ePublishing
ΕΘΝΙΚΟ ΚΕΝΤΡΟ ΤΕΚΜΗΡΙΩΣΗΣ
NATIONAL DOCUMENTATION CENTRE

Συνέδρια της Ελληνικής Επιστημονικής Ένωσης Τεχνολογιών Πληροφορίας & Επικοινωνιών στην Εκπαίδευση

Τόμ. 1 (2008)

4ο Συνέδριο Διδακτική Πληροφορικής

Μέθοδος για την Ιεραρχική Αξιολόγηση Γνώσεων Προγραμματισμού

Ι. Μπέλλου, Τ. Α. Μικρόπουλος

Βιβλιογραφική αναφορά:

Μπέλλου Ι., & Μικρόπουλος Τ. Α. (2023). Μέθοδος για την Ιεραρχική Αξιολόγηση Γνώσεων Προγραμματισμού . *Συνέδρια της Ελληνικής Επιστημονικής Ένωσης Τεχνολογιών Πληροφορίας & Επικοινωνιών στην Εκπαίδευση*, *1*, 111–120. ανακτήθηκε από https://eproceedings.epublishing.ekt.gr/index.php/cetpe/article/view/5856

https://epublishing.ekt.gr | **e-Εκδότης:** EKT | **Πρόσβαση:** 25/02/2026 15:53:57

Μέθοδος για την Ιεραρχική Αξιολόγηση Γνώσεων Προγραμματισμού

Ι. Μπέλλου[1], Τ. Α. Μικρόπουλος[2]

[1]Σχολική Σύμβουλος Πληροφορικής
ibellou@sch.gr
[2]Εργαστήριο Εφαρμογών Εικονικής Πραγματικότητας στην Εκπαίδευση,
Πανεπιστήμιο Ιωαννίνων
amikrop@uoi.gr

Περίληψη

Η εργασία προτείνει την 'Ιεραρχική Αξιολόγηση Προγραμματισμού' (Hierarchical Assessment of Programming, HAP), μια μέθοδο για την ποιοτική αξιολόγηση της επίλυσης προγραμματιστικών προβλημάτων. Η 'Ιεραρχική Αξιολόγηση Προγραμματισμού' βασίζεται στα υπάρχοντα θεωρητικά μοντέλα για την ανάπτυξη της σκέψης στον προγραμματισμό, υιοθετεί ορισμένες αρχές της γνωστικής ταξινομίας μαθησιακών αποτελεσμάτων SOLO και λαμβάνει υπόψη τη διάκριση της γνώσης σε δηλωτική, δομική, διαδικαστική και στρατηγική. Ακολουθώντας αναλυτική και συνθετική διαδικασία για τη συντακτική και εννοιολογική γνώση στον προγραμματισμό, συγκεκριμενοποιεί και διατυπώνει τα κριτήρια αξιολόγησης της γνώσης προγραμματισμού, διαμορφώνοντας και παρουσιάζοντας πέντε ιεραρχικά επίπεδα, που ορίζονται με άξονες αφενός την ανάπτυξη αλγοριθμικής σκέψης για την επίλυση προβλήματος, και αφετέρου τις δεξιότητες στη γλώσσα προγραμματισμού.

Λέξεις κλειδιά: *Ιεραρχική Αξιολόγηση Προγραμματισμού, ταξινομία SOLO, συντακτική, εννοιολογική, στρατηγική γνώση*

Abstract

This work proposes the 'Hierarchical Assessment of Programming' (HAP), a method for the qualitative assessment of solving programming problems. The 'Hierarchical Assessment of Programming' is based on existing theoretical models for knowledge development in programming, adopts certain principles of the SOLO learning outcomes taxonomy, and takes into account the declarative, structural, procedural and strategic types of knowledge. Following analytic and synthetic process for the syntactic and conceptual knowledge in programming, our model instantiates and expresses assessment criteria for the programming knowledge, by configuring five hierarchical levels, which are determined both by the development of algorithmic thinking and programming skills.

Keywords: *Hierarchical Assessment of Programming, SOLO taxonomy, syntactic, conceptual, strategic knowledge*

1. Εισαγωγή

Η αξιολόγηση της μαθησιακής διαδικασίας και ειδικότερα των μαθησιακών αποτελεσμάτων αποτελεί κύριο ζητούμενο στην εκπαιδευτική διαδικασία, για κάθε

γνωστικό αντικείμενο και βαθμίδα. Το θέμα αποκτά μεγάλο ερευνητικό και πρακτικό ενδιαφέρον στη διδακτική της πληροφορικής και συγκεκριμένα στη διδασκαλία του προγραμματισμού, ως νέου πεδίου, που απαιτεί ταυτόχρονα πνευματικές και τεχνικές δεξιότητες υψηλού επιπέδου και διαφορετικού τύπου. Οι κατευθυντήριες γραμμές και η μεθοδολογία της αξιολόγησης οφείλει να ακολουθεί τις εκάστοτε ψυχοπαιδαγωγικές προσεγγίσεις, τη σχετική εκπαιδευτική έρευνα, το πλαίσιο προγράμματος σπουδών και το αναλυτικό πρόγραμμα, με τους σκοπούς και τους στόχους του.

Από τη δεκαετία του 1980 οι ερευνητές ασχολούνται με τη γνώση στον προγραμματισμό (Bayman & Mayer, 1988), προτείνουν πλαίσια για την ανάλυσή της (McGill & Volet, 1997), μοντέλα για την επίλυση προβλημάτων με προγραμματισμό (Deek et al., 1999; de Raadt, 2007), και υλοποιούν εμπειρικές μελέτες σχετικές με τη διδασκαλία και τη μάθηση του προγραμματισμού (Van Gorp & Grissom, 2001; de Raadt, 2007). Μια εκτεταμένη βιβλιογραφική επισκόπηση στο διεθνή και στον ελληνικό χώρο δείχνει ότι ενώ υπάρχουν προτάσεις για την ανάπτυξη της γνώσης και αποτελέσματα από εμπειρικές μελέτες στον προγραμματισμό, λείπει ένα πλαίσιο αξιολόγησης της γνώσης και των μαθησιακών αποτελεσμάτων, με σαφή κριτήρια και συνέπεια ως προς τις θεωρητικές προσεγγίσεις και τα συμπεράσματα των μελετών. Η έλλειψη αυτή είναι ουσιαστική σε έναν τομέα όπως ο προγραμματισμός όπου συνδυάζονται δύο σημαντικοί παράγοντες, η αλγοριθμική σκέψη με τη γνώση γλώσσας προγραμματισμού. Μόνο ένας σωστός και ολοκληρωμένος συνδυασμός τους μπορεί να οδηγήσει στο επιθυμητό αποτέλεσμα της επίλυσης του προβλήματος με προγραμματισμό.

Η παρούσα εργασία προτείνει την 'Ιεραρχική Αξιολόγηση Προγραμματισμού', ένα μοντέλο ποιοτικής αξιολόγησης της λύσης προβλημάτων σε προγραμματιστικά περιβάλλοντα, βασιζόμενο στους δύο ουσιαστικούς παράγοντες, την αλγοριθμική σκέψη και τις προγραμματιστικές δεξιότητες, σε ένα εποικοδομητικό πλαίσιο.

2. Η γνώση στον προγραμματισμό

Ο στόχος του μαθητή κατά την επίλυση ενός προγραμματιστικού προβλήματος είναι να αναπτύξει μια εκτελέσιμη διαδικασία από μια φορμαλιστική μηχανή (Κόμης, 2005). Το αποτέλεσμα έχει ιδιαίτερη αξία όταν το πρόβλημα είναι αυθεντικό, δεν έχει δηλαδή προηγουμένως διδαχθεί (Τζιμογιάννης, 2003). Η λύση ενός προβλήματος με μια μηχανή απαιτεί αφενός τεχνικές δεξιότητες, που αφορούν κυρίως μια γλώσσα προγραμματισμού, και αφετέρου νοητικές διαδικασίες υψηλού επιπέδου, που αναφέρονται τόσο στον αλγόριθμο που πρέπει να σχεδιασθεί, όσο και στη γλώσσα προγραμματισμού για την υλοποίησή του και την εκτέλεση του προγράμματος.

Κατά την επίλυση ενός προβλήματος ο μαθητής κατασκευάζει νοητικά μοντέλα που αποτελούνται συνήθως από πολλαπλές αναπαραστάσεις και περιλαμβάνουν διαφορετικούς τύπους γνώσης όπως δηλωτική, δομική, διαδικαστική, στρατηγική,

λειτουργική, συναισθηματική, κοινωνική. Η δηλωτική, η διαδικαστική και η στρατηγική γνώση απαιτούνται κυρίως για την επίλυση προβλημάτων. Αυτοί είναι και οι τύποι γνώσης που αξιοποιούνται από τους ερευνητές για τη δημιουργία εννοιολογικών πλαισίων για την ανάλυση της γνώσης κατά τον προγραμματισμό (McGill & Volet, 1997; Τζιμογιάννης, 2003).

Ως δηλωτική (declarative) αναφέρεται η γνώση σχετικά με ένα αντικείμενο, ένα γεγονός, μία έννοια. Αποτελείται από τις πληροφορίες που αφορούν το αντικείμενο, το γεγονός ή την έννοια και αποτελεί το υπόβαθρο για την ανάπτυξη σκεπτικού πάνω σε ένα θέμα που απαιτεί τη διαχείριση του αντικειμένου, του γεγονότος ή της έννοιας. Η διαδικαστική (procedural) γνώση είναι η γνώση του τρόπου με τον οποίο αξιοποιείται η δηλωτική γνώση για την επιχειρηματολογία σε ένα θέμα, την επίλυση ενός προβλήματος, τη λήψη μιας απόφασης. Η δηλωτική και η διαδικαστική γνώση σχετίζονται μεταξύ τους. Είναι όμως συνηθισμένο το φαινόμενο κατά το οποίο κάποιος έχει πρωτογενή γνώση γύρω από ένα θέμα αλλά αδυνατεί να την αξιοποιήσει, κατέχει δηλαδή αδρανή γνώση. Φαίνεται ότι για την οικοδόμηση της γνώσης είναι σημαντικό το πέρασμα από τη δηλωτική στη διαδικαστική γνώση και γι' αυτό απαιτείται κάποιου τύπου συνδετικός κρίκος. Αυτός είναι ένας άλλος τύπος γνώσης, η δομική (structural) (Jonassen et al., 1993). Η δομική γνώση, ή αλλιώς γνωστική δομή (cognitive structure), αναφέρεται στη γνώση του τρόπου σύνδεσης, συσχετισμού και ολοκλήρωσης των εννοιών που εμπλέκονται στο υπό μελέτη θέμα. Η γνώση των σχέσεων μεταξύ εννοιών, η δεξιότητα περιγραφής και αναπαράστασής τους και η συνολική οργάνωσή τους, δομεί το περιβάλλον που οδηγεί στη διαδικαστική γνώση και τελικά στη μάθηση. Η ανάπτυξη σκεπτικού σχετικά με τις σχέσεις εννοιών, η αξιολόγηση αυτών των σχέσεων και ο αναστοχασμός έχουν ως συνέπεια τη δημιουργία ενός πλαισίου διατεταγμένων κατηγοριών ανάλυσης και κατανόησης δεδομένων και την ανάπτυξη μεταγνωστικών δεξιοτήτων.

Με βάση προσεγγίσεις της γνωστικής ψυχολογίας και την εκπαιδευτική έρευνα στη διδασκαλία του προγραμματισμού, οι McGill και Volet προτείνουν το 1997 ένα εννοιολογικό πλαίσιο για τις γνώσεις που οικοδομούνται κατά τον προγραμματισμό. Διακρίνουν και συνδυάζουν τρεις τύπους γνώσης που δίνονται από την προσέγγιση της ψυχολογίας και τρεις τύπους γνώσης από την πλευρά της εκπαιδευτικής έρευνας. Από το χώρο της ψυχολογίας αναφέρουν τη δηλωτική, τη διαδικαστική και τη στρατηγική (υπό συνθήκη – conditional) γνώση με την ερμηνεία που τους δίνεται στην προηγούμενη παράγραφο. Από την πλευρά της εκπαιδευτικής έρευνας προτείνουν τη συντακτική, την εννοιολογική και τη στρατηγική γνώση. Συνδυάζοντας τους παραπάνω τύπους προτείνουν ένα μοντέλο δύο διαστάσεων που δημιουργεί τις συνιστώσες της γνώσης στον προγραμματισμό. Οι τέσσερις κατηγορίες γνώσης που προκύπτουν είναι η δηλωτική – συντακτική, δηλωτική – εννοιολογική, διαδικαστική – συντακτική, διαδικαστική – εννοιολογική. Η στρατηγική γνώση αναφέρεται ως το ανώτερο επίπεδο γνώσης, εκτεταμένης θεώρησης, κατά το οποίο ο προγραμματιστής μπορεί να αντιμετωπίσει με πληρότητα

ένα νέο, αυθεντικό πρόβλημα. Οι συγγραφείς συμφωνούν με την εργασία της Linn (1985) η οποία συνδυάζει και ιεραρχεί πνευματικές δεξιότητες και γνώσεις για τον προγραμματισμό λαμβάνοντας υπόψη και την αλγοριθμική σκέψη εκτός από τις δεξιότητες στον προγραμματισμό που εξαρτώνται από κάθε γλώσσα. Όμως δε φαίνεται να περιλαμβάνουν στο μοντέλο των γνώσεων που προτείνουν την αλγοριθμική σκέψη, εκτός από τη γενική τους πρόταση 'ικανότητα σχεδίασης λύσεων σε προγραμματιστικά προβλήματα'. Αυτό προκύπτει και από τα παραδείγματα που παραθέτουν στο άρθρο τους, τα οποία δεν περιλαμβάνουν παρά προγραμματιστικές τεχνικές και γνώσεις. Το μοντέλο των McGill και Volet συμφωνεί με αντίστοιχα εμπειρικά δεδομένα για τους τύπους γνώσης στον προγραμματισμό, αλλά δεν μπορεί να αξιοποιηθεί ως ένα μοντέλο αξιολόγησης, αφού δεν παρουσιάζει ιεράρχηση, ούτε κριτήρια αξιολόγησης. Όμως και το μοντέλο της Linn, παρότι είναι ιεραρχημένο, δεν μπορεί να αξιοποιηθεί για αξιολόγηση γιατί είναι γενικό και δεν προτείνει συγκεκριμένα κριτήρια. Η 'αλυσίδα της γνωστικής εκπλήρωσης' όπως την αναφέρει, περιλαμβάνει τα εξής τρία γενικά βήματα:

1. Απόκτηση γνώσης σχετικά με τα χαρακτηριστικά της γλώσσας προγραμματισμού
2. Απόκτηση δεξιοτήτων σχεδίασης που περιλαμβάνουν διαδικαστική γνώση
3. Ανάπτυξη δεξιοτήτων επίλυσης προβλήματος.

Επιπλέον οι McGill και Volet, παρότι βασίζουν το μοντέλο τους στους τύπους γνώσης, δεν αναφέρουν τη δομική γνώση που θεωρείται απαραίτητη για τη μετάβαση από τη δηλωτική στη διαδικαστική γνώση. Αυτή αναφέρεται σε ένα άλλο πλαίσιο συνθηκών για τη γνωστική δραστηριότητα στον προγραμματισμό (Green et al., 1990; Κόμης, 2005). Και αυτό επίσης το πλαίσιο θεωρείται γενικό και μη ιεραρχημένο και δεν μπορεί να αξιοποιηθεί για την κατάταξη του κάθε τύπου γνώσης και την αξιολόγηση της συνολικής γνώσης στον προγραμματισμό.

Οι Τζιμογιάννης (2003) και Κόμης (2005) προσαρμόζουν και μεταφέρουν το μοντέλο των McGill και Volet παρουσιάζοντας διαφοροποιημένα παραδείγματα για ορισμένους τύπους γνώσης. Η ουσιαστική όμως προσαρμογή αφορά στην ενσωμάτωση στο μοντέλο στοιχείων αλγοριθμικής σκέψης, του πεδίου δηλαδή της επίλυσης προβλημάτων. Και πάλι, θεωρούμε ότι το μοντέλο περιλαμβάνει και συνδυάζει επιτυχημένα τους τύπους γνώσης για τον προγραμματισμό, αλλά δεν παρουσιάζει ιεράρχηση (Κόμης, 2007) ούτε κριτήρια και επομένως δεν μπορεί να αξιοποιηθεί για την αξιολόγηση της γνώσης.

Ο George (2000) αναφέρει τη σπουδαιότητα δημιουργίας των κατάλληλων νοητικών μοντέλων για την επίλυση προγραμματιστικών προβλημάτων. Ενώ οι μαθητές μπορεί να έχουν γνώση του συντακτικού μιας γλώσσας, χωρίς τα κατάλληλα νοητικά μοντέλα δεν μπορούν να κατανοήσουν πολύπλοκες διεργασίες και να λύσουν προβλήματα. Η διαπίστωση αυτή υπονοεί ουσιαστικά τους τύπους γνώσης και παραπέμπει σε ένα μοντέλο όπως αυτό των McGill και Volet (1997).

Ο Robins και οι συνεργάτες του επαναθέτουν το θέμα της γνώσης και των στρατηγικών υπονοώντας τη δηλωτική και τη διαδικαστική γνώση στον προγραμματισμό (2003). Ακολουθούν και αυτοί τα δεδομένα από τους χώρους της γνωστικής ψυχολογίας και της εκπαιδευτικής έρευνας και προτείνουν ένα πλαίσιο για τις συνιστώσες που σχετίζονται με τον προγραμματισμό (γνώση, στρατηγικές, μοντέλα) και την αξιολόγηση προγραμματιστικών γνώσεων (σχεδίαση, δημιουργία, αξιολόγηση). Το πλαίσιο συγκεκριμενοποιεί ως κάποιο βαθμό το αντίστοιχο των McGill και Volet, δεν παρέχει όμως μια μεθοδολογία αξιολόγησης.

Μια πρόταση για την ιεράρχηση γνωστικών επιπέδων με σκοπό την αξιολόγηση μαθησιακών αποτελεσμάτων αποτελεί η ταξινομία SOLO (Structure of the Observed Learning Outcomes) των Biggs και Collis (1982) που αξιοποιείται σε ποικίλα γνωστικά αντικείμενα. Κατά τη SOLO, τα γνωστικά επίπεδα των μαθητών διακρίνονται σε πρώτο επίπεδο προδομικό ή πρώιμο, δεύτερο μονοπαραγοντικό ή μονοδομικό, τρίτο πολυπαραγοντικό ή παραθετικό ή πολυδομικό, τέταρτο συσχετιστικό ή συνδυαστικό ή συνθετικό και πέμπτο επίπεδο θεωρητικής γενίκευσης ή εκτεταμένης θεώρησης. Η ταξινομία έχει χρησιμοποιηθεί στην αξιολόγηση μαθησιακών αποτελεσμάτων και στον προγραμματισμό, μετά από τροποποίηση και ερμηνεία των πέντε επιπέδων (de Raadt, 2007). Στον προγραμματισμό ως πρώτο επίπεδο θεωρείται η έλλειψη απάντησης, ως προδομικό η ουσιαστική έλλειψη γνώσης προγραμματιστικών δομών, ως μονοδομικό επίπεδο η περιγραφή ενός τμήματος του κώδικα, πολυδομικό η περιγραφή κώδικα γραμμή προς γραμμή και ως συσχετιστικό το σύνολο του απαιτούμενου κώδικα. Πέμπτο επίπεδο δεν προτείνεται. Όπως φαίνεται και από την περιγραφή των επιπέδων, αυτά παραμένουν γενικά και αόριστα, αφορούν σε τμήματα του κώδικα χωρίς να αναφέρονται τύποι γνώσης, παρότι η ταξινομία SOLO είναι γνωστική και αποτελεί μια εποικοδομητική προσέγγιση (Μπέλλου, 2003). Επίσης, η ταξινομία δεν κάνει διάκριση μεταξύ δεξιοτήτων επίλυσης προβλήματος (αλγοριθμική σκέψη) και δεξιοτήτων στη γλώσσα προγραμματισμού. Οι ίδιοι περιορισμοί εντοπίζονται και σε πρόσφατες μελέτες μεγάλης κλίμακας από τους Lister και Thompson οι οποίοι όμως παραμένουν σε προβλήματα κατά την κατανόηση έτοιμου κώδικα και όχι κατά την ανάπτυξη αλγορίθμου και τη συγγραφή του κατάλληλου κώδικα από μαθητές (Thompson et al., 2006; Lister et al., 2006).

Οι Κουτσάκας και Roberts αξιολογούν γνώσεις στον προγραμματισμό χρησιμοποιώντας την ταξινομία SOLO, αναλύοντας απαντήσεις μαθητών ως προς τέσσερις συνιστώσες (2005). Αναφέρουν τη χωρητικότητα της λειτουργικής μνήμης ή το βαθμό προσοχής των μαθητών, τη συσχετιστική λειτουργία, τη συνέπεια και περάτωση, και τη δομή της απάντησης. Οι συνιστώσες θεωρούνται γενικές και δύσκολα μετρήσιμες, καθώς δεν αναφέρεται ο τρόπος μέτρησής τους για την κατάταξη των απαντήσεων των μαθητών.

Όπως προκύπτει και από την πρόσφατη βιβλιογραφική επισκόπηση (de Raadt, 2007) δε φαίνεται να προτείνεται μια συγκεκριμένη μεθοδολογία αξιολόγησης της γνώσης

στον προγραμματισμό. Οι θεωρητικές προσεγγίσεις στη διδακτική του προγραμματισμού αναφέρονται κατά κύριο λόγο στην ανάπτυξη μοντέλων γνώσης, τα οποία σταματούν ουσιαστικά να εμφανίζονται την τελευταία δεκαετία και εξαρτώνται άμεσα από τις γλώσσες προγραμματισμού.

3. Ιεραρχική αξιολόγηση προγραμματισμού

Η παρούσα εργασία προτείνει την 'Ιεραρχική Αξιολόγηση Προγραμματισμού' (ΗΑΡ), ένα μοντέλο για την ποιοτική αξιολόγηση των γνώσεων στον προγραμματισμό. Το μοντέλο ακολουθεί την εποικοδομητική θεώρηση και το πλαίσιο των McGill και Volet, το οποίο επεκτείνει λαμβάνοντας υπόψη και τη δομική γνώση, φιλοδοξώντας να δημιουργήσει ένα εργαλείο για την εκπαιδευτική έρευνα και πράξη. Οδηγούμενο από την ανάγκη για μια ιεραρχική δομή αξιολόγησης, το μοντέλο ΗΑΡ βασίζεται στη λογική της ταξινομίας SOLO και προτείνει πέντε αξιολογικά επίπεδα, δανειζόμενο ορισμένες από τις αρχές του.

Ως αφετηρία για τη δημιουργία του ΗΑΡ χρησιμοποιείται η διάκριση των γνώσεων που αφορούν τον προγραμματισμό σε δύο κατηγορίες. Η πρώτη αφορά στο συνδυασμό συντακτικής – γραμματικής και εννοιολογικής γνώσης για το χειρισμό της γλώσσας προγραμματισμού και η δεύτερη αφορά στο συνδυασμό εννοιολογικής και στρατηγικής γνώσης του προγραμματισμού, δηλαδή στην αλγοριθμική σκέψη. Σύμφωνα και με την εκπαιδευτική έρευνα στη διδακτική της πληροφορικής αυτοί οι δύο τύποι γνώσης θεωρούνται συμπληρωματικοί και ιεραρχημένοι, με τη δεύτερη να βρίσκεται σε υψηλότερο επίπεδο (McGill & Volet, 1997). Ο Πίνακας 1 παρουσιάζει την ανάλυση κάθε μιας από τις δύο κατηγορίες της προγραμματιστικής γνώσης. σε ιεραρχημένα επίπεδα.

Πίνακας 1: *Επίπεδα ιεράρχησης των δύο συνιστωσών του ΗΑΡ*

Γνώση στη γλώσσα προγραμματισμού	**Αλγοριθμική σκέψη**
1. Λανθασμένη χρήση εντολών	1. Κατανόηση του προβλήματος εν μέρει
2. Ορθή χρήση απλών εντολών, όπως εισόδου – εξόδου, δηλωτικών, εκχώρησης	2. Κατανόηση του προβλήματος, αλλά ελλιπής αλγόριθμος
3. Ορθή χρήση λογικών δομών, όπως επιλογής και επανάληψης, κατανόηση λειτουργίας εντολών, μεταβλητών και μνήμης	3. Σχεδιασμός της λύσης σε γενικό πλαίσιο, χωρίς να λαμβάνονται υπόψη οριακές περιπτώσεις και περιορισμοί

4. Δεξιότητες προηγούμενου επιπέδου και επιπλέον επιλογή κατάλληλων δομών επανάληψης, συναρτήσεων, διαδικασιών	4. Οργάνωση και απόδοση της λύσης του προβλήματος με πλήρη και ικανοποιητικά δομημένη προσέγγιση
5. Βαθιά γνώση και φιλοσοφία της γλώσσας προγραμματισμού	5. Επιλογή και υλοποίηση του βέλτιστου αλγόριθμου

Με βάση τα παραπάνω, από τους δυνατούς συνδυασμούς προκύπτουν πέντε ιεραρχικά επίπεδα μαθησιακών αποτελεσμάτων, που συγκροτούν το μοντέλο 'Ιεραρχική Αξιολόγηση Προγραμματισμού'.

1^{ο} επίπεδο, προδομικό

Ο μαθητής δεν έχει κατανοήσει το πρόβλημα, ούτε γνωρίζει τη σύνταξη των εντολών. Εναλλακτικά, δεν απαντά. Ουσιαστικά δεν έχει δομήσει ακόμα επιστημονικά αποδεκτή γνώση σχετική με το ζητούμενο θέμα.

2^{ο} επίπεδο, επιμέρους κατανόησης

Ο μαθητής έχει αντιληφθεί εν μέρει το πρόβλημα και έχει αποδώσει με ορθό τρόπο, σύμφωνο με τη γραμματική και το συντακτικό της χρησιμοποιούμενης γλώσσας, απλές εντολές όπως εισόδου – εξόδου, δήλωσης και εκχώρησης, που απαιτούνται για τη διατύπωση της λύσης σε μορφή προγράμματος.

3^{ο} επίπεδο, προσεγγιστικής κατανόησης

Ο μαθητής έχει αντιληφθεί σε γενικές γραμμές το πρόβλημα και έχει αποδώσει μια αποδεκτή λύση με ορθό τρόπο (γραμματικό – συντακτικό), τόσο όσον αφορά στις απλές εντολές, όσο και στις λογικές δομές όπως επανάληψης και επιλογής, που απαιτεί το πρόγραμμα. Φαίνεται ότι γνωρίζει τον τρόπο λειτουργίας της κάθε εντολής καθώς και τη λειτουργία των μεταβλητών.

4^{ο} επίπεδο, συνδυαστικό

Ο μαθητής έχει κατανοήσει ακόμα και τα επιμέρους προβλήματα και έχει σχεδιάσει ικανοποιητικά δομημένη λύση του όλου προβλήματος. Χρησιμοποιεί εντολές και δομές, όπως στο προηγούμενο επίπεδο, αλλά επιπλέον επιλέγει την κατάλληλη δομή επανάληψης και τις κατάλληλες συναρτήσεις και διαδικασίες. Συνδυάζει με επιτυχία και τις δύο συνιστώσες του προγραμματισμού, την αλγοριθμική σκέψη και τη γνώση της γλώσσας.

5^{ο} επίπεδο, εκτεταμένης θεώρησης

Ο μαθητής – προγραμματιστής έχει επιλέξει και υλοποιήσει το βέλτιστο αλγόριθμο για κάθε περίπτωση και γνωρίζει τη φιλοσοφία της συγκεκριμένης γλώσσας προγραμματισμού που χρησιμοποιεί, πιθανά με δική του επιλογή.

Το πέμπτο επίπεδο αφορά στην υψηλότερη κατηγορία γνώσης, συμφωνώντας τόσο με την εκπαιδευτική έρευνα στην πληροφορική, που την ονομάζει στρατηγική γνώση, όσο και με τις γνωστικές θεωρίες στις οποίες αναφέρεται ως υπό συνθήκη γνώση. Σε αυτό το επίπεδο ο μαθητής θεωρείται ότι έχει αναπτύξει μεταγνωστικές δεξιότητες.

Σημειώνεται ότι μεταξύ δύο επιπέδων μπορούν να υπάρξουν μεταβατικά στάδια, στα οποία έχει εξελιχθεί η γνώση σε μια από τις δύο συνιστώσες του κάθε επιπέδου, είτε η δηλωτική και διαδικαστική γνώση της γλώσσας, είτε η οργάνωση και απόδοση του αλγόριθμου. Στις περιπτώσεις αυτές εξετάζεται η προσέγγιση της γνώσης στο αμέσως ανώτερο επίπεδο και αξιολογείται ανάλογα. Για παράδειγμα αν ο μαθητής έχει αντιληφθεί σε γενικές γραμμές το πρόβλημα και έχει αποδώσει μια αποδεκτή λύση με ορθό τρόπο, όμως μόνο όσον αφορά στις απλές εντολές, ενώ στις λογικές δομές όπως επανάληψης και επιλογής που απαιτεί το πρόγραμμα παρουσιάζει αδυναμία σύνταξης, κατατάσσεται στο ενδιάμεσο 2ο προς 3ο επίπεδο ΗΑΡ.

Ως παράδειγμα χρήσης του μοντέλου της ιεραρχικής αξιολόγησης προγραμματισμού αναφέρεται το πρόβλημα καταμέτρησης των άριστων (έστω βαθμός μεγαλύτερος και ίσος του 18), καλών (έστω βαθμός 15 ως 18) και μέτριων (έστω βαθμός απο10 ως 15) μαθητών ενός τμήματος που αποτελείται από 25 μαθητές. Δίνεται ο μέσος όρος του βαθμού κάθε μαθητή, ελέγχεται και ο μαθητής καταμετράται σε μια από τις τρεις κατηγορίες. Δειγματικές απαντήσεις ανά επίπεδο παρουσιάζονται στον Πίνακα 2.

Πίνακας 2: *Παράδειγμα χρήσης της ιεραρχικής αξιολόγησης προγραμματισμού*

1ο επίπεδο, προδομικό

```
ΠΡΟΓΡΑΜΜΑΒαθμοί μαθητών
Μεταβλητές
ΑΡΧΗ



ΤΕΛΟΣ
```

2ο επίπεδο, επιμέρους κατανόησης

```
ΠΡΟΓΡΑΜΜΑ Βαθμοίμαθητών
Μεταβλητές
ΑΚΕΡΑΙΕΣ: Β, άριστοι, καλοί, μέτριοι
ΑΡΧΗ
ΓΙΑ i από 1 ΜΕΧΡΙ 25
ΓΡΑΨΕ 'Δώσε βαθμό:'
ΔΙΑΒΑΣΕ Β
ΑΝ Β>18 ΤΟΤΕ
ΑΝ Β>10 ΤΟΤΕ
```

3ο επίπεδο, προσεγγιστικής κατανόησης

```
ΠΡΟΓΡΑΜΜΑ Βαθμοίμαθητών
ΜΕΤΑΒΛΗΤΕΣ
  ΠΡΑΓΜΑΤΙΚΕΣ: Β
  ΑΚΕΡΑΙΕΣ: i, α, μ, κ
ΑΡΧΗ
α<-0
μ<-0
κ<-0
ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 25
ΓΡΑΨΕ 'Δώσε', i,
'βαθμό:'
ΔΙΑΒΑΣΕ Β
```

4ο επίπεδο, συνδυαστικό

```
ΠΡΟΓΡΑΜΜΑ Βαθμοίμαθητών
ΜΕΤΑΒΛΗΤΕΣ
  ΠΡΑΓΜΑΤΙΚΕΣ: Β
  ΑΚΕΡΑΙΕΣ: i, α, κ, μ
ΑΡΧΗ
  α <- 0
  κ <- 0
  μ <- 0
  ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 25
   ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
     ΓΡΑΨΕ 'Δώσε ', i, 'ο βαθμό:'
     ΔΙΑΒΑΣΕ Β
```

```
AN B>18 ΤΟΤΕ
α<-α+1
ΑΛΛΙΩΣ_ΑΝ B>=15 ΤΟΤΕ
κ<-κ+1
ΑΛΛΙΩΣ_ΑΝ B>=10 ΤΟΤΕ
μ<-μ+1
ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΓΡΑΨΕ α, μ, β
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

```
ΜΕΧΡΙΣ_ΟΤΟΥ  Β  >= 10 ΚΑΙ Β <= 20
AN B < 15 ΤΟΤΕ
  μ <- μ + 1
ΑΛΛΙΩΣ_ΑΝ Β < 18 ΤΟΤΕ
  κ <- κ + 1
ΑΛΛΙΩΣ
  α <- α + 1
ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΓΡΑΨΕ 'ΑΡΙΣΤΟΙ', α
ΓΡΑΨΕ 'ΚΑΛΟΙ', κ
ΓΡΑΨΕ 'ΜΕΤΡΙΟΙ', μ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ Βαθμοί Μαθητών
```

5ο επίπεδο, εκτεταμένης θεώρησης

Δεν παρατίθεται, αφού το πρόβλημα είναι απλό και μπορεί να αποτελεί μια διαδικασία ενός σύνθετου προγράμματος, στο οποίο θα ήταν δυνατό να αναδειχτεί καλύτερα η οργάνωση πέμπτου επιπέδου. Παρατηρείται όμως ότι στο πρόγραμμα του 4ου επιπέδου αν στην είσοδο δεδομένων πληκτρολογηθεί χαρακτήρας αντί αριθμού, η εκτέλεση σταματά. Το πρόγραμμα εκτεταμένης θεώρησης θα προέβλεπε κάθε παρόμοια περίπτωση. Επιπλέον, ο μαθητής θα μπορούσε να επιλέξει τη γλώσσα προγραμματισμού που εξυπηρετεί περισσότερο το συγκεκριμένο πρόβλημα.

Το προτεινόμενο μοντέλο αναφέρεται προς το παρόν σε διαδικαστικές – αλγοριθμικές γλώσσες. Μελετάται η προσαρμογή του για άλλου τύπου προγραμματισμό.

Βιβλιογραφία

Bayman, P., & Mayer, R. (1988). Using conceptual models to teach BASIC computer programming. *Journal of Educational Psychology*, 80(3), 291-298.

Biggs, J. B., & Collis, K. F. (1982). *Evaluating the quality of learning. The SOLO taxonomy*. NY: Academic Press.

de Raadt, M. (2007). A Review of Australasian Investigations into Problem Solving and the Novice Programmer. *Computer Science Education*, 17(3), 201 – 213.

Deek, F. P., Turoff, M., & McHugh, J. A. (1999). A Common Model for Problem Solving and Program Development. *IEEE Transactions on Education*, 42(4), 331-336.

George, C. E. (2000). Experiences with novices: The importance of graphical representations in supporting mental models. In A. F. Blackwell & E. Bilotta (Eds), 12th Workshop of the Psychology of Programming Interest Group (pp. 33-44). Cozenza Italy.

Green, T., Hoc, J. M., Samurcay, R., & Gilmore, D. (1990). *Psychology of Programming*. London: Academic Press.

Jonassen, D. H., Beissner, K., & Yacci, M. A. (1993). *Structural knowledge: Techniques for representing, conveying, and acquiring structural knowledge*. NJ: LEA.

Linn, M. C. (1985). The cognitive consequences of programming instruction in classrooms. *Educational Researcher*, 14(5), 14-16, 25-29.

Lister, R., Simon, B., Thompson, E., Whalley, J. L., & Prasad, C. (2006). Not seeing the forest for the trees: novice programmers and the SOLO taxonomy. *Proceedings of the 11th annual SIGCSE conference on Innovation and Technology in Computer Science Education* (pp. 118-122). Bologna, Italy: ACM Press.

McGill, T. J., & Volet, S. E. (1997). A Conceptual Framework for Analyzing Students' Knowledge of Programming. *Journal of Research on Computing in Education*, 29, 276-297.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, 13(2), 137-172.

Thompson, E., Whalley, J. L., Lister, R., & Simon, B. (2006). Code Classification as Learning and Assessment Exercise for Novice Programmers. *Proceedings of the 19th Annual Conference of the National Advisory Committee on Computing Qualifications (NACCQ)* (pp. 291-298). Wellington, New Zealand: NACCQ.

Van Gorp, M. J., & Grissom, S. (2001). An Empirical Evaluation of Using Constructive Classroom Activities to Teach Introductory Programming. *Computer Science Education*, 11(3), 247-260.

Κόμης, Β. (2005). *Εισαγωγή στη διδακτική της πληροφορικής*. Αθήνα: Κλειδάριθμος

Κόμης, Β. (2007). Προσωπική επικοινωνία

Κουτσάκας, Φ., & Roberts, R. (2005). Μελέτη του αντίκτυπου της χρήσης του περιβάλλοντος δυναμικής προσομοίωσης εκτέλεσης κώδικα DYNALAB στη μαθησιακή διαδικασία του προγραμματισμού των Η/Υ μέσω της ποιοτικής αξιολόγησης των μαθησιακών αποτελεσμάτων μαθητών. Στο Α. Γιαλαμά, Ν. Τζιμόπουλος, Α. Χλωρίδου (επ.) *Αξιοποίηση των Τεχνολογιών της Πληροφορίας και της Επικοινωνίας στη Διδακτική Πράξη, Πρακτικά 3ου Πανελλήνιου Συνεδρίου των Εκπαιδευτικών για τις ΤΠΕ*, (σελ. 50-60). Σύρος: Εκδόσεις Ν. Τεχνολογιών

Μπέλλου, Ι. (2003). *Εικονικές πραγματικότητες στη Γεωγραφική Εκπαίδευση: Σχεδιασμός, ανάπτυξη, εφαρμογή και αξιολόγηση ενός διδακτικού πακέτου για τη διδασκαλία και μάθηση γεωγραφικών εννοιών*. Πανεπιστήμιο Θεσσαλίας: διδακτορική διατριβή.

Τζιμογιάννης, Α. (2003). Η διδασκαλία του προγραμματισμού στο ενιαίο λύκειο: προς ένα ολοκληρωμένο πλαίσιο με στόχο την ανάπτυξη δεξιοτήτων επίλυσης προβλημάτων. Στο *Πρακτικά 2ου Πανελλήνιου Συνεδρίου των Εκπαιδευτικών για τις ΤΠΕ: 'Αξιοποίηση των Τεχνολογιών της Πληροφορίας και των Επικοινωνιών στη Διδακτική Πράξη'*, Σύρος, σελ. Α/706-720.