteria are the following:

- **Students' satisfaction** about the e-learning program. More concretely, we want to gather information about the feelings, thoughts and satisfaction of the learners' about the adaptivity and effectiveness of the e-learning education environment.
- **Students' performance** on the knowledge domain. Particularly, we want to measure the extent to which the learners gain knowledge on computer programming.
- The changes that were caused on the **individual state** of the students. In other words, we want to assess the effect of the e-learning program on the behavior and thoughts of students about computer programming and distance learning.
- The **results** of the e-learning program to students' progress. It concerns the effects of the e-learning program to students' progress on their further studies.
- The **validity of the conclusions** drawn by the student model concerning the aspects of the students' characteristics.
- The **validity of the adaptation** decision making of the student model.

7.2 The method

PeRSIVA combines the four levels of the Kirkpatrick's evaluation model with the two layers of the layered evaluation framework. The name PeRSIVA comes from the initials of the evaluation criteria's names (Performance, Results, Satisfaction, Individual state, Validity of conclusions, Adaptation's validity). The reason for the combination of the above two evaluation methods is that the assess of satisfaction coincides with the Kirkpatrick's evaluation of reaction level, the measurement of students' performance is similar to the Kirkpatrick's evaluation level of learning and the students' state/behavior and progress can be matched with the Kirkpatrick's evaluation levels of behavior and results, accordingly. Furthermore, the validity of the student model can be evaluated applying the two layers of the layered evaluation framework, which focus on the user modeling process and on adaptation decision-making process. Another significant reason for the use of this blended evaluation model is the fact that there is no a standard agreed measurement framework for evaluating the effectiveness, accuracy and validity of a student model. Thus, two well-known evaluation methods, like the above two, were used. Consequently, PeRSIVA includes:

I. Assessing the learners' satisfaction about the e-learning environment. For gathering this kind of information a questionnaire (Questionnaire A, section 7.5) was used. The questions were close-ended based on Likert scale with five responses ranging from "Very much" (5) to "Not at all" (1). The questions were divided into two sections based on the type of information we were interested in. The questions of the first section were related to the effectiveness of the training program. The second section was aimed at evaluating the adaptivity of the system.

II. Measuring the students' performance by conducting an experiment with an experimental group (the group of students which used the ELaC environment) and a control group (the group of students which used a similar educational environment from which the student model was absent).

III. Assessing the changes on the students' state/behavior about computer programming and e-learning. For gathering this kind of information a questionnaire (Questionnaire B, section 7.5) was used. The questions were close-ended based on Likert scale with five responses ranging from "Very much" (5) to "Not at all" (1). The questions were divided into three sections based on the type of information we were interested in. The questions of the first section were related to the students' perception about computer programming. The second section was aimed at evaluating the students' state towards e-learning. The third section included questions related to students' motivation to be involved in e-learning programs.

IV. Assessing the effects of the e-learning program on the students' progress concerning their further studies. For assessing this criterion a questionnaire (Questionnaire C, section 7.5) was used, which included five close-ended questions based on Likert scale with five responses ranging from "Very much" (5) to "Not at all" (1).

V. Assessing the validity of the conclusions drawn by the student model concerning the aspects of the students' characteristics by assessing the results of the system's use in relation with the different backgrounds of the students. More concretely, the group of the thirty-eight students that used ElaC was divided into three categories according to their background knowledge. These three categories are arts, science fields (other than computer science) and computer science related fields. To arts belong students, which have studies in the field of human, social, political and educational sciences. To science fields belong students, which have studies in the field of mathematic, physic, business and economic sciences. To computer science-related fields belong students, which have studies in the field of computer, engineering and system sciences. It is obvious that students, which belong to the last category, have a previous knowledge on computer programming, so their progress should be better than the students of the other two categories, and also they should be advised to read a chapter fewer times than the others. The next category whose students should have a good learning pace is the category of science fields, since their studies offer them a way of thinking that is close to the programming logic. Whilst, the students of the arts' category should be advised to read more times the learning material, until to learn the knowledge domain, since they often have no idea about computer programming. Consequently, for each background category, the average reading times of each domain concept and the percentage of learners that are advised to return to a previous concept in order to revise it were measured and these values were compared in order to assess if the conclusions drawn by the system concerning the characteristics of student model are valid.

VI. Assessing the validity of adaptation decisions, which affect the students' learning pace by improving the quality of the learner's interaction with the system. Therefore, the adaptation decision about the return of a learner to a domain concept in order to revise it and the adaptation decision about the inference of the system that a learner should not read a particular domain concept at all, have to be assessed. The decision that a learner should return to a concept in order to revise it arises from the learner's performance on this domain concept. So, the only method to assess the validity of this kind of adaptation decision is to ask the learners' opinion about the appropriateness or necessity of these returns. Thus, a questionnaire (Questionnaire D, section 7.5) with close-ended questions based on Likert scale with five responses ranging from "Very much" (5) to "Not at all" (1), was used. Furthermore, for assessing the validity of the system's inference that a learner should not read a particular domain concept at all, the percentage of times that a learner needed finally to read a domain concept that the system had advised her/him not to read was measured.

7.3 The evaluation population

ElaC was used by a group of thirty-eight students (group A) of a postgraduate conversion course program in the field of informatics at the University of Piraeus. After their participation in the training program, the learners completed the questionnaires A and D that are displayed in section 7.5. After, almost, two years, learners were asked to answer to the questionnaires B and C that are displayed in section 7.5. The reason, for which the learners were provided with these two questionnaires after such a long period, is that the evaluation of behavior and the evaluation of results levels of the Kirkpatrick's model need at least a two-year evaluation period (Jeremić, Jovanović & Gasěvić, 2009). The answers of the above four questionnaires helped to assess students' satisfaction, the changes on students' state/behavior, the results on students' progress on their further studies and the validity of adaptation decision making.

Moreover, students' performance was measured and was compared with the performance of another group of thirty-seven students (group B) of the same postgraduate program, which used a similar educational system from which the student model was absent. Both systems had the same knowledge domain, which is divided into 31 concepts, but the second system delivers the concepts of the learning material in sequence without taking into account how the learner's performance on a domain concept may affect her/his knowledge of another concept.

Learners of both groups had different ages, varying from 22 to 50, and backgrounds. Examples of such backgrounds are physics, mathematics, computer science, education, human and social science. The number of students, which belong to either each age category or background category, is the same for both groups (table 28). The reason for this is the fact that the homogeneity of the experiment's samples simplifies the experiment's performing. Furthermore, the distribution of students' knowledge of other languages for both groups is depicted in table 29. The learners of both groups used the corresponding systems without attending any complementary course on programming, over a period of six months.

Table 28: Distribution of students' ages and backgrounds

Ages	**22-25**	**26-29**	**30-35**	**36-39**	**40-45**	**46-50**
	23.68%	28.95%	31.58%	2.63%	7.90%	5.26%
Background	**Arts**		**Science (other than computers)**		**Computer science related**	
	31.58%		18.42%		50%	

Table 29: Distribution of students' knowledge of other languages

Language	Java	Pascal	Basic
Group A	23.68%	31.58%	34.21%
Group B	18.42%	39.47%	29%

7.4 Results

7.4.1 Satisfaction

Learners' satisfaction about the adaptivity and effectiveness of ELaC is positive. The results of the questionnaire reveal that the users are very satisfied with the educational environment and its contribution to the learning process. The results of the questionnaire are depicted in figure 53. This information is easy to collect, but does not tell enough about the learning success.

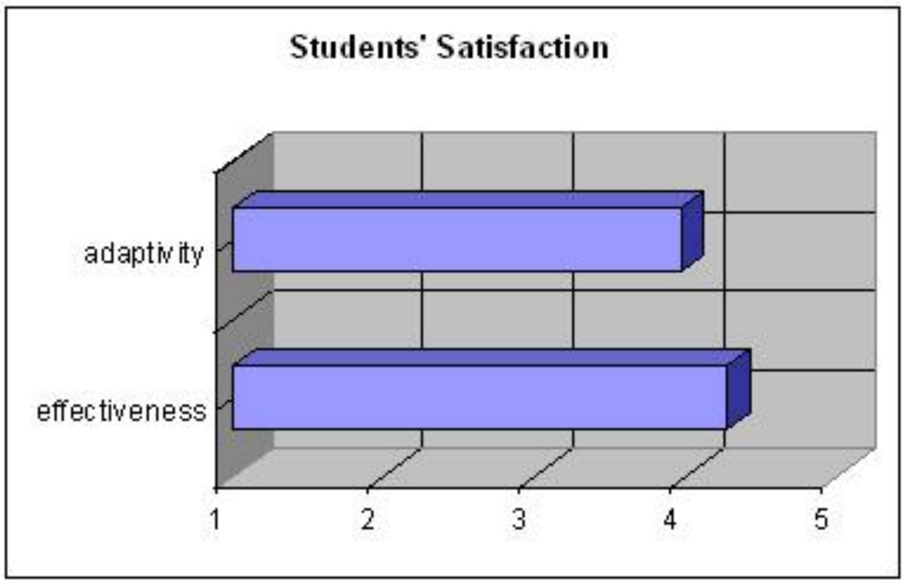


Figure 53: Students' satisfaction

7.4.2 Performance

One of the main goals of the system is to adapt dynamically the teaching sequence to the users' individual level of knowledge. In this sense the evaluation's aim is to evaluate the individualization of the teaching rather than evaluating the success of the teaching method alone. In other words, it is evaluated how the student was taught and whether s/he learned successfully rather than just whether s/he learned successfully. As such, a classical pre-test and post-test methodology could not be sufficient for the aimed evaluation. The learners' performance in a final exam, as well as the mean value of times that a learner needed to read a domain concept in order to learn it, were measured and compared.

According to Grubišić et al. (2009) experiment used in the e-learning systems' effectiveness evaluation change the independent variable (tutoring strategy) while measuring the depended variable (effects on learning). In the presented work, the experiment has two independent variables and two depended variables. The independent variables are the tutoring system and the learner's background. The dependent variables are the learner's performance and the times that a learner needed to read a domain concept in order to learn it. Due to the fact that the experiment involves more than one dependent variables at a time, the multivariate analysis of variance (MANOVA) should be used to analyze the experiment's data. MANOVA allows researchers to test hypotheses regarding the effect of one or more independent variables on two or more dependent variables (Carver & Nash, 2009). It is used to determine whether there are any differences between independent groups on more than one continuous dependent variable. The results of the experiment are depicted in table 30.

According to them the mean value of performance of group A (93.9) is higher than the corresponding mean value of group B (84.5). Furthermore, students who used ELaC needed to read each domain concept less times (1.06) on average than the students of group B (1.46). Although, ElaC advises a learner to revise a concept whenever it realizes that s/he has forgot it during the learning program, the mean of reading times of each domain concept is lower. They are almost 27.4% lower than the reading times of group B. The reason for this is the knowledge dependencies that are kept by the system. Due to them, there are concepts that were not read at all. ELaC inferred that they were already known; taking into account the learner's performance and progress on depended with them domain concepts, and did not advise her/him to read them.

Table 30: Descriptive statistics

	group	background	Mean	Std. Deviation	N
performance	group A	arts	93,4292	,68495	12
		computer science related	94,2605	,78642	19
		science (other than computers)	93,7243	,63416	7
		Total	93,8992	,80626	38
	group B	arts	84,0100	,59829	12
		computer science related	84,7500	,87703	19
		science (other than computers)	84,6800	,76450	7
		Total	84,5034	,83155	38
no_of_reads	group A	arts	1,4525	,18815	12
		computer science related	,7953	,12420	19
		science (other than computers)	1,0771	,14625	7
		Total	1,0547	,32784	38
	group B	Arts	1,8092	,36092	12
		computer science related	1,2805	,19708	19
		science (other than computers)	1,3386	,25030	7
		Total	1,4582	,35588	38

One of the assumptions of the MANOVA is homogeneity of covariances, which is tested by Box's Test of Equality of Covariance Matrices. If the "Sig." value is less than .001 then the assumption of homogeneity of covariances was violated. In the presented experiment this assumption is not violated (sig. = .210) (table 31), so Wilks' Lambda test can be conducted. Wilks' Lambda test is used to determine whether the differences of the means were statistically significant. Lambda varies between 1 and zero. The ideal value of it is to be near zero. Furthermore, if the "Sig." value is less than .0005, then it means that the dependent variable has a significant effect on the independent variable. The lambda value of the independent variables “group” and “background” is 0.026

and 0.244 correspondingly. Also, the “Sig.” value of the conducted experiment is .000 for both dependent variables (table 32). Therefore, learners' performance and/or their times of reading of a concept were significantly dependent on which system they had used and on their background.

Table 31: Box's Test of Equality of Covariance Matrices

Box's M	21,007
F	1,273
df1	15
df2	6555,789
Sig.	,210

Table 32: Wilks' Lambda Effect

Effect	Lambda	F	Hypothesis df	Error df	Sig.
Group	,026	1298,628	2,000	69,000	,000
background	,244	35,357	4,000	138	,000

To determine, however, how the dependent variables differ for the independent variables and if the dependent variables have a significant effect on both or only on one independent variable, tests of Between-Subjects Effects Table (table 33) has to be checked. But, firstly, the homogeneity of variances has to be checked. This is indicated by the “sig.” value of the Levene's Test of Equality of Error Variances, which has to be higher than .05. Both the 'performance' and 'no_of_reads' scores have homogeneity of variances, since their “sig.” value is .942 and 0.132 correspondingly. From the results that are depicted in table 31, it is concluded that the tutoring system has a statistically significant effect on both performance (sig. < .0005) and times of reading (sig. < .0005). However, the learner's background has a significant effect only on the times of reading. The “sig” value of the performance for background is 0.001 that is higher than 0.0005. That indicates that the learner's background does not affect her/his performance significantly.

Table 33: Tests of Between-Subjects Effects

Source	Dependent Variable	F	Sig.
Group	performance	2456,052	,000
	no_of_reads	79,348	,000
background	performance	7,946	,001
	no_of_reads	94,765	,000

In order to be informed about the significance of the effect of background on the times of reading of a concept, a univariate analysis was conducted for the two groups separately. The results are depicted in table 34. According to them, for group A the mean values of the times of reading were statistically significantly different between arts and computer science related backgrounds (sig. < .0005), arts and sciences other than computers (sig. < .0005), and computer science related backgrounds and sciences other than computers (sig. < .0005). On the other hand, for group B the corresponding mean values were statistically significantly different between arts and computer science related backgrounds (sig. < .0005), but not between arts and sciences other than computers (sig. =0.003), and computer science related backgrounds and sciences other than computers (sig. < 0.009). This implies that ELaC is able to recognize each individual learner's needs and to adapt dynamically to her/his learning pace.

Table 34: Multiple Comparisons

Dependent Variable	(I) background	(J) background	Group A		Group B	
			Mean Difference (I-J)	Sig.	Mean Difference (I-J)	Sig.
no_of_reads	arts	computer science related	,6572	,000	,6529	,000
		science (other than computers)	,3754	,000	,3594	,003
	computer science related	arts	-,6572	,000	-,6529	,000
		science (other than computers)	-,2819	,000	-,2935	,009
	science (other than computers)	arts	-,3754	,000	-,3594	,003
		computer science related	,2819	,000	,2935	,009

7.4.3 Students' individual state

Learners' state and behavior towards the computer programming and distance learning have positively changed. The results of the questionnaire reveal that the students' perception about computer programming and e-learning has improved, and also these two domains have drawn learners' interest. They seem more willing to be involved in computer programming and e-learning programs. For more representative and accurate results, the evaluation results were separated in two categories. The first category concerns the learners that had no previous knowledge on computer programming and the second category concerns the learners that knew already at least one programming language. This separation was done because it is logical that the learners, who already know a programming language, have already a positive state toward computer programming, and thus, changes in their state may be less significant.

Furthermore, usually a learner, who has previous knowledge on computer programming, is involved with computer science, and therefore the subject of e-learning is less “strange” to her/him. The questionnaire B that was answered by the learners and the mean of students’ answers are displayed in section 7.5. The results of the questionnaire are depicted in figures 54 and 55.

The results show that the state towards computer programming and e-learning of the learners, who had no previous knowledge on programming, was improved by 84.6% and 84.4% respectively. While their willingness to be engaged in e-learning programs, was increased by 76.8%. Similarly, the state of the learners who knew already another programming language at least, towards computer programming and e-learning was improved by 75.4% and 79.6% respectively. Also, their motivation to be involved in e-learning programs was increased by 74%.

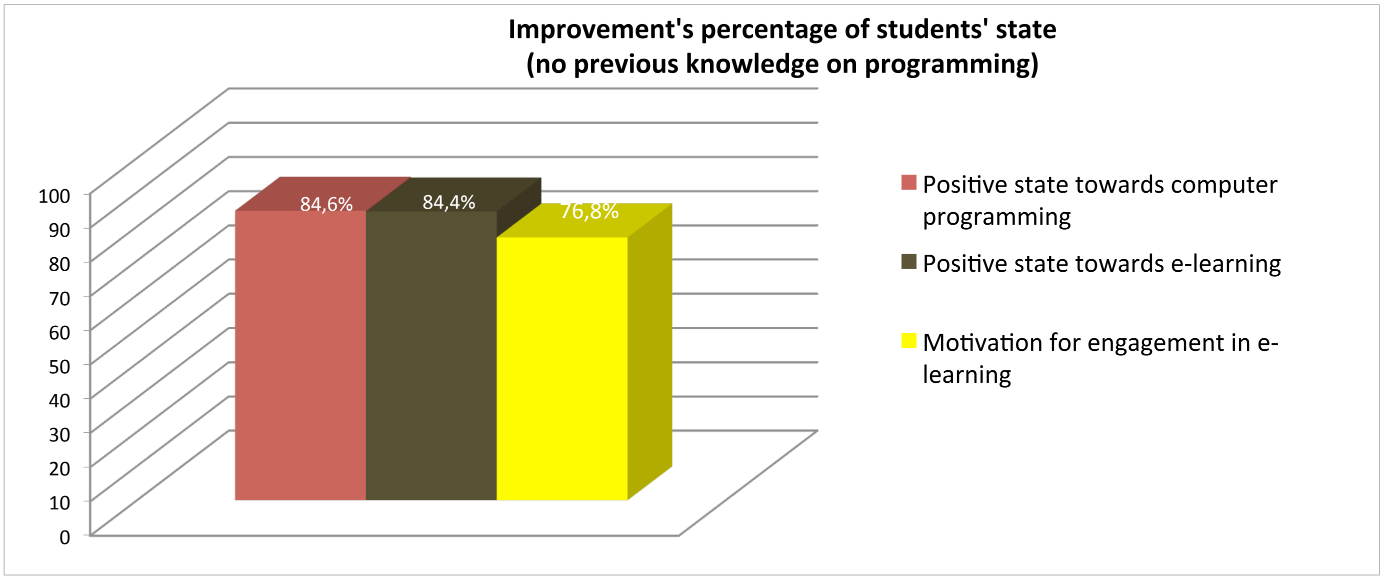


Figure 54: Changes on individual state of students with no previous knowledge on computer programming

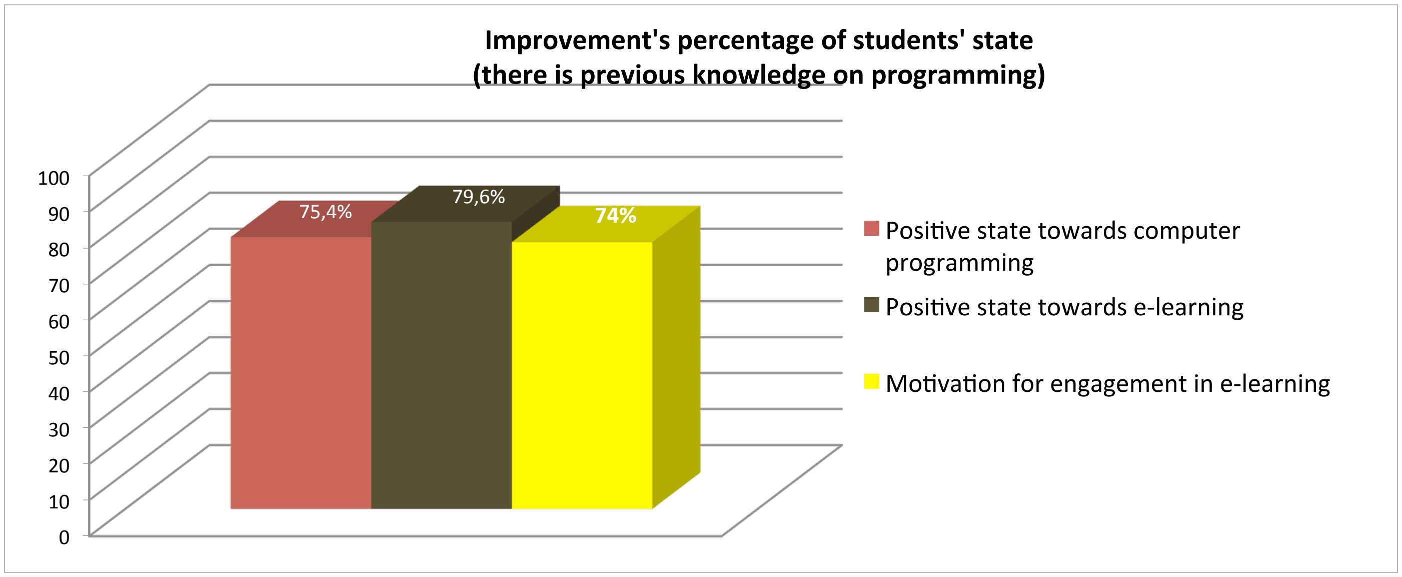


Figure 55: Changes on individual state of students with previous knowledge on computer programming

7.4.5 Results concerning students' progress

The results of the e-learning program to the learners' progress on their further studies are satisfactory. The results of the questionnaire reveal that the e-learning program helped the users. The questionnaire C that was answered by the learners and the evaluation degree are displayed in section 7.5, while the results are depicted in figure 56. For gathering this information, a more advanced empirical research should be done. However the fact that the students, who participated in the empirical evaluation of this study, have already graduated in combination with the fact that the evaluation of results needs at least a two year evaluation period, limited the evaluation framework to this questionnaire.

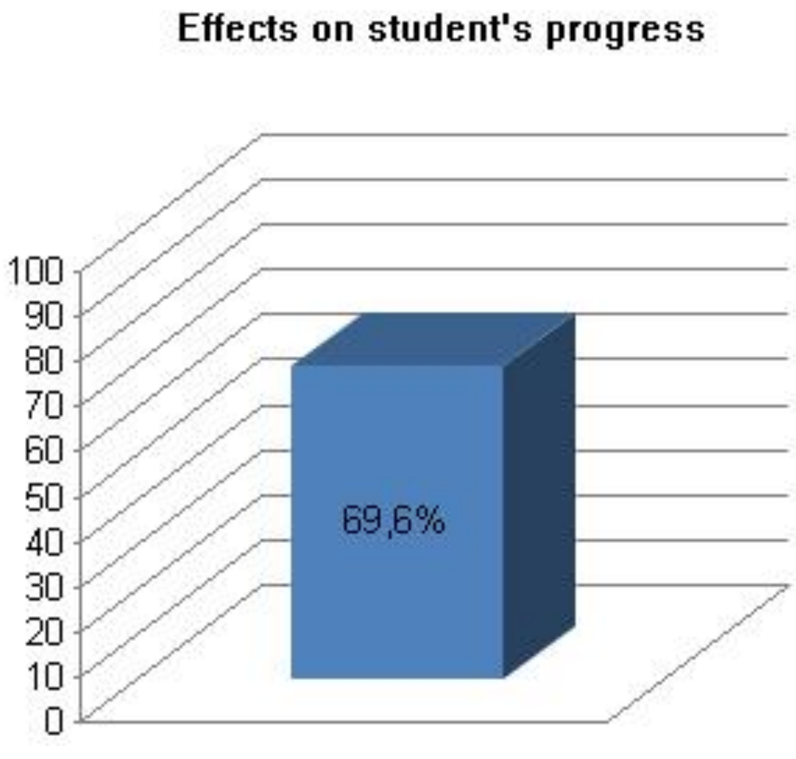


Figure 56: Results on learners' progress

7.4.6 Validity of the conclusions

The conclusions that are drawn by the system concerning the aspects of students' characteristics seem to be satisfactory valid. According to the results the system advises the learners with studies on arts fields to read a domain concept more times than the learners who had been involved with the logic of programming before (Fig. 57). Furthermore, the system advises the learners with studies on arts to return to a domain concept in order to revise it more times than the other learners (Fig. 58). This is evident, since learners with no previous knowledge and experience on computer programming, have difficulty in assimilating the learning material of the particular knowledge domain. The percentage of learners with background on computer science related fields is very low (2.97). It is logical, since many of the learners with background on arts had already been involved in computer programming and thus it is easy to deal with the learning material of the system's knowledge domain.

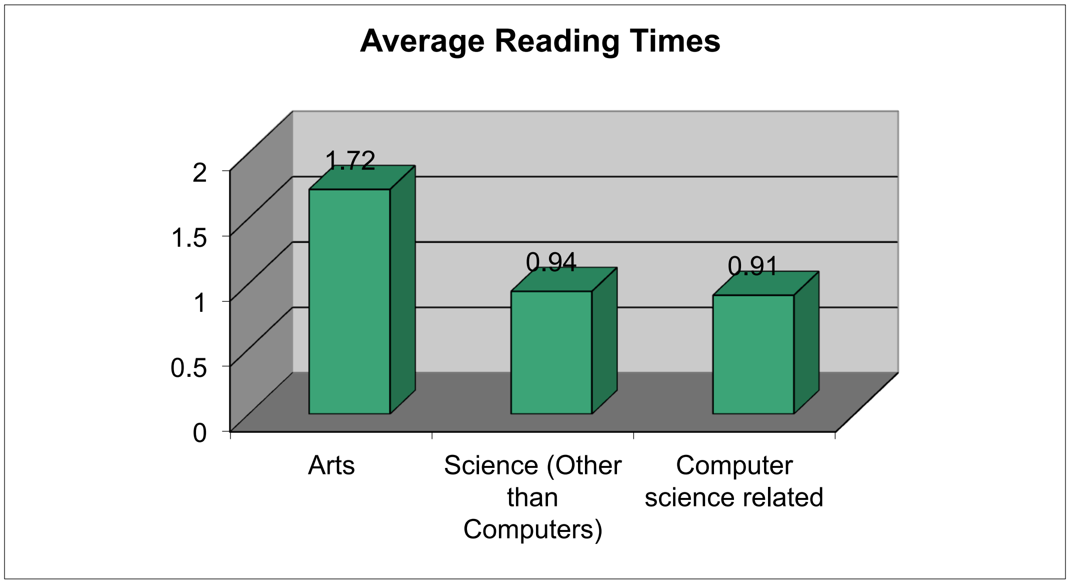


Figure 57: mean of reading times per background category

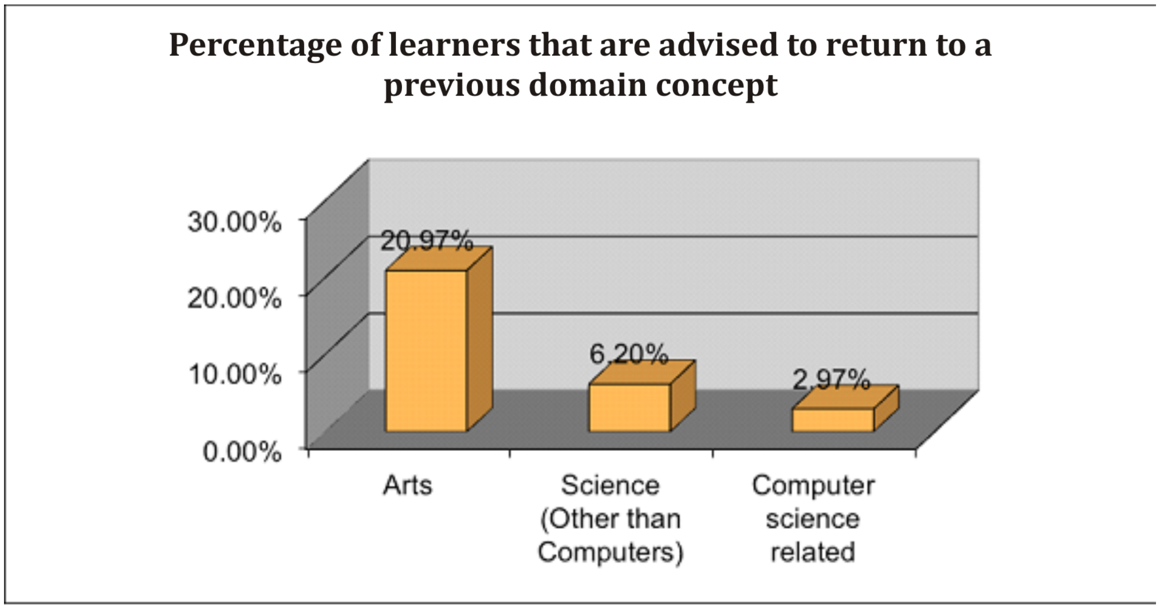


Figure 58: Mean of times of returns to a concept per background category

7.4.7 Validity of adaptation decision-making

The results of the validity of the system's decision making were positive. It has been answered that the percentage of time that the learners spent to revise a previous domain concept is at all waste of time (Questionnaire D, section 7.5). Furthermore, it seems that the system's decision to lead a learner to revise a previous domain concept, correspond satisfactory to the learners' needs and help them in the learning process. In addition, the percentage of times that a learner needed, finally, to read a domain concept that the system had advised her/him not to read it is 12%. This percentage is sufficiently satisfactory to be able to lead to the conclusion that the decisions, which are made by the system based on the student model, are valid.

7.5 Questionnaires

Questionnaire A

	Questions	Evaluation Degree
Effectiveness	Does the educational software meet your expectations?	4
	Does the educational software help you understanding the logic of programming?	4
	Do you think that this educational software is useful as an educational "tool"?	4
	Do you think that the use of this educational software is a waste of time?	1
	After the end of the educational process, do you feel that you have assimilated all the subjects that you are taught?	4.34
Adaptivity	Does the program correspond to your knowledge level each time?	4
	Does the program correspond to your educational needs level each time?	4
	How time do you spend on issues that you already known?	2
	Does the test adapt to your educational needs?	3.78
	Do you thing that each time you go to a next level, you have known adequately all the subjects of the previous chapters?	4.1
	Does your return to a previous level, that happened each time the system discovered that you made errors of previous chapters, help you learning programming?	3.86

Questionnaire B

	Questions	Evaluation Degree	
		No previous knowledge	**Yes previous knowledge**
State on computer programming	Does the educational software affect positively your perception about computer programming?	4,4	3,8
	Does the educational software draw your interest on computer programming?	4,28	3,8
	Does the educational software motivate you to be involved in computer programming?	4	3,7
State on e-learning	Does the educational software help you to understand the subject of computers in education?	4,4	3,95
	Does the educational software affect positively your perception about distance learning?	4,04	4
Eengagement in e-learning	Does the educational software motivate you to deal with distance education?	4,04	4
	Does the educational software motivate you to join other e-learning programs?	3,64	3,4

Questionnaire C

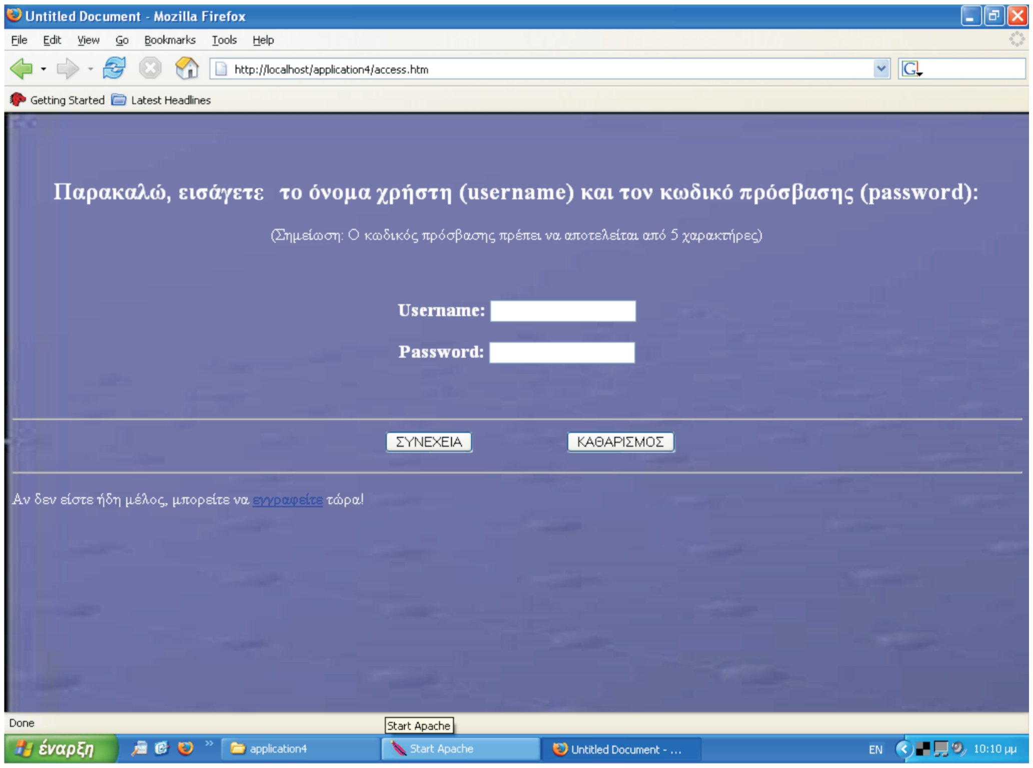
Questions	Evaluation Degree
Does the educational software help you understanding better the logic of programming?	3,82
Does the educational software help you to learn other programming languages?	3,16
Does the educational software help you in your studies?	3,62
Does the educational software help you understanding better other lessons of computer science?	3,49
Does the educational software help you in the elaboration of tasks and activities considering your studies?	3,29

Questionnaire D

Questions	Evaluation Degree
Do you think that your returns to a previous chapter in order to revise it are a waste of time?	1
Do the returns to a previous chapter correspond to your need for revision?	4
Does your return to a previous level, that happened each time the system discovered that you made errors of previous chapters, help you learning programming?	3.86

8. Screenshots of the ELaC

- **Registration and log-in**


Untitled Document - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/application4/access.htm
Getting Started Latest Headlines
Done
Start Apache
έναρξη
application4
Start Apache
Untitled Document - ...
EN
10:10 μμ

Παρακαλώ, εισάγετε το όνομα χρήστη (username) και τον κωδικό πρόσβασης (password):

(Σημείωση: Ο κωδικός πρόσβασης πρέπει να αποτελείται από 5 χαρακτήρες)

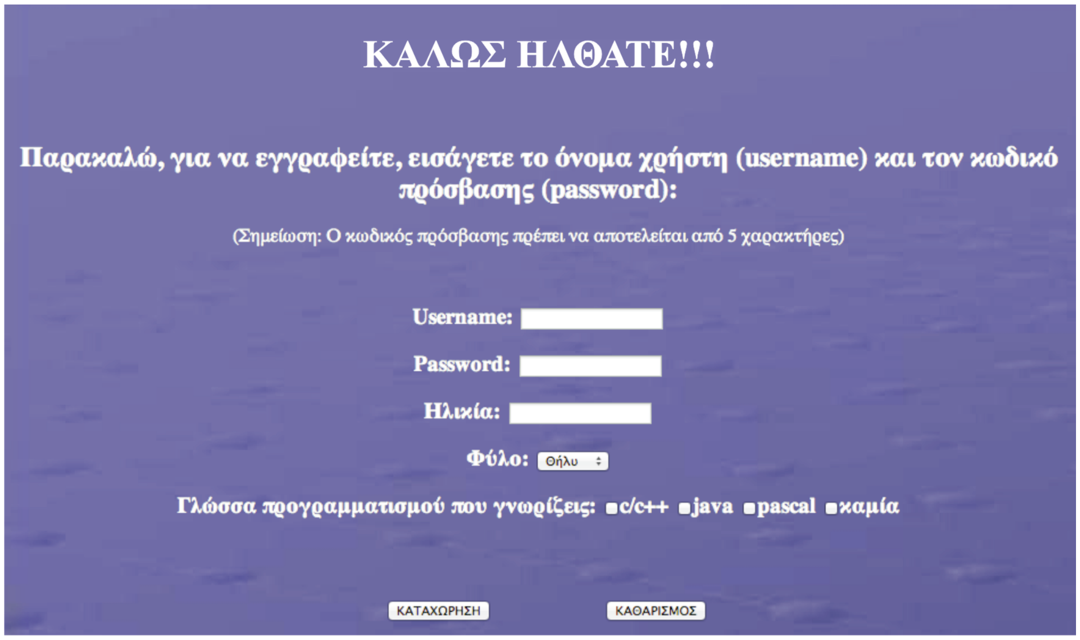
Username:

Password:

ΣΥΝΕΧΕΙΑ ΚΑΘΑΡΙΣΜΟΣ

Αν δεν είστε ήδη μέλος, μπορείτε να εγγραφείτε τώρα!

Figure 59: Log-in form



ΚΑΛΩΣ ΗΛΘΑΤΕ!!!

Παρακαλώ, για να εγγραφείτε, εισάγετε το όνομα χρήστη (username) και τον κωδικό πρόσβασης (password):

(Σημείωση: Ο κωδικός πρόσβασης πρέπει να αποτελείται από 5 χαρακτήρες)

Username:

Password:

Ηλικία:

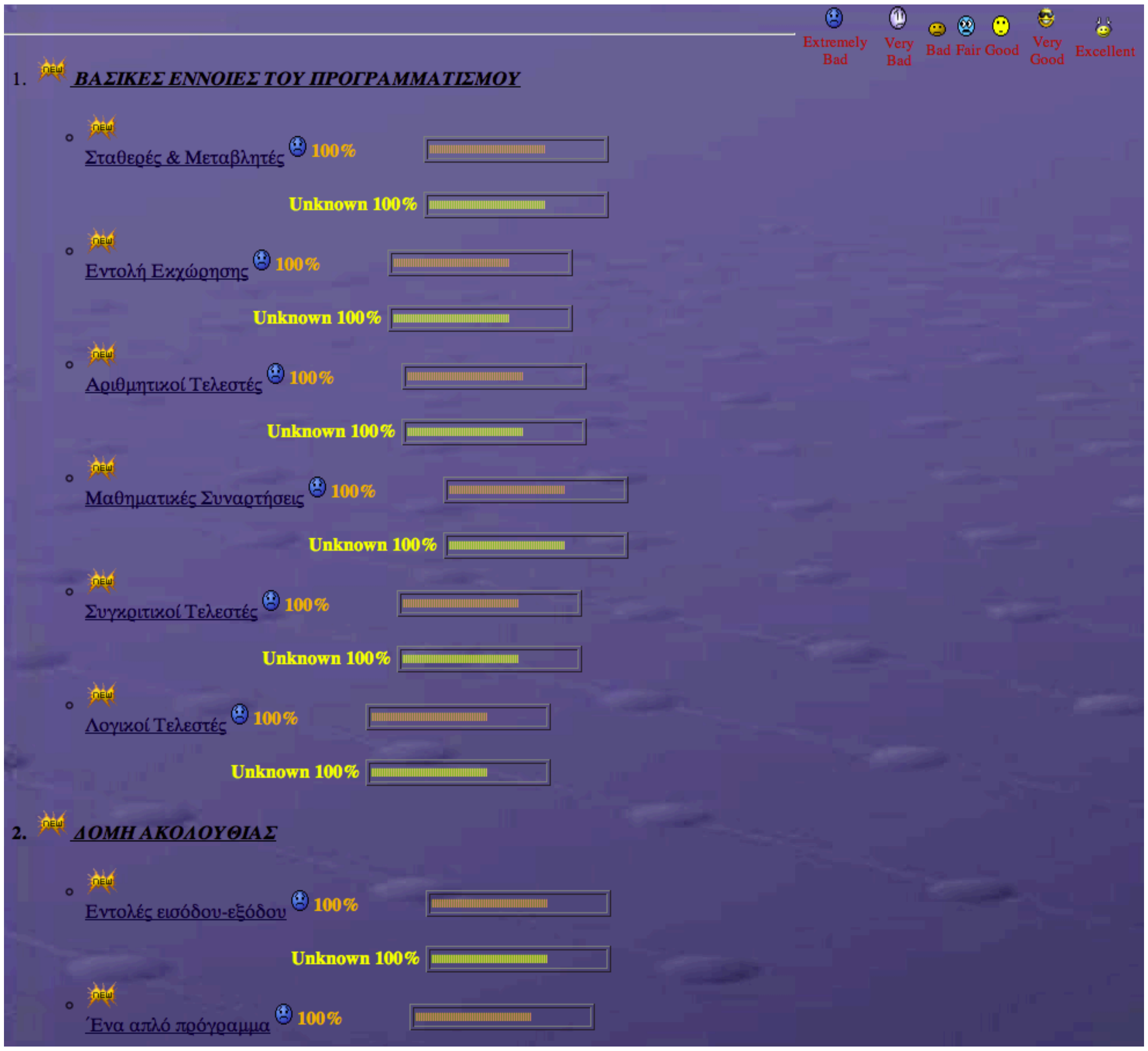
Φύλο: Θήλυ

Γλώσσα προγραμματισμού που γνωρίζεις: c/c++ java pascal καμία

ΚΑΤΑΧΩΡΗΣΗ ΚΑΘΑΡΙΣΜΟΣ

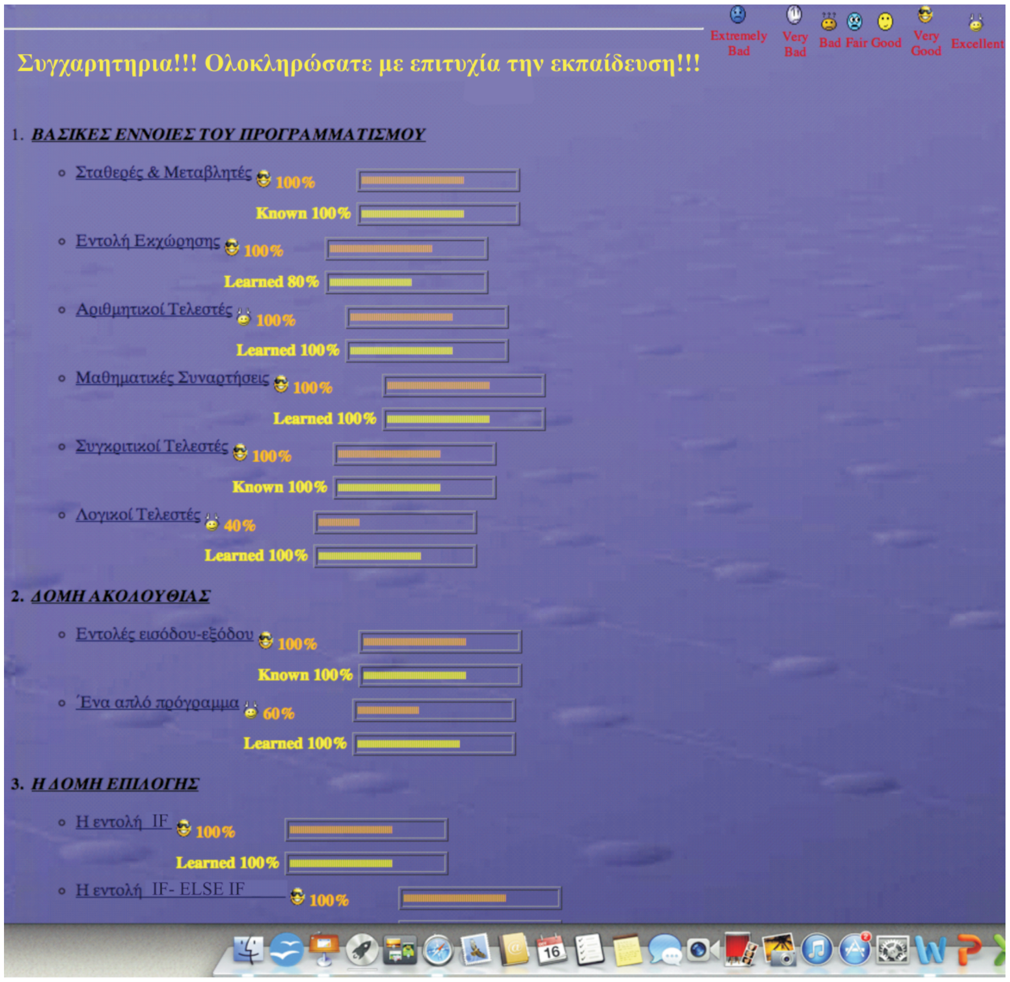
Figure 60: Registration form

- **Domain concepts**



1. ***ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ***
   - Σταθερές & Μεταβλητές 100%
     Unknown 100%
   - Εντολή Εκχώρησης 100%
     Unknown 100%
   - Αριθμητικοί Τελεστές 100%
     Unknown 100%
   - Μαθηματικές Συναρτήσεις 100%
     Unknown 100%
   - Συγκριτικοί Τελεστές 100%
     Unknown 100%
   - Λογικοί Τελεστές 100%
     Unknown 100%
2. ***ΔΟΜΗ ΑΚΟΛΟΥΘΙΑΣ***
   - Εντολές εισόδου-εξόδου 100%
     Unknown 100%
   - Ένα απλό πρόγραμμα 100%

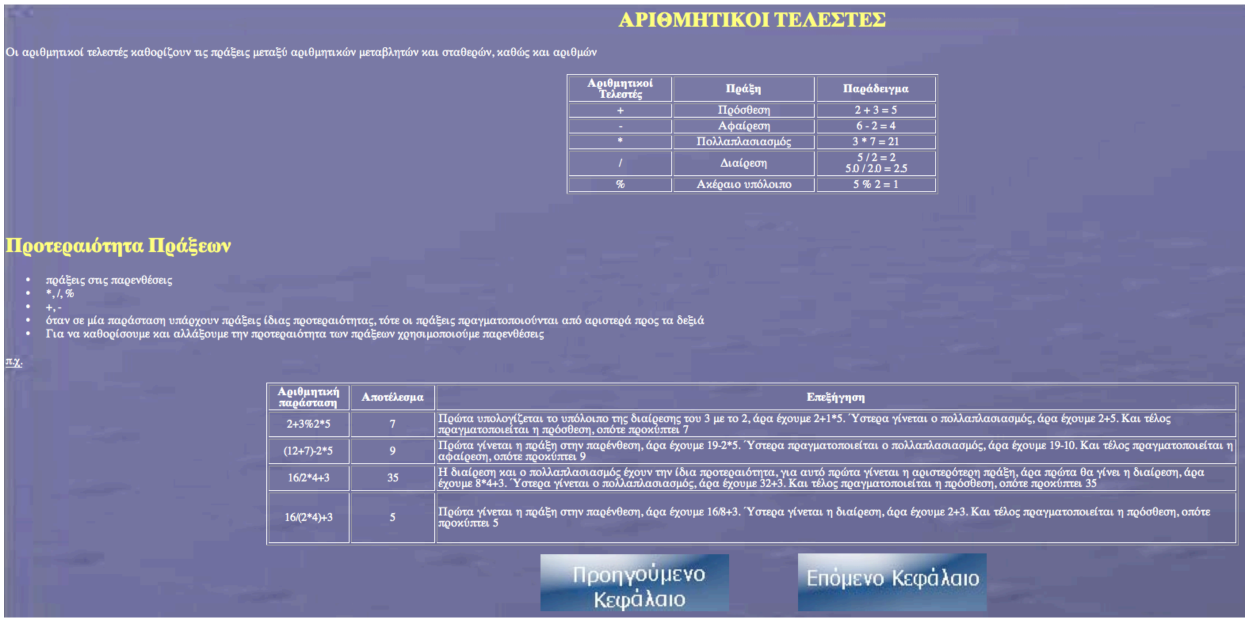
Figure 61: Domain concepts of knowledge stereotype 1



Συγχαρητηρια!!! Ολοκληρώσατε με επιτυχία την εκπαίδευση!!!

1. ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ
   - Σταθερές & Μεταβλητές 100%
     Known 100%
   - Εντολή Εκχώρησης 100%
     Learned 80%
   - Αριθμητικοί Τελεστές 100%
     Learned 100%
   - Μαθηματικές Συναρτήσεις 100%
     Learned 100%
   - Συγκριτικοί Τελεστές 100%
     Known 100%
   - Λογικοί Τελεστές 40%
     Learned 100%
2. ΔΟΜΗ ΑΚΟΛΟΥΘΙΑΣ
   - Εντολές εισόδου-εξόδου 100%
     Known 100%
   - Ένα απλό πρόγραμμα 60%
     Learned 100%
3. Η ΔΟΜΗ ΕΠΙΛΟΓΗΣ
   - Η εντολή IF 100%
     Learned 100%
   - Η εντολή IF- ELSE IF 100%

Figure 62: Successful completion of the learning process of all the learning material


ΑΡΙΘΜΗΤΙΚΟΙ ΤΕΛΕΣΤΕΣ

Οι αριθμητικοί τελεστές καθορίζουν τις πράξεις μεταξύ αριθμητικών μεταβλητών και σταθερών, καθώς και αριθμών

Αριθμητικοί Τελεστές	Πράξη	Παράδειγμα
+	Πρόσθεση	2 + 3 = 5
-	Αφαίρεση	6 - 2 = 4
*	Πολλαπλασιασμός	3 * 7 = 21
/	Διαίρεση	5 / 2 = 2 5.0 / 2.0 = 2.5
%	Ακέραιο υπόλοιπο	5 % 2 = 1

Προτεραιότητα Πράξεων

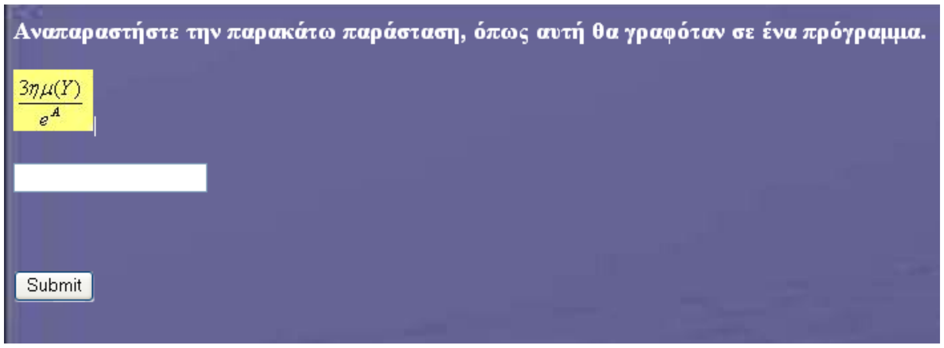
- πράξεις στις παρενθέσεις
- *, /, %
- +, -
- όταν σε μία παράσταση υπάρχουν πράξεις ίδιας προτεραιότητας, τότε οι πράξεις πραγματοποιούνται από αριστερά προς τα δεξιά
- Για να καθορίσουμε και αλλάξουμε την προτεραιότητα των πράξεων χρησιμοποιούμε παρενθέσεις

π.χ.

Αριθμητική παράσταση	Αποτέλεσμα	Επεξήγηση
2+3%2*5	7	Πρώτα υπολογίζεται το υπόλοιπο της διαίρεσης του 3 με το 2, άρα έχουμε 2+1*5. Ύστερα γίνεται ο πολλαπλασιασμός, άρα έχουμε 2+5. Και τέλος πραγματοποιείται η πρόσθεση, οπότε προκύπτει 7
(12+7)-2*5	9	Πρώτα γίνεται η πράξη στην παρένθεση, άρα έχουμε 19-2*5. Ύστερα πραγματοποιείται ο πολλαπλασιασμός, άρα έχουμε 19-10. Και τέλος πραγματοποιείται η αφαίρεση, οπότε προκύπτει 9
16/2*4+3	35	Η διαίρεση και ο πολλαπλασιασμός έχουν την ίδια προτεραιότητα, για αυτό πρώτα γίνεται η αριστερότερη πράξη, άρα πρώτα θα γίνει η διαίρεση, άρα έχουμε 8*4+3. Ύστερα γίνεται ο πολλαπλασιασμός, άρα έχουμε 32+3. Και τέλος πραγματοποιείται η πρόσθεση, οπότε προκύπτει 35
16/(2*4)+3	5	Πρώτα γίνεται η πράξη στην παρένθεση, άρα έχουμε 16/8+3. Ύστερα γίνεται η διαίρεση, άρα έχουμε 2+3. Και τέλος πραγματοποιείται η πρόσθεση, οπότε προκύπτει 5

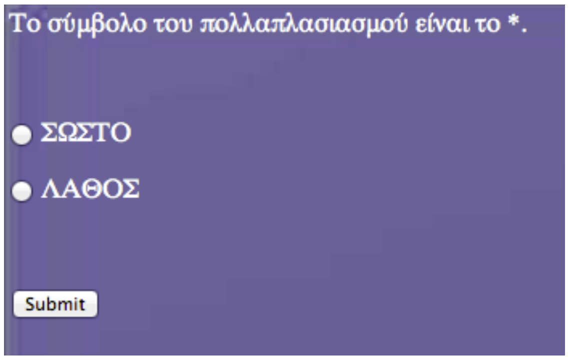
Figure 63: Arithmetic operators

- **Exercises and questions of tests**


Αναπαραστήστε την παρακάτω παράσταση, όπως αυτή θα γραφόταν σε ένα πρόγραμμα.

$$\frac{3\eta\mu(Y)}{e^{A}}$$

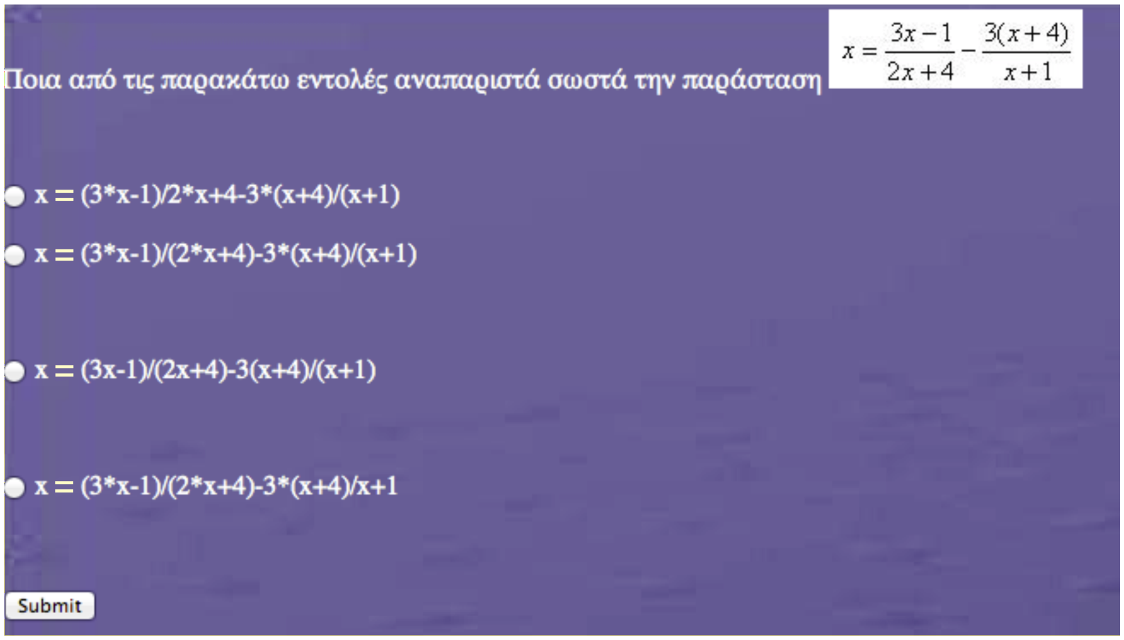
Figure 64: Fill in the gaps exercise


Το σύμβολο του πολλαπλασιασμού είναι το *.

● ΣΩΣΤΟ

● ΛΑΘΟΣ

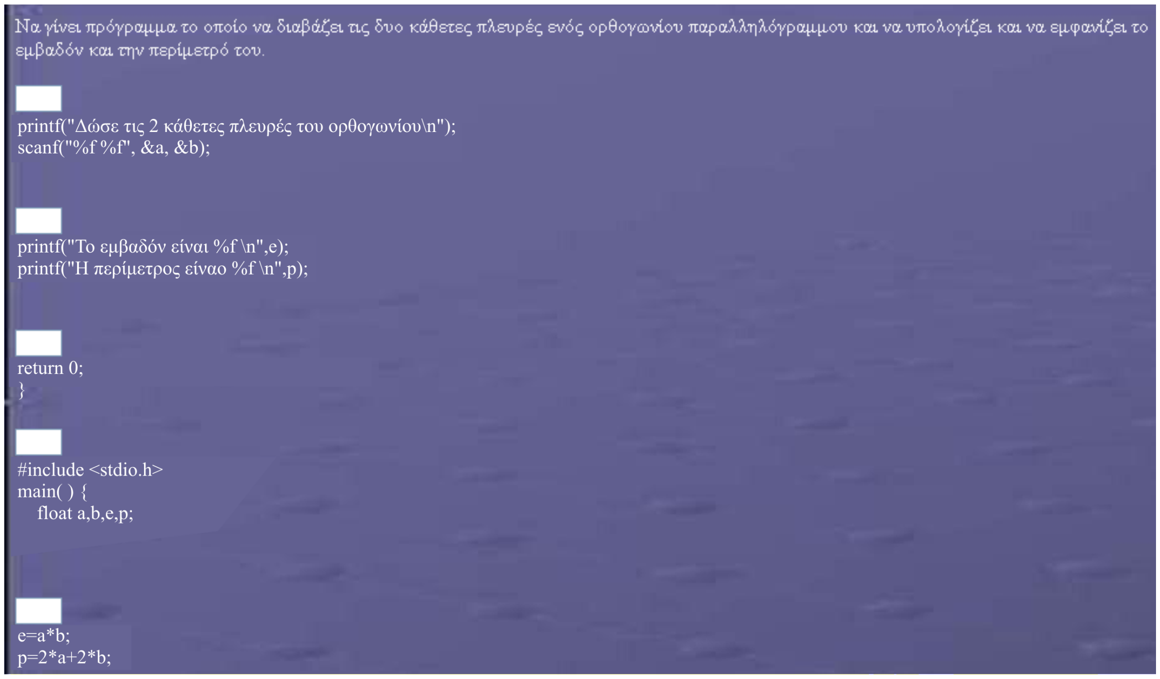
Figure 65: Right-wrong exercise



Ποια από τις παρακάτω εντολές αναπαριστά σωστά την παράσταση $x = \frac{3x-1}{2x+4} - \frac{3(x+4)}{x+1}$

- x = (3*x-1)/2*x+4-3*(x+4)/(x+1)
- x = (3*x-1)/(2*x+4)-3*(x+4)/(x+1)
- x = (3x-1)/(2x+4)-3(x+4)/(x+1)
- x = (3*x-1)/(2*x+4)-3*(x+4)/x+1

Figure 66: Multiple-choice exercise



Να γίνει πρόγραμμα το οποίο να διαβάζει τις δυο κάθετες πλευρές ενός ορθογωνίου παραλληλόγραμμου και να υπολογίζει και να εμφανίζει το εμβαδόν και την περίμετρό του.

```
printf("Δώσε τις 2 κάθετες πλευρές του ορθογωνίου\n");
scanf("%f %f", &a, &b);
```

```
printf("Το εμβαδόν είναι %f \n",e);
printf("Η περίμετρος είναο %f \n",p);
```

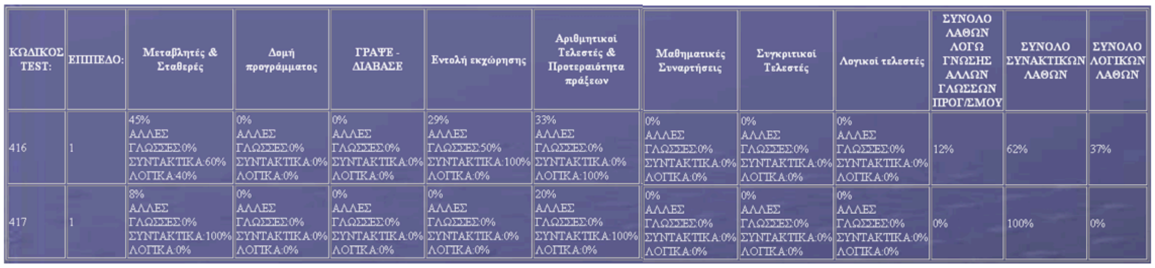
```
return 0;
}
```

```
#include <stdio.h>
main( ) {
    float a,b,e,p;
```

```
e=a*b;
p=2*a+2*b;
```

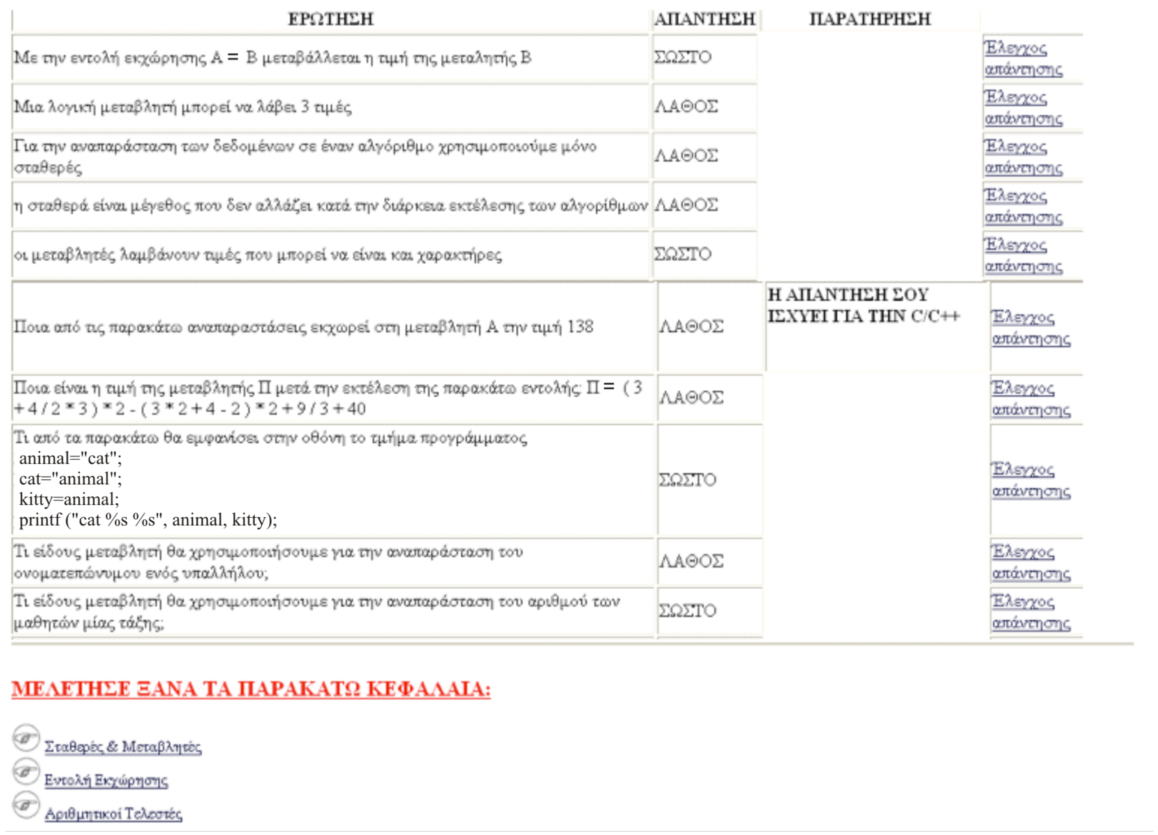
Figure 67: Put the algorithm pieces in the right order exercise

- **Results of the tests**



ΚΩΔΙΚΟΣ TEST:	ΕΠΙΠΕΔΟ:	Μεταβλητές & Σταθερές	Δομή προγράμματος	ΓΡΑΨΕ - ΔΙΑΒΑΣΕ	Εντολή εκχώρησης	Αριθμητικοί Τελεστές & Προτεραιότητα πράξεων	Μαθηματικές Συναρτήσεις	Συγκριτικοί Τελεστές	Λογικοί τελεστές	ΣΥΝΟΛΟ ΛΑΘΩΝ ΛΟΓΩ ΓΝΩΣΗΣ ΑΛΛΩΝ ΓΛΩΣΣΩΝ ΠΡΟΓ/ΣΜΟΥ	ΣΥΝΟΛΟ ΣΥΝΑΚΤΙΚΩΝ ΛΑΘΩΝ	ΣΥΝΟΛΟ ΛΟΓΙΚΩΝ ΛΑΘΩΝ
416	1	45% ΑΛΛΕΣ ΓΛΩΣΣΕΣ:0% ΣΥΝΤΑΚΤΙΚΑ:60% ΛΟΓΙΚΑ:40%	0% ΑΛΛΕΣ ΓΛΩΣΣΕΣ:0% ΣΥΝΤΑΚΤΙΚΑ:0% ΛΟΓΙΚΑ:0%	0% ΑΛΛΕΣ ΓΛΩΣΣΕΣ:0% ΣΥΝΤΑΚΤΙΚΑ:0% ΛΟΓΙΚΑ:0%	29% ΑΛΛΕΣ ΓΛΩΣΣΕΣ:50% ΣΥΝΤΑΚΤΙΚΑ:100% ΛΟΓΙΚΑ:0%	33% ΑΛΛΕΣ ΓΛΩΣΣΕΣ:0% ΣΥΝΤΑΚΤΙΚΑ:0% ΛΟΓΙΚΑ:100%	0% ΑΛΛΕΣ ΓΛΩΣΣΕΣ:0% ΣΥΝΤΑΚΤΙΚΑ:0% ΛΟΓΙΚΑ:0%	0% ΑΛΛΕΣ ΓΛΩΣΣΕΣ:0% ΣΥΝΤΑΚΤΙΚΑ:0% ΛΟΓΙΚΑ:0%	0% ΑΛΛΕΣ ΓΛΩΣΣΕΣ:0% ΣΥΝΤΑΚΤΙΚΑ:0% ΛΟΓΙΚΑ:0%	12%	62%	37%
417	1	8% ΑΛΛΕΣ ΓΛΩΣΣΕΣ:0% ΣΥΝΤΑΚΤΙΚΑ:100% ΛΟΓΙΚΑ:0%	0% ΑΛΛΕΣ ΓΛΩΣΣΕΣ:0% ΣΥΝΤΑΚΤΙΚΑ:0% ΛΟΓΙΚΑ:0%	0% ΑΛΛΕΣ ΓΛΩΣΣΕΣ:0% ΣΥΝΤΑΚΤΙΚΑ:0% ΛΟΓΙΚΑ:0%	0% ΑΛΛΕΣ ΓΛΩΣΣΕΣ:0% ΣΥΝΤΑΚΤΙΚΑ:0% ΛΟΓΙΚΑ:0%	20% ΑΛΛΕΣ ΓΛΩΣΣΕΣ:0% ΣΥΝΤΑΚΤΙΚΑ:100% ΛΟΓΙΚΑ:0%	0% ΑΛΛΕΣ ΓΛΩΣΣΕΣ:0% ΣΥΝΤΑΚΤΙΚΑ:0% ΛΟΓΙΚΑ:0%	0% ΑΛΛΕΣ ΓΛΩΣΣΕΣ:0% ΣΥΝΤΑΚΤΙΚΑ:0% ΛΟΓΙΚΑ:0%	0% ΑΛΛΕΣ ΓΛΩΣΣΕΣ:0% ΣΥΝΤΑΚΤΙΚΑ:0% ΛΟΓΙΚΑ:0%	0%	100%	0%

Figure 68: overall results-progress



ΕΡΩΤΗΣΗ	ΑΠΑΝΤΗΣΗ	ΠΑΡΑΤΗΡΗΣΗ	
Με την εντολή εκχώρησης A = B μεταβάλλεται η τιμή της μεταλητής B	ΣΩΣΤΟ		Έλεγχος απάντησης
Μια λογική μεταβλητή μπορεί να λάβει 3 τιμές	ΛΑΘΟΣ		Έλεγχος απάντησης
Για την αναπαράσταση των δεδομένων σε έναν αλγόριθμο χρησιμοποιούμε μόνο σταθερές	ΛΑΘΟΣ		Έλεγχος απάντησης
η σταθερά είναι μέγεθος που δεν αλλάζει κατά την διάρκεια εκτέλεσης των αλγορίθμων	ΛΑΘΟΣ		Έλεγχος απάντησης
οι μεταβλητές λαμβάνουν τιμές που μπορεί να είναι και χαρακτήρες	ΣΩΣΤΟ		Έλεγχος απάντησης
Ποια από τις παρακάτω αναπαραστάσεις εκχωρεί στη μεταβλητή A την τιμή 138	ΛΑΘΟΣ	Η ΑΠΑΝΤΗΣΗ ΣΟΥ ΙΣΧΥΕΙ ΓΙΑ ΤΗΝ C/C++	Έλεγχος απάντησης
Ποια είναι η τιμή της μεταβλητής Π μετά την εκτέλεση της παρακάτω εντολής: Π = (3 + 4 / 2 * 3) * 2 - (3 * 2 + 4 - 2) * 2 + 9 / 3 + 40	ΛΑΘΟΣ		Έλεγχος απάντησης

Τι από τα παρακάτω θα εμφανίσει στην οθόνη το τμήμα προγράμματος
animal="cat";
cat="animal";
kitty=animal;
printf ("cat %s %s", animal, kitty);
	ΣΩΣΤΟ		Έλεγχος απάντησης

Τι είδους μεταβλητή θα χρησιμοποιήσουμε για την αναπαράσταση του ονοματεπώνυμου ενός υπαλλήλου;	ΛΑΘΟΣ		Έλεγχος απάντησης
Τι είδους μεταβλητή θα χρησιμοποιήσουμε για την αναπαράσταση του αριθμού των μαθητών μίας τάξης;	ΣΩΣΤΟ		Έλεγχος απάντησης

ΜΕΛΕΤΗΣΕ ΞΑΝΑ ΤΑ ΠΑΡΑΚΑΤΩ ΚΕΦΑΛΑΙΑ:

Σταθερές & Μεταβλητές
Εντολή Εκχώρησης
Αριθμητικοί Τελεστές

Figure 69: Results of the test

Conclusions and contribution to the science

The target in this research was to create a novel approach that combines fuzzy logic techniques for offering individualized instruction and personalized support in adaptive educational systems. The presented system provides adaptation of the instructional material, taking into account the individuality of learners in terms of background, skills and pace of learning. It automatically models the learning or forgetting process of a student. In particular, the system advises the student model and constructs automatically state-chart diagrams to keep track of cognitive state transitions of learners wit respect to their progress or no-progress. Thereby, it reveals if a student learns or not, if s/he forgets and reasons these states. Therefore, the system allows each individual learner to complete the e-learning course at their own pace, taking decisions about the concepts of the learning material that have to be delivered to them, the concepts that need revision and the concepts that are known and do not need further reading. In this way, the system helps learners to save time and effort during the learning process.

The operation of the system is based on the knowledge domain representation that is implemented through a combination of hierarchical trees and Fuzzy Cognitive Maps. This kind of knowledge domain representation helps to manage to represent either the order in which the domain concepts of the learning material have to be taught and organized, or the knowledge dependencies that exist among the domain concepts. This is significant because the knowledge level of a domain concept increases or decreases due to changes on the knowledge level of a related domain concept. The presented knowledge domain representation approach constitutes a driver for an adaptive and/or personalized tutoring system for delivering the learning material to each individual learner dynamically, taking into account her/his learning needs and different learning pace. The design of the learning material and the definition of the individual domain concepts that it includes, are based on the knowledge and experience of domain experts. Furthermore, the contribution of domain experts is significant for the definition of the knowledge dependencies that exist among the domain concepts of the learning material and their “strength o impact” on each other. Evidently, the knowledge domain representation has to be combined with a well-designed student model, which will be responsible for how the system will utilize the information that is included in the knowledge domain module in order to make the right decisions for offering personalized instruction and support.

The main innovation of the presented approach is the student model. It is a hybrid student model that combines an overlay model and stereotypes with fuzzy logic techniques. In particular, the student model is based on an overlay model, which represents the knowledge level of a learner. The determination of the student's knowledge level of each domain concept, as well as the updating of the student model are based on the fuzzy logic technique that have been incorporated into the student model. Fuzzy sets are used in order to describe how well each individual domain concept is known and learned. In addition, the student model includes a mechanism of rules over the fuzzy sets, which is triggered after any change of the value of the knowledge level of a domain concept and updates the values of the knowledge level of all the related domain concepts with this. According to the learner's knowledge level and errors, the system attached her/him to the appropriate stereotype. The transition of a learner from one stereotype to another depicts the state of the learner. In other words, the transition of a learner from one stereotype to another reveals if s/he has learned or not a domain concept, if s/he has forgot a concept or if s/he has assimilated it.

The presented novel approach of knowledge domain representation and student modeling has been fully implemented in a web-based educational application, which teaches the programming language 'C'. The student model of the particular system has two layers. The first layer includes a weighted overlay model. The second layer includes a three-dimensional stereotype model. The first dimension consists of eight stereotypes that represent the learner's knowledge level; the second dimension consists of two stereotypes that concern the type of programming errors (logical or syntactic) and the third dimension concerns prior knowledge of the student on other programming languages.

Due to the fact that student’s actions, misconceptions and needs are imprecise information, fuzzy logic has been chosen to manage the uncertainty and to describe human descriptions of knowledge and of student’s cognitive abilities. Therefore, the incorporated fuzzy technique is responsible for the determination of the student's knowledge level of each domain concept, the updating of the student model and the decision-making about the instruction model that the system has to follow for each individual learner. The system identifies the alterations on the state of students' knowledge level, recognizes the misconceptions and needs of a learner, and reasons them. It tracks the cognitive state transitions of learners by constructing automatically state-chart diagrams. Thereby, the system recognizes if a student learns or not, if s/he reads or not, if s/he has difficulty in understanding, if s/he forgets, if s/he has confused with other programming languages that s/he has previously learned.

The implemented novel educational system that teaches the programming language 'C' has been evaluated. In particular, an evaluation method, which is called PeRSIVA, has been used to assess the effectiveness, accuracy and validity of the presented student model. PeRSIVA can be applied for the evaluation of the student model of any adaptive and/or personalized tutoring system, since it has been designed under the framework of two well-known evaluation methods: the Kirkpatrick’s model and the layered evaluation framework for student model assessing.

PeRSIVA’s application was based on close-ended questionnaires and on experimental research. The questionnaire survey was performed in two stages. In particular, two questionnaires were answered immediately after the end of the training program, while the other two questionnaires were answered two years later. The two years waiting time for the follow-up evaluation could have as a result the responses to have affected by students’ personal factors. It is known that there is no objective way to deal with it. However, the large amount of students (53) of the experimental group, their answers in the questionnaires of the first stage and the objective experimental research enhance the evaluation results.

The system's evaluation revealed that the combination of fuzzy sets with overlay model and stereotypes contributes, significantly, to the adaptation of the learning process to the learning pace of each individual learner. The results of the evaluation demonstrated learning improvements in students and adaptation success to students' needs. They revealed that the presented novel approach of student modeling improves the efficiency of the adaptation of the instructional process; by adapting instantly the sequence of learning lessons, as well as it helps learners to succeed a better performance. It was showed that the presented novel approach of student modeling improves significantly the student's performance. Furthermore, the majority of the learners were very satisfied with the educational program. They obtained a more positive state and behavior towards computer programming and distance learning. The assessment results showed, also, that the e-learning program helped learners to their further studies satisfactory. Moreover, the presented student model improves the validity of both, the conclusions that are drawn by the system concerning the aspects of students' characteristics and the adaptation decision-making. Consequently, the presented novel hybrid student model contributes significantly to the adaptation process and helps the system to provide a personalized and effective educational process for learning of computer programming.

The ability of the presented system to recognize the alteration of the student's learning states and dynamically adapt the presentation of the learning material accordingly, renders the particular approach a novel generic tool for adaptive learning. The encouraging results motivate the implementation of the presented novel approach of tutoring system that combines fuzzy techniques to educational environments of knowledge domains other than computer programming. Indeed, it is within the future plans of the author to develop an authoring tool that will integrate the presented adaptation and reasoning mechanism and will be used for the creation of adaptive tutoring systems in a variety of domains.

Furthermore, the model that was presented in section B is applicable to systems, in which the user's changeable state and/or preferences are affected by the existing dependencies among the system's elements (like concepts, preferences, events, choices). Thereafter, the particular model could be implemented in adaptive systems other than adaptive tutoring system. For example, it could be used in an e-shop, where the preference of an online shopper for particular products can be used in order to guess and propose her/him other products that the user is likely to be interested in. In the table 37 the correlation of an e-shop and an adaptive e-learning system is presented concerning the thesis' model.

Table 35: Correlation of an e-shop and an adaptive e-learning system concerning the application of the thesi's model

	e-shop	e-learning
Nodes	Products	Domain concepts
Arcs	Preferences' dependencies	Knowledge dependencies
Fuzzy Sets	Descriptions of a preference (e.g. 'uninterested', 'interested', 'liked', 'preferred')	Descriptions of knowledge level (e.g. 'unknown', 'insufficiently known', 'known', 'learned')
Changeable states	Preferences	Knowledge level

Below the contribution of this research to science is described:

- **Contribution to Intelligent Tutoring Systems**

  The novelty of the research to the technology of Intelligent Tutoring Systems renders to the addition of two novel modules in the typical ITS architecture. These two modules are the cognitive state transitions module that constructs automatically state-chart diagrams to keep track of cognitive state transitions with respect to their progress or non-progress; and the fuzzy knowledge definer module that uses fuzzy logic techniques to identify and update the overall student's knowledge level, after a change has occurred on her/his knowledge level of a domain concept. These two modules improve the adaptivity of a tutoring system, since they allow to the system to model automatically the learning and forgetting process.

- **Contribution to Knowledge domain Representation**

The knowledge domain representation is the base for the representation of the learner's knowledge, which is usually performed as a subset of the knowledge domain. However, the representation of the learner's knowledge is a moving target. The student's knowledge level of a domain concept usually is affected by her/his knowledge level of other related domain concepts. For example, if a learner excels at a domain concept, it implies that s/he does not need to read some other relative domain concepts or that a depended domain concept is already all or partly known for her/his. Furthermore, if a learner has misconceptions on a domain concept, it implies that s/he has to revise a prerequisite relative domain concept. That is the reason for using FCMs to represent the knowledge dependencies that exist among the domain concepts of the knowledge domain material, as well as the "strength of impact" of them on each other.

The combination of hierarchical trees and FCMs, which is used, manages to represent the knowledge level dependencies among the domain concepts of the learning material, except of the logical relationships among them. This is significant because the knowledge level of a domain concept may increase or reduce due to changes on the knowledge level of a related domain concept. The particular knowledge domain representation approach helps the system to recognize either the domain concepts that are already partly or completely known for a learner or the domain concepts that s/he has forgot, taking into account the learner's knowledge level of the related concepts of the learning material. Therefore, the presented knowledge domain representation approach contributes to the improvement of the navigation support that an adaptive and/or personalized learning system provides. It constitutes an ideal way for representing the knowledge domain of an adaptive and/or personalized tutoring system in a more realistic way. It constitutes a driver for the student model of an adaptive and/or personalized system for delivering the learning material to each individual learner dynamically, taking into account her/his learning needs and her/his different learning pace.

- **Contribution to Student Modeling**

The target of this research was to show how fuzzy sets can be combined with user stereotypes and the overlay model to promote adaptivity and personalization in educational applications. The system's evaluation revealed that the combination of fuzzy sets with overlay and stereotypes models contributes significantly to the adaptation of the learning process to the learning pace of each individual learner. This approach improves the efficiency of the adaptivity of the instructional process, identifying each time the appropriate domain concepts that correspond to the learner's knowledge level and educational needs. In this way, the presented novel approach helps the learners that already know aspects of computer programming to save time and effort during the learning process. Furthermore, the presented approach helps the system to recognize the students’ state of forgetting what they have already learned in previous interactions, and adapts the presentation of material accordingly. It leads the educational system to make inferences about the changes of the students’ knowledge level and thus make useful adaptation decisions, offering personalized instruction and support.

The ability of the presented novel hybrid student model to recognize the alterations of the student’s learning states and dynamically adapt the presentation of the learning material accordingly, renders the particular student modeling approach a novel generic tool for adaptive learning. The gain from the presented approach is significant as fuzzy logic can be used in combination with overlay and stereotype models to provide adaptivity and personalization in other interactive systems in addition to educational applications. The application of this innovative approach is possible where the user’s changeable state and/or preferences are affected by the existing dependencies among the system’s elements (like concepts, preferences, events, choices).

- **Contribution to programming tutoring systems**

Programming tutoring systems teaches computer programming to learners providing adaptivity. Mainly, these systems adapt the learning process dynamically to the student's knowledge level and needs. However, they do not consider how the learner's performance in a domain concept affects the learner's knowledge level of other related domain concepts of the learning material. They do not model the learning or forgetting process of a student. Consequently, the gain of the presented approach is that allows each learner to complete the e-training course at their own pace giving the ability to the system to adapt dynamically to each individual learner’s needs by scheduling the sequence of lessons instantly. In particular, the presented approach of this research gives to the tutoring system the ability to recognize the learner's knowledge level and the changes that occur in the state of her/his knowledge of a domain concept and update the student's overall knowledge level according to the knowledge dependencies between the learning material’s domain concepts and the learner's progress. The presented system recognizes when a new domain concept is completely unknown to the learner, or when it is partly known due to the learner having previous related knowledge. Furthermore, it recognizes when a previously-known domain concept has been completely or partly forgotten by the learner. Thus it models either the possible increase or decrease of the learner’s knowledge. Furthermore, each time it checks if the learner's errors were due to possible confusion with features of another previously-known programming language. In this way, the system allows each learner to complete the e-learning course at their own pace, taking decisions about which concepts have to be delivered, which concepts need revision and which concepts are known and do not need further reading.

- **Contribution to fuzzy logic**

  The presented adaptation and reasoning approach combines fuzzy theory with the overlay model. Moreover, it employs a novel inference mechanism that dynamically updates user stereotypes using fuzzy sets. It should be noted that the overlay model and stereotypes constitute two widely used methods for user modeling. The gain from this novel combination is significant as the students' level of knowledge is represented in a more realistic way by automatically modeling the learning or forgetting process of a student with respect to the FCMs and thus the system can provide individualized adaptive advice. The application of this approach is not limited to adaptive instruction, but it can also be used in other systems with changeable user states, such as e-shops, where consumers' preferences change over the time and affect one another. For example, it can be used to reduce customer information overload by recommending products that are likely to be of interest to them, considering their preferences and the dependencies that exist between products' choices (in accordance to e-learning, users' preferences correspond to users' knowledge level and products' choices correspond to the domain concepts). Therefore, the particular module constitutes a novel generic fuzzy tool, which offers dynamic adaptation to users' needs and preferences of adaptive systems.

Further research

This Ph. D. thesis describes a novel approach of fuzzy student model that automatically models the learning or forgetting process of a student, allowing her/him to complete the educational program in her/his own abilities and pace. The encouraging results of the application and implementation of the particular approach to an integrated e-learning environment for computer programming language ‘C’, motivate the implementation of the particular model to educational environments of other knowledge domains, also. Therefore, future work includes the development of an authoring tool, which will allow the construction of e-learning courses of any knowledge domain that will integrate the adaptation and student modeling techniques of the presented approach. Furthermore, as it is referred to the conclusions, the presented fuzzy student model is applicable to systems in which the user’s changeable state and/or preferences are affected by the existing dependencies among the system’s elements (like concepts, preferences, choices). As a consequence, the proposed authoring tool can be extended to provide the ability to construct modules, which integrate the presented novel approach for adaptive systems other than adaptive tutoring systems (like e-shops).

Another interesting field of further research is to extend the presented novel fuzzy approach making it able to model affective and meta-cognitive learner’s characteristics, also. In particular, motivation, self-efficacy and emotional state of the student’s affect the learning process and the student’s performance. Therefore, it would be very interested to add more student modeling techniques to the presented approach in order to model more learners’ characteristics.

Furthermore, the integration of artificial neural networks is an interesting project for future work. The integration of neural networks to the presented student model will allow the system to learn about the learner’s cognitive states, the transitions between them and the reasons for these. In other words, the system will simulate the thinking and decision-making processes of an expert. Thereby, the system will be able to make more valid adaptation decisions.

References

Aguilar, J. (2005). A survey about fuzzy cognitive maps papers (Invited Paper). *International Journal of Computational Cognition*, 3 (2), 27-33.

Albano, G. (2011). Knowledge, Skills, Competencies: A Model for Mathematics E-Learning. *In Proc. of the 18th International Conference on Telecommunications*, Ayia Napa, Cyprus (pp. 214-225).

Alepis, E., and Virvou, M. (2011). Automatic generation of emotions in tutoring agents for affective e-learning in medical education. *Expert Systems with Applications*, 38, 9840-9847.

Alepis, E., Virvou, M., and Kabassi., K. (2008). Mobile education: Towards affective bi-modal interaction for adaptivity. *In Proc. of the 3rd International Conference on Digital Information Management, ICDIM 2008*, London (pp. 51-56).

Aleven, V., McLaren, B.M., and Sewall, J. (2009). Scaling Up Programming by Demonstration for Intelligent Tutoring Systems Development: An Open-Access Web Site for Middle School Mathematics Learning. *IEEE Transactions on Learning Technologies*, 2 (2), 64-78.

Al-Hmouz, A., Shen, J., Yan, J., and Al-Hmouz, R. (2010). Enhanced Learner Model for Adaptive Mobile Learning. *In Proc. of the 12th International Conference on Information Integration and Web-based Applications & Services*, Paris, France (pp. 783-786).

Al-Hmouz, A., Shen, J., Yan, J., and Al-Hmouz, R. (2011). Modeling mobile learning system using ANFIS*. In Proc. of the 11th IEEE International Conference on Advanced Learning Technologies (ICALT 2011)*, Athens, USA (pp. 32-36).

Alves, P., Amaral, L., and Pires, J. (2008). Case-Based Reasoning Approach to Adaptive Web-Based Educational Systems. *In Proc. Eighth IEEE International Conference on Advanced Learning Technologies (ICALT '08)*, Santander, Cantabria, Spain (pp. 260-261).

Anderson, J. R., and Reiser, B. J. (1985) The lisp tutor: it approaches the effectiveness of a human tutor. *Byte*, 10(4), 159-175.

Anderson, J. R., Boyle, D. G., and Reiser, B. J. (1985). Intelligent tutoring systems. Science, 228, 456-462.

Anderson, J. R., Boyle, D. F., and Yost, G. (1985) The geometry tutor. In Proc. of the 9th International Joint Conference on Artificial Intelligence, Los Angeles, CA (pp. 1-7).

Anderson, J. R., Corbett, A. T., Koedinger, K. R., and Pelletier, R. (1995). Cognitive Tutors: Lessons Learned. *The Journal of the Learning Sciences*, 4 (2), 167-207.

Arnau, D., Arevalillo-Herráez, M., Puig, L., and González-Calero, J.A. (2013). Fundamentals of the design and the operation of an intelligent tutoring system for the learning of the arithmetical and algebraic way of solving word problems. *Computers & Education*, 63, 119–130.

Arroyo, I., Meheranian, H., and Woolf, B.P. (2010). Effort-based tutoring: An empirical approach to intelligent tutoring. *In Proc. of the 3rd International Conference on Educational Data Mining*, Pittsburgh, PA, USA (pp.1-10).

Azadeh, A., Ziaei, B., and Moghaddam, M. (2012). A hybrid fuzzy regression-fuzzy cognitive map algorithm for forecasting and optimization of housing market fluctuations. *Expert Systems with Applications*, 39 (1), 298-315.

Bai, S. M., and Chen, S. M. (2006a). Automatically constructing grade membership functions for students' evaluation for fuzzy grading systems. *In Proceedings of the 2006 world automation congress*, Budapest, Hungary.

Bai, S. M., and Chen, S. M. (2006b). A new method for students' learning achievement using fuzzy membership functions. *In Proceedings of the 11th conference on artificial intelligence*, Kaohsiung, Taiwan, Republic of China.

Bai, S. M., and Chen, S. M. (2008). Evaluating students' learning achievement using fuzzy membership functions and fuzzy rules. *Expert Systems with Applications*, 34, 399–410.

Badaracco, M., and Martinez, L. (2011). An intelligent tutoring system architecture for competency-based learning. *In Proc. of the 15th international conference on Knowledge-Based and Intelligent Information and Engineering Systems,* Kaiserslautern, Germany (pp. 124–133).

Badaracco, M., and Martinez, L. (2013). A fuzzy linguistic algorithm for adaptive test in Intelligent Tutoring System based on competences. *Expert Systems with Applications*, 40 (8), 3073–3086.

Baghaei, N., Mitrovic, A., and Irwin, W. (2005). A constraint-based tutor for learning Object-Oriented analysis and design using UML. *Proc. Int. Conf. on Computers in Education 2005*, Singapore (pp. 9-16).

Baker, R.S. (2007). Modeling and Understanding Students' Off-Task Behavior in Intelligent Tutoring Systems. *In Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*, San Jose, California, USA (pp. 1059-1068).

Baker, R.S., Corbett, A.T., Koedinger, K.R., and Wagner, A.Z. (2004). Off-Task Behavior in the Cognitive Tutor Classroom: When Students 'Game the System. *In Proc. of the SIGCHI Conf. on Human Factors in Computing Systems (CHI '04)*, Vienna, Austria (pp. 383-390).

Baker, R.S., Goldstein, A.B., and Heffernan, N.T. (2010). Detecting the Moment of Learning. *In Proc. of the ACM International Conference on Interactive Tabletops and Surfaces*, Saarbrücken, Germany (pp. 25–34).

Balakrishnan, A. (2011). On Modeling the Affective Effect on Learning. *In Proc. of the 5th international conference on Multi-Disciplinary Trends in Artificial Intelligence*, Hyderabald, India (pp. 225-235).

Barak, M. (2010). Motivating self-regulated learning in technology education. *International Journal of Technology and Design Education*, 20 (4), 381–401.

Baschera, G.M., and Gross, M. (2010). Poisson-Based Inference for Perturbation Models in Adaptive Spelling Training. *International Journal of Artificial Intelligence in Education*, 20, 1–3.

Baykasoglu, A., Durmusoglu, Z.D.U., and Kaplanoglu, V. (2011). Training Fuzzy Cognitive Maps via Extended Great Deluge Algorithm with applications. *Computers in Industry*, 62 (2), 187-195.

Beena, P., and Ganguli, R. (2011). Structural Damage Detection using Fuzzy Cognitive Maps and Hebbian Learning. *Applied Software Computing*, 11 (1), 1014-1020.

Bishop ,C.C., and Wheeler, D. (1994). The Myers-Briggs Personality Type and Its Relationship to Computer Programming. *Journal of Research on Computing in Education*, 26, 358–371.

Biswas, R. (1995). An application of fuzzy sets in students' evaluation. *Fuzzy Sets and Systems*, 74(2), 187–194.

Bloom, B. S. (1956). *Taxonomy of Educational Objectives*. Handbook I: The Cognitive Domain. New York: David McKay Co Inc.

Bloom, B. S. (1984). The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher*, 13 (6), 4–16.

Bontcheva, K., and Wilks, Y. (2005). Tailoring Automatically Generated Hypertext. *User Modeling and User-Adapted Interaction*, 15 (1-2), 135 – 168.

Bourdeau, J., and Grandbastien, M. (2010). Modeling tutoring knowledge. In R. Nkambou, J. Bourdeau, & R. Mizoguchi (Eds.). *Advances in intelligent tutoring systems* (pp. 123–143).

Breuker, J., and Muntjewerff, A. (1999). Ontological Modeling for Designing Educational Systems. *In Proc. of the Workshop on Ontologies for Intelligent Educational Systems, 9th International Conference on Artificial Intelligence in Education (AIED 1999)*, Le Mans, France.

Brown, J. S., and Burton, R. R. (1975). Multiple representation of knowledge for tutorial reasoning. In D. Bobrow, & A. Collins (Eds.), *Representation and Understanding: Studies in Cognitive Science*.

Brusilovsky, P, and Anderson, J. (1998). ACT-R electronic bookshelf: An adaptive system for learning cognitive psychology on the Web. *In Proc. of the WebNet'98, World Conference of the WWW, Internet, and Intranet*, Orlando, Florida, USA, (pp. 92-97).

Brusilovsky, P., and Millán, E. (2007). User models for adaptive hypermedia and adaptive educational systems. In P. Brusilovsky, A. Kobsa, & W. Neidl (eds.), *The Adaptive Web: Methods and Strategies of Web Personalization* (pp. 3-53).

Brusilovsky, P. Karagiannidis, C., and Sampson, D. (2004). Layered evaluation of adaptive learning systems. *Int. J. Cont. Engineering Education and Lifelong Learning*, 14(4/5), 402-421.

Bunt, A., and Conati, C. (2003). Probabilistic Student Modelling to Improve Exploratory Behaviour. *User Modeling and User-Adapted Interaction*, 13(3), 269-309.

Burstein, M. H., and Collins, A.M. (1988). Modeling a theory of Human Plausible Reasoning. In T. O'Shea and V. Sgurev (eds), *Artificial Intelligence III: Methodology, Systems, Applications* (pp. 21-28).

Burstein, M. H., Collins, A., and Baker, M. (1991). Plausible Generalisation: Extending a model of Human Plausible Reasoning. *Journal of the Learning Sciences*, 3, 319-359.

Butz, C.J., Hua, S., and Maguire, R.B. (2006). A Web-based Bayesian Intelligent Tutoring System for Computer Programming. *Web Intelligence and Agent Systems*, 4 (1), 77–97.

Butz, B.P., Duarte, M., and Miller, S.M. (2006). An intelligent tutoring system for circuit analysis. *IEEE Transactions on Learning Technologies*, 49 (2), 216-223.

Carmona, C., and Conejo, R. (2004). A learner model in a distributed environment*. In Proc. of the 3rd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH'2004)*, Eindhoven, the Netherlands (pp. 353-359).

Carmona, C., Castillo, G., and Millán, E. (2008). Designing a Dynamic Bayesian Network for Modeling Students' Learning Styles. *In Proc. of the 8th IEEE International Conference on Advanced Learning Technologies (ICALT 2008)*, Santander, Cantabria, Spain (pp. 346-350).

Carver, R., and Nash, J.G. (2009). *Doing Data Analysis with SPSS*. United States: Cengage Learning, Inc.

Castillo, O., and Melin, P. (2004). Adaptive noise cancellation using type-2 fuzzy logic and neural networks. *In Proc. of IEEE FUZZ Conference*, Budapest, Hungary (pp. 1093-1098).

Castillo, O., and Melin, P. (2008). Intelligent systems with interval type-2 fuzzy logic. *International Journal of Innovative Computing, Information and Control*, 4 (4), 771-783.

Castillo, O., Huesca, G., and Valdez, F. (2005). Evolutionary computing for optimizing type-2 fuzzy systems in intelligent control of non-linear dynamic plants. *In Proc. of North American Fuzzy Information Processing Society (NAFIPS)*, Ann Arbor, MI (pp. 247–251).

Castillo, G., Gama, J., and Breda, A.M. (2009). An Adaptive Predictive Model for Student Modeling. *In Advances in Web-Based Education: Personalized Learning Environments*. USA: Information Science Publishing, 2009, Chapter IV, pp. 70-92.

Cetintas, S., Si, L., Xin, Y.P., and Hord C. (2010). Automatic Detection of Off-Task Behaviors in Intelligent Tutoring Systems with Machine Learning Techniques. *IEEE Transactions on Learning Technologies*, 3 (3), 228-236.

Chang, D. F., and Sun, C. M. (1993). Fuzzy assessment of learning performance of junior high school students. *In Proc. of the 1993 first national symposium on fuzzy theory and applications*, Hsinchu, Taiwan, Republic of China (pp. 1–10).

Chen, S. M., and Lee, C. H. (1999). New methods for students' evaluating using fuzzy sets. *Fuzzy Sets and Systems*, 104(2), 209–218.

Chen, L., and Shen, R. (2011). FAQ System in Specific Domain Based on Concept Hierarchy and Question Type*. In Proc. of the International Conference on Computational and Information Sciences (ICCIS)*, Chengdu, Sichuan, China.

Cheng, C. H., and Yang, K. L. (1998). Using fuzzy sets in education grading system. *Journal of Chinese Fuzzy Systems Association*, 4(2), 81–89.

Cheung, R., Wan, C., and Cheng, C. (2010). An ontology-based framework for personalized adaptive learning. *In Proc. of the 9th International Conference on Web-based Learning (ICWL 2010)*, Shanghai, China (pp.52–61).

Chiang, T. T., and Lin, C. M. (1994). Application of fuzzy theory to teaching assessment. *In Proc. of the 1994 second national conference on fuzzy theory and applications*, Taipei, Taiwan, Republic of China (pp. 92–97).

Chieu, V.M., Luengo, V., Vadcard, L., and Tonetti, J. (2010). Student Modeling in Orthopedic Surgery Training: Exploiting Symbiosis between Temporal Bayesian Networks and Fine-grained Didactic Analysis. *Journal of Artificial Intelligence in Education*, 20 (3), 269-301.

Chin, D.N. (2001). Empirical Evaluation of the User Models and User-Adapted Systems. *User Modelling and User-Adapted Interaction*, 11, 181-194.

Chrysafiadi, K., and Virvou, M. (2008). Personalized Teaching of a Programming language over the web: Stereotypes and rule-based mechanisms. *In Proc. of the 8th Joint Conference on Knowledge-Based Software Engineering*, Piraeus, Greece (pp. 484-492).

Chrysafiadi K. & Virvou M. (2012). Evaluating the Integration of Fuzzy Logic into the Student Model of a Web-Based Learning Environment. *Experts Systems with Applications*, 39 (18), 13127-13134.

Chytas, P., Glykas, M., and Valiris, G. (2011). A proactive balanced scorecard. *International Journal of Information Management: The Journal for Information Professionals*, 31 (5), 460-468.

Clancey, W. (1988). The role of qualitative models in instruction. In J. Self (eds.), *Artificial Intelligence and Human Learning*, Chapman and Hall Computing.

Clemente, J., Ramírez, J., and de Antonio, A. (2011). A proposal for student modeling based on ontologies and diagnosis rules. *Expert Systems with Applications*, 38 (7), 8066-8078.

Codara, L. (1998). *Le mappe cognitive*. Roma: Carrocci Editore.

Cohen, P. A., Kulik, J. A., Kulik, C. L .C. (1982). Educational outcomes of tutoring: a meta-analysis of findings. *American Educational Research Journal*, 19 (2), 237–248.

Collins, A., and Michalski, R. (1989). The Logic of Plausible Reasoning: A core Theory. Cognitive Science, *Elsevier Science*, The Netherlands, 13, 1-49.

Conati, C. (2009). Intelligent Tutoring Systems: New Challenges and Directions. *In Proc. of the 21st International Joint Conference on Artificial Intelligence*, San Francisco, CA, USA (pp.2-7).

Conati, C., and Maclaren, H. (2009). Empirically Building and Evaluating a Probabilistic Model of User Affect. *User Modeling and User-Adapted Interaction*, 19 (3), 267–303.

Conati, C., and Zhou, X. (2002). Modeling students' emotions from cognitive appraisal in educational games. *In Proc. of the 6th International Conference on Intelligent Tutoring Systems*, Biarritz, France and San Sebastian, Spain (pp. 944-954).

Conati, C., Gertner, A., and Vanlehn, K. (2002). Using Bayesian networks to manage uncertainty in student modeling. *User Modeling and User-Adapted Interaction*, 12 (4), 371–417.

Conde, A., López de Ipiña, K., Larrañaga, M., Garay-Vitoria, N., Irigoyen, E., Ezeiza, A., and Rubio, J. (2009). LAGUNTXO: A rule-based intelligent tutoring system oriented to people with intellectual disabilities. *In Proc. of the 2nd World Summit on the Knowledge Society: Visioning and Engineering the Knowledge Society*. A Web Science Perspective (pp. 186–195).

Corbett, A. (2001). Cognitive computer tutors: solving the two-sigma problem. *In Proc. of the 8th International Conference on User Modeling*, Sonthofen, Germany (pp. 137–147).

Craiger, J.P., Goodman, D.F., Weiss, R.J., and Butler, A. (1996). Modeling organizational behavior with fuzzy cognitive maps. *J Comput Intell and Organizations*, 1,120-123.

Crowley, R.S., and Medvedeva, O. (2006). An intelligent tutoring system for visual classification problem solving. *Artificial Intelligence in Medicine*, 36, 85-117.

des Jardins, M., Ciavolino, A., Deloatch, R., and Feasley, E. (2011). Playing to Program: Towards an Intelligent Programming Tutor for RUR-PLE. *In Proc. of the Second Symposium on Educational Advances in Artificial Intelligence*, San Francisco, California.

de Raadt, M. (2007). A review of Australasian investigations into problem solving and the novice programmer. *Computer Science Education*, 17 (3), 201-213.

de Souza, M., and Ferreira, M. (2002). Designing reusable rule-based architectures with design patterns. *Expert Systems with Applications*, 23 (4), 395–403.

Dempster, J. (2004). Evaluating e-learning developments: An overview. Retrieved 23/07/08 from warwick.ac.uk/go/cap/resources/ eguides

Desmarais, M.C., and Baker, R.S. (2012). A review of recent advances in learner and skill modeling in intelligent learning environments. *User Modeling and User-Adapted Interaction*, 22 (1-2), 9-38.

Devedzic, V. (2006). *Semantic Web and Education* (Monograph). New York : Springer, Berlin Heidelberg.

Di Lascio, L., Gisolfi, A., and Nappi, A. (2005). Medical differential diagnosis through type-2 fuzzy sets. *In Proc. of IEEE FUZZ Conference*, Reno, NV (pp. 371–376).

D'Mello, S., Olney, A., Williams, C., and Hays, P. (2012). Gaze tutor: A gaze-reactive intelligent tutoring system. *International Journal of Human-Computer Studies*, 70 (5), 377–398.

Doctor, F., Hagras, H., and Callaghan, V. (2004). A type-2 fuzzy embedded agent for ubiquitous computing environment. *In Proc. of 2004 IEEE International Conference on Fuzzy Systems*, Budapest, Hungary (pp. 1105-1110).

Doctor, F., Hagras, H., and Callaghan, V. (2005). A type-2 fuzzy embedded agent to realise ambient intelligence in ubiquitous computing environments. *Information Sciences*, 171 (4), 309–334.

Dodds, P., Fletcher, J., (2004). Opportunities for new "smart" learning environments enabled by next-generation web capabilities. *Journal of Educational Multimedia and Hypermedia*. 13 (4), 391–404.

Dolog, P., Henze, N., Nejdl, W., Sintek, M. (2004). The personal reader: Personalizing and enriching learning resources using semantic web technologies. *In Proc. of the 3rd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, Eindhoven, Netherlands (pp. 85–94).

Drigas, A., Argyri, K., Vrettaros, J. (2009). Decade Review (1999-2009): Artificial Intelligence Techniques in Student Modeling. *In Proc. of the 2nd World Summit on the Knowledge Society (WSKS 2009)*, Chania, Crete, Greece (pp. 552-564).

Durrani, S., and Durrani, D. S. (2010). Intelligent Tutoring Systems and cognitive abilities. *In Proceedings of Graduate Colloquium on Computer Sciences (GCCS), Department of Computer Science*, FAST-NU Lahore, 1.

Echauz, J. R., and Vachtsevanos, G. J. (1995). Fuzzy grading system. *IEEE Transactions on Education*, 38(2), 158–165.

Esteva, F., and Godo, L. (2006). Towards the generalization of Mundici's gamma functor to IMTL algebras: the linearly ordered case. *Algebraic and Proof-theoretic Aspects of Non-classical Logics*, 127–137.

Faraco, R.A., Rosatelli, M.C., and Gauthier, F.A.O. (2004). An Approach of Student Modeling in a Learning Companion System. *in Proc. IX IBERAMIA*, Puebla, Mexico (pp. 891-900).

Feedman, R. (2000). What is an Intelligent Tutoring System?. *Intelligence*, 11(3), 15-16.

Felder, R.M., and Silverman L.K., (1988). Learning and Teaching Styles. *Engineering Education*, 78 (7), 674-681.

Felder, R.M., and Soloman, B.A. (2003). Learning styles and strategies.(2003). URL last accessed on 2012-06-28. http://www.ncsu.edu/felder-public/ILSdir/styles.htm

Figueroa, J., Posada, J., Soriano, J., Melgarejo, M., and Rojas, S. (2005). A type-2 fuzzy controller for tracking mobile objects in the context of robotic soccer games. *In Proc. of IEEE FUZZ Conference*, Reno, NV (pp. 359–364).

Flavell, J.H. (1976). Metacognitive Aspects of Problem Solving. In L.B. Resnick, L.B. (eds.), *The Nature of Intelligence* (pp. 231–236). Erlbaum: Hillsdale.

Gaudioso, E., Montero, M., and Hernandez-del-Olmo, F. (2012). Supporting teachers in adaptive educational system through predictive models: A proof of concept. *Expert Systems with Applications*, 39 (1), 621-625.

Gena, C. (2005). Methods and techniques for the evaluation of user-adaptive systems. *The knowledge Engineering Review*, 20(1), 1-37.

Gena, C., and Weibelzahl, S. (2007). Usability engineering for the adaptive web. In P. Brusilovsky, P., A. Kobsa, & W. Nejdl (Eds.), *The Adaptive Web: Methods and Strategies of Web Personalization* (pp. 720-762). New York: Springer-Verlag

Geng, X., Qin, S.,Chang, H., and Yang, Y. (2011). A hybrid knowledge representation for the domain model of intelligent flight trainer. *In Proc. 2011 IEEE International Conference on Cloud Computing and Intelligence Systems*, Beijing, China (pp. 29 – 33).

Glaser, R., Lesgold, A., and Lajoie, S. (1987). Toward a cognitive theory for the measurement of achievement. In R. Ronning, J. Glover, J.C. Conoley & J. Witt (Eds.), *The influence of cognitive psychology on testing and measurement: The Buros-Nebraska symposium on measurement and testing* (pp. 41- 85). Hillsdale. NJ: Erlbaum.

Glushkova, T. (2008). Adaptive model for user knowledge in the e-learning system. *In Proc. of the International Conference on Computer Systems and Technologies* (CompSysTech'08).

Glykas, G. (2010). *Fuzzy Cognitive Maps: Advances in Theory, Methodologies, Tools and Applications*. Springer Verlag, Berlin Heidelberg.

Godo, L.I., Esteva, F., Garcı´a, P., and Agustı, J. (1991). A formal semantical approach to fuzzy logic. *In International Symposium on Multiple Valued Logic* (pp. 72–79).

Goel,. G., Lallé, S., and Luengo, V. (2012). Fuzzy Logic Representation for Student Modelling Case Study on Geometry. *In Proc. of the 11th International Conference on Intelligent Tutoring Systems*, Chania, Greece (pp. 428-433).

Goguadze, G., Sosnovsky, S.A., Isotani, S., and McLaren, B.M. (2011a). Evaluating a bayesian student model of decimal misconceptions. *In the Proceedings of the 4th International Conference on Educational Data Mining*, Eindhoven, the Netherlands (pp. 301-306).

Goguadze, G., Sosnovsky, S., Isotani, S., and McLaren, B.M. (2011b). Towards a Bayesian Student Model for Detecting Decimal Misconceptions. *In Proc. 19th Int. Conf. on Computers in Education*, Chiang Mai, Thailand (pp. 34-41).

Gonzalez, C., Burguillo, J.C., and Llamas, M. (2006). A Qualitative Comparison of Techniques for Student Modeling in Intelligent Tutoring Systems. *In Proc. of the 36th Frontiers in Education Conference* (pp.13 – 18).

Graesser, A. C., Conley, M. W., & Olney, A. M. (2012). Intelligent tutoring systems. In S. Graham, & K. Harris (Eds.), *APA Educational Psychology Handbook: Vol. 3. Applications to Learning and Teaching* (pp. 451-473). Washington, DC: American Psychological Association.

Graesser, A.C., Chipman, P., Haynes, B.C., and Olney, A. (2005). AutoTutor: an intelligent tutoring system with mixedinitiative dialogue. *IEEE Transactions on Learning Technologies*, 48 (4), 612-618.

Grigoriadou, M., Kornilakis, H., Papanikolaou, K.A., and Magoulas, G.D. (2002). Fuzzy Inference for Student Diagnosis in Adaptive Educational Hypermedia. *In Proc. of the 2nd Hellenic Conference on AI: Methods and Applications of Artificial Intelligence*, Thessaloniki, Greece (pp. 191-202).

Grubišić, A., Stankov, S., Rosić, M., and Žitko, B. (2009). Controlled experiment replication in evaluation of e-learning system's educational influence. *Computers & Education*, 53 (3), 591-602.

Guangbing, Y., Kinshuk, K., and Graf, S. A practical Student Model for a Location-Aware and Context-Sensitive Personalized Adaptive Learning System. *In Proc. of the 2nd International Conference on Technology for Education*, Bombay, Mumbai (pp. 130-133).

Hajek, P. (1998). Metamathematics of Fuzzy Logic. Dordrecht: Kluwer.

Hagras, H. (2004a) A type-2 fuzzy logic controller for autonomous mobile robots. *In Proc. of IEEE FUZZ Conference*, Budapest, Hungary.

Hagras, H. (2004b). A hierarchical type-2 fuzzy logic control architecture for autonomous mobile robots. *IEEE Transactions on Fuzzy Systems*, 12 (4), 524–539.

Hanafizadeh, P., and Aliehyaei, R. (2011). The Application of Fuzzy Cognitive Map in Soft System Methodology. *Systemic Practice and Action Research*, 24, 325-354.

Henze, N.,and Nejdl, W. (1999). Student modeling for KBS Hyperbook system using Bayesian networks. *Technical report, University of Hannover [online].* Available: http://www.kbs.unihannover.de/paper/99/adaptivity.html

Hernández, Y., Sucar, L., and Arroyo-Figueroa, G. (2010). Evaluating an Affective Student Model for Intelligent Learning Environments. *In Proc. of the 12th Ibero-American conference on Advances in artificial intelligence (IBERAMIA'10)*, Bahía Blanca, Argentina (pp. 473-482).

Holland, J., Mitrovic, A., and Martin B. (2009). J-Latte: a constraint-based tutor for java. *In Proc. of the 17th International Conference on Computers in Education*, Hong Kong (pp. 142–146).

Hwang, C.L., and Yoon, K. (1981). Multiple attribute decision making: methods and applications. *Lecture notes in economics and mathematical systems 186*. Springer, Berlin

Hwang, C., and Rhee, F.C.-H. (2004). An interval type-2 fuzzy spherical shells algorithm, *In Proc. of IEEE FUZZ Conference*, Budapest, Hungary (pp. 1117-1122).

Iakovidis, D.K., and Papageorgiou, E. (2011). Intuitionistic Fuzzy Cognitive Maps for Medical Decision Making. *IEEE Transactions on Information Technology in Biomedicine*, 15 (1), 100-107.

Inventado, P.S., Legaspi, R., The Duy Bui, and Suarez., M. (2010) Predicting Student's Appraisal of Feedback in an ITS Using Previous Affective States and Continuous Affect Labels from EEG Data. *In Proc. of the 18th International Conference on Computers in Education*, Putrajaya, Malaysia (pp. 71-75).

Jameson, A. (1996). Numerical uncertainty management in user and student modeling: an overview of systems and issues. *User Modeling and User-Adapted Interaction*, 5(3–4), 193–251.

Jeremić, Z., Jovanović, J. & Gasěvić, D. (2009). Evaluating an Intelligent Tutoring System for Design Patterns: the DEPTHS Experience. *Educational Technology & Society*, 12(2), 111–130.

Jeremić, Z., Jovanović, J. & Gasěvić, D. (2012). Student modeling and assessment in intelligent tutoring of software patterns. *Expert Systems with Applications*, 39(1), 210-222.

Jetter, A., and Schweinfort, W. (2011). Building scenarios with Fuzzy Cognitive Maps: An exploratory study of solar energy. *Futures*, 43 (1), 52-66.

Jia, B., Zhong, S., Zheng, T., and Liu, Z. (2010). The Study and Design of Adaptive Learning System Based on Fuzzy Set Theory. In Z., A.D. Cheok, W. Müller, X. Zhang, & K. Wong (Eds.), *Transactions on Edutainment* IV (pp. 1–11). Berlin, Heidelberg: Springer-Verlag.

Jili, C., Kebin, H., Feng, W., & Huixia, W. (2009). E-learning Behavior Analysis Based on Fuzzy Clustering. *In Proc. Third International Conference on Genetic and Evolutionary Computing (WGEC '09)* (pp. 863-866).

Jin, W., Barnes, T., Stamper, J., Eagle, M.J., Johnson, M.W., and Lehmann, L. (2012). Program Representation for Automatic Hint Generation for a Data-Driven Novice Programming Tutor. *In Proc of. ITS'12 Proceedings of the 11th international conference on Intelligent Tutoring Systems*, Chania, Crete, Greece (pp. 304-309).

John, R.I, and Innocent, P.R. (2005). Modeling uncertainty in clinical diagnosis using fuzzy logic. *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernet*. 35, 1340–1350.

Johnson, W.L., Rickel, J.W., Lester, J.C. (2000). Animated Pedagogical Agents: Face-to-Face Interaction in Interactive Learning Environment. *International Journal of Artificial Intelligence in Education*,11, 47–78.

Jonassen, D.H., and Grabowski, B.L. (1993). *Handbook of Individual Differences, Learning and Instruction*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Jurado, F., Santos, O.C., Redondo, M.A., Boticario, J.G., and Ortega, M. (2008). Providing Dynamic Instructional Adaptation in Programming Learning. *In Proc. of the 3rd international workshop on Hybrid Artificial Intelligence Systems*, Burgos, Spain (pp. 329–336).

Kabassi, K., and Virvou, M. (2004). Personalised Adult e-Training on Computer Use based on Multiple Attribute Decision Making. *Interacting with Computers*, 16 (1), 115-132.

Kaklauskas, A., Zavadskas, E., and Ditkevičius, R. (2006). An intelligent tutoring system for construction and real estate. *In Proc. of Cooperative Design, Visualization, and Engineering (CDVE 2006)*, Mallorca, Spain (pp. 174–181).

Kannappan, A., Tamilarasi, A., and Papageorgiou, E.I. (2011). Analyzing the performance of fuzzy cognitive maps with non-linear hebbian learning algorithm in predicting autistic disorder. *Expert Systems with Applications*, 38 (3), 1282-1292.

Karnik, N.N, and Mendel, J.M. (2001). Centroid of a type-2 fuzzy set. *Information Sciences*, 132, 195–220.

Karnik, N.N, Mendel, J.M. and Liang, Q. (1999). Type-2 Fuzzy Logic Systems. *IEEE Transactions on Fuzzy Systems*, 7 (6), 643-658.

Kass, R. (1991). Building a user model implicitly from a cooperative advisory dialog. *User Modeling and User-Adapted Interaction*, 1, 203-258.

Kassim, A.A, Kazi, S. A., and Ranganath, S. (2004). A Web-based intelligent learning environment for digital systems. *Int. J. Eng. Educ.*, 20(1), 13-23.

Katsionis, G., and Virvou, M. (2004). A cognitive theory for affective user modelling in a virtual reality educational game. *In Proc. of 2004 IEEE International Conference on Systems, Man, and Cybernetics* (pp. 1209-1213).

Kavčič, A. (2004a). Fuzzy Student Model in InterMediActor Platform. *In Proc. of the 26th International Conference on Information Technology Interfaces*, Croatia (pp. 297-302).

Kavčič, A. (2004b). Fuzzy User Modeling for Adaptation in Educational Hypermedia. *IEEE Trans. on Systems, Man and Cybernetics. Part C: Applications and Reviews*, 34 (4), 439-449.

Kay, J. (2000). Stereotypes, student models and scrutability. *In Proc. of the 5th International Conference on Intelligent Tutoring Systems*, Montréal, Canada (pp. 19-30).

Kazi, H., Haddawy, P., and Suebnukarn, S. (2012). Employing UMLS for generating hints in a tutoring system for medical problem-based learning. *Journal of Biomedical Informatics*, 45, 557–565.

Keles, A., Ocak, R., Keles, A., and Gülcü, A. (2009). Zosmat: Web-based intelligent tutoring system for teaching–learning process. *Expert Systems with Applications*, 36 (2-1), 1229–1239.

Khamis, M. (2011) IDEAL: an Intelligent Distributed Experience-based Adaptive Learning Model. *Journal of Arts and Humanities*, 20 (1).

Khuwaja, R., Evens, M., Rovick, A., & Michael, J. A. (1992). Knowledge representation for an intelligent tutoring system based on a multilevel causal model. *In Proc. of the Second International Conference on Intelligent Tutoring Systems, ITS '92*, Montreal, Canada (pp. 217-224).

Kirkpatrick, D.L. (1979). Techniques for evaluating training programs. *Training and Development Journal*, 33(6), 78-92.

Knight, P. (2002). *Conspiracy Nation: the Politics of Paranoia in Postwar America.* New York and London: New York University Press.

Koedinger, K.R., Anderson, J. R., Hadley, W. H., and Mark, M. A. (1997). Intelligent Tutoring Goes to School in the Big City. *Int. J. Artificial Intelligence in Education*, 8, 30-43.

Kofod-Petersen, A., Petersen, S.A., Bye, G.G., Kolås, L., and Staupe, A. (2008). Learning in an ambient intelligent environment – towards modeling learners through stereotypes. *Revue d'Intelligence Artificielle*, 22 (5), 569-588.

Kosba, E., Dimitrova, V., and Boyle R. (2003). Using Fuzzy Techniques to Model Students in Web-Based Learning Environment. *In Proc. of the 7th International Conference on Knowledge-Based Intelligent Information & Engineering Systems*, United Kingdom (pp. 222-229).

Kosba, E., Dimitrova, V., and Boyle R. (2005). Using Student and Group Models to Support Teachers in Web-Based Distance Education. *In Proc. of the International Conference on User Modeling*, Edinburgh (pp. 124-133).

Kosko, B. (1986). Fuzzy cognitive maps. *Int J Man–Mach Stud*, 24 (1), 65-75.

Kosko, B. (1992). *Neural Networks and Fuzzy Systems*. Upper Saddle River, NJ: Prentice Hall.

Kosko, B. (1999). *Fuzzy Engineering*. New Jersey: Prentice Hall.

Kumar, A. (2005). Rule-based adaptive problem generation in programming tutors and its Evaluation. *In Proc. of the 12th International Conference on Artificial Intelligence in Education*, Amsterdam (pp. 35-43).

Kumar, A. (2006a). Using Enhanced Concept Map for Student Modeling in Programming Tutors*. In Proc. of the 19th International Florida Artificial Intelligence Research Society Conference*, Melbourne Beach (pp. 527-532).

Kumar, A. (2006b). A Scalable Solution for Adaptive Problem Sequencing and its Evaluation. *In Proc. of the 4th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH'2006)*, Dublin, Ireland (pp. 161-171).

Kyriacou, D. (2008). A Scrutable User Modelling Infrastructure for enabling life-long User Modelling. *In Proc. of the 5th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, Hannover, Germany (pp. 421-425).

Latham, A., Crockett, K.A., McLean, D. and Edmonds, B. (2010). Predicting Learning Styles in a Conversational Intelligent Tutoring System. *In Proc. of the 9th International Conference on Web-based Learning (ICWL 2010)*, Shanghai, China (pp. 131-140).

Law, C. K. (1996). Using fuzzy numbers in education grading system. *Fuzzy Sets and Systems*, 83(3), 311–323.

Le, N.T., and Menzel, W. (2009). Using Weighted Constraints to Diagnose Errors in Logic Programming–The Case of an Ill-defined Domain. *Journal on Artificial Intelligence in Education*, 19 (2), 382-400.

Le, N.-T., Menzel, W., and Pinkwart, N., (2009). Evaluation of a constraint-based homework assistance system for logic programming. *In Proc. of the 17th International Conference on Computers in Education*, Hong Kong.

Lee, N., Bae, J.K., and Koo, C. (2011). A case-based reasoning based multi-agent cognitive map inference mechanism: An application to sales opportunity assessment. *Information Systems Frontiers*, 14 (3), 653-668.

Lehman, B., Matthews. M., D'Mello, S., and Person, N. (2008). What are you feeling? Investigating student affective states during expert human tutoring sessions. *In Proc. of the 9th international conference on Intelligent Tutoring Systems (ITS 2008)*, Montreal (pp 50–59).

Leon, M., Napoles, G., Garcia, M., Bello, R., and Vanhoof, K. (2011). Two steps Individuals Travel Behavior through Fuzzy Cognitive Maps Pre-definition and Learning. *In Proc. of the 10th Mexican International Conference on Artificial Intelligence*, Puebla, Mexico.

Li, N., Cohen, W.W., Koedinger, K.R., and Matsuda, N. (2011). A Machine Learning Approach for Automatic Student Model Discovery. *In Proc. of conf. on Educational Data Mining (EDM 2011)*, Eindhoven, the Netherlands (pp. 31-40).

Liang, Q., and Mendel, J.M. (2000). Interval Type-2 Fuzzy Logic Systems: Theory and Design. *IEEE transactions on Fuzzy Systems*, 8 (5), 535-550.

Liang, Q., and Wang, L. (2005). Sensed signal strength forecasting for wireless sensors using interval type-2 fuzzy logic system. *In Proc. of the IEEE FUZZ Conference*, Reno, NV (pp. 25–30).

Limongelli, C., Sciarrone, F., Temperini M., and Vaste, G. (2009). Adaptive learning with the LS-Plan System: A Field Evaluation. *IEEE Trans. On Learning Technologies*, 2 (3), 203-215.

Lin, P.Z., Hsu, C.F, and Lee, T.T. (2005). Type-2 fuzzy logic controller design for buck DC–DC converters. *In Proc. of the IEEE FUZZ Conference*, Reno, NV (pp. 365–370).

Liu, C.-I. (2008). Using bayesian networks for student modeling. In R. M. Viccari, P. Augustin-Jaques, & R. Verdin (Eds.), *Agent-based tutoring systems by cognitive and affective modeling* (pp. 97–113). Igi Global.

Liu, Z., and Wang, H. (2007). A Modeling Method Based on Bayesian Networks in Intelligent Tutoring System. *In Proc. of the 11th International Conference on Computer Supported Cooperative Work in Design*, Melbourne, Australia (pp. 967 – 972).

Lu, C.H., Ong, C.S., and Hsu,W.L. (2007). Using an ITS as an arithmetic assistant for teachers 3-year review. *Journal of Internet Technology*, 8 (4), 389-398.

Lu, C.-H., Wu, C.- W., Wu, S.-H. Chiou, G.-F., and Hsu, W.-L. (2005). Ontological Support in Modeling Learners' Problem Solving Process. *Educational Technology & Society*, 8 (4), 64-74.

Lynch, C., Hagras, H., and Callaghan, V. (2005). Embedded type-2 FLC for real-time speed control of marine and traction diesel engines. *In Proc. of IEEE FUZZ Conference*, Reno, NV (pp. 347–352).

Ma, J., & Zhou, D. (2000). Fuzzy set approach to the assessment of student-centered learning. *IEEE Transactions on Education*, 43(2), 237–241.

Mahnane, L., Laskri, M.T., and Trigano, P. (2012). An adaptive Hypermedia System Integrating Thinking Style (AHS-TS): Model and Experiment. *Inter. Journal of Hybrid Information Technology*, 5 (1), 11-28.

Malik, K.S., Prakash, N., and Rizvi, S. (2011). Ontology Creation towards an Intelligent Web: Some Key Issues Revisited. *International Journal of Engineering and Technology* 3 (1), 44-52.

Martin, B. (1999). Constraint-based Student Modeling: Representing Student Knowledge. *In Proc. of the 3rd New Zealand Computer Science Research Students' Conference*, Hamilton NZ (pp. 22-29).

Martins, A. C., Faria, L., Vaz de Carvalho, C., and Carrapatoso, E. (2008). User Modeling in Adaptive Hypermedia Educational Systems. *Educational Technology & Society*, 11 (1), 194-207.

Mayo, M.J. (2001). Bayesian Student Modelling and Decision-Theoretic Selection of Tutorial Actions in Intelligent Tutoring Systems. PhD Thesis. Retrieved 29 June 2012, from http://www.cosc.canterbury.ac.nz/research/reports/PhdTheses/2001/phd_0102.pdf

Mayo, M., and Mitrovic, A. (2001). Optimising ITS Behavior with Bayesian Networks and Decision Theory. *International Journal of Artificial Intelligence in Education*, 12, 124-153.

McGill, T.J., and Volet, S.E. (1997). A conceptual framework for analyzing students' knowledge of programming. *Journal of Research on Computing in Education*, 29 (3), 276-297.

Melin, P., and Castillo,O. (2003a). A new method for adaptive model-based control of non-linear plants using type-2 fuzzy logic and neural networks. *In Proc. of the IEEE FUZZ Conference*, St. Louis, MO (pp. 420–425).

Melin, P., and Castillo, O. (2003b). A new approach for quality control of sound speakers combining type-2 fuzzy logic and the fractal dimension. *In Proc. of the International Conference NAFIPS 2003*, Chicago, USA (pp. 20–25).

Melis, E., and Siekmann, J.H. (2004). ActiveMath: An Intelligent Tutoring System for Mathematics. *In Proc. of the 7th International Conference*, Zakopane, Poland (pp. 91-101).

Mendel, J.M. (2001). *Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions*. NJ: Prentice-Hall.

Mendel, J.M. (2003). Type-2 Fuzzy Sets: Some Questions and Answers. *IEEE Connections, Newsletter of the IEEE Neural Networks Society*, 1, 10-13.

Mendez, G.M., and Castillo, O. (2005). Interval type-2 TSK fuzzy logic systems using hybrid learning algorithm. *In Proc. of the IEEE FUZZ Conference*, Reno, NV (pp. 230–235).

Meshref, H., and Mohamed, I.N. (2012). Intelligent tutoring systems: a new proposed structure. *In Proc. the International Conference on Advances in Computing, Communications and Informatics*, Chennai, India (pp. 1182-1186).

Miao, Y., and Liu Z.Q. (2000). On causal inference in fuzzy cognitive maps. *IEEE Transactions on Fuzzy Systems*, 8 (1), 107-119.

Miao, Y., Miao, C., Tao, X., Shen, Z., and Liu, Z. (2010) Transformation of cognitive maps. *IEEE Transactions on Fuzzy Systems*, 18 (1), 114-124.

Michaud, L. N., and McCoy, K. F. (2004). Empirical derivation of a sequence of user stereotypes for language learning. *User Modeling and User-Adapted Interaction*, 14, 317-350.

Millán, E., and Perez de la Cruz, J.-L. (2002). A Bayesian Diagnostic Algorithm for Student Modeling. *User Modeling and User-Adapted Interaction*, 12(2/3), 281-330.

Millán, E., Loboda, T., Pérez-de-la-Cruz, J.L. (2010). Bayesian networks for student model engineering. *Computers & Education*, 55(4), 1663-1683.

Mitrovic, A. (2003). An Intelligent SQL Tutor on the Web. *International Journal of Artificial Intelligence in Education*, 13 (2-4), 173-197.

Mitrovic, A., and Martin, B. (2006). Evaluating the Effects of Open Student Models on Learning. *In Proc. of the 2nd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems* (pp. 296-305).

Mitrovic, A., and Ohlsson, S. (1999). Evaluation of a constraint-based tutor for a database language. *International Journal of Artificial Intelligence in E*ducation, 10(3-4), 238-256.

Mitrovic, A., Marting, B., & Mayo, M. (2002). Using Evaluation to Shape ITS Design: Results and Experiences with SQL-Tutor. *User Modelling and User-Adapted Interaction*, 12(2/3), 243-279

Mitrovic, A., Martin, B., and Suraweera, P. (2007). Intelligent Tutors for All: Constraint-Based Approach. *IEEE Intelligent Systems*, 22 (4), 38-45.

Mitrovic, A., Mayo, M., Suraweera, P., and Martin, B. (2001). Constraint-based Tutors: a Success Story. *In Proc. of 14th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE-2001)*, Budapest (pp. 931-940).

Mitrovic, A., Suraweera, P., Martin, B., Weerasinghe, A. (2004). DB-Suite: Experiences with Three Intelligent, Web-based Database Tutors. *Journal of Interactive Learning Research*, 15, 409-432.

Mitrovic, A., Williamson, C., Bebbington, A., Mathews, M., Suraweera, P., Martin, B., Thomson, D., and Holland, J. (2011). Thermo-Tutor: An Intelligent Tutoring System for thermodynamics. *In Proc. of the 2011 IEEE Global Engineering Education Conference (EDUCON)* (pp. 378 – 385).

Mizumoto, M., and Tanaka, K. (1976). Some properties of fuzzy sets of type 2. *Information and Control*, 31 (4), 312-340.

Mulwa, C., Lawless, S., Sharp, M., & Wade, V. (2011). The evaluation of adaptive and personalized information retrieval systems: a review. *Int. J. Knowledge and Web Intelligence*, 2(2/3), 138-156.

Muñoz, K., Mc Kevitt, P., Lunney, T., Noguez, J., and Neri, L. (2010). PlayPhysics: An emotional Game Learning Environment for Teaching Physics. *In Proc. of the 4th international conference on Knowledge science, engineering and management (KSEM' 10)*, Belfast, Northern Ireland, UK (pp. 400-411).

Muñoz, K., Mc Kevitt, P., Lunney, T., Noguez, J., and Neri, L. (2011). An emotional student model for game-play adaptation. *Entertainment Computing*, 2 (2), 133–141.

Muñoz-Merino, P.J., Molina, M.F., Muñoz-Organero, M., and Kloos, C.D. (2012), An adaptive and innovative question-driven competition-based intelligent tutoring system for learning. *Expert Systems with Applications* ,39, 6932–6948.

Myneni, L., and Narayanan, N.H. (2012). ViPS - An Intelligent Tutoring System for Exploring and Learning Physics through Simple Machines. *In Proc. of the 4th International Conference on Computer Supported Education*, Porto, Portugal, (pp. 73-82).

Niewiadomski, A., Kacprzyk, J., Ochelska, J., and, Szczepaniak, P.S. (2006). Interval-valued linguistic summaries of databases. *Control & Cybernetics*, 35 (2), 415-443.

Nguyen, L., and Do, P. (2008). Learner Model in Adaptive Learning. *In Proc. of World Academy of Scıence, Engıneerıng and Technology* (pp. 396-401).

Nguyen, L., and Do, P. (2009). Combination of Bayesian Network and Overlay Model in User Modeling. *In Proc. of the 9th International Conference on Computational Science*, Baton Rouge, Louisiana, USA (pp. 5-14).

Nguyen, C.D., Vo, K.D., Bui, D.B., and Nguyen, D.T. (2011). An ontology-based IT student model in an educational social network. *In Proc. of the 13th International Conference on Information Integration and Web-based Applications and Services (iiWAS '11),* Bali, Indonesia (pp. 379-382).

Nkambou, R. (2010). Modeling the domain: An introduction to the expert module. In R. Nkambou, J. Bourdeau, & R. Mizoguchi (Eds.). *Advances in intelligent tutoring systems* (Vol. 308, pp. 15–32). Berlin/ Heidelberg: Springer.

Novak, J. (1990). Concept mapping: a useful tool for science education. *Journal of Research in Science Teaching,* 27 (10), 937–949.

Novak, V. (2006). Which logic is the real fuzzy logic? *Fuzzy Sets and Systems*, 157, 635–641.

Novak, J.D., and Cañas, A.J. (2006). The Theory Underlying Concept Maps and How To Construct and Use Them. *Institute for Human and Machine Cognition*. Accessed 24 Nov 2008.

Novak, J. D., and Musonda, D. (1991). A twelve-year longitudinal study of science concept learning. *American Educational Research Journal*, 28(1), 117-153.

Novak, J. D. and Cañas, A. J. (2008). The Theory Underlying Concept Maps and How to Construct Them, Technical Report IHMC CmapTools 01-2006, Revised 01-2008, Florida Institute for Human and Machine Cognition; retrieved from http://cmap.ihmc.us/Publications/ResearchPapers/TheoryUnderlyingConceptMaps.pdf on August 28, 2008.

Novak, V., Perfilieva, I., and Mockor, J. (1999). *Mathematical Principles of Fuzzy Logic*. Boston/Dordrecht: Kluwer.

Nwana, h.s. (1990). Intelligent Tutoring Systems: an Overview. *Artificial Intelligence Review*, 4, 251-277.

Nykänen, O. (2006). Inducing Fuzzy Models for Student Classification. *Educational Technology and Society*, 9 (2), 223-234.

Ohlsson, S. (1994). Constraint-based Student Modeling. In J.E. Greer, G.I. McCalla (Eds.), *Student Modeling: the Key to Individualized Knowledge--based Instruction* (pp. 167-189). Berlin: Springer-Verlag.

Ohlsson, S. (1996). Learning from Performance Errors. *Psychological Review*, 103(2), 241-262.

Ohlsson, S., and Mitrovic, A. (2006). Constraint-Based Modeling: An Introduction. *In Proc. of the the 28th Annual Conference of the Cognitive Science Society, Vancouver, Canada.*

Olney, A.M. (2009). GnuTutor: An open source intelligent tutoring system based on AutoTutor. *In Proc. of the AAAI Fall Symposium on Cognitive and Metacognitive Educational Systems*, Menlo Park, CA (pp. 70 - 75).

O'Shea, T,, Bornat, R., Du Boulay, B., and EisenstadL, M. (1984). Tools for creating intelligent computer tutors. *In Artificial and Human Intelligence* (eds.) Elsevier, North Holland (pp. 181-199).

Ortony, A., Clore, G. L. and Collins, A. (1988). *The Cognitive Structure of Emotions*. Cambridge: Cambridge University Press.

Ozen, T., Garibaldi, J.M., and Musikasuwan, S. (2004). Preliminary investigations into modeling the variation in human decision making. *In Proc. of 10th Information Processing and Management of Uncertainty in Knowledge Based Systems (IPMU 2004),* Perugia, Italy (pp. 641–648).

Panagiotopoulos, I., Seremeti, L., and Kameas, A. (2010) PROACT: An Ontology-Based Model of Privacy Policies in Ambient Intelligence Environments. *In Proc. of the 14th Panhellenic Conference on Informatics (PCI 2010)* (pp. 124–129).

Panagiotopoulos, I., Kalou, A.K., Pierrakeas, C., and Kameas, A. (2012). An Ontological Approach for Domain Knowledge Modeling and Management in E-Learning Systems. *First AI in Education Workshop: Innovations and Applications*, 2, 95-104.

Papageorgiou, E. (2011a). Review study on Fuzzy Cognitive Maps and their applications during the last decade. *In Proc. of the IEEE International Conference on Fuzzy Systems*. Taipei, Taiwan.

Papageorgiou, E. (2011b). A new methodology for Decisions in Medical Informatics using Fuzzy Cognitive Maps based on Fuzzy Rule-Extraction techniques. *Applied Soft Computing*, 11, 500–513.

Papanikolaou, K.A., Grigoriadou, M., Kornilakis, H., and Magoulas, G.D. (2003). Personalizing the Interaction in a Web-based Educational Hypermedia System: the case of INSPIRE. *User Modeling and User-Adapted Interaction*, 13 (3), 213-267.

Parvez, S.M., and Blank, G.D. (2008). Individualizing Tutoring with Learning Style Based Feedback. *In Proc. of the 9th international conference on Intelligent Tutoring Systems*, Montreal, 291 – 301.

Pekrun, R., Frenzel, A.C., Goetz, T., and Perry, R. P. (2007). The control value theory of achievement emotions: an integrative approach to emotions in education. In P.A. Shutz, R. Pekrun (Eds.), *Emotion in Education* (pp. 13–36). London: Elsevier.

Peña, A., and Kayashima, M. (2011). Improving Students' Meta-cognitive Skills within Intelligent Educational Systems: A Review. *In Proc. of the 6th international conference on Foundations of augmented cognition: directing the future of adaptive systems* (pp. 442–451).

Peña, A., and Sossa, H. (2010). Semantic Representation and Management of Student Models: An Approach to Adapt Lecture Sequencing to Enhance Learning. *In Proc. of the 9th Mexican international conference on Advances in artificial intelligence: Part I,* Pachuca, Mexico (pp. 175–186).

Pearl, J. (1988). *Probabilistic Reasoning in Expert Systems: Networks of Plausible Inference*. San Francisco: Morgan Kaufmann Publishers, Inc.

Pearl, J. (1996). Decision Making Under Uncertainty. *ACM Computing Surveys*. 28 (1), 89-92.

Peylo, C., Teiken, W., Rollinger, C., and Gust, H. (2000). An ontology as domain model in a web-based educational system for prolog. *In Proc. of the 13th Intern. Florida Artificial Intelligence Research Society Conference*, Orlando, Florida, USA.

Popescu, E. (2009) Diagnosing Students' Learning Style in an Educational Hypermedia System. Cognitive and Emotional Processes in Web-based Education: Integrating Human Factors and Personalization. *Advances in Web-Based Learning Book Series, IGI Global*, pp. 187–208.

Popescu, E., Badica C., and Moraret, L.(2009). WELSA: An Intelligent and Adaptive Web-based Educational System*. In Proc. of the 3rd Symposium on Intelligent Distributed Computing*, Ayia Napa, Cyprus (pp. 175-185).

Popescu, E., Badica C., and Moraret, L.(2010). Accommodating Learning Styles in an Adaptive Educational System. *Informatica*, 34, 451–462.

Psotka, J., Massey, L.D., Mutter, S.A. (1988). *Intelligent Tutoring Systems: Lessons Learned*. Lawrence Erlbaum Associates.

Pramitasari, L., Hidayanto, A.N., Aminah, S., Krisnadhi, A.A., and Ramadhanie, M.A. (2009). Development of Student Model Ontology for Personalization in an E-Learning System based on Semantic Web. *In Proc. of the International Conference on Advanced Computer Science and Information Systems (ICACSIS 2009)*, Universitas Indonesia, Depok, Indonesia (pp. 434–439).

Rahimi, S., Cobb, M., Zhou, A., Ali, D., Yang, H., and Petry, F.E. (2003). An inexact inferencing strategy for spatial objects with determined and indeterminate boundaries. *In Proc. of IEEE FUZZ Conference*, St. Louis, MO (pp. 778–783).

Reghis, M., and Roventa, E. (1998). Classical and Fuzzy Concepts in Mathematical Logic and Applications. *CRC-Press*..

Rhee, F.C.-H., and Hwang, C. (2001). A type-2 fuzzy c-means clustering algorithm. *In Proc. of IEEE FUZZ Conference*, Melbourne, Australia (pp. 1926–1929).

Rhee, F.C.-H., and Hwang, C. (2002a). An interval type-2 fuzzy perceptron. *In Proc. of IEEE FUZZ Conference*, Honolulu, HI.

Rhee, F.C.-H., and Hwang, C. (2002b). An interval type-2 fuzzy K-nearest neighbor, *In Proc. of IEEE FUZZ Conference*, Honolulu, HI (pp. 802–807).

Rich, E. (1979). User Modelling via Stereotypes. Cognitive Science, 3 (4), 329-354.

Rivers, R. (1989). Embedded user models – where next? *Interacting with Computers*, 1, 14-30.

Roberge, A. (2005). RUR: A Python learning environment. http://rur-ple.sourceforge.net/.

Rodrigo, M., Baker, R., Maria, L., Sheryl, L., Alexis, M., Sheila, P., Jerry, S., Leima, S., Jessica, S., and Sinath, T. (2007). Affect and usage choices in simulation problem solving environments. *In Proc. of the 13th Int. Conf. on Artificial Intelligence in Education*, Marina Del Ray, CA, USA (pp.145–152).

Rodriguez-Castro, B., Glaser, H. and Carr, L. (2010) How to Reuse a Faceted Classification and Put it on the Semantic Web. *In Proc. of the 9th International Semantic Web Conference (ISWC)*, Shanghai, China.

Rodriguez-Repiso L, Setchi R, Salmeron JL (2007) Modelling IT Projects success with Fuzzy Cognitive Maps*. Expert Syst Appl*, 32 (2), 543-559.

Sabourin, J., Mott, B., and Lester, J.C. (2011). Modeling Learner Affect with Theoretically Grounded Dynamic Bayesian Networks. *In Proc. of the 4th international conference on Affective computing and intelligent interaction*, Memphis, Tennessee (pp. 286-295).

Salim, N., and Haron, N. (2006). The Construction of Fuzzy Set and Fuzzy Rulef or Mixed Approach in Adaptive Hypermedia Learning System. *In Proc. of the 1st international conference on Technologies for E-Learning and Digital Entertainment (Edutainment 2006)*, Hangzhou, China (pp. 183–187).

Salomon, G. (1990). Studying the flute and the orchestra: Controlled vs. classroom research on computers. *International Journal of Educational Research*, 14, 521-532.

Schiaffino, S., Garcia, P., and Amandi, A. (2008). eTeacher: Providing personalized assistance to e-learning students. *Computers & Education*, 51(4), 1744–1754.

Schramm, J., Strickroth, S., Le, N.T., and Pinkwart, N. (2012). Teaching UML Skills to Novice Programmers Using a Sample Solution Based Intelligent Tutoring System. *In Proc. of the Twenty-Fifth International Florida Artificial Intelligence Research Society Conference*, Marco Island, Florida.

Self, J. A. (1988). Student models: what use are they? *In Artificial Intelligence Tools in Education*, North Holland, Amsterdam (pp. 73-86).

Self, J. A. (1990). Bypassing the intractable problem of student modelling. In C. Frasson & G. Gauthier (Eds.), *Intelligent Tutoring Systems: At the crossroads of AI and Education* (pp. 107-123). Norwood, NJ: Ablex.

Sepulveda, R., Castillo, O., Melin, P., Rodriguez-Diaz, A., and Montiel, O. (2005). Integrated development platform for intelligent control based on type-2 fuzzy logic. *In Proc. of North American Fuzzy Information Processing Society (NAFIPS)*, Ann Arbor, MI (pp. 607–610).

Sevarac, Z. (2006). Neuro Fuzzy Reasoner for Student Modeling. *In Proc. Sixth International Conference on Advanced Learning Technologies,* Kerkrade, The Netherlands (pp. 740-744).

Shakouri, H.G., and Menhaj, M. (2008). A systematic fuzzy decision-making process to choose the best model among a set of competing models. *IEEE Trans. on Systems, Man and Cybernetics, Part A: Systems and Humans*, 38 (5), 1118–1128.

Shakouri, H.G., and Tavassoli, N. (1998). Implementation of a hybrid fuzzy system as a decision support process: A FAHP-FMCDM-FIS composition. *Expert Systems with Applications*, 39 (3), 3682-3691.

Shapiro, J. A. (2005). An algebra subsystem for diagnosing students' input in a physics tutorin system. *International Journal of Artificial Intelligence in Education*, 15(3), 205-228.

Shu,H., and Liang, Q. (2005). Wireless sensor network lifetime analysis using interval type-2 fuzzy logic systems. *In Proc. of IEEE FUZZ Conference*, Reno, NV (pp. 19–24).

Shute, V. J., and Psotka, J. (1996). Intelligent tutoring systems: past, present and future. In D. Jonassen (Ed.), *Handbook of research on educational communications and technology* (pp. 570–600). New York: Macmillan.

Siddapa, M., and Manjunath, A.S. (2007). Knowledge representation using multilevel hierarchical model in intelligent tutoring system. *In Proc. of the Third International Conference on Advances in Computer Science and Technology*, Thailand.

Siddappa, M., Manjunath, A.S., and Kurian, M.Z. (2009). Design, Implementation and Evaluation of Intelligent Tutoring System for Numerical Methods (ITNM). *International Conference on Computational Intelligence and Software Engineering*, Wuhan, China (pp. 1 – 7).

Sison, R., and Shimura, M. (1998). Student modeling and machine learning. *International Journal of Artificial Intelligence in Education*, 9, 128-158.

Sleeman, D., and Brown, J. (1982). *Intelligent Tutoring Systems*. New York: Academic Press.

Smith, S. (1998). Tutorial on. Retrieved 15 Mar, 2002, from http://www.cs.mdx.ac.uk/staffpages/serengul/table.of.contents.htm

Song, H., Miao, C., Roel, W., and Shen, Z. (2012). Implementation of Fyzzy Cognitive Maps Bsed on Fuzzy Neural Network and Application in Prediction of Time Series. *IEEE Transactions on Fuzzy Systems*, 18 (2), 233-250.

Song, H., Miao, C., Roel, W., Shen, Z., and D'Hondt, M. (2011). An Extension to Fuzzy Cognitive Maps for Classification and Prediction. *IEEE Trans. on Fuzzy Systems*, 19 (1), 116-135.

Sosnovsky, S., and Dicheva, D. (2010). Ontological technologies for user modeling. *International Journal of Metadata, Semantics and Ontologies*, 5 (2), 32-71.

Spada, H. (1993). How the Role of Cognitive Modeling for Computerized Instruction is changing. *In Proc. of AI-ED'93, World Conference on Artificial Intelligence in Education*, Edinburgh, Scotland (pp. 21-25).

Specht, M., and Klemke, R. (2001). ALE - Adaptive Learning Environment. *In Proc. of WebNet'2001*, Orlando, Florida (pp. 1155-1160).

Stach, W., Kurgan, L., Pedrycz, W., and Reformat, M. (2005). Genetic learning of fuzzy cognitive maps. *Fuzzy Set Syst*, 153, 371-401.

Staff, C. (2001) HyperContext: A framework for adaptive and adaptable hypertext. *Ph.D. Thesis. University of Sussex*.

Stansfield, J.C., Carr, B., and 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.

Stathacopoulou, R., Magoulas, G.D., Grigoriadou, M., and Samarakou, M. (2005). Neuro-fuzzy Knowledge Processing in Intelligent Learning Environments for Improved Student Diagnosis. *Information Sciences*, 170 (2-4), 273-307.

Stula, M., Stipanicev, D., and Bodrozic, L. (2010). Intelligent Modeling with Agent-Based Fuzzy Cognitive Map. *International Journal on Intelligent Systems*, 25 (10), 981-1004.

Suarez-Cansino, J., and Hernandez-Gomez, A. (2008). Adaptive Testing System Modeled Through Fuzzy Logic, *In Proc. of the second WSEAS Int. Conf on Computer Engineering and Applications (CEA'08)*, Acapulco, Mexico (pp. 85-89).

Stylios, C.D., and Groumpos, P.P. (2004). Modeling complex systems using fuzzy cognitive maps. *IEEE Transactions on Systems Man and Cybernetics: Part A*, 34 (1), 155-162.

Sucar, L.E., and Noguez, J. (2008). Student modeling. In: O. Pourret, P. Naom, & B. Marcot (Eds.), *Bayesian Networks: A Practical Guide to Applications* (pp. 173-185). West Sussex: J. Wiley & Sons.

Suraweera, P., and Mitrovic, A. (2004). An intelligent tutoring system for Entity-Relationship modelling. *Artificial Intelligence in Education*, 14(3-4), 375-417.

Surjono, H. and Maltby, J. (2003). Adaptive educational hypermedia based on multiple student characteristics. *In Proc. of the 2nd international conference on Web-based learning*, Melbourne, Australia (pp. 442-449).

Tan, WW., and Lai, J. (2004). Development of a type-2 fuzzy proportional controller. *In Proc. of IEEE FUZZ Conference*, Budapest, Hungary.

Thomson, D., and Mitrovic, A. (2009). Towards a negotiable student model for constraint-based ITSs. *In Proc. 17th International Conference on Computers in Education*, Hong Kong (pp. 83-90).

Ting, C-Y, and Phon-Amnuaisuk, S. (2012). Properties of Bayesian student model for INQPRO. *Applied Intelligence*, 36 (2), 391–406.

Tourtoglou, K., and Virvou, M. (2008). User Stereotypes Concerning Cognitive, Personality and Performance Issues in a Collaborative Learning Environment for UML. *In Proc. of the 1st International Symposium on Intelligent Interactive Multimedia Systems and Services (KES-IIMSS 2008)*, Piraeus, Greece (pp. 385-394).

Tourtoglou, K., and Virvou, M. (2012). An intelligent recommender system for trainers and trainees in a collaborative learning environment for UML. *Journal of Intelligent Decision Technologies*, 6(2), 79-95.

Tretiakov, A., Sridharan, B., and Kinshuk, (2005). Conceptual Modelling of Web-Based Tutoring Systems. *In Proc. of the International Conference on Computers in Education*, Altona & Melbourne , Australia (pp. 2051-2058).

Tsaganoua, G., Grigoriadou, M., Cavoura, T., and Koutra, D. (2003). Evaluating an intelligent diagnosis system of historical text comprehension. *Expert Systems with Applications*, 25, 493-502.

Tsiriga, V., and Virvou, M. (2002). Initializing the student model using stereotypes and machine learning. *In Proc. of the 2002 IEEE inter. Conf. on System, Man and Cybernetics* (pp. 404-409).

Tsiriga, V., and Virvou, M. (2003a). Modelling the Student to Individualise Tutoring in a Web-Based ICALL. *Inter. Journal of Continuing Engineering Education and Lifelong Learning*, 13 (3-4), 350-365.

Tsiriga, V., and Virvou, M. (2003b). Evaluation of an intelligent web-based language tutor. *In Proc. of the 7th International Conference on Knowledge-Based Intelligent Information and Engineering Systems (KES 2003)*, Oxford, UK (pp. 275–281).

Tsiriga, V., and Virvou, M. (2003c). Initializing Student Models in Web-based ITSs: a Generic Approach. *In Proc. of the 3rd International Conference on Advanced Learning Technologies (ICALT 2003)*, Athens, Greece (pp. 42–46).

VanLehn, K. (2011). The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational Psychologist*, 46 (4), 197–221.

VanLehn, K., Graesser, A., Jackson, G., Jordan, P., Olney, A., and Rose, C.P. (2007). When are tutorial dialogues more effective than reading? *Cognitive Science*, 31 (1), 3–62.

VanLehn, K., Lynch, C., Schulze, K., Shapiro, J.A., Shelby, R.., Taylor, L., Treacy, D., Weinstein, A., and Wintersgill, M. (2005). The Andes Physics Tutoring System: Lessons Learned. *International Journal on Artificial Intelligence in Education*, 15, 147-204.

van Vliet, M., Kok, K., and Veldkamp, T. (2010). Linking stakeholders and modellers in scenario studies: The use of Fuzzy Cognitive Maps as a communication and learning tool. *Futures*, 42 (1), 1-14.

Vasandani, V., and Govindaraj, T. (1995). Knowledge organization in intelligent tutoring systems for diagnostic problem solving in complex dynamic domains. *IEEE Transactions on Systems, Man and Cybernetics*, 25 (7), 1076-1096.

Vélez, J., Fabregat, R., Nassiff, S., Petro, J., and Fernandez, A. (2008). User Integral Model in Adaptive Virtual Learning Environment. *In Proc. of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education*, Las Vegas, Nevada, United States (pp. 3275-3284).

Vesin, B., Ivanović, M., Klašnja-Milićević, A., and Budimac, Z. (2011). Rule-based Reasoning for Building Learner Model in Programming Tutoring System. *In Proc. of the 10th international conference on Advances in Web-Based Learning* (pp. 154-163).

Viccari, R.M., Flores, C.D., Seixas, L., Gluz, J.C., and Coelho, H. (2008). AMPLIA: A Probabilistic Learning Environment. *International Journal of Artificial Intelligence in Education*, 18 (4), 347-373.

Virvou, M., and Kabassi, K. (2002). F-SMILE: An intelligent multi-agent learning environment. *In Proc. of IEEE international conference on advanced learning technologies 2002 (ICALT'02)* (pp. 144-149).

Virvou, M., Katsionis, G., and, Manos, K. (2005). Combining Software Games with Education: Evaluation of its Educational Effectiveness. *Educational Technology and Society*, 8 (2), 54-65.

Virvou, M., Lampropoulos, A.S. and Tsihrintzis, G.A. (2006). Inma: a knowledge-based authoring tool for music education*. In Proc. of the 10th international conference on Knowledge-Based Intelligent Information and Engineering Systems*, United Kingdom (pp. 376-383).

Wang, H.Y., and Chen, S.M. (2006a). New methods for evaluating the answerscripts of students using fuzzy sets. *In Proc. of the 19th international conference on industrial, engineering & other applications of applied intelligent systems*, Annecy, France (pp. 442–451).

Wang, H.Y., and Chen, S.M. (2006b). New methods for evaluating students' answerscripts using fuzzy numbers associated with degrees of confidence. *In Proc. of the 2006 IEEE international conference on fuzzy systems*, Vancouver, BC, Canada (pp. 5492–5497).

Wang., X., Yang, Y., and Wen, X. (2009). Study on Blended Learning Approach for English Teaching*. In Proc. of the 2009 IEEE International Conference on Systems, Man, and Cybernetics* (pp. 4641-4644).

Webb, G. (1998). Preface to UMUAI Special Issue on Machine Learning for User Modeling. *User Modeling and User-Adapted Interaction*, 8, 1-3

Webb, G., Pazzani, M., and Billsus, D. (2001). Machine Learning for User Modeling. *User Modeling and User-Adapted Interaction*, 11, 19-29.

Weerasinghe, A., and Mitrovic, A. (2011). Facilitating adaptive tutorial dialogues in EER-tutor. *In Proc. of the 15th international conference on Artificial intelligence in education, Auckland*, New Zealand (pp. 630-631).

Weibelzahl, S., and Weber, G. (2003). Evaluation of the Inference Mechanism of Adaptive Learning Systems. *In Proc. of the 9th international conference on User modeling*, Johnstown, PA, USA (pp. 154-168).

Welch, M., and Brownell, K. (2000). The development and evaluation of a multimedia course on educational collaboration. *Journal of Educational Multimedia and Hypermedia, 9 (3)*, 169-194.

Wenger, E. (1987). *Artificial intelligence and tutoring systems*. Morgan Kaufmann Publishers, Inc..

Weon, S., and Kim, J. (2001). Learning achievement evaluation strategy using fuzzy membership function. *In Proc. of the 31st ASEE/IEEE frontiers in education conference*, Reno, NV (pp. 19–24).

Wills, K., John, R.I., and Lake, S. (2004). Combining categories in nursing assessment using interval valued fuzzy sets. *In Proc. of 10th Information Processing and Management of Uncertainty in Knowledge Based Systems (IPMU 2004)*, Perugia, Italy.

Wilson, E., Karr, C. L., and Freeman, L. M. (1998). Flexible, adaptive, automatic fuzzy-based grade assigning system. *In Proc. of the 1998 north American fuzzy information processing society (NAFIPS) conference* (pp. 334–338).

Winter, M., Brooks, C., and Greer, J. (2005) .Towards best practices for semantic web student modeling. *In Proc. of the 12th Int. Conf. on Artificial Intelligence in Education*, Amsterdam, the Netherlands (pp. 694-701).

Wisher, R. A., and Fletcher, J. D. (2004). The case for advanced distributed learning. *Information & Security: An International Journal*, 14,17–25.

Woolf, B.P. (2009). *Building intelligent interactive tutors*. Burlington, MA: Morgan Kaufman.

Wu, M. H. (2003). Research on applying fuzzy set theory and item response theory to evaluate learning performance. *Master Thesis, Department of Information Management, Chaoyang University of Technology*, Wufeng, Taichung County, Taiwan, Republic of China.

Wu, D., and Tan, W.W. (2004). A type-2 fuzzy logic controller for the liquid-level process. *In Proc. of IEEE FUZZ Conference*, Budapest, Hungary (pp. 953–958).

Xu, D., Wang, H., and Su, K. (2002). Intelligent Student Profiling with Fuzzy Models. *In Proc. of the 35th Hawaii International Conference on System Sciences*.

Yang, G., Kinshuk, K., and Graf, S. (2010). A practical student model for a location-aware and context-sensitive Personalized Adaptive Learning System. *In Proc. of the IEEE Technology for Education Conference, Bombay*, India (pp. 130-133).

Yu, S., and Zhiping, L. (2009). Ontology-based domain knowledge representation. *In Proc. of the 4th International Conference on Computer Science & Education*, Nanning, China (pp. 174 – 177).

Zadeh, L.A. (1965). Fuzzy sets. *Information and Control*, 8 (3), 338-353.

Zadeh, L.A. (1973). Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transaction on Systems Man and Cybernetics*, 3, 28–44.

Zadeh, L.A. (1975). Fuzzy logic and approximate reasoning. *Synthese*, 30, 407–428.

Zadeh, L.A. (1979). A theory of approximate reasoning, in: J. Hayes, D. Michie, L.I. Mikulich (Eds.), *Machine Intelligence 9*, Halstead Press, New York (pp. 149–194).

Zadeh,L.A. (1996). Fuzzy logic=Computing with words. *IEEE Transactions on Fuzzy Systems*, 4 (2), 103-111.

Zadeh, L.A. (2001). A new direction in AI – toward a computational theory of perceptions. *AI Magazine*, 22 (1), 73–84.

Zadeh, L.A. (2008). Is there a need for fuzzy logic? *Information Sciences*, 178, 2751–2779.

Zapata-Rivera, D. (2007). Indirectly visible Bayesian student models. *In Proc. of the 5th UAI Bayesian Modelling Applications Workshop*, Vancouver, BC, Canada.

Zhang, XM., and Han, H. (2005). An empirical testing of user stereotypes of informational retrieval systems. *Information Processing & Management*, 41 (3), 651-664.

Zeng, J. Liu, Z.-Q. (2004). Interval type-2 fuzzy hidden markov models. In Proc. of IEEE FUZZ Conference, Budapest, Hungary.

List of figures

List of tables

Appendix

Papers that were published during the elaboration of my Ph.D. thesis:

- **In international journals**

[1] Chrysafiadi, K. & Virvou, M. (2014). *Fuzzy Logic for adaptive instruction in an e-learning environment for computer programming. IEEE Transactions on Fuzzy Systems, Article in press*.

[2] Chrysafiadi, K. & Virvou, M. (2013). *PeRSIVA: An Empirical Evaluation Method of a Student Model of an Intelligent e-Learning Environment for Computer Programming*. Computers and Education, Volume 68, pp. 322-333.

[3] Chrysafiadi, K. & Virvou, M. (2013). *Student modeling approaches: A literature review for the last decade*. Expert Systems with Applications, Volume 40, Issue 11, pp. 4715-4729.

[4] Chrysafiadi, K. & Virvou, M. (2013). *A knowledge representation approach using fuzzy cognitive maps for better navigation support in an adaptive learning system.* SpringerPlus, Volume 2, Issue 1, pp. 1-13.

[5] Chrysafiadi, K. & Virvou, M. (2013). *Dynamically Personalized E-Training in Computer Programming and the Language C.* IEEE Transactions on Education, Volume 56, Issue 4, pp. 385-392.

[6] Chrysafiadi, K. & Virvou, M. (2012). *Evaluating the integration of fuzzy logic into the student model of a web-based learning environment.* Expert Systems with Applications, Volume 39, Issue 18, pp. 13127-13134.

- **In international conferences**

[1] Chrysafiadi, K. & Virvou, M. (2013). *Fuzzy Logic for Dynamic Adaptation: Represent and Manage changeable user states*. In Proceeding of the Information, Intelligence, Systems and Applications, IISA 2013, Piraeus, Greece, July 10-12.

[2] Chrysafiadi, K. & Virvou, M. (2012). *Using Fuzzy Cognitive Maps for the Domain Knowledge Representation of an Adaptive e-Learning System*. In Proceeding of the 10th Conference on Knowledge-Based Software Engineering, JCKBSE 2012, Rodos, Greece, August 23-26, pp. 257-265.

[3] Chrysafiadi, K. & Virvou, M. (2011). *Adaptive Navigation in a Web Educational System Using Fuzzy Techniques*. In Proceeding of the 4th International Conference on Intelligent Interactive Multimedia Systems and Services, KES IIMSS 2011, Piraeus, Greece, July 20-22, pp. 81-90.

[4] Chrysafiadi, K. & Virvou, M. (2010)*. Modeling Student's Knowledge on Programming Using Fuzzy Techniques.* In Proceedings of the 3rd International Symposium on Intelligent and Interactive Multimedia: Systems and Services, KES IIMSS 2010, Baltimore, USA, July 28-30.

[5] Chrysafiadi, K. & Virvou, M. (2009). *Usability factors for an intelligent tutoring system on computer programming*. In Proceedings of the 2nd International Symposium on Intelligent Interactive Multimedia Systems and Services, KES IIMSS 2009, Mogliano Veneto, Italy, July 16-17 July.

[6] Chrysafiadi, K. & Virvou, M. (2008). *Personalized teaching of a programming language over the web: Stereotypes and rule-based mechanisms*. In Proceedings of the 8th Joint Conference on Knowledge - Based Software Engineering 2008, JCKBSE 08, Pireaus, Greece, August 25-28, pp. 484-492.

[7] Virvou, M. & Chrysafiadi, K. (2006). *A web-based educational application for teaching of programming: Student modeling via stereotypes*. In Proceedings of the 6th International Conference on Advanced Learning Technologies, ICALT 2006, Kerkrade, The Netherlands, pp. 117-119.