Αυτοματοποιημένη αξιολόγηση προγραμμάτων Ψευδογλώσσας

Γ. Μαυροχαλυβίδης

Καθηγητής Πληροφορικής ΠΕ19, gmeap07@gmail.com

Περίληψη

Η αυτοματοποιημένη αξιολόγηση προγραμμάτων στοχεύει στον αυτοματοποιημένο έλεγχο, στην αξιολόγηση και στη βαθμολόγηση προγραμμάτων που υποβάλλονται από φοιτητές/μαθητές, αποφεύγοντας τα σφάλματα, τις δυσκολίες αλλά και τον υποκειμενικό χαρακτήρα της χειρωνακτικής αξιολόγησης από άτομα/καθηγητές.Το περιβάλλον που αναπτύχθηκε επιτρέπει την υποβολή προγραμμάτων από μαθητές, την αυτοματοποιημένη διόρθωση τους και την καταχώρηση των αποτελεσμάτων της αξιολόγησής τους. Οι μαθητές έχουν τη δυνατότητα πριν την υποβολή της απάντησης να εξασκηθούν στο προγραμματιστικό περιβάλλον της ψευδογλώσσας. Το σύστημα γνωστοποιεί στον μαθητή τόσο το βαθμό που έλαβε όσο και τα τυχόν λάθη που προέκυψαν κατά την εκτέλεση του προγράμματός του. Οι εκπαιδευτικοί αναρτούν τις ασκήσεις και ενημερώνονται για τις επιδόσεις των μαθητών τους. Έχουν τη δυνατότητα να δουν το πρόγραμμα που υπέβαλλε ο μαθητής και τα τυχόν λάθη του ή μόνο το αποτέλεσμα της αξιολόγησης. Μπορούν έτσι να εστιάσουν στους μαθητές τους διαπιστώνοντας άμεσα το επίπεδο γνώσεών τους και τις επιμέρους ανάγκες εκπαίδευσής τους.

Λέξεις κλειδιά: *αυτοματοποιημένη αξιολόγηση προγραμμάτων, επικύρωση, δυναμική ανάλυση.*

1. Εισαγωγή

Η κατανόηση της αλγοριθμικής και η εμπέδωση του προγραμματισμού απαιτούν πολλή πρακτική εξάσκηση (Truong Nghi, 2005). Για να βελτιώσουν οι μαθητές/φοιτητές τα προγράμματά τους, χρειάζονται άμεση ανατροφοδότηση (feedback). Η πρακτική εξάσκηση είναι απαραίτητο να συνδυάζεται με την αξιολόγηση όσο το δυνατόν πιο πολλών προγραμμάτων, να εντοπίζονται τα τυχόν λάθη (συντακτικά ή λογικά), να επικυρώνεται (ή όχι) ο κώδικας (validation) και να υπάρχει ανάδραση (Venables & Haywood, 2003).

Οι παραδοσιακές χειρωνακτικές, μακροσκοπικές μέθοδοι αξιολόγησης των προγραμμάτων των μαθητών/φοιτητών τείνουν να αντικατασταθούν από αυτοματοποιημένα συστήματα αξιολόγησης (APAS Automated programming assessment system) που αναπτύσσονται με την χρήση νέων τεχνολογιών. Τα προγράμματα μπορούν να ελέγχονται, να αξιολογούνται και να βαθμολογούνται αυτοματοποιημένα (Ala-Mutka & Kirsti, 2005). Επιπλέον την τελευταία δεκαετία παρατηρείται σε παγκόσμιο επίπεδο μία τάση ενσωμάτωσης στην εκπαιδευτική διαδικασία των δυνατοτήτων που εισήγαγε το διαδίκτυο (Internet) και ο παγκόσμιος ιστός (www).

Το πληροφοριακό σύστημα αυτοματοποιημένης αξιολόγησης αλγορίθμων σε ψευδογλώσσα που αναπτύχθηκε και παρουσιάζεται στην παρούσα εργασία, είναι στη διάθεση των εκπαιδευτικών και των μαθητών τους και μέσω αυτού οι μεν εκπαιδευτικοί έχουν τη δυνατότητα να παρακολουθούν το επίπεδο των γνώσεων και τις αδυναμίες των μαθητών τους μέσω ικανού αριθμού ασκήσεων που θα τους αναθέτουν, οι δε μαθητές έχουν τη δυνατότητα της εξάσκησης σε κατάλληλο προγραμματιστικό περιβάλλον, αλλά και της υποβολής για αυτοματοποιημένη αξιολόγηση και άμεση επικύρωση των προγραμμάτων τους. Το σύστημα παρέχει στους μαθητές την απαραίτητη ανατροφοδότηση.

Η πρόσβαση στην εκπαιδευτική πλατφόρμα γίνεται μέσω του διαδικτύου στη διεύθυνση www.algorithmos.info.

Το σύστημα μεταξύ άλλων υποστηρίζει:

- την είσοδο του μαθητή και του εκπαιδευτικού στο σύστημα κατόπιν αυθεντικοποίησης.
- τη δυνατότητα του εκπαιδευτικού να αναρτά ασκήσεις και να τις διαχειρίζεται.
- τη δυνατότητα εξάσκησης σε κατάλληλο προγραμματιστικό περιβάλλον.
- την υποβολή του πηγαίου κώδικα του μαθητή σε ψευδογλώσσα και την καταχώρησή του σε βάση δεδομένων.
- τον έλεγχο του πηγαίου κώδικα για συντακτικά σφάλματα και την ενημέρωση για τον εντοπισμό τους.
- την εκτέλεση του προγράμματος με δεδομένα εισόδου που έχουν προκαθοριστεί από τον εκπαιδευτικό και δεν είναι γνωστά στους μαθητές.
- τον εντοπισμό πιθανών λαθών κατά την εκτέλεση των προγραμμάτων.
- την αυτόματη αξιολόγηση των προγραμμάτων με βάση τη σύγκριση των αποτελεσμάτων της εκτέλεσης του προγράμματος σε σχέση με τα αναμενόμενα αποτελέσματα όπως έχουν προκαθοριστεί από τον εκπαιδευτικό που ανάρτησε την άσκηση.
- την καταχώρηση των αποτελεσμάτων της αξιολόγησης σε βάση δεδομένων και την βαθμολόγησή τους (υπό προϋποθέσεις).
- την κοινοποίηση των αποτελεσμάτων της αξιολόγησης/βαθμολόγησης στον μαθητή και τον εκπαιδευτικό.

2. Έλεγχος και εγκυροποίηση λογισμικού

Ο έλεγχος του λογισμικού ορίζεται ως *" η διαδικασία κατά την οποία εξετάζεται το λογισμικό με χρήση ειδικά σχεδιασμένων τεχνικών και με σκοπό την εύρεση και διόρθωση σφαλμάτων στην υλοποίησή του"* (Βεσκούκης, 2000).

Η εγκυροποίηση του λογισμικού περιλαμβάνει τις έννοιες της ***επαλήθευσης (verification)*** και ***επικύρωσης (validation).***

Οι διαδικασίες ***επαλήθευσης (verification)*** εφαρμόζονται συνεχώς κατά τη διάρκεια του κύκλου ανάπτυξης και ορίζονται ως *"οι διαδικασίες αξιολόγησης ενός συστήματος ή ενός τμήματος κάποιου συστήματος με στόχο να προσδιοριστεί κατά πόσο τα προϊόντα μιας συγκεκριμένης φάσης ανάπτυξης ικανοποιούν τις προδιαγραφές που είχαν τεθεί στην αρχή της φάσης".*

Αντίθετα, *"οι διαδικασίες* ***επικύρωσης (validation)*** *εκτελούνται στο τέλος της διαδικασίας ανάπτυξης και ελέγχουν κατά πόσο το λογισμικό που αναπτύχθηκε συμφωνεί με τις αρχικές απαιτήσεις και τις προσδοκίες αυτών που θα το χρησιμοποιήσουν"* (Καμέας, 2000).

Οι διαδικασίες αυτές απαντούν στις εξής ερωτήσεις: (Somerville, 2007)

- Επαλήθευση: αναπτύσσουμε σωστά το λογισμικό;
- Επικύρωση: αναπτύξαμε το σωστό λογισμικό;

2.1 Γενικό πλαίσιο ελέγχου

Επειδή οι δραστηριότητες εγκυροποίησης είναι πολύ σημαντικές στον κύκλο ανάπτυξης του λογισμικού, έχουν προταθεί πολλές τεχνικές επαλήθευσης και επικύρωσης. Οι τεχνικές αυτές χωρίζονται σε δύο κατηγορίες:

- στις ***τυπικές τεχνικές (formal techniques)*** επαλήθευσης και επικύρωσης, οι οποίες παράγουν μια αυστηρή μαθηματική απόδειξη ότι το λογισμικό λειτουργεί σύμφωνα με τις προδιαγραφές του. Το αποτέλεσμα μιας τέτοιας διαδικασίας είναι συνήθως δυαδικό: το δοκιμαζόμενο τμήμα κώδικα ικανοποιεί ή δεν ικανοποιεί τις προδιαγραφές του.

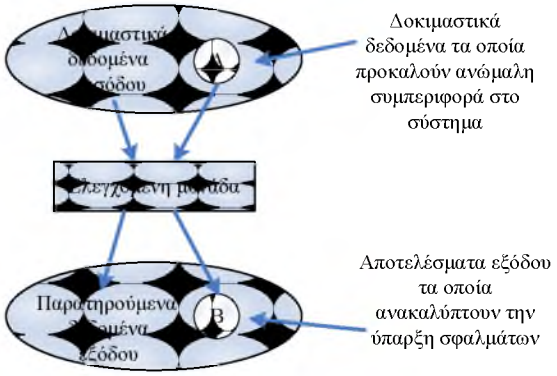
- στις ***μη τυπικές τεχνικές* (informal techniques)** επαλήθευσης και επικύρωσης, οι οποίες δεν μπορούν να αποδείξουν την ορθότητα του λογισμικού, αλλά επικεντρώνονται στην ανακάλυψη και αντιμετώπιση όσο το δυνατόν περισσότερων λαθών.

Οι μη τυπικές τεχνικές επαλήθευσης και επικύρωσης είναι πιο φθηνές και εύκολες στην εφαρμογή, γι᾽ αυτό και είναι οι περισσότερο διαδεδομένες. Ανάλογα με τη μέθοδο ανακάλυψης λαθών που χρησιμοποιούν διακρίνονται σε:

- ***στατικές τεχνικές*** επαλήθευσης και επικύρωσης (static techniques), οι οποίες ασχολούνται με την εξέταση διαφόρων αναπαραστάσεων του συστήματος λογισμικού. Τέτοιες αναπαραστάσεις είναι οι προδιαγραφές, το σχέδιο του συστήματος και ο κώδικας των προγραμμάτων.
- ***δυναμικές τεχνικές*** επαλήθευσης και επικύρωσης (dynamic techniques), με τις οποίες εξετάζεται ο τρόπος υλοποίησης και λειτουργίας ενός λογισμικού. Η πιο σημαντική δυναμική τεχνική είναι ο *έλεγχος του προγράμματος (program testing).*

2.2 Στρατηγική του αδιαφανούς κουτιού (black box testing)

Στις τεχνικές του ελέγχου αδιαφανούς κουτιού, για να σχεδιαστούν οι περιπτώσεις ελέγχου χρησιμοποιούνται οι λειτουργικές απαιτήσεις ή οι προδιαγραφές ενός τμήματος λογισμικού. Το τμήμα αυτό θεωρείται σαν ένα «***αδιαφανές κουτί***», του οποίου η συμπεριφορά προσδιορίζεται από την εξέταση των εισόδων και των εξόδων που αντιστοιχούν σε αυτές.



Εικόνα 1: *Στρατηγική μαύρου κουτιού (Somerville I., 2007)*

2.3 Αυτοματοποιημένα συστήματα αξιολόγησης προγραμμάτων

Τα συστήματα αυτοματοποιημένης αξιολόγησης προγραμμάτων εξελίσσονται για περισσότερο από 50 χρόνια και παρέχουν σημαντική υποστήριξη σε πολλά τμήματα της Επιστήμης Υπολογιστών παγκοσμίως.

Συστήματα πρώτης γενιάς: Πρώιμα συστήματα

Η πρώτη αναφορά τέτοιου συστήματος αναφέρεται από τον Hollingsworth το 1960 (Hollinosworth, 1960). Τα συστήματα αξιολόγησης εξελίχθηκαν ακολουθώντας την ανάπτυξη των συστημάτων προγραμματισμού. Ο Naur (Naur, 1964) και οι Forsythe, Wirth (Forsythe & Wirth, 1965), παρουσίασαν συστήματα βαθμολόγησης για προγράμματα σε Algol. Νέες ιδέες εισήγαγαν οι Hext and Winnings (Hext & Winings, 1969). Η αξιολόγηση των προγραμμάτων γινόταν, συγκρίνοντας την έξοδο της εκτέλεσης του προγράμματος του φοιτητή, με την αναμενόμενη έξοδο που ήταν αποθηκευμένη στο σύστημα.

Συστήματα δεύτερης γενιάς: Tool-oriented συστήματα

Τα συστήματα δεύτερης γενιάς ήταν "βασισμένα σε εργαλεία" (tool–based). Χρησιμοποιούσαν τη γραμμή εντολών (command-line) ή GUI διεπαφές και εργαλεία προγραμματισμού.

Οι Isaacson και Scott (Isaacson & Scott, 1989) παρουσίασαν ένα σύστημα περιγράφοντας μια προσέγγιση του ελέγχου "βασισμένη σε σενάρια". Το σενάριο μεταγλώττιζε το πρόγραμμα και το εκτελούσε χρησιμοποιώντας ένα σύνολο προκαθορισμένων δεδομένων εισόδου. Τα αποτελέσματα της μεταγλώττισης και της

εκτέλεσης καταχωρίζονταν σε ένα αρχείο το οποίο μπορούσε να ελέγξει ο εκπαιδευτικός.

Ο Reek (Reek, 1989) ανέπτυξε ένα διαφορετικό σύστημα. Το σύστημα TRY επέτρεπε στους μαθητές να δοκιμάσουν τα προγράμματά τους χρησιμοποιώντας ένα πρόγραμμα εξάσκησης. Τα αποτελέσματα της εκτέλεσης του προγράμματος καταχωρίζονταν σε μια βάση.

Το project Kassandra (Von Matt, 1994) υποστήριζε τον έλεγχο προγραμμάτων σε Matlab και Maple (μαθηματικές γλώσσες), καθώς και σε Oberon, διάδοχο του Modula-2. Η αξιολόγηση γινόταν και πάλι συγκρίνοντας την έξοδο της εκτέλεσης με τα δεδομένα που είχε προκαθορίσει ο καθηγητής και ήταν αποθηκευμένα στη βάση.

Το σύστημα ASSYST (Jackson & Usher, 1997) αξιολογούσε τα προγράμματα με διάφορα κριτήρια. Το ASSYST έλεγχε αν ο κώδικας ήταν σωστός (συγκρίνοντας την έξοδο του προγράμματος με ένα σύνολο προκαθορισμένων δεδομένων), αν ήταν αποτελεσματικός ως προς τη χρήση του χρόνου της CPU και χρησιμοποιούσε μετρικές για να ελέγξει την πολυπλοκότητα και το ύφος. Μία από τις σημαντικές συνεισφορές αυτού του project ήταν η αντίληψη ότι ένα αυτοματοποιημένο σύστημα αξιολόγησης μπορεί επίσης να είναι και ένα σύστημα βαθμολόγησης - υποστήριξης.

Το σύστημα BOSS αναπτύχθηκε στο Πανεπιστήμιο του Warwick στην Αγγλία. Οι αρχικές προδιαγραφές του ήταν παρόμοιες με εκείνες του ASSYST (Joy& Luck, 1998).

Στο Πανεπιστήμιο του Nottingham αναπτύχθηκε το σύστημα Ceilidh (Benford, Burke, Foxley, Gutteridge & Zin 1993) που υποστήριζε αξιολόγηση βασισμένη σε υπολογιστή, παρέχοντας εργαλεία ανάπτυξης, εκτέλεσης και διαχείρισης.

Συστήματα τρίτης γενιάς: Web-oriented συστήματα

Τα συστήματα αξιολόγησης τρίτης γενιάς κάνουν χρήση των εξελίξεων της τεχνολογίας του διαδικτύου και υιοθετούν ολοένα και πιο εξελιγμένες μεθόδους ανάλυσης.

Το CourseMarker (Higgins, Symeonidis & Tsintsifas, 2002), (Higgins, Hegazy, Symeonidis & Tsintsifas, 2003), ήταν η εξέλιξη του συστήματος Ceilidh. Τα κριτήρια αξιολόγησης ήταν: η μορφή του προγράμματος (τυπογραφική, λεξιλογική δομή και παρουσία ιδιαίτερων χαρακτηριστικών), η εκτέλεση του προγράμματος με δεδομένα ελέγχου (δυναμικός έλεγχος), η πολυπλοκότητα του προγράμματος και η αποτελεσματικότητα εκτέλεσης (χρόνος που δαπανάται για την εκτέλεση).

Το σύστημα BOSS συνέχισε επίσης να αναπτύσσεται (Joy, Griffitis & Boyatt, 2005). Παρείχε πλέον GUI, ήταν εγκατεστημένο σε web server και η πρόσβαση γινόταν μέσω διαδικτύου με web-browser εφαρμογές.

Ο Daly και οι συνεργάτες ανέπτυξαν στο Dublin City University, ένα σύστημα αξιολόγησης προσανατολισμένο σε Java που ονομάζεται RoboProf (Daly, 1999), (Daly & Waldron, 2004). Το σύστημα παρουσιάζει διαδοχικά ασκήσεις

προγραμματισμού στον browser και ο φοιτητής καλείται να πληκτρολογήσει τον κώδικα - απάντηση σε ένα πλαίσιο κειμένου. Το πρόγραμμα υποβάλλεται, αξιολογείται και επιστρέφει το αποτελέσματα. Εάν επικυρωθεί, ο φοιτητής καλείται να απαντήσει στο επόμενο πρόβλημα.

Στην Ελλάδα έχουν αναπτυχθεί διάφορα συστήματα που υποστηρίζουν μαθήματα προγραμματισμού σε Πανεπιστημιακά ιδρύματα. Για τη δευτεροβάθμια εκπαίδευση δεν είχε αναπτυχθεί μέχρι σήμερα κάποιο περιβάλλον αυτοματοποιημένης αξιολόγησης. Η παρούσα υλοποίηση αποτελεί την πρώτη απόπειρα δημιουργίας τέτοιου συστήματος.

2.4 Πλεονεκτήματα και μειονεκτήματα των αυτοματοποιημένων συστημάτων αξιολόγησης προγραμμάτων

Τα σημαντικότερα πλεονεκτήματα των αυτοματοποιημένων συστημάτων αξιολόγησης προγραμμάτων είναι η εξοικονόμηση χρόνου, η ακρίβεια και η αντικειμενικότητα στη διαχείριση, στον έλεγχο και στην αξιολόγηση/βαθμολόγηση των προγραμμάτων και η δυνατότητα άμεσης ανατροφοδότησης. Οι μαθητές ενημερώνονται για τα λάθη που έκαναν ώστε να μην τα επαναλάβουν και οι εκπαιδευτικοί μπορούν να παρακολουθούν τις επιδόσεις των μαθητών τους (Khirulnizam & Nordin, 2007).

Τα μειονεκτήματα εντοπίζονται σε προβλήματα σχεδιασμού, όπως φιλικότητα προς τον χρήστη και έλλειψης υποστήριξης πολλών γλωσσών προγραμματισμού.

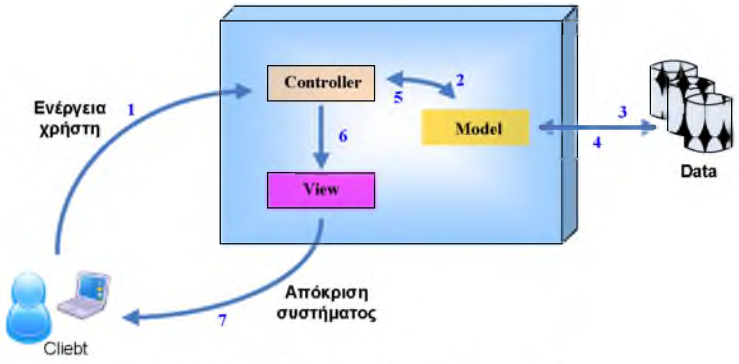
3. Αρχιτεκτονική της εφαρμογής

Για την υλοποίηση του συστήματος αναπτύχθηκε μια δυναμική εφαρμογή ιστού (Web application). Η εφαρμογή χρησιμοποιεί το μοντέλο client – server. Χρησιμοποιώντας το μοντέλο σχεδιασμού MVC (Model – View – Controller) το πρόγραμμα χωρίζεται σε τρία βασικά μέρη :

1. Το μοντέλο **(Model)** είναι υπεύθυνο για την κυρίως λογική της εφαρμογής, αντλεί τα δεδομένα από τη βάση και τα προωθεί μέσω του ελεγκτή στην προβολή.

2. Η προβολή **(View)** παράγει μια παρουσίαση των δεδομένων του μοντέλου και αντιστοιχεί στον σχεδιασμό των ιστοσελίδων, δηλαδή στον τρόπο με τον οποίο αυτές παρουσιάζονται στο χρήστη (γραφικά, διάταξη, χρώματα κτλ.).

3. Ο ελεγκτής **(Controller)** διαχειρίζεται και κατευθύνει τις αιτήσεις που κάνει ο χρήστης. Αναλαμβάνει τη σύνδεση του μοντέλου με την παρουσίαση και την επικοινωνία με τον χρήστη για την επίτευξη της διασύνδεσης αυτού με την παρουσίαση.



Εικόνα 2: *Το μοντέλο της εφαρμογής*

Το σύστημα σε γενικές γραμμές λειτουργεί ως εξής:

Οι εκπαιδευτικοί (μετά τη δημιουργία και ενεργοποίηση του λογαριασμού τους), μπορούν να εγγράψουν τους μαθητές τους δημιουργώντας τμήματα. Οι εκπαιδευτικοί έχουν την ευθύνη της διαχείρισης των λογαριασμών των μαθητών τους (ενεργοποίηση, απενεργοποίηση, διαγραφή, αλλαγή στοιχείων κ.ά.). Μπορούν να ανεβάζουν ασκήσεις δίνοντας τις εισόδους και τις αναμενόμενες εξόδους (που δεν είναι σε γνώση των μαθητών) και να τις διαχειρίζονται (ενεργοποίηση, απενεργοποίηση, διαγραφή, αλλαγή των εισόδων και των εξόδων, κ.ά.).

Οι μαθητές καλούνται να απαντήσουν στις ασκήσεις υποβάλλοντας το πρόγραμμά τους. Πριν να το υποβάλλουν έχουν τη δυνατότητα να κάνουν πρακτική εξάσκηση σε κατάλληλο προγραμματιστικό περιβάλλον που έχει προσαρμοστεί κατάλληλα (Στέργου Σ., 2010). Μετά την υποβολή της απάντησης, το πρόγραμμά εκτελείται με τις εισόδους που έχει προκαθορίσει ο εκπαιδευτικός και τα παραγόμενα αποτελέσματα συγκρίνονται με τα αναμενόμενα που επίσης είναι προκαθορισμένα από τον εκπαιδευτικό. Ο μαθητής ενημερώνεται άμεσα για το αποτέλεσμα της αξιολόγησης και για τα τυχόν λάθη που προέκυψαν. Η κύρια υπηρεσία που παρέχεται στους μαθητές είναι η επικύρωση (validation) ή όχι των προγραμμάτων τους (το εάν δηλαδή το πρόγραμμά τους κάνει πραγματικά αυτό που ισχυρίζεται ότι κάνει).

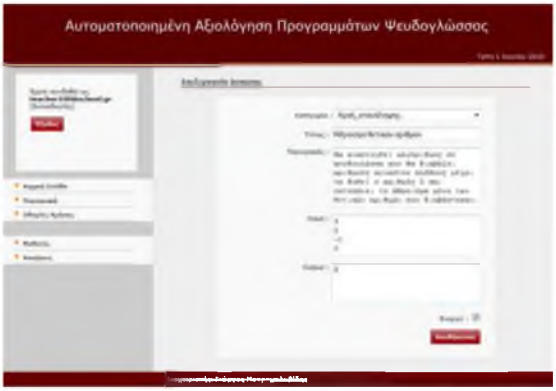
Για την αυτοματοποιημένη αξιολόγηση των αλγορίθμων χρησιμοποιείται η στρατηγική του μαύρου κουτιού (Somerville, 2007) και ο έλεγχος του αλγόριθμου μέσω δυναμικής ανάλυσης.

Ο αλγόριθμος αξιολογείται αυτόματα με 3 κριτήρια :

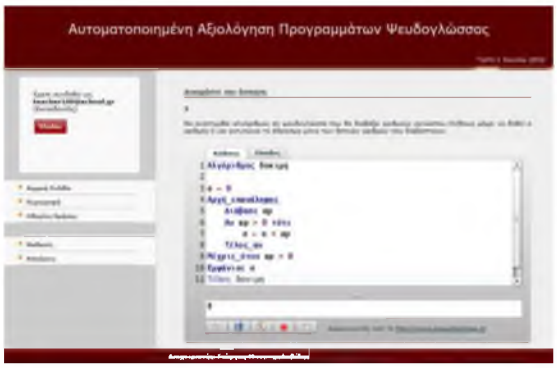
- αν υπήρξαν ή όχι συντακτικά σφάλματα
- αν υπήρξαν ή όχι άλλα σφάλματα (π.χ. διαίρεση με 0, ατέρμων βρόχος κ.ά)
- αν τα αποτελέσματα της εκτέλεσης του προγράμματος συμπίπτουν ή όχι με τα αναμενόμενα όπως τα έχει προκαθορίσει ο εκπαιδευτικός.

4. Υλοποίηση - Γραφικό περιβάλλον διεπαφών

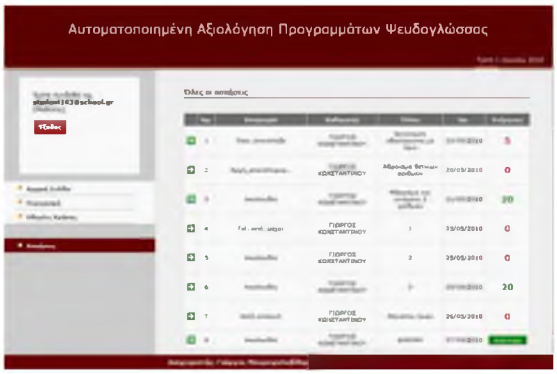
Ενδεικτικά παρουσιάζονται 4 οθόνες του γραφικού περιβάλλοντος διεπαφής:



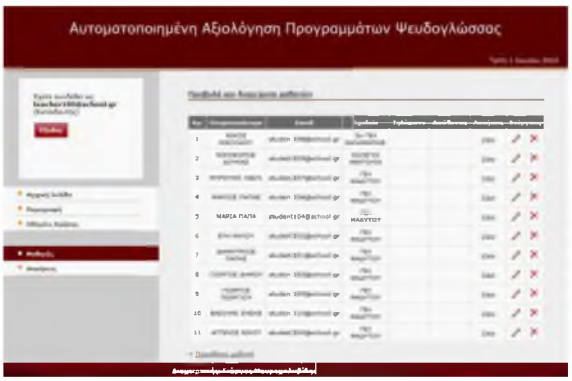
Εικόνα 3: *Οθόνη ανάρτησης – τροποποίησης άσκησης*



Εικόνα 4: *Οθόνη προβολής – δοκιμής κώδικα*



Εικόνα 5: *Οθόνη προβολής αποτελεσμάτων αξιολόγησης*



Εικόνα 6: *Οθόνη προβολής και διαχείρισης μαθητών*

5. Συμπεράσματα

Η πλατφόρμα λειτουργεί από τον Σεπτέμβριο του 2010 και χρησιμοποιείται από δεκάδες εκπαιδευτικούς και μαθητές. Στο διάστημα αυτό αξιολογήθηκε με συνεχή επαφή με τους χρήστες αλλά και μέσω ερωτηματολογίων.

Η εφαρμογή κρίθηκε από εκπαιδευτικούς και μαθητές θετικά. Φαίνεται πως τείνει να εκπληρώσει τον εκπαιδευτικό της σκοπό και να αποτελέσει ένα χρήσιμο εργαλείο για τη διδασκαλία των αλγορίθμων.

Βιβλιογραφία

Ala-Mutka & Kirsti M. (2005), A Survey of Automated Assessment Approaches for Programming Assignments, *Computer Science Education 15*, 83-102.

Benford, S., Burke, E., Foxley, E., Gutteridge, N. & Zin, A. M. (1993), Experience using the Ceilidh system, *Proceedings of the 1st All-Ireland conference on the teaching of computing*, 32-35.

Daly, C. (1999), RoboProf and an introductory computer programming course, *Proceedings of the 4th Annual SIGCSE/SIGCUE ITiCSE Conference on Innovation and Technology in Computer Science Education*. 155-158.

Daly, C. & Waldron J. (2004), Assessing the assessment of programming ability, *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education*, 210-213.

Forsythe, G. E. & Wirth, N. (1965), Automatic grading programs, *Commun ACM 8, 5*, 275-529.

Hext, J. B. & Winings, J. W. (1969), An Automatic Grading Scheme for Simple Programming Exercises, *ACM Volume 12 , Issue 5*, 272 – 275.

Higgins Colin, Symeonidis Pavlos & Tsintsifas Athanasios (2002), The Marking System for CourseMaster, *Annual Joint Conference Integrating Technology into*

Computer Science Education Proceedings of the 7th annual conference on Innovation and technology in computer science education, 46 - 50

Higgins C., Hegazy T., Symeonidis P., & Tsintsifas A. (2003), The CourseMaster CBA system: Improvements over Ceilidh, *J Edu. Inf Technol. 8, 3*, 287-304.

Hollinosworth, J. (1960), Automatic graders for programming classes, *ACM Volume 3, Issue 10*, 528-529

Isaacson P. C. & Scott T. A. (1989), Automating the execution of student programs, *SIGCSE Bull. 21, 2*, 15-22.

Jackson D & Usher M. (1997) Grading student programing using ASSYST, *Technical Symposium on Computer Science Education, Proceedings of the 28th SIGCSE (San Jose, CA)*, 335-339.

Joy Mike, Griffitis Nathan, & Boyatt Russell (2005). The BOSS Online Submission and Assessment System, *Journal on Educational Resources in Computing (JERIC), Volume 5, Issue 3, Article No.: 2*

Joy, M. & Luck, M. (1998), Effective electronic marking for on-line assessment, *In Proceedings of the 6th Annual Conference on the Teaching of Computing/3rd Annual Conference on Integrating Technology into Computer Science Education*, 134-138.

Khirulnizam A. Rahman, & Nordin Md. Jan (2007), A Review on the Static Analysis Approach in the Automated Programming Assessment Systems, *National Conference on Programming 07, Kuala Lumpur, Malaysia.*

Naur, P. (1964). Automatic grading of students' ALGOL programming, *BIT 4*, 177-188.

Reek K. A. (1989). The TRY system – or – how to avoid testing student programs, *SIGCSE Bull. 21, 1*, 112-116.

Somerville Ian, (2007). *Software engineering* (8th Edition), Addison-Wesley, ISBN: 9780321313799

Truong Nghi (2005). ELP – A Web Environment For Learning To Program, *ACM SIGCSE Bulletin, Vol 37, Issue 3.*

Truong Nghi, Roe Paul & Bancroft Peter (2005), Automated feedback for "fill in the gap" programming exercises, *ACM, Proceedings of the 7th conference on Australasian computing education, vol. 106*, 117 - 126 .

Truong Nghi, Roe Paul & Bancroft Peter, (2004), Static Analysis of Students' Java Programs, *Proceedings of the 6th conference on Australasian computing education, vol. 30*, 317-325.

Venables A. & Haywood L. (2003), Programming students NEED instant feedback!, *Paper read at Conferences in Research and Practice in Information Technology,*

Proceedings of the fifth Australasian conference on computing education - Volume 20, 267 – 272.

Von Matt U., (1994). *Kassandra:* The automatic grading system, *Tech. Rep.UMIACS-TR-94-59, Institute for Advanced Computer Studies, Dept. of Computer Science, University of Maryland.*

Βεσκούκης Β. (2000). *Τεχνολογία λογισμικού,* Ελληνικό Ανοιχτό Πανεπιστήμιο, Πάτρα.

Καμέας Α. (2000). *Εγκυροποίηση λογισμικού,* Ελληνικό Ανοιχτό Πανεπιστήμιο, Πάτρα.

Στέργου Ε., *"Διερμηνευτής ψευδογλώσσας"*, Ανακτήθηκε το 10/12/2010 από http://www.pseudoglossa.gr/