ePublishing

Συνέδρια της Ελληνικής Επιστημονικής Ένωσης Τεχνολογιών Πληροφορίας & Επικοινωνιών στην Εκπαίδευση

Τόμ. 1 (2002)

3ο Συνέδριο ΕΤΠΕ «Οι ΤΠΕ στην Εκπαίδευση»

Εκπαιδευτικά Προγραμματιστικά Περιβάλλοντα βασισμένα στους Εκδότες Σύνταξης: Ανάλυση Εργαλείων και Επισκόπηση Αποτελεσμάτων Αξιολόγησης

Στέλιος Ξυνόγαλος, Μάγια Σατρατζέμη

Βιβλιογραφική αναφορά:

Ξυνόγαλος Σ., & Σατρατζέμη Μ. (2026). Εκπαιδευτικά Προγραμματιστικά Περιβάλλοντα βασισμένα στους Εκδότες Σύνταξης: Ανάλυση Εργαλείων και Επισκόπηση Αποτελεσμάτων Αξιολόγησης. *Συνέδρια της Ελληνικής Επιστημονικής Ένωσης Τεχνολογιών Πληροφορίας & Επικοινωνιών στην Εκπαίδευση*, *1*, 087–096. ανακτήθηκε από https://eproceedings.epublishing.ekt.gr/index.php/cetpe/article/view/8873

https://epublishing.ekt.gr | **e-Εκδότης:** EKT | **Πρόσβαση:** 27/02/2026 05:40:24

Εκπαιδευτικά Προγραμματιστικά Περιβάλλοντα βασισμένα στους Εκδότες Σύνταξης: Ανάλυση Εργαλείων και Επισκόπηση Αποτελεσμάτων Αξιολόγησης

Ξυνόγαλος Στέλιος
Υποψήφιος Διδάκτωρ, Τμήμα Εφαρμοσμένης Πληροφορικής, Πανεπιστήμιο Μακεδονίας
Αναπληρωτής Καθηγητής Πληροφορικής, Β'/θμια Εκπαίδευση
Εγνατία 156, Τ.Θ. 1591, 54006, Θεσσαλονίκη, Ελλάδα
stelios@uom.gr

Σατρατζέμη Μάγια
Επίκουρος Καθηγήτρια
Τμήμα Εφαρμοσμένης Πληροφορικής, Πανεπιστήμιο Μακεδονίας
Εγνατία 156, Τ.Θ. 1591, 54006, Θεσσαλονίκη, Ελλάδα
maya@uom.gr

ΠΕΡΙΛΗΨΗ

Ο στόχος των εκπαιδευτικών προγραμματιστικών περιβαλλόντων που ενσωματώνουν εκδότες σύνταξης είναι η στήριξη της διαδικασίας της διδασκαλίας και εκμάθησης του προγραμματισμού. Η μέχρι τώρα έρευνα έχει δείξει ότι οι εκδότες σύνταξης αντιμετωπίζουν αποτελεσματικά το πρόβλημα της επικέντρωσης στις συντακτικές λεπτομέρειες μιας γλώσσας προγραμματισμού. Παρ' όλα αυτά έχει διαπιστωθεί ότι προκαλούν και κάποιες δυσκολίες στους αρχάριους χρήστες. Στην παρούσα εργασία παρουσιάζονται τα χαρακτηριστικά που διαφοροποιούν τα εκπαιδευτικά προγραμματιστικά περιβάλλοντα με εκδότες σύνταξης και τα παιδαγωγικά οφέλη που παρέχουν. Επιπλέον, αναλύονται προγραμματιστικά περιβάλλοντα που βασίζονται σε εκδότες δομής και εικονικούς-γραφικούς εκδότες χρησιμοποιώντας τα παραπάνω χαρακτηριστικά. Τέλος, συνοψίζονται τα αποτελέσματα των εμπειρικών μελετών που έχουν διεξαχθεί με στόχο την αξιολόγηση της μαθησιακής διαδικασίας όταν χρησιμοποιούνται προγραμματιστικά περιβάλλοντα βασισμένα σε εκδότες σύνταξης και γίνονται προτάσεις για τη δημιουργία πιο αποτελεσματικών εκδοτών σύνταξης.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: *εκπαιδευτικά προγραμματιστικά περιβάλλοντα (educational programming environments), εκδότες σύνταξης (syntax-directed editors), εκδότες δομής (structure editors), εικονικές γλώσσες προγραμματισμού (iconic programming languages), αξιολόγηση (evaluation).*

ΕΙΣΑΓΩΓΗ

Τις τελευταίες δεκαετίες έχουν γίνει εκτεταμένες έρευνες με αντικείμενο τις δυσκολίες που αντιμετωπίζουν οι σπουδαστές κατά την εισαγωγή τους στον προγραμματισμό. Η σχετική έρευνα κατέδειξε ότι το μεγάλο ρεπερτόριο εντολών και η πολυπλοκότητα των γλωσσών προγραμματισμού, η αναγκαστική επικέντρωση των σπουδαστών στις συντακτικές λεπτομέρειες της γλώσσας προγραμματισμού και η πληθώρα των δυσνόητων μηνυμάτων λάθους οδηγούν στην απογοήτευση των σπουδαστών και την «αποτυχία» τους στα μαθήματα προγραμματισμού. Το ερώτημα που τίθεται όμως είναι: Ο στόχος των μαθημάτων εισαγωγής στον προγραμματισμό είναι

η διδασκαλία μιας γλώσσας προγραμματισμού ή η χρήση της για την κατανόηση των αρχών του προγραμματισμού και την υλοποίηση αλγορίθμων, την απόκτηση επομένως ικανοτήτων επίλυσης προβλημάτων;

Αρκετοί ερευνητές αναζήτησαν λύσεις προκειμένου να αντιμετωπίσουν το πρόβλημα της επικέντρωσης των αρχάριων προγραμματιστών στις συντακτικές λεπτομέρειες μια γλώσσας προγραμματισμού. Οι πιο αξιόλογες λύσεις που προτάθηκαν είναι η χρήση προγραμματιστικών μικρόκοσμων (Brusilovsky, 1997) και εκδοτών σύνταξης (Goldenson, 1989; MilIer, 1994). Στην παρούσα εργασία γίνεται μια προσπάθεια παρουσίασης των βασικών χαρακτηριστικών των εκδοτών σύνταξης και των πλεονεκτημάτων που έχουν τα εκπαιδευτικά περιβάλλοντα προγραμματισμού που ενσωματώνουν τέτοιους εκδότες, εκδότες δηλαδή στους οποίους η ανάπτυξη των προγραμμάτων γίνεται με τη χρήση προτύπων και την επιλογή από μενού συντακτικά σωστών εντολών για κάθε ημιτελές τμήμα του προγράμματος. Επίσης, γίνεται μια σύντομη ανάλυση εκπαιδευτικών προγραμματιστικών περιβαλλόντων με εκδότες σύνταξης βάση των προαναφερθέντων χαρακτηριστικών και μια σύνοψη των αποτελεσμάτων αξιολόγησης τέτοιων περιβαλλόντων. Τέλος, παρουσιάζονται τα συμπεράσματα από τη μέχρι τώρα χρήση και αξιολόγηση των εκπαιδευτικών προγραμματιστικών περιβαλλόντων με εκδότες σύνταξης.

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΚΑΙ ΠΑΙΔΑΓΩΓΙΚΑ ΟΦΕΛΗ

Στην ενότητα αυτή περιγράφονται σύντομα τα βασικά χαρακτηριστικά των εκπαιδευτικών προγραμματιστικών περιβαλλόντων που βασίζονται σε εκδότες σύνταξης και τα παιδαγωγικά οφέλη που παρέχουν. Τα χαρακτηριστικά που αναφέρονται στη συνέχεια αποτελούν υποσύνολο των χαρακτηριστικών που περιγράφουν οι Khwaja και Urban για τη σχεδίαση και τη χρήση βασισμένων σε εκδότες σύνταξης περιβαλλόντων (Khwaja & Urban, 1993). Συγκεκριμένα, παρουσιάζονται εκείνα τα χαρακτηριστικά που κρίνονται σημαντικά για τα εκπαιδευτικά προγραμματιστικά περιβάλλοντα:

1. *Εξωτερική αναπαράσταση* (external representation): σε έναν εκδότη σύνταξης μπορεί να χρησιμοποιούνται πρότυπα που βασίζονται είτε σε κείμενο είτε σε κάποιου είδους εικονικές-συμβολικές αναπαραστάσεις. Με βάση αυτό το χαρακτηριστικό ένας εκδότης σύνταξης μπορεί να χαρακτηριστεί ως *εκδότης δομής* ή *εικονικός-γραφικός εκδότης*.
2. *Παραγόμενο έγγραφο* (target document): στην περίπτωση των προγραμματιστικών περιβαλλόντων το έγγραφο που παράγεται είναι ο κώδικας του προγράμματος σε μια συγκεκριμένη γλώσσα προγραμματισμού ή κάποιου είδους εικονική-συμβολική αναπαράσταση, για παράδειγμα ένα λογικό διάγραμμα.
3. *Τύποι λαθών που εντοπίζονται κατά την ανάπτυξη* (types of errors): ένας εκδότης σύνταξης εντοπίζει πάντα *συντακτικά λάθη*, ενώ κάποιοι εκδότες έχουν και τη δυνατότητα εντοπισμού *στατικών σημασιολογικών λαθών*.
4. *Διαχείριση λαθών* (error handling): ένας εκδότης σύνταξης μπορεί να διαχειρίζεται τα λάθη που εντοπίζονται με έναν από τους παρακάτω τρόπους ή με ένα συνδυασμό αυτών:
    - απαιτεί τη διόρθωση του λάθους τη στιγμή που εντοπίζεται,
    - αναφέρει άμεσα το λάθος, αλλά δεν απαιτεί την άμεση διόρθωσή του,
    - αυτόματη διόρθωση του λάθους.
5. *Τμηματική μεταγλώττιση – δημιουργία συντακτικού δένδρου* (Incremental system): όταν ένας εκδότης σύνταξης ενσωματώνει αυτή τη δυνατότητα δεν χρειάζεται να γίνεται επαναδημιουργία του συντακτικού δένδρου ή μεταγλώττιση κάθε φορά που ο χρήστης θέλει να εκτελέσει ένα πρόγραμμα.
6. *Επίπεδο εκτέλεσης* (execution level): ένας εκδότης σύνταξης μπορεί να επιτρέπει την τμηματική ή μόνο την πλήρη εκτέλεση ενός προγράμματος. Βέβαια υπάρχει περίπτωση ένας

εκδότης σύνταξης να μην επιτρέπει καθόλου την εκτέλεση, οπότε χρησιμοποιείται μόνο για την ανάπτυξη συντακτικά σωστών προγραμμάτων.

7. *Διαθέσιμα εργαλεία*: όπως αποσφαλματωτές, ανιχνευτές εκτέλεσης των προγραμμάτων και παρουσίαση του προγράμματος με διάφορες απόψεις.

Ο βασικός στόχος των εκπαιδευτικών προγραμματιστικών περιβαλλόντων με εκδότες σύνταξης είναι η *αντιμετώπιση του προβλήματος της επικέντρωσης της προσοχής των σπουδαστών στις συντακτικές λεπτομέρειες* της χρησιμοποιούμενης γλώσσας προγραμματισμού. Άλλα παιδαγωγικά οφέλη, κάποια από τα οποία απορρέουν από το προηγούμενο είναι:
Αντιμετώπιση του προβλήματος της επικέντρωσης σε χαμηλού επιπέδου λεπτομέρειες γενικότερα: εκμάθηση κανόνων για τη μορφοποίηση ενός προγράμματος έτσι ώστε να είναι ευανάγνωστο, εξοικείωση με δύσχρηστους εκδότες κειμένου και σωστή διαχείριση του κύκλου ζωής λογισμικού (Goldenson, 1989).
Επικέντρωση της προσοχής στη διαδικασία επίλυσης προβλημάτων και την απόκτηση ανάλογων ικανοτήτων (Crews & Ziegler 1999; Calloni & Bagert, 1994).
Επικέντρωση της προσοχής σε θέματα δομής και σχεδίασης: εξαλείφοντας το πρόβλημα της επικέντρωσης σε λεπτομέρειες χαμηλού επιπέδου δίνεται η δυνατότητα ανάθεσης πιο απαιτητικών, σε θέματα δομής και σχεδίασης, εργασιών στους σπουδαστές (Goldenson, 1989).

Βέβαια, έχει διαπιστωθεί ότι οι εκδότες σύνταξης αρκετές φορές επιβάλλουν κάποιες δυσκολίες ευχρηστίας στους αρχάριους χρήστες. Οι δυσκολίες αυτές ποικίλουν και εξαρτώνται από την υλοποίηση του εκδότη σύνταξης. Ανεξάρτητα από την υλοποίηση του εκδότη σύνταξης όμως, τα περισσότερα προβλήματα εμφανίζονται κατά τη διόρθωση παρά κατά την ανάπτυξη των προγραμμάτων. Όπως είναι γνωστό ο πιο εύκολος και προφανής τρόπος πραγματοποίησης αλλαγών σε ένα πρόγραμμα συνήθως απαιτεί την προσωρινή παραβίαση των συντακτικών κανόνων της γλώσσας προγραμματισμού. Οι εκδότες σύνταξης όμως επιτρέπουν την ανάπτυξη αυστηρά σωστού - συντακτικά - πηγαίου κώδικα με αποτέλεσμα η πραγματοποίηση αλλαγών να είναι ιδιαίτερα δύσκολη. Δυσκολίες προκαλεί αρκετές φορές και η εισαγωγή εντολών μεταξύ άλλων.

ΑΝΑΛΥΣΗ ΕΡΓΑΛΕΙΩΝ

Στον Πίνακα 1 αναλύονται τέσσερα εκπαιδευτικά προγραμματιστικά περιβάλλοντα που βασίζονται σε εικονικούς-γραφικούς εκδότες και στον Πίνακα 2 αναλύονται τέσσερα εκπαιδευτικά προγραμματιστικά περιβάλλοντα που βασίζονται σε εκδότες δομής. Η ανάλυση γίνεται βάση των χαρακτηριστικών που αναφέρθηκαν στην προηγούμενη ενότητα, ενώ τα στοιχεία προέρχονται από την αναφερόμενη βιβλιογραφία.

Όλα τα προγραμματιστικά περιβάλλοντα που αναλύονται στον Πίνακα 1 χρησιμοποιούν ένα συντακτικά καθοδηγούμενο γραφικό ενδιάμεσο που δεν επιτρέπει στο σπουδαστή να κάνει λάθη. Τα τέσσερα προγραμματιστικά περιβάλλοντα που αναλύονται αν και βασίζονται σε εικονικούς εκδότες διαφέρουν σε αρκετά σημεία. Κατ' αρχήν τα περιβάλλοντα KidSim (Smith, 1994) και Youngster (Studer et al., 1995) απευθύνονται σε μαθητές, ενώ τα BACII/BACCII++ (Calloni & Bagert 1994; Calloni & Bagert 1997) και FLINT (Crews & Ziegler 1998; Crews & Ziegler 1999) σε φοιτητές. Συγκεκριμένα, το KidSim απευθύνεται σε παιδιά ηλικίας 5 έως 18 ετών και το Youngster σε μαθητές της Α'θμιας εκπαίδευσης. Αντίθετα τα BACII/BACCII++ αναπτύχθηκαν για να χρησιμοποιηθούν για τη διδασκαλία των μαθημάτων CS1 και CS2, και το FLINT για να χρησιμοποιηθεί στα πλαίσια του μαθήματος CS1. Βέβαια αυτό δε σημαίνει ότι τα περιβάλλοντα αυτά δεν μπορούν να χρησιμοποιηθούν και στη Β'θμια εκπαίδευση, στα προγράμματα σπουδών της οποίας υποδεικνύεται σαφέστατα η διδασκαλία της αλγοριθμικής και όχι μιας συγκεκριμένης γλώσσας προγραμματισμού. Από τα τέσσερα περιβάλλοντα που αναλύονται στον Πίνακα 1, τα

BACII/BACCII++ είναι τα μόνα που δεν παρέχουν τη δυνατότητα εκτέλεσης των προγραμμάτων, στην ουσία αλγορίθμων, που αναπτύσσονται. Ωστόσο αυτές οι εικονικές γλώσσες προγραμματισμού, μετά το τέλος της ανάπτυξης ενός αλγορίθμου παρέχουν τη δυνατότητα αυτόματης παραγωγής συντακτικά σωστού πηγαίου κώδικα - σε μορφή ASCII - σε Pascal, C, Fortran, Basic και C++. Τέλος, το KidSim σε αντίθεση με τα άλλα περιβάλλοντα χρησιμοποιείται για τη δημιουργία *συμβολικών προσομοιώσεων* (symbolic simulations), ελεγχόμενων δηλαδή από τον υπολογιστή μικρόκοσμων που αποτελούνται από αντικείμενα ή αλλιώς πράκτορες (agents) που κινούνται μέσα στο μικρόκοσμο αλληλεπιδρώντας σύμφωνα με τους κανόνες που έχει καθορίσει ο δημιουργός του. Ο καθορισμός της αρχικής κατάστασης του μικρόκοσμου και των κανόνων πραγματοποιείται με *προγραμματισμό δι' επιδείξεως* (programming by demonstration) και *γραφικούς κανόνες* (graphical rewrite rules).

	BACCII/ BACCII++ (Calloni & Bagert, 1994-1997)	FLINT (Crews & Ziegler, 1998 & 1999)	KidSim (Smith, 1994)	Youngster (Studer et al., 1995)
Εξωτερική αναπαράσταση	Γραφική: διάγραμμα ροής	Γραφική: διάγραμμα ροής	Γραφική: γραφικοί κανόνες	Γραφική
Παραγόμενο έγγραφο	Διάγραμμα ροής	Ανεξάρτητο γλώσσας - δημιουργία διαγραμμάτων ροής	Χρησιμοποιούνται γραφικοί κανόνες & προγραμματισμός δι' επιδείξεως	Οι εντολές έχουν τη μορφή κομματιών puzzle
Τύποι λαθών που εντοπίζονται	Δεν επιτρέπει λάθη	Δεν επιτρέπει λάθη	Δεν επιτρέπει λάθη	Δεν επιτρέπει λάθη
Δυνατότητα εκτέλεσης	Καμία	Τμηματική	Ο χρήστης παρακολουθεί την εξέλιξη της κατάστασης ενός μικρόκοσμου.	Υπάρχει δυνατότητα εκτέλεσης, αλλά δεν προσδιορίζεται το επίπεδό της.
Διαθέσιμα εργαλεία	Παραγωγή συντακτικά σωστού πηγαίου κώδικα (σε μορφή ASCII) σε Pascal, C, Fortran, Basic, C++.	Ερμηνευτής – Αποσφαλματωτής διαγραμμάτων ροής	Προγραμματισμός δι' επιδείξεως. Συμβολική προσομοίωση μικρόκοσμων.	Ερμηνευτής

Πίνακας 1: Ανάλυση εκπαιδευτικών προγραμματιστικών περιβαλλόντων με εικονικούς-γραφικούς εκδότες.

Τα προγραμματιστικά περιβάλλοντα με εκδότες δομής που αναλύονται στον Πίνακα 2 δεν παρουσιάζουν τόσο μεγάλες διαφορές, με εξαίρεση το Cornell Program Synthesizer (Teitelbaum, 1981) που είχε περιορισμένες δυνατότητες. Συγκεκριμένα, τα προγράμματα που αναπτύσσονταν στο περιβάλλον αυτό δεν ξεπερνούσαν τις 24 γραμμές και δεν υπήρχε η δυνατότητα ανάπτυξης υποπρογραμμάτων. Ωστόσο, το περιβάλλον αυτό μαζί με κάποια άλλα που αναπτύχθηκαν εκείνη την περίοδο αποτέλεσε την αφετηρία για την ανάπτυξη της τεχνολογίας των εκδοτών δομής. Τα περιβάλλοντα Genies (Miller, 1994) που βασίστηκαν στα περιβάλλοντα που αναπτύχθηκαν στα πλαίσια του project Gnome είναι από τα καλύτερα προγραμματιστικά περιβάλλοντα που βασίζονται σε εκδότες δομής. Τα περιβάλλοντα Genies είναι τα μοναδικά προγραμματιστικά περιβάλλοντα με εκδότη δομής που παρέχουν τη δυνατότητα επεξεργασίας ενός προγράμματος και ως κείμενο. Βέβαια σ' αυτή την περίπτωση απαιτείται άμεση διόρθωση των λαθών που εντοπίζονται. Ένα άλλο, επίσης μοναδικό χαρακτηριστικό των περιβαλλόντων Genies είναι η παροχή πολλαπλών αναπαραστάσεων ενός προγράμματος και η *δυναμική οπτικοποίηση των*

δεδομένων (dynamic data visualization). Ωστόσο, τα περιβάλλοντα Genies αναπτύχθηκαν για πλατφόρμα Macintosh. Αντίθετα, το προγραμματιστικό περιβάλλον Alice Pascal (Templeton) τρέχει σε PC (περιβάλλον Dos). Επιπλέον, υπάρχει δυνατότητα απόκτησης του Alice Pascal και του εγχειριδίου χρήσης – εγχειριδίου της γλώσσας προγραμματισμού δωρεάν από το διαδίκτυο.

	Cornell Program Synthesizer (Teitelbaum,1981)	Gnome (Miller, 1994)	MacGnome Genies (Miller, 1994; Chandhok)	Alice Pascal (Templeton)
Εξωτερική αναπαράσταση	Κειμενική	Κειμενική	Κειμενική	Κειμενική
Παραγόμενο έγγραφο	PL/CS: υποσύνολο της PL/I	Pascal, Fortran, Lisp, Karel the Robot	Pascal, Karel the Robot	Pascal
Τύποι λαθών που εντοπίζονται	Συντακτικά, Στατικά σημασιολογικά	Συντακτικά, Στατικά σημασιολογικά	Συντακτικά	Συντακτικά, Σημασιολογικά
Διαχείριση λαθών	Αναφέρει άμεσα το λάθος, αλλά δεν απαιτεί την άμεση διόρθωσή του	Απαιτείται άμεση διόρθωση του λάθους	Απαιτείται άμεση διόρθωση του λάθους όταν το πρόγραμμα επεξεργάζεται ως κείμενο. Κάποια λάθη αναφέρονται όταν ο χρήστης ζητάει να εκτελεστεί το πρόγραμμα.	Αναφέρει άμεσα το λάθος, αλλά δεν απαιτεί την άμεση διόρθωσή του
Δυνατότητα εκτέλεσης	Τμηματική έως πλήρης	Ο εκτελέσιμος κώδικας παραγόταν μέσω του ερμηνευτή Berkeley Unix Pascal.	Τμηματική έως πλήρης	Τμηματική έως πλήρης
Διαθέσιμα εργαλεία	Ερμηνευτής, αποσφαλματωτής	Ιεραρχική παρουσίαση προγράμματος: μία διαδικασία κάθε φορά με τις επικεφαλίδες μόνο των εμφωλευμένων υποπρογραμμάτων	Δυνατότητα επεξεργασίας ενός προγράμματος και ως κείμενο. Πολλαπλές αναπαραστάσεις ενός προγράμματος: Κώδικας, Λίστα επικεφαλίδων υποπρογραμμάτων, Κώδικας μιας διαδικασίας ή συνάρτησης, Στοίβα ενεργών υποπρογραμμάτων, Δενδροειδής παρουσίαση. Ανίχνευση & αποσφαλμάτωση. Οπτικοποίηση δεδομένων.	Αποσφαλματωτής. Ερμηνευτής. Συμβατό με Turbo Pascal

Πίνακας 2: Ανάλυση εκπαιδευτικών προγραμματιστικών περιβαλλόντων με εκδότες δομής.

ΕΠΙΣΚΟΠΗΣΗ ΤΩΝ ΑΠΟΤΕΛΕΣΜΑΤΩΝ ΤΗΣ ΑΞΙΟΛΟΓΗΣΗΣ

ΕΜΠΕΙΡΙΚΕΣ ΜΕΛΕΤΕΣ

Αρκετές εμπειρικές μελέτες έχουν διεξαχθεί έχοντας ως αντικείμενο τη διερεύνηση της επίδρασης των προγραμματιστικών περιβαλλόντων με εκδότες σύνταξης και των εμπορικά διαθέσιμων προγραμματιστικών περιβαλλόντων στην επίδοση των σπουδαστών σε μαθήματα εισαγωγής στον προγραμματισμό. Τουλάχιστον τέσσερις μελέτες διεξήχθησαν στα πλαίσια υποχρεωτικών μαθημάτων σε Κολέγια και Πανεπιστήμια. Τρεις από αυτές διεξήχθησαν στα πλαίσια μαθημάτων «Εισαγωγής στον Προγραμματισμό» (CS1) τα οποία βασίζονταν στο κατηγορηματικό παράδειγμα προγραμματισμού, χρησιμοποιώντας την Pascal (σε δύο) (Calloni & Bagert, 1994; Goldenson, 1989) και την C++ (σε ένα) ως γλώσσα προγραμματισμού (Calloni & Bagert, 1997). Μια μελέτη διεξήχθηκε στα πλαίσια του μαθήματος «Δομές Δεδομένων» (CS2) το οποίο βασίζονταν στο αντικειμενοστραφές παράδειγμα προγραμματισμού, χρησιμοποιώντας την C++ ως γλώσσα προγραμματισμού (Calloni & Bagert, 1997). Τα στατιστικά σημαντικά αποτελέσματα και τα συμπεράσματα που εξήχθησαν από τις μελέτες συνοψίζονται ως εξής:

1. Οι σπουδαστές που χρησιμοποίησαν συντακτικά καθοδηγούμενα προγραμματιστικά περιβάλλοντα, όπως τα BACII & BACCII++ (Calloni & Bagert, 1994; Calloni & Bagert, 1997) και Genies (Goldenson, 1989), παρουσίασαν υψηλότερη επίδοση τόσο στις εργασίες όσο και στις τελικές εξετάσεις. Οι (Gondelson et al., 1990) επίσης αναφέρουν ότι τεστ που διεξήχθησαν σε αρκετά Πανεπιστήμια και σχολεία της Δευτεροβάθμιας Εκπαίδευσης στα πλαίσια υποχρεωτικών μαθημάτων, χρησιμοποιώντας τα περιβάλλοντα Genies, έδειξαν σημαντικές διαφορές στην επίδοση των σπουδαστών.
2. Η χρήση των εικονικών περιβαλλόντων BACCII και BACCII++ στα μαθήματα CS1 και CS2 στο Texas Tech University έδειξε ότι το περιβάλλον είχε μεγαλύτερη επίδραση στο μάθημα CS2, το οποίο βασίζονταν στο αντικειμενοστραφές παράδειγμα προγραμματισμού. Οι Calloni και Bagert συμπέραναν *"the iconic representations play more to the strengths of the object-oriented paradigm than to the imperative design issues…"*, σελ. 266, (Calloni & Bagert, 1997). Επιπλέον, οι βαθμοί των σπουδαστών παρουσίασαν μικρότερη απόκλιση από το μέσο βαθμό σε όλες τις κατηγορίες (προγράμματα, εργαστήρια, εξετάσεις), γεγονός που υποδηλώνει μια πιο ομοιόμορφη μάθηση.
3. Το πιο απρόσμενο αποτέλεσμα των μελετών με τα περιβάλλοντα BACCII και BACCII++ ήταν η μεγαλύτερη κατανόηση, από την ομάδα των φοιτητών που τα χρησιμοποίησαν, της σύνταξης της Pascal και της C++ αντίστοιχα σε σχέση με τους φοιτητές που χρησιμοποίησαν συμβατικά περιβάλλοντα (Calloni & Bagert 1994; Calloni & Bagert 1997). Οι Calloni και Bagert απέδωσαν το αποτέλεσμα αυτό στο γεγονός ότι οι σπουδαστές έβλεπαν συνεχώς συντακτικά σωστά προγράμματα.
4. Οι διδάσκοντες που χρησιμοποίησαν τα περιβάλλοντα Genies στο Carnegie Mellon University και σε άλλα εκπαιδευτικά ιδρύματα συμπέραναν από τις ερωτήσεις των σπουδαστών ότι οι τελευταίοι επικέντρωναν την προσοχή τους σε ουσιαστικά προβλήματα δομής παρά σε συντακτικές λεπτομέρειες (Goldenson, 1989).

Τα προγραμματιστικά περιβάλλοντα με εκδότες σύνταξης ή αλλιώς εκδότες γλωσσών (language-based editors) αρκετές φορές προσφέρουν *κειμενικές ή γραφικές αναπαραστάσεις της ιεραρχικής δομής* των μπλοκ ή τμημάτων (modules) ενός προγράμματος. Η αλληλεπίδραση με αυτές τις αναπαραστάσεις βασίζεται κατά κανόνα σε τεχνικές άμεσης διαχείρισης ενώ για την παρουσίασή τους χρησιμοποιούνται μενού που μπορεί να έχουν τη μορφή ενός *γραφήματος δομής/δένδρου (structure chart model)* ή μιας *λίστας* με τα ονόματα των μπλοκ/τμημάτων ενός προγράμματος στοιχημένα έτσι ώστε να αποδίδουν τη δομή του. Ένα σχετικό πείραμα (Toleman et al., 1992) έδειξε ότι:

1. Δεν υπάρχει διαφορά, στο χρόνο που απαιτείται για την κατανόηση μιας δεδομένης ερώτησης που αφορά την ιεραρχική δομή και την εύρεση του ζητούμενου στοιχείου στην δομή, μεταξύ των δυο υλοποιήσεων του μενού (μορφή δένδρου και λίστας).
2. Οι συμμετέχοντες προτίμησαν τη γραφική αναπαράσταση από τη βασισμένη σε κείμενο λίστα με αναλογία 2:1. Ωστόσο, η προτίμηση αυτή δεν επηρέασε την απόδοσή τους.

Οι υποστηρικτές των εκπαιδευτικών προγραμματιστικών περιβαλλόντων με εικονικούς εκδότες – τα οποία μπορεί να βασίζονται σε μια εικονική γλώσσα προγραμματισμού ή σε ένα περιβάλλον ανάπτυξης λογικών διαγραμμάτων ανεξαρτήτου γλώσσας – ισχυρίζονται ότι τα συγκεκριμένα περιβάλλοντα είναι πιο αποτελεσματικά για τους αρχάριους από τα αντίστοιχα περιβάλλοντα στα οποία η εκφορά των αλγορίθμων είναι κειμενική. Για να στηρίξουν τη θέση τους οι περισσότεροι ερευνητές (Calloni & Bagert, 1994; Crews & Ziegler, 1998) αναφέρονται στα αποτελέσματα των μελετών διακεκριμένων ερευνητών και κυρίως στα αποτελέσματα των πειραμάτων του Scanlan που είχαν ως αντικείμενο τη σύγκριση των *δομημένων διαγραμμάτων ροής (structured flowcharts)* και του ψευδοκώδικα ως βοηθήματα κατανόησης αλγορίθμων. Τα συμπεράσματα των πειραμάτων του (Scanlan, 1989) συνοψίζονται ως εξής:

1. Οι φοιτητές χρειάστηκαν λιγότερο χρόνο για την κατανόηση των δομημένων διαγραμμάτων ροής.
2. Οι φοιτητές έκαναν λιγότερα λάθη, είχαν μεγαλύτερη αυτοπεποίθηση, είδαν τους αλγόριθμους λιγότερες φορές και χρειάστηκαν λιγότερο χρόνο για να απαντήσουν στις ερωτήσεις όταν χρησιμοποιούσαν διαγράμματα ροής.
3. Η αποτελεσματικότητα των διαγραμμάτων ροής αυξάνεται όσο αυξάνεται η πολυπλοκότητα των αλγορίθμων.

Στα ίδια συμπεράσματα οδήγησε και ανάλογο πείραμα με αντικείμενο τη σύγκριση των δομημένων διαγραμμάτων ροής και προγραμμάτων σε Qbasic (Crews & Ziegler, 1998). Παρόλο που τα αποτελέσματα των πειραμάτων αυτών και ο ίδιος ο Scanlan ενθαρρύνουν τη χρήση γραφικών αναπαραστάσεων, πρέπει να επισημάνουμε ότι σε κανένα από τα πειράματα αυτά δεν εξετάστηκε η χρησιμότητα των δομημένων διαγραμμάτων ροής κατά το στάδιο της σχεδίασης αλγορίθμων. Αντίθετα, τα πειράματα που διεξήγαγαν οι (Shneiderman et al., 1977) με στόχο τον έλεγχο της χρησιμότητας των *λεπτομερών διαγραμμάτων ροής (detailed flowcharts)* ως εργαλείων για την ανάπτυξη (composition), κατανόηση (comprehension), αποσφαλμάτωση (debugging) και τροποποίηση (modification) προγραμμάτων έδειξαν ότι δεν υπάρχει σημαντικά στατιστική διαφορά στην απόδοση μεταξύ των σπουδαστών που χρησιμοποιούν διαγράμματα ροής και εκείνων που χρησιμοποιούν κώδικα για τις παραπάνω δραστηριότητες.

ΑΝΑΛΥΣΗ ΕΡΩΤΗΜΑΤΟΛΟΓΙΩΝ

Δεδομένα από ερωτηματολόγια συγκεντρώθηκαν στο Carnegie Mellon University και σε πέντε άλλες τοποθεσίες από μαθητές και καθηγητές που χρησιμοποίησαν τα περιβάλλοντα Genies. Η ανάλυση των ερωτηματολογίων έδειξε ότι (Goldenson, 1989):

1. Οι σπουδαστές που χρησιμοποιούν τα περιβάλλοντα Genies δεν παρουσιάζουν τις τυπικές δυσκολίες που αντιμετωπίζουν συνήθως οι αρχάριοι, όπως προβλήματα με τη σύνταξη της Pascal και τη χρήση υποπρογραμμάτων σε μεγάλα προγράμματα.
2. Ακόμα και όταν οι σπουδαστές έχουν τη δυνατότητα να επεξεργαστούν ένα πρόγραμμα ως κείμενο, όπως στα περιβάλλοντα Genies, προτιμούν να το επεξεργαστούν χρησιμοποιώντας έναν εκδότη δομής.
3. Οι σπουδαστές που χρησιμοποιούν τα περιβάλλοντα Genies είναι πιο ικανοποιημένοι με το περιβάλλον τους από άλλους σπουδαστές που ερωτήθηκαν, ενώ εκείνοι που έχουν χρησιμοποιήσει περισσότερα από ένα περιβάλλοντα προτιμούν τα περιβάλλοντα Genies.

ΑΝΑΛΥΣΗ ΚΑΤΑΓΕΓΡΑΜΜΕΝΩΝ ΕΝΕΡΓΕΙΩΝ ΤΩΝ ΣΠΟΥΔΑΣΤΩΝ

Η ανάλυση ενός τεράστιου όγκου δεδομένων που συγκεντρώθηκαν καταγράφοντας τις ενέργειες των σπουδαστών ενώ χρησιμοποιούσαν τα περιβάλλοντα Genies οδήγησε τους ερευνητές στα ακόλουθα συμπεράσματα (Miller et al., 1994):

1. Τα μενού δομής, όπως ήταν αναμενόμενο, χρησιμοποιούνται περισσότερο στην αρχή του εξαμήνου, αλλά ακόμα και έμπειροι χρήστες χρησιμοποιούν τα μενού συχνά.
2. Οι σπουδαστές δεν χρησιμοποίησαν πολλά από τα «δυνατά» χαρακτηριστικά των περιβαλλόντων Genies. Τα χαρακτηριστικά που χρησιμοποίησαν ήταν κατά κύριο λόγο εκείνα που χρησιμοποιήθηκαν και στην τάξη. Το γεγονός αυτό αποδόθηκε εν μέρει στην έλλειψη προσοχής σε θέματα διεπιφάνειας χρήστη.
3. Το σύστημα βοήθειας χρησιμοποιήθηκε ελάχιστα.
4. Οι σπουδαστές που υιοθέτησαν ένα τμηματικό τρόπο προγραμματισμού είχαν καλύτερη επίδοση.
5. Η δυναμική εκτέλεση των προγραμμάτων ωφελεί ακόμα και τους πιο «ταλαντούχους» σπουδαστές.
6. Από την ανάλυση των καταγεγραμμένων ενεργειών των σπουδαστών έγινε φανερό ότι η παρατήρηση του τρόπου αλληλεπίδρασης χρήστη - συστήματος αποκαλύπτει προβλήματα που δεν αναμενόταν από τους ερευνητές.

ΕΜΠΕΙΡΙΕΣ ΔΙΔΑΣΚΟΝΤΩΝ

Οι εμπειρίες όσων έχουν χρησιμοποιήσει προγραμματιστικά περιβάλλοντα με εκδότες σύνταξης, τις οποίες θεωρούμε εξίσου σημαντικές με τα αποτελέσματα της αξιολόγησης, είναι ιδιαίτερα ενθαρρυντικές:

1. Οι διδάσκοντες που χρησιμοποίησαν τα περιβάλλοντα Genies δεν ήθελαν να μεταβούν σε άλλα περιβάλλοντα (Miller et al., 1994).
2. Οι Goldenson et al. αναφερόμενοι στους εκδότες δομής δηλώνουν: *"At worst their environmental semantics can be as difficult to learn as the language syntax they are meant to avoid."*, σελ.267, (Goldenson et al., 1990). Στη χειρότερη δηλαδή περίπτωση, η εξοικείωση με ένα εκδότη δομής μπορεί να είναι τόσο δύσκολη όσο η εκμάθηση της σύνταξης της γλώσσας προγραμματισμού που υποστηρίζει.
3. Οι εκδότες δομής λύνουν όχι μόνο το πρόβλημα της επικέντρωσης σε χαμηλού επιπέδου συντακτικές λεπτομέρειες, αλλά και σε λεπτομέρειες χρήσης του εκδότη και του λειτουργικού συστήματος (Goldenson, 1989). Ο ισχυρισμός επομένως ότι ένας εκδότης σύνταξης επιβάλλει δυσκολίες ευχρηστίας δεν επαληθεύεται, τουλάχιστον για τα περιβάλλοντα Genies.

ΣΥΜΠΕΡΑΣΜΑΤΑ

Από τις εμπειρικές μελέτες, την ανάλυση ερωτηματολογίων και καταγεγραμμένων ενεργειών των σπουδαστών και τις εμπειρίες διδασκόντων και σπουδαστών σε διάφορα εκπαιδευτικά ιδρύματα προκύπτουν τα εξής:

- Η εξάλειψη του προβλήματος της επικέντρωσης στις συντακτικές λεπτομέρειες μιας οποιασδήποτε γλώσσας προγραμματισμού δίνει τη δυνατότητα επικέντρωσης σε πιο σημαντικά θέματα, όπως σε θέματα σχεδίασης, δομής και ανάπτυξης ικανοτήτων επίλυσης προβλημάτων.
- Η επίδοση των σπουδαστών που χρησιμοποιούν προγραμματιστικά περιβάλλοντα με εκδότες σύνταξης είναι καλύτερη από εκείνη των σπουδαστών που χρησιμοποιούν συμβατικά περιβάλλοντα, ακόμα και όσον αφορά τη σύνταξη της χρησιμοποιούμενης γλώσσας προγραμματισμού.

- Τόσο οι διδάσκοντες όσο και οι σπουδαστές που έχουν χρησιμοποιήσει προγραμματιστικά περιβάλλοντα με εκδότες σύνταξης τα προτιμούν από τα συμβατικά περιβάλλοντα.
- Τα προβλήματα ευχρηστίας που παρατηρήθηκαν σε κάποιους εκδότες σύνταξης μπορούν να αποφευχθούν με προσεκτική σχεδίαση των εκδοτών. Στη χειρότερη περίπτωση η εξοικείωση με τον εκδότη μπορεί να είναι τόσο επίπονη όσο η εξοικείωση με τη σύνταξη της γλώσσας προγραμματισμού που υποστηρίζει. Η καλύτερη επιλογή φαίνεται ότι είναι η χρήση ενός εκδότη που συνδυάζει την ευκολία χρήσης ενός «καλού» εκδότη κειμένου και της «ευφυΐας» ενός εκδότη σύνταξης.

Όσον αφορά την αποτελεσματικότητα των εκδοτών δομής σε σχέση με τους εικονικούς εκδότες δεν υπάρχουν διαθέσιμα στοιχεία. Στην ουσία το θέμα έγκειται στο κατά πόσο τα διαγράμματα ροής υποστηρίζουν τους αρχάριους στην ανάπτυξη, κατανόηση, αποσφαλμάτωση και τροποποίηση προγραμμάτων. Όπως όμως αναφέρθηκε τα αποτελέσματα των σχετικών μελετών, με χαρακτηριστικότερα παραδείγματα τις μελέτες των (Scanlan, 1989; Shneiderman, 1977), είναι αντικρουόμενα.

Στα παραπάνω συμπεράσματα θα θέλαμε να προσθέσουμε και την προσωπική εμπειρία που προκύπτει από σχετική έρευνα που διενεργείται αυτή την περίοδο με φοιτητές τμήματος Πληροφορικής, σχετικά με τη χρήση εκπαιδευτικού περιβάλλοντος προγραμματισμού βασισμένου σε εκδότη σύνταξης. Αυτή τη στιγμή δεν υπάρχουν διαθέσιμα τα τελικά στοιχεία για να υποστηρίξουμε τη θέση μας, καθόσον η έρευνα αυτή βρίσκεται σε εξέλιξη, αλλά από τα μέχρι τώρα στοιχεία πιστεύουμε ότι η καλύτερη επιλογή για ένα εκπαιδευτικό προγραμματιστικό περιβάλλον είναι η *χρήση ενός λιγότερο «αυστηρού» εκδότη σύνταξης*. Τα βασικά χαρακτηριστικά ενός τέτοιου εκδότη είναι:

- Παροχή προτύπων και ενός μενού εντολών από το οποίο ο χρήστης μπορεί να επιλέξει όλες τις διαθέσιμες εντολές και δομές ελέγχου, έτσι ώστε να αποφεύγεται η επικέντρωση στις συντακτικές λεπτομέρειες και να μειώνεται η διανοητική πολυπλοκότητα που απαιτεί η εκφορά ενός αλγορίθμου στη γλώσσα προγραμματισμού.
- Δυνατότητα άμεσης εισαγωγής εντολών μεταξύ άλλων και διαγραφής εντολών ακόμα και αν η ενέργεια αυτή συνεπάγεται την παραβίαση των συντακτικών κανόνων της γλώσσας προγραμματισμού, έτσι ώστε η πραγματοποίηση αλλαγών να μην απαιτεί μια εκτεταμένη σειρά ενεργειών από τον χρήστη.
- Όσον αφορά τους τύπους των λαθών που εντοπίζονται (σημείο 3 της ενότητας Χαρακτηριστικά και Παιδαγωγικά Οφέλη) πιστεύουμε ότι είναι σημαντικό ένας εκδότης σύνταξης να εντοπίζει όχι μόνο συντακτικά, αλλά και λογικά-σημασιολογικά λάθη, στο βαθμό βέβαια που αυτό είναι εφικτό. Ωστόσο, τα σημασιολογικά και ενδεχομένως τα λογικά λάθη που εντοπίζονται κατά την ανάπτυξη ενός προγράμματος δεν πρέπει να αναφέρονται άμεσα και σε καμία περίπτωση δεν πρέπει να διορθώνονται αυτόματα (σημείο 4 της ενότητας Χαρακτηριστικά και Παιδαγωγικά Οφέλη). Είναι σημαντικό οι σπουδαστές να εξοικειωθούν με τα μηνύματα λάθους που αφορούν σε θέματα κατανόησης βασικών εννοιών, καθώς επίσης και με τη διαδικασία της αποσφαλμάτωσης. Εξάλλου, όπως αναφέρεται στη συνέχεια η πρόσβαση στο συντακτικό δένδρο επιτρέπει την αναφορά μηνυμάτων λάθους που χαρακτηρίζονται από πληρότητα και σαφήνεια.
- Αξιοποίηση της απεριόριστης πρόσβασης στο συντακτικό δένδρο για την αναφορά όχι μόνο λαθών, αλλά και προειδοποιήσεων που:
    - προσδιορίζουν τη γραμμή που πραγματικά βρίσκεται το λάθος,
    - επεξηγούν το λάθος με σαφήνεια και
    - προτείνουν ένα τρόπο διόρθωσής του.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Brusilovsky, P., Calabrese, E., Hvorecky, J., Kouchnirenko, A. & Miller P. (1997), Mini-languages: a way to learn programming principles, *Education and Information Technologies 2*, 65-83

Calloni, B. A. & Bagert, D. J. (1997), Iconic Programming Proves Effective for Teaching the First Year Programming Sequence, *ACM, SIGSCE '97 CA, USA*, 262-266

Calloni, B. and Bagert, D. (1994), Iconic Programming in BACCII vs. Textual Programming: which is a better learning environment?, *ACM, SIGSCE '94 3/94, Phoenix AZ*, 188-192

Chandhok, R., Garlan, D., Meter, G., Miller, P. & Pane, J. Karel Genie User's Manual, The MacGnome Project, Carnegie Mellon University.

Crews, T. & Ziegler, U. (1998), The Flowchart Interpreter for Introductory Programming Courses, *In Proceedings of FIE '98 Conference*, 307-312

Goldenson, D. et al. (1990), Roundtable on Structure Editing: Teachers' Experience Using Carnegie Mellon's GENIE Programming Environments (panel session*), Proceedings of the 21st SIGCSE technical symposium on Computer Science Education*, 267

Goldenson, D. R. (1989), The Impact of Structure Editing on Introductory Computer Science Education: The Results So Far, *SIGCSE Bulletin, Vol. 21, No. 3, September 1989*, 26-29

Khwaja, A. A. & Urban, J. E. (1993), Syntax-directed Editing Environments: Issues and Features, *Proceedings of the 1993 ACM, SIGGAP symposium on Applied Computing, Indianapolis, USA*, 230-237

MilIer, P., Pane, J., Meter, G. & Vorthmann. S., (1994) Evolution of Novice Programming Environments: The Structure Editors of Carnegie Mellon University, *Interactive Learning Environments 4 (2), 140-158, also available as Technical Report: Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213-3890*

Scanlan, D. (1989), Structured flowcharts outperform pseudocode: an experimental comparison, *IEEE Software, Vol 6, No 5, Sept. 1989*, 28-36

Shneiderman, B., Mayer, R., McKay, D. & Heller, P. (1977) Experimental Investigations of The Utility of Detailed Flowcharts in Programming, *Communications of the ACM, Vol. 20, No. 6*, 373-381

Smith, D.C., Cypher, A. & Sprohrer, J. (1994), KIDSIM: Programming Agents Without a Programming Language, *Communications of the ACM, Vol.37, No.7,* 55-67

Studer, S., Taylor, J. & Macie, K. (1995), YOUNGSTER: A simplified introduction to computing removing the details so that a child may program, *ACM, SIGSCE '95 3/95 Nashville, TN USA*, 102-105

Teitelbaum, T., Reps, T. (1981) The Cornell Program Synthesizer: A Syntax-Directed Programming Environment, *Communications of the ACM, Vol. 24, No. 9*, 563-573.

Templeton, B. Alice Pascal. *http://www.templetons.com/brad/alice.html*.

Toleman, M. A., Welsh, J. & Chapman, A. L. (1992), An empirical investigation of menu design in language-based editors, *ACM SDE*, 41-46

Ziegler, U. & Crews, T. (1999), An Integrated Program Development Tool for Teaching and Learning How to Program, *ACM, SIGSCE '99 3/99 New Orleans, LA, USA*, 276-280.