Ατομική Πτυχιακή Εργασία

ΔΗΜΙΟΥΡΓΙΑ ΚΑΙ ΓΡΑΦΙΚΗ ΑΠΕΙΚΟΝΙΣΗ / ΔΙΑΧΕΙΡΙΣΗ ΓΡΑΦΟΥ ΕΞΑΡΤΗΣΕΩΝ ΑΠΟ ΚΩΔΙΚΑ JAVA

Καλογήρου Χρίστος

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Σεπτέμβριος 2006

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΔΗΜΙΟΥΡΓΙΑ ΚΑΙ ΓΡΑΦΙΚΗ ΑΠΕΙΚΟΝΙΣΗ / ΔΙΑΧΕΙΡΙΣΗ ΓΡΑΦΟΥ ΕΞΑΡΤΗΣΕΩΝ ΑΠΟ ΚΩΔΙΚΑ JAVA

Χρίστος Καλογήρου

Επιβλέπων Καθηγητής
Ανδρέας Ανδρέου

Η Ατομική αυτή Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Σεπτέμβριος 2006

Ευχαριστίες

Θα ήθελα κατ’ αρχή να ευχαριστήσω τον επιβλέποντα καθηγητή μου, Δρ. Αντρέα Ανδρέου για τη βοήθεια που μου έχει προσφέρει για την περάτωση της εργασίας αυτής.
Επίσης θα ήθελα να ευχαριστήσω τον κ. Τάσο Σοφοκλέους για την βοήθεια του.

Ειδικά θα ήθελα να ευχαριστήσω την συμφοιτήτρια μου Γιάννα Ιωακείμ για την πολύτιμη βοήθεια που μου πρόσφερε καθ’ όλη την διάρκεια της ανάπτυξης του εργαλείου, εξηγώντας μου την λειτουργία του εργαλείου που ανέπτυξε και βοηθώντας με στην κατανόηση του.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1. Γενικά

Η κοινωνία του σήμερα έχει χαρακτηριστεί από πολλούς σαν «Η Κοινωνία της Πληροφορίας». Στην ζωή του καθενός μπήκαν για τα καλά οι ηλεκτρονικοί υπολογιστές και ότι αυτοί συνεπάγονται. Πλέον σχεδόν τα πάντα βασίζονται στην χρήση των ηλεκτρονικών υπολογιστών και στα διάφορα προγράμματα που τρέχουν σε αυτούς (λογισμικά).

Λόγω της αναγκαιότητας των προγραμμάτων αυτών και της ζωτικής σημασίας που αρκετές φορές έχουν, δίνεται ιδιαίτερη έμφαση στην αξιοπιστία των προγραμμάτων και μέσω ενδελεχών ελέγχων γίνεται προσπάθεια για την βελτίωση αυτής.

Ο έλεγχος του λογισμικού είναι μια από της πιο σημαντικές διαδικασίες στην όλη πορεία παραγωγής λογισμικού. Το μεγαλύτερο κόστος ενός λογισμικού, τόσο σε χρήματα, όσο και σε χρόνο δαπανιέται σε αυτόν. Σε αρκετές περιπτώσεις, ο χρόνος που απαιτείται για τον έλεγχο κάποιου λογισμικού είναι μεγαλύτερος και από τον χρόνο που δαπανιέται για την δημιουργία του. Γι' αυτό, έγιναν αρκετές προσπάθειες για αυτοματοποίηση της διαδικασίας ελέγχου των λογισμικών.

Σε προηγούμενη εργασία υλοποιήθηκε ένα εργαλείο, το οποίο παράγει από ένα πρόγραμμα τον Γράφο Ροής Ελέγχου του προγράμματος (Control Flow Graph), με σκοπό την βελτίωση της διαδικασίας ελέγχου και της συντήρησης του προγράμματος. Ο Γράφος Ροής Ελέγχου δεν είναι από μόνος του αρκετός για να γίνει ένας ολοκληρωμένος έλεγχος κάποιου προγράμματος. Χρειάζεται κάποιος να ξέρει και τις εξαρτήσεις μεταξύ των διαφόρων εντολών και δηλώσεων του προγράμματος, ούτως ώστε να γίνεται πιο σωστή παραγωγή των test cases με τα οποία θα γίνεται ο έλεγχος του λογισμικού.

Γι' αυτό, σε αυτή την διπλωματική εργασία είχαμε ως στόχο την δημιουργία ενός εργαλείου, το οποίο θα παίρνει σαν είσοδο τον Γράφο Ροής Ελέγχου που παράγει το εργαλείο της κ. Ιωάννας Ιωακείμ και θα παρουσιάζει σαν έξοδο τον Γράφο Εξαρτήσεων του προγράμματος.

2. Υποκίνηση Έρευνας

Όπως αναφέρθηκε πιο πάνω, γίνονται αρκετές προσπάθειες για αυτοματοποίηση της διαδικασίας ελέγχου των λογισμικών. Μια τέτοια προσπάθεια ξεκίνησε με την εφαρμογή που ανέπτυξε ο κ. Αναστάσιος Σοφοκλέους και συνεχίστηκε με την εφαρμογή της κ. Ιωάννας Ιωακείμ. Οι δύο αυτές εφαρμογές αυτοματοποιούν σε μεγάλο βαθμό τον έλεγχο διαφόρων προγραμμάτων, παράγοντας διάφορα test cases και τον Γράφο Ροής Ελέγχου (κανονικό και ομαδοποιημένο).

Παρόλα αυτά, δεν παράγουν τον Γράφο Εξαρτήσεων του προγράμματος που αναλύουν. Έτσι, σε αυτή την εργασία θα δημιουργήσουμε μια εφαρμογή, η οποία θα στηρίζεται στις δυο εφαρμογές που αναπτύχθηκαν, και η οποία θα παράγει τον Γράφο Εξαρτήσεων του προγράμματος.

Με την ανάπτυξη της εφαρμογής αυτής, θα αυτοματοποιείται ακόμα περισσότερο η διαδικασία ελέγχου και θα διευκολύνεται ο έλεγχος και η συντήρηση διαφόρων λογισμικών.

3. Σκιαγράφηση Έρευνας

Η εργασία αυτή στηρίχτηκε πάνω στο εργαλείο που ανέπτυξε η κ. Ιωακείμ, το οποίο παράγει τον Γράφο Ροής Ελέγχου ενός προγράμματος. Το εργαλείο που παρουσιάζεται σε αυτή την εργασία, παίρνει σαν είσοδο του τον Γράφο Ροής Ελέγχου που παράγεται και με την κατάλληλη επεξεργασία, παράγει τον Γράφο Εξαρτήσεων του προγράμματος που αναλύεται.

Η δομή αυτής της μελέτης έχει την ακόλουθη μορφή:

- Κεφάλαιο 2: Γράφοι
  Γίνεται μια σύντομη αναφορά στην ιστορία της θεωρίας των γράφων και ακολούθως παρουσίαση των κύριων τύπων τους, καθώς και μερικές από τις χρήσεις τους. Τελειώνουμε με μια αναφορά στις χρήσεις των γράφων στην Τεχνολογία Λογισμικού και παρουσίαση του Γράφου Ροής Ελέγχου και του Γράφου Εξαρτήσεων.

- Κεφάλαιο 3: Παρουσίαση Προηγούμενης Δουλειάς
  Γίνεται μια συνοπτική παρουσίαση της δουλειάς που έγινε από την κ. Ιωάννα Ιωακείμ και τον κ. Αναστάσιο Σοφοκλέους.

- Κεφάλαιο 4: Εφαρμογή αυτόματης παραγωγής Γράφου Εξαρτήσεων
  Γίνεται μια παρουσίαση του αλγορίθμου με τον οποίο δημιουργήθηκε ο Γράφος Εξαρτήσεων με ενδεικτικά παραδείγματα κώδικα.

- Κεφάλαιο 5: Παρουσίαση εργαλείου
  Γίνεται μια περιγραφή της λειτουργίας του εργαλείου και του τρόπου με τον οποίο ο χρήστης μπορεί να το χρησιμοποιήσει. Επίσης παρουσιάζεται ένα παράδειγμα ανάλυσης προγράμματος και μετρήσεις για το χρόνο εκτέλεσης διαφόρων προγραμμάτων.

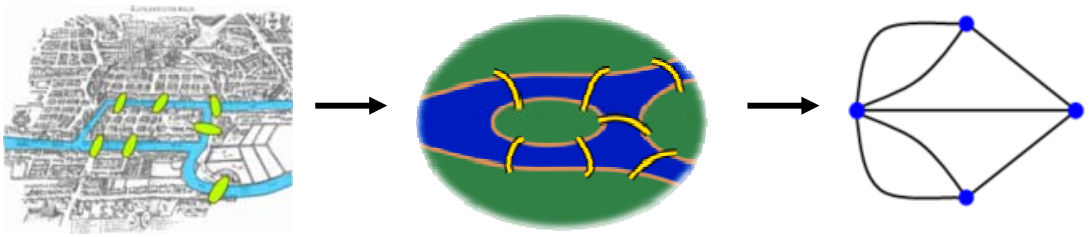
- Κεφάλαιο 6: Προτάσεις για μελλοντική εργασία
  Γίνεται μερικές προτάσεις για μελλοντικές προσθήκες ή και βελτιώσεις καθώς και για εργασίες που θα στηρίζονται στην παρούσα εργασία.

Κεφάλαιο 2

Γράφοι

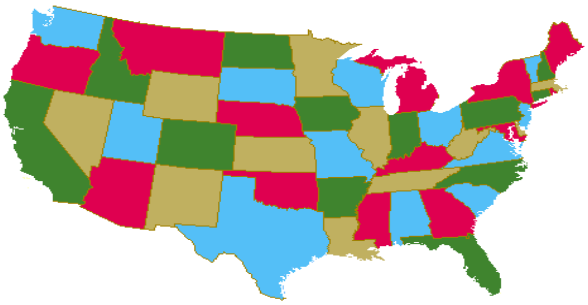
1. Εισαγωγή στην Θεωρία των Γράφων

Η βασική ιδέα της Θεωρίας των Γράφων παρουσιάστηκε το 1736 από τον Ελβετό φυσικομαθηματικό Leonhard Euler σε μια μελέτη του για το πρόβλημα Seven Bridges of Königsberg (Σχήμα 1) . Σκοπός του προβλήματος αυτού, να βρεθεί ένα μονοπάτι που να διασχίζει όλες τις γέφυρες και να καταλήγει στο σημείο εκκίνησης, περνώντας μόνο μία φορά πάνω από κάθε γέφυρα, πράγμα αδύνατο όπως απέδειξε ο Euler.



(Σχήμα 1: Οι 7 γέφυρες του Königsberg)

Το πρόβλημα που οδήγησε στην ανάπτυξη της Θεωρίας των Γράφων, προτάθηκε το 1852 από τον Νοτιοαφρικάνο μαθηματικό Francis Guthrie και αφορούσε το χρωματισμό ενός χάρτη με τέσσερα χρώματα (σχήμα 2), με την προϋπόθεση οι γειτονικές χώρες να έχουν διαφορετικό χρώμα (four color problem). Κατά την προσπάθεια για την επίλυση του προβλήματος αυτού, το οποίο επιλύθηκε το 1976 από τους Kenneth Appel και Wolfgang Haken, αρκετοί μαθηματικοί εφεύραν αρκετούς βασικούς θεωρητικούς όρους και έννοιες για τους Γράφους.

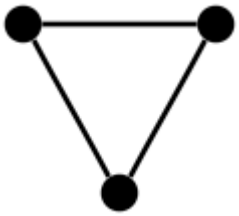


(Σχήμα 2: Χρωματισμένος χάρτης)

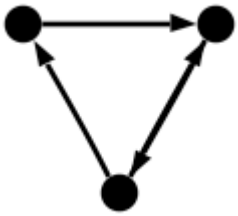
2. Τύποι Γράφων

Με τον όρο Γράφος, δηλώνουμε μια δομή που αποτελείται από ένα διατεταγμένο ζεύγος G =(V, E), όπου G ο Γράφος, V ένα σετ από κόμβους (vertices) και E ένα σετ από ακμές (edges), οι οποίες ενώνουν τους κόμβους μεταξύ τους.

Οι Γράφοι χωρίζονται κυρίως σε δύο κατηγορίες, τους μη κατευθυνόμενους (σχήμα 3) και τους κατευθυνόμενους (σχήμα 4).



(Σχήμα 3: Μη κατευθυνόμενος γράφος)



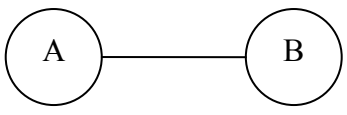
(Σχήμα 4: Κατευθυνόμενος Γράφος)

2.1 Μη κατευθυνόμενοι Γράφοι

Οι μη κατευθυνόμενοι γράφοι, χαρακτηρίζονται σαν μια διατεταγμένη τριάδα G = (V, E, f), στην οποία:

- V: ένα σύνολο κόμβων
- E: ένα σύνολο ακμών
- f: μια συνάρτηση, η οποία συσχετίζει μία ακμή από το σύνολο των ακμών E με δύο κόμβους από το σύνολο των κόμβων, V.

Η ακμή που ενώνει τους δύο κόμβους δεν έχει μία συγκεκριμένη κατεύθυνση, έτσι η σχέση μεταξύ των κόμβων είναι αμφίδρομη. Δηλαδή η σχέση που έχει ο κόμβος A ως προς τον κόμβο B είναι η ίδια με την σχέση που έχει ο κόμβος B ως προς τον A (σχήμα 5).



(Σχήμα 5: Ένωση δυο κόμβων σε μη κατευθυνόμενο γράφο)

2.2 Κατευθυνόμενοι Γράφοι

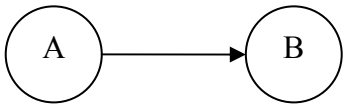
Οι κατευθυνόμενοι γράφοι, χαρακτηρίζονται σαν μια διατεταγμένη τριάδα G = (V, A, f), στην οποία:

V: ένα σύνολο κόμβων

A: ένα σύνολο κατευθυνόμενων ακμών

f: μια συνάρτηση η οποία συσχετίζει μια ακμή από το σύνολο A με δύο κόμβους από το σύνολο V.

Η ακμή που ενώνει τους δύο κόμβους, λόγω του ότι έχει μια συγκεκριμένη κατεύθυνση, δηλώνει μία συγκεκριμένη σχέση μεταξύ των δύο κόμβων, η οποία όμως δεν ισχύει αμφίδρομα. Η σχέση που αναπαρίσταται, ισχύει μόνο προς την κατεύθυνση που ακολουθεί η ακμή (σχήμα 6).



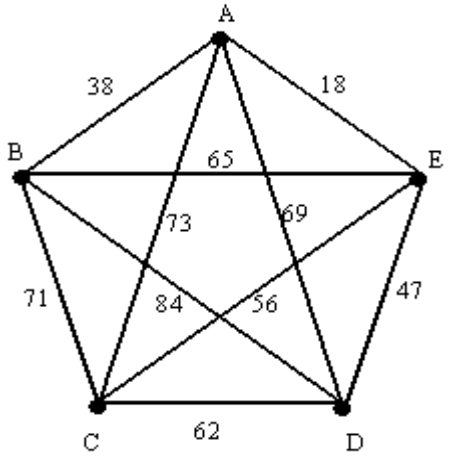
(Σχήμα 6: Ένωση δυο κόμβων σε κατευθυνόμενο γράφο)

3. Χρήσεις Γράφων

Η χρήση των γράφων είναι διαδεδομένη σε αρκετούς τομείς και χρησιμοποιούνται για παρουσίαση και επίλυση διαφόρων προβλημάτων. Μερικά παραδείγματα χρήσεων των γράφων είναι:

- Χρωματισμός χάρτη: Οι κόμβοι του γράφου αναπαριστούν τις διάφορες περιοχές του χάρτη και οι ακμές που ενώνουν δύο κόμβους, δηλώνουν γειτνίαση των περιοχών του χάρτη που αναπαριστούν.

- Συγκοινωνίες: Στις συγκοινωνίες γίνεται χρήση των γράφων για επίλυση προβλημάτων υπολογισμού του κόστους για κατασκευή δρόμων ή σιδηροδρομικών συνδέσεων μεταξύ πόλεων. Οι γράφοι αναπαριστούν τις πόλεις και οι ακμές που τους ενώνουν παρουσιάζουν την σύνδεση μεταξύ των δύο πόλεων. Πάνω στις ακμές ανατίθενται διάφορα βάρη (Σχήμα 7)ούτως ώστε να παρουσιάζουν το κόστος της σύνδεσης, για να γίνονται οι υπολογισμοί για την πιο οικονομική σύνδεση.
  Επίσης χρησιμοποιούνται και για υπολογισμό του κοντινότερου μονοπατιού μεταξύ δύο πόλεων, καθώς και την εύρεση του πιο κοντινού γείτονα (Nearest Neighbor).



(Σχήμα 7: Γράφος με βάρη στις ακμές)

- Δίκτυα Υπολογιστών: Γίνεται χρήση γράφων για παρουσίαση της τοπολογίας ενός δικτύου υπολογιστών. Οι κόμβοι παρουσιάζουν τα τερματικά (υπολογιστές) και οι ακμές την συνδεσμολογία μεταξύ τους.

4. Γράφοι στην Τεχνολογία Λογισμικού

Κατά την διαδικασία ανάπτυξης κάποιου λογισμικού, ο προγραμματιστής (ή η προγραμματιστική ομάδα) που είναι υπεύθυνος για την ανάπτυξη, επιβάλλει κάποιο συγκεκριμένο τρόπο σχεδιασμού και υλοποίησης του λογισμικού. Πρέπει όμως να βεβαιωθεί ότι ο τρόπος αυτό θα είναι κατανοητός, ούτως ώστε οι προγραμματιστές που θα αναλάβουν τον έλεγχο και την συντήρηση του λογισμικού να μπορούν να κατανοήσουν και να ακολουθήσουν τον συγκεκριμένο τρόπο σχεδιασμού.

Με την ανάπτυξη μεγάλων και πολύπλοκων λογισμικών, η ανάγκη για καλή περιγραφή και ανάλυση της δομής των λογισμικών έγινε ακόμα μεγαλύτερη. Γι' αυτό, έγινε εισαγωγή διαφόρων ειδών γράφων στην διαδικασία σχεδιασμού, ανάπτυξης, ελέγχου και συντήρησης των λογισμικών.

Οι κυριότεροι γράφοι που χρησιμοποιούνται από τους προγραμματιστές είναι ο Γράφος Ροής Ελέγχου (Control Flow Graph) και ο Γράφος Εξαρτήσεων Προγράμματος (Program Dependence Graph). Με την χρήση των γράφων αυτών, οι προγραμματιστές μπορούν να δείξουν την λειτουργία και τον τρόπο που δουλεύει το λογισμικό και έτσι διευκολύνεται ο έλεγχος του λογισμικού, η συντήρηση του καθώς και η περαιτέρω ανάπτυξη του.

Ακολουθεί μια σύντομη παρουσίαση των δύο αυτών γράφων:

4.1. Γράφος Ροής Ελέγχου (Control Flow Graph)

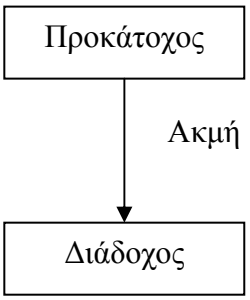
4.1.1. Περιγραφή

Ο Γράφος Ροής Ελέγχου (Control Flow Graph), είναι μια αφηρημένη δομή δεδομένων (abstract data structure) που παρουσιάζει γραφικά την ροή του ελέγχου μέσα σε ένα πρόγραμμα ή μια διαδικασία (procedure).

Αποτελείται από κόμβους (vertices) και ακμές (edges). Οι κόμβοι του Γράφου Ροής Ελέγχου, αντιπροσωπεύουν τις βασικές εκφράσεις του προγράμματος

(δηλώσεις και εντολές) και οι ακμές, οι οποίες είναι κατευθυνόμενες (directed), την πιθανή ροή του ελέγχου ανάμεσα στους κόμβους.

Κάθε κόμβος του Γράφου Ροής Ελέγχου, είναι ενωμένος με κάποιο άλλο κόμβο μέσω μίας ακμής. Η ένωση των δύο κόμβων, υποδηλώνει την ροή του ελέγχου του προγράμματος από τον ένα κόμβο στον άλλο. Ο κόμβος από τον οποίο φεύγει η ακμή ονομάζεται προκάτοχος (predecessor) και αυτός στον οποίο καταλήγει η ακμή ονομάζεται διάδοχος (successor) (σχήμα 8).



(Σχήμα 8: Ένωση κόμβων)

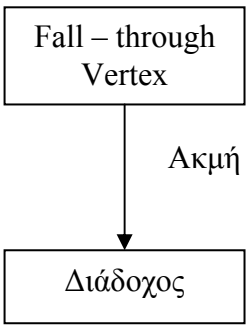
Υπάρχουν τρεις τύποι κόμβων στο Γράφο Ροής Ελέγχου:

- Οι κόμβοι που έχουν ένα διάδοχο (fall – through vertices) και οι οποίοι αντιπροσωπεύουν εντολές ή δηλώσεις του προγράμματος (σχήμα 9),

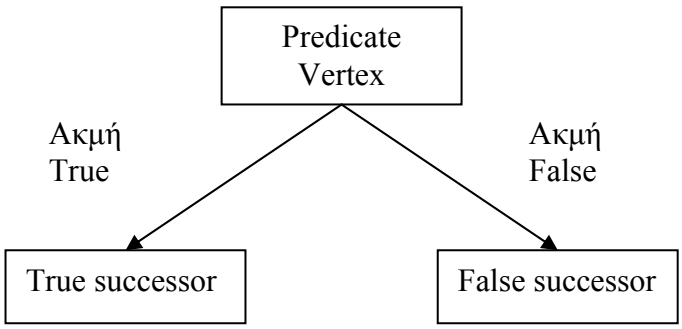
- οι κόμβοι που έχουν δύο διαδόχους (predicate vertices), οι οποίοι ελέγχουν κάποια συνθήκη και η ροή του προγράμματος ακολουθεί την ανάλογη πορεία προς τον αντίστοιχο διάδοχο (true successor ή false successor) σε σχέση με την συνθήκη που ελέγχεται (σχήμα 10) και

- ο ειδικός κόμβος εξόδου του προγράμματος (EXIT vertex), ο οποίος δεν έχει κανένα διάδοχο και σηματοδοτεί το τέλος του Γράφου Ροής Ελέγχου (σχήμα 11).

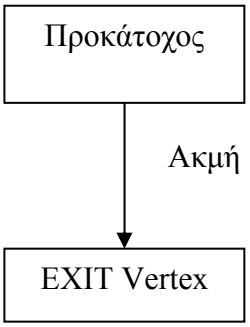
Κάθε κόμβος έχει τουλάχιστον ένα προκάτοχο, με εξαίρεση τον κόμβο – ρίζα (ENTRY vertex) του Γράφου Ροής Ελέγχου, ο οποίος σηματοδοτεί την αρχή του Γράφου και δεν έχει προκάτοχο (σχήμα 12) .



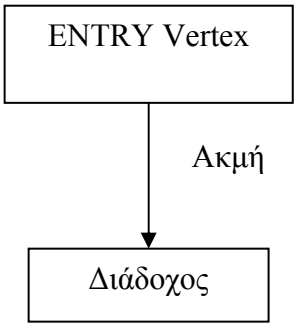
(Σχήμα 9:
Fall – through Vertex)



(Σχήμα 10:
Predicate Vertex)

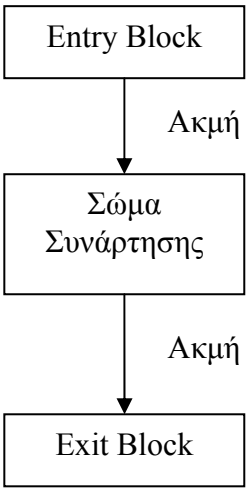


(Σχήμα 11:
EXIT Vertex)



(Σχήμα 12:
ENTRY Vertex)

Μέσα στο Γράφο Ροής Ελέγχου, υπάρχουν δύο ειδικοί τύποι κόμβων, οι οποίοι δηλώνουν την είσοδο και την έξοδο συναρτήσεων (Entry Block και Exit Block). Αυτοί οι ειδικοί κόμβοι που δεν αντιπροσωπεύουν εκφράσεις του προγράμματος, δηλαδή δεν περιέχουν κανένα κομμάτι κώδικα από το πρόγραμμα (σχήμα 13).



(Σχήμα 13: Entry - Exit Block)

Οι ακμές του Γράφου Ροής Ελέγχου, αναπαριστούν την πιθανή ροή του ελέγχου ανάμεσα στους κόμβους του Γράφου, δηλαδή ανάμεσα στις βασικές εκφράσεις και τις εντολές του προγράμματος. Οι ακμές είναι κατευθυνόμενες (directed edges) και ξεκινούν από τον κόμβο – προκάτοχο (predecessor) και καταλήγουν στον κόμβο – διάδοχο (successor).

Από κάθε κόμβο φεύγει μία μόνο ακμή, εκτός από τους κόμβους οι οποίοι ελέγχουν κάποια συνθήκη (predicate vertices) και σε κάθε κόμβο καταλήγει μόνο μία ακμή, εκτός από τους κόμβους στους οποίου καταλήγουν τα μονοπάτια από τους predicate vertices, καθώς και στους ίδιους τους predicate vertices μέσω ακμών που επιστρέφουν σε αυτούς (περιπτώσεις for, while. – back edges).

4.1.2. Δημιουργία

Ο Γράφος Ροής Ελέγχου δημιουργείται από τους προγραμματιστές για να παρουσιάζεται γραφικά η ροή ελέγχου ενός λογισμικού, ούτως ώστε να είναι εύκολη η κατανόηση του τρόπου που λειτουργεί και έτσι να διευκολύνεται ο έλεγχος και η συντήρηση του λογισμικού.

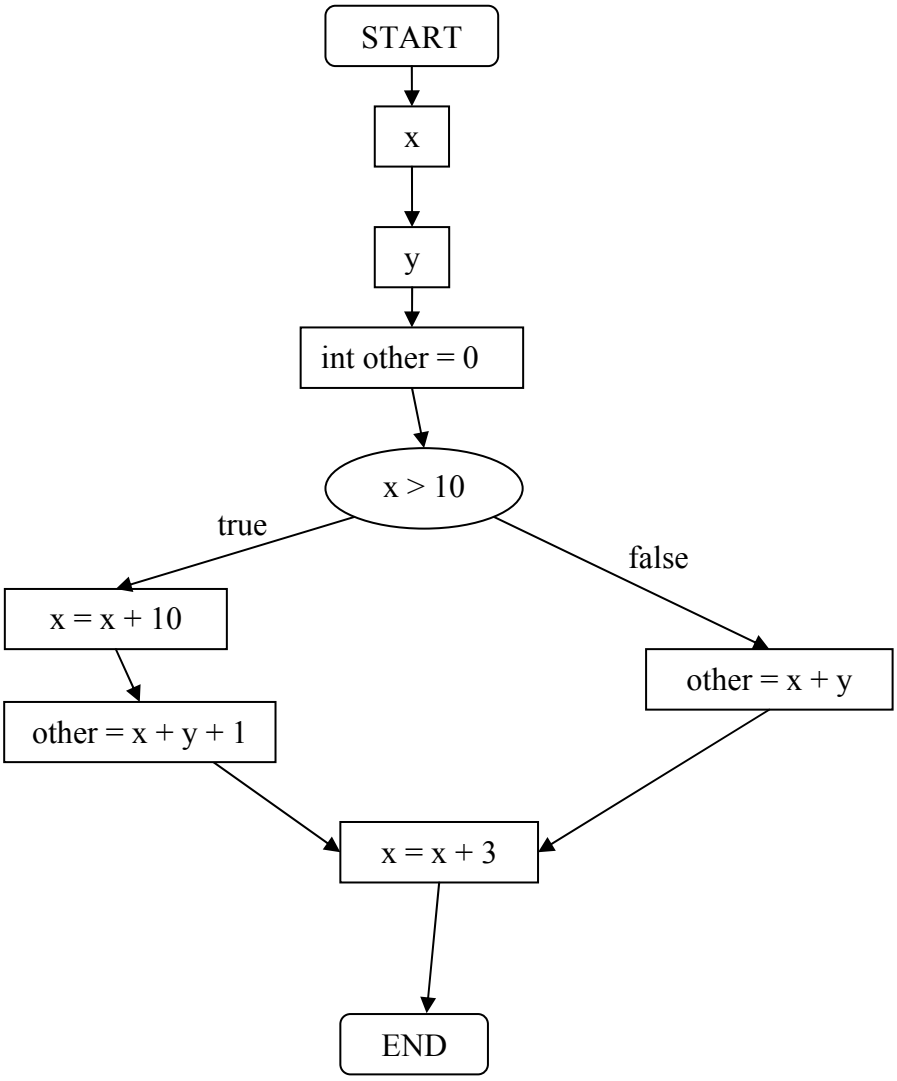
Η παραγωγή του γίνεται κατευθείαν από τον κώδικα του λογισμικού (source code) είτε με χρήση κάποιου μεταγλωττιστή που μεταφράζει τον κώδικα σε γράφο ροής είτε με χρήση ειδικών εργαλείων που παράγουν τον γράφο ροής.

Παρακάτω βλέπουμε ένα παράδειγμα Γράφου Ελέγχου Ροής για το ακόλουθο πρόγραμμα (σχήμα 14):

```
public class Program1 {
public Program1 () {
}
public void methodA(int x, int y)
{
int other = 0;
if(x>10){
x = x + 10;
other = x + y + 1;
}
else{
other = x + y;
}
x = x+ 3;
}
```

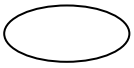
(Σχήμα 14: Πρόγραμμα για παραγωγή Γράφου Ροής Ελέγχου)

Από το πιο πάνω πρόγραμμα (σχήμα 14), παράγεται ο Γράφος Ροής Ελέγχου που ακολουθεί (σχήμα 15):



(Σχήμα 15: Γράφος Ροής Ελέγχου προγράμματος σχήματος 14)

Υπόμνημα:

-  : Κόμβος που αντιπροσωπεύει βασικές εκφράσεις
-  : Κόμβος ελέγχου συνθήκης

4.2. Γράφος Εξαρτήσεων Προγράμματος (Program Dependence Graph)

4.2.1. Περιγραφή

Ο Γράφος Εξαρτήσεων είναι και αυτός, όπως ο Γράφος Ροής Ελέγχου, μια αφηρημένη δομή δεδομένων, η οποία παρουσιάζει γραφικά τις εξαρτήσεις μεταξύ των βασικών εκφράσεων (δηλώσεις και εντολές) ενός προγράμματος ή μιας διαδικασίας. Η παρουσίαση αυτών των εξαρτήσεων γίνεται μέσω κόμβων και ακμών, όπου οι κόμβοι αντιπροσωπεύουν τις βασικές εκφράσεις και οι ακμές την εξάρτηση μεταξύ τους.

Η συνένωση των κόμβων του Γράφου Εξαρτήσεων γίνεται με τον ίδιο τρόπο όπως και αυτών του Γράφου Ροής Ελέγχου, δηλαδή η ακμή φεύγει από τον κόμβο – προκάτοχο και καταλήγει στον κόμβο – διάδοχο (σχήμα 8).
Οι ακμές που ενώνουν τους κόμβους του Γράφου Εξαρτήσεων δεν είναι όλες του ίδιου τύπου, όπως στον Γράφο Ροής Ελέγχου, αλλά υπάρχουν τρία (3) διαφορετικά ήδη ακμών:

- Ακμές εξάρτησης ροής (Control Dependence Edges): Είναι οι ακμές που παρουσιάζουν την εξάρτηση μεταξύ των βασικών εκφράσεων του προγράμματος. Ενώνουν μεταξύ του τους κόμβους του Γράφου Εξαρτήσεων, ανάλογα με την σχέση εξάρτησης τους. Οι ακμές αυτές ξεκινούν πάντα από τον κόμβο εισόδου (ENTRY Vertex (σχήμα 12)) ή από ένα predicate vertex (σχήμα 10).

- Ακμές εξάρτησης δεδομένων (Data Dependence Edges): Είναι οι ακμές που παρουσιάζουν την εξάρτηση των βασικών εκφράσεων του προγράμματος πάνω στις μεταβλητές του προγράμματος. Ενώνουν τους κόμβους που αντιπροσωπεύουν την δήλωση των μεταβλητών (Variable Declaration) με τους κόμβους, οι οποίοι περιέχουν χρήση της μεταβλητής από την οποία προέρχεται η ακμή.

- Summary Edges: Είναι οι ακμές που παρουσιάζουν την εξάρτηση των βασικών εκφράσεων του προγράμματος πάνω στις παραμέτρους του προγράμματος. Ενώνουν τους κόμβους που αντιπροσωπεύουν τις παραμέτρους με τους κόμβους, οι οποίοι περιέχουν χρήση της παραμέτρου από την οποία προέρχεται η ακμή.

Οι κόμβοι που αποτελούν τον Γράφο Εξαρτήσεων, είναι οι ίδιοι που αποτελούν τον Γράφο Ροής Ελέγχου, εκτός από τον κόμβο Εξόδου (EXIT Vertex (σχήμα 11)). Στην περίπτωση που το πρόγραμμα μας αποτελείται από πολλές μεθόδους και ο Γράφος Εξαρτήσεων που δημιουργούμε ονομάζεται Γράφος Εξαρτήσεων Συστήματος (System Dependence Graph) και περιλαμβάνει επιπλέον κόμβους και ακμές, σε σχέση με τον Γράφο Εξαρτήσεων Προγράμματος (σχήμα 16).

Οι επιπλέον κόμβοι που υπάρχουν στον γράφο είναι οι ακόλουθοι:

- Call Vertex: Αναπαριστά την κλήση κάποιας μεθόδου του προγράμματος
- Actual – in Vertex: Αναπαριστά την δράση που παίρνει η μέθοδος που καλεί μία άλλη μέθοδο για να μεταφέρει την πραγματική παράμετρο.
- Formal – in Vertex: Αναπαριστά την δράση που παίρνει η μέθοδος που καλείται για να πάρει την τιμή της παραμέτρου.
- Actual – out Vertex: Αναπαριστά την δράση που παίρνει η μέθοδος που κάλεσε την άλλη μέθοδο για να πάρει πίσω την τιμή επιστροφής της παραμέτρου
- Formal – out Vertex: Αναπαριστά την δράση που παίρνει η μέθοδος που κλήθηκε για να επιστρέψει την τιμή επιστροφής της παραμέτρου

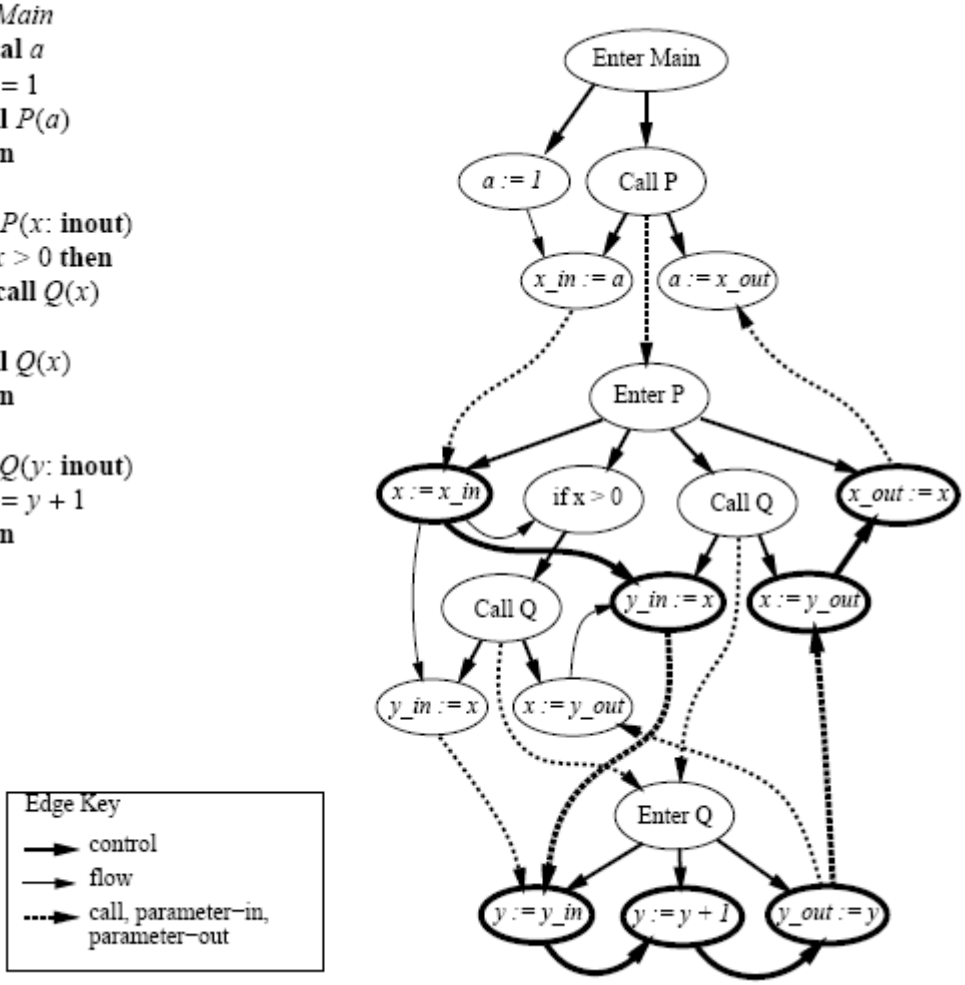
Οι επιπλέον ακμές που υπάρχουν στον γράφο είναι:

- Parameter – in Edge: Ενώνει τον κόμβο Actual - in με τον Formal – in.
- Parameter – out Edger: Ενώνει τον κόμβο Actual - out με τον Formal – out.

```
proc Main
   local a
   a := 1
   call P(a)
return

proc P(x: inout)
   if x > 0 then
      call Q(x)
   fi
   call Q(x)
return

proc Q(y: inout)
   y := y + 1
return
```



(Σχήμα 16: Γράφος Εξαρτήσεων Συστήματος (System Dependence Graph))

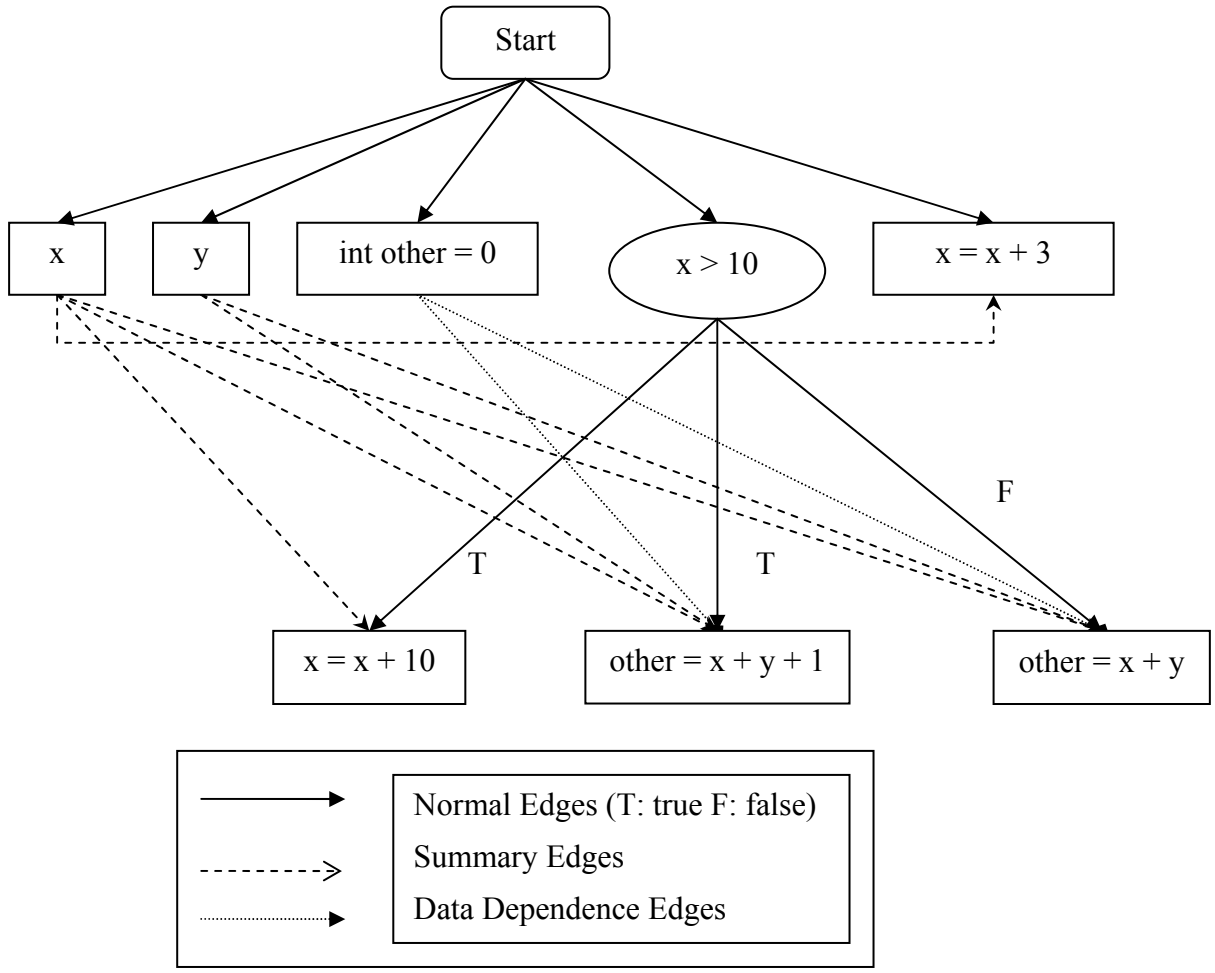
4.2.2. Δημιουργία

Για την δημιουργία του Γράφου Εξαρτήσεων ενός προγράμματος, χρησιμοποιείται ο Γράφος Ροής Ελέγχου που παράχθηκε για αυτό το πρόγραμμα. Οι αλγόριθμοι που υπάρχουν για την παραγωγή του Γράφου εξαρτήσεων έχουν σαν είσοδο τους τον Γράφο Ροής Ελέγχου ενός προγράμματος και, μετά από την ανάλογη επεξεργασία του, σαν έξοδο τον Γράφο Εξαρτήσεων του προγράμματος (σχήμα 17).

```
algorithm   ConstructSDG
input       Control flow graph (CFG) of Program P
output      System dependence graph (SDG) of P
declare
 begin ConstructSDG
1.  foreach class C
2.      Identify the methods that need new representations
3.      foreach method m declared in C
4.          Compute control dependences for m
5.          Preprocess m with usual data-flow analysis
6.          Expand objects in m
7.          Compute data dependences for objects
8.      endfor
9.      foreach method m in the base classes
10.         if m is "marked" then
11.             Copy old procedure dependence graph
12.             Adjust callsites
13.         else
14.             Reuse m's old procedure dependence graph
15.         endif
16.     endfor
17. endfor
18. Connect()
19. BuildSummaryEdges()
 end ConstructSDG
```

(Σχήμα 17: Ένας από τους αλγορίθμους δημιουργίας Γράφου Εξαρτήσεων)

Έχοντας το πρόγραμμα του σχήματος 14 και τον Γράφο Ροής Ελέγχου που παράγεται από το πρόγραμμα αυτό (σχήμα 15), δημιουργούμε τον Γράφο Εξαρτήσεων του προγράμματος (σχήμα 18)



(Σχήμα 18: Γράφος Εξαρτήσεων προγράμματος σχήματος 14)

Στον Γράφο Εξαρτήσεων που δημιουργήσαμε, βλέπουμε τους τρεις (3) τύπους ακμών που υπάρχουν, καθώς και την απουσία του κόμβου εξόδου (EXIT Vertex). Το παράδειγμα του σχήματος 18, αναφέρεται σε ένα πρόγραμμα, το οποίο αποτελείται μόνο από μία μέθοδο. Σε περίπτωση που το πρόγραμμα μας περιέχει περισσότερες μεθόδους τότε, όπως προαναφέραμε, παράγουμε τον Γράφο Εξαρτήσεων Συστήματος (σχήμα 16).

Κεφάλαιο 3

Παρουσίαση Προηγούμενης Δουλειάς

1. Εισαγωγή

Η προηγούμενη δουλειά, που έγινε από την κ. Ιωάννα Ιωακείμ, είχε ως πρωταρχικό σκοπό της την αυτόματη - δυναμική παραγωγή του Γράφου Ροής Ελέγχου ενός προγράμματος.

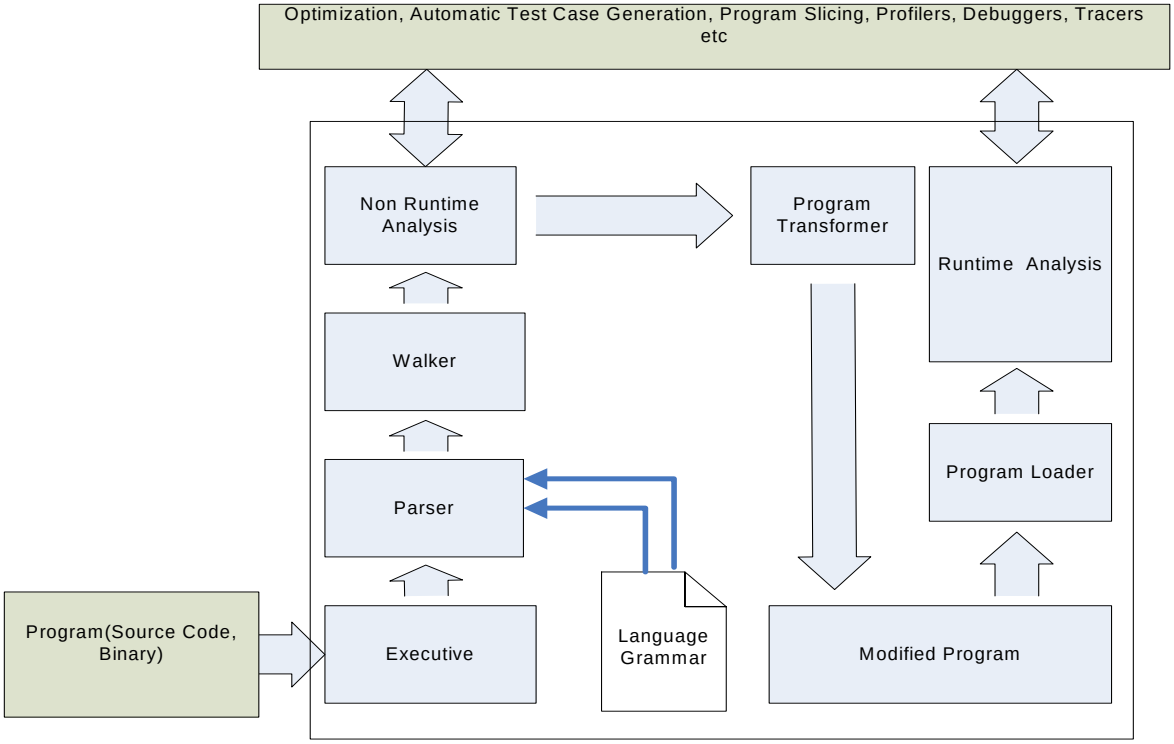
Δημιουργήθηκε ένα εργαλείο, το οποίο παίρνει σαν είσοδο του ένα πρόγραμμα (γραμμένο σε JAVA) και μετά από την απαραίτητη ανάλυση, παράγει τον Γράφο Ροής Ελέγχου του συγκεκριμένου προγράμματος.

Στο κεφάλαιο αυτό, θα γίνει μια σύντομη περιγραφή της λειτουργίας του εργαλείου που αναπτύχθηκε για την παραγωγή του Γράφου Ροής Ελέγχου, καθώς και της δουλειάς του κ. Αναστάσιου Σοφοκλέους, πάνω στην οποία στηρίχτηκε η εργασία της κ. Ιωακείμ.

2. Παρουσίαση και περιγραφή προηγούμενης δουλειάς

Για να γίνει κατορθωτή η αυτόματη παραγωγή του Γράφου Ροής Ελέγχου ενός προγράμματος, είναι απαραίτητη η ανάλυση του προγράμματος αυτού ούτως ώστε να εντοπιστούν διάφορες πληροφορίες από την συμπεριφορά του.

Η ανάλυση αυτή, γίνεται με την χρήση ενός αναλυτή προγραμμάτων, ο οποίος υποστηρίζει ανάλυση προγραμμάτων γραμμένων στην γλώσσα προγραμματισμού JAVA. Η αρχιτεκτονική του αναλυτή αυτού (σχήμα 1), αποτελείται από εννέα (9) στρώματα, όπου το κάθε στρώμα είναι υπεύθυνο για μια λειτουργία:



(Σχήμα 1: Αρχιτεκτονική Αναλυτή προγραμμάτων)

Τα εννέα στρώματα του αναλυτή είναι:

- IOExecutive: Αναγνωρίζει και γράφει το αρχείο του κώδικα
- Parser: Αναλύει τον κώδικα του προγράμματος, βάσει μιας γραμματικής της γλώσσας του προγράμματος (JAVA)
- Walker: Δημιουργεί τους κόμβους και τις ακμές για τον Γράφο Ροής Ελέγχου
- Non Runtime: Παρέχει τη διεπαφή για περαιτέρω ενότητες ανάλυσης.
- Program Transformer: Υπεύθυνο για μια πιο προηγμένη ανάλυση του προγράμματος
- Modified Program: Αποτελεί το στρώμα μετασχηματιστών προγράμματος

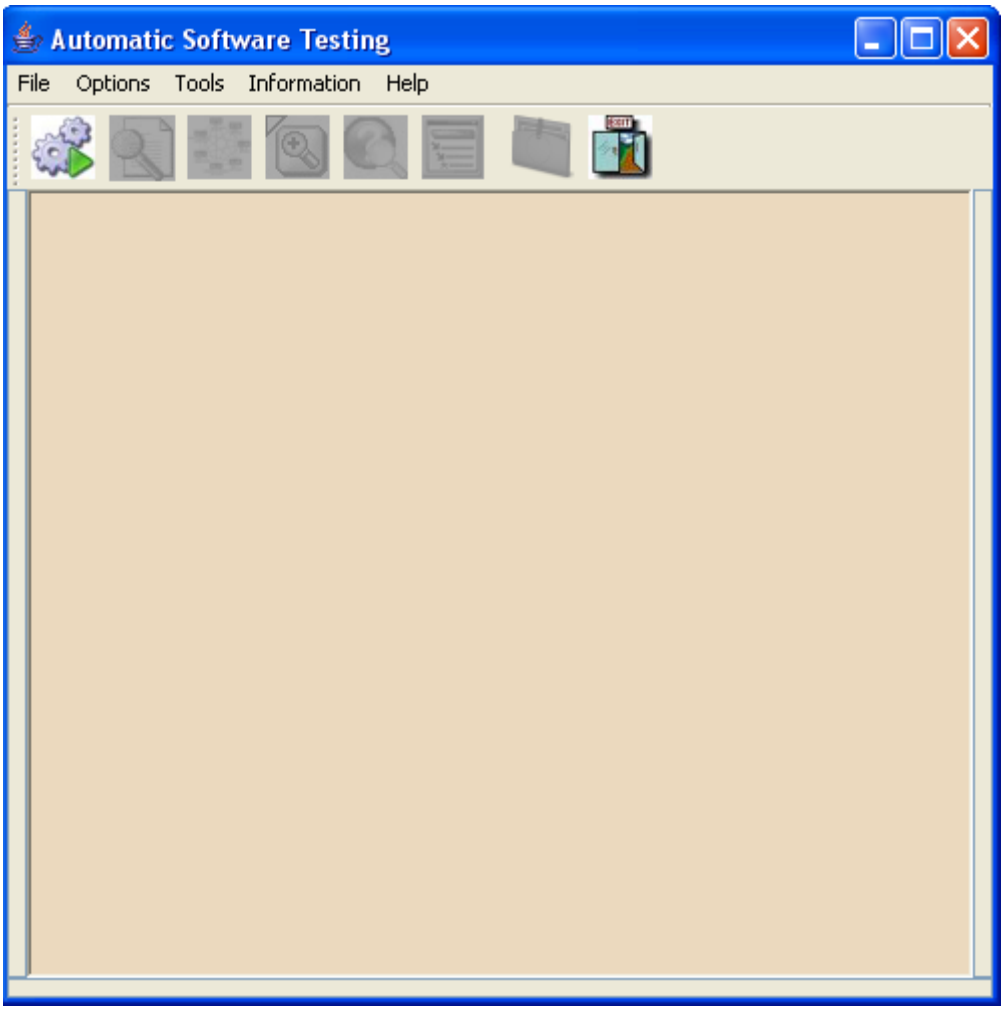
- IOWriter: Υπεύθυνο για το επαναγράψιμο των αρχείων του κώδικα
- Program Loader: Υπεύθυνο για φόρτωμα των κλάσεων του προγράμματος
- Runtime Analysis: Υπεύθυνο για δυναμική επαλήθευση των πόρων που καλύπτει ένας χρήστης.

Στο εργαλείο που δημιουργήθηκε από την κ. Ιωακείμ, ο χρήστης μπορεί μέσω της γραφικής διεπαφής να εκτελέσει εύκολα τις λειτουργίες του εργαλείου:

- Μπορεί να επιλέξει το πρόγραμμα που θα αναλύσει
- Μπορεί να δει τον πηγαίο κώδικα του προγράμματος που επέλεξε
- Μπορεί να δει διάφορες πληροφορίες για το πρόγραμμα που επέλεξε
- Μπορεί να δημιουργήσει τον ομαδοποιημένο Γράφο Ροής Ελέγχου του προγράμματος
- Μπορεί να δημιουργήσει τον κανονικό Γράφο Ροής Ελέγχου του προγράμματος
- Μπορεί να δει πληροφορίες για κάποιο συγκεκριμένο κόμβο του κανονικού Γράφου Ροής Ελέγχου
- Μπορεί να δει τον υπογράφο που αντιστοιχεί στις εντολές ενός κόμβου του ομαδοποιημένου Γράφου Ροής Ελέγχου

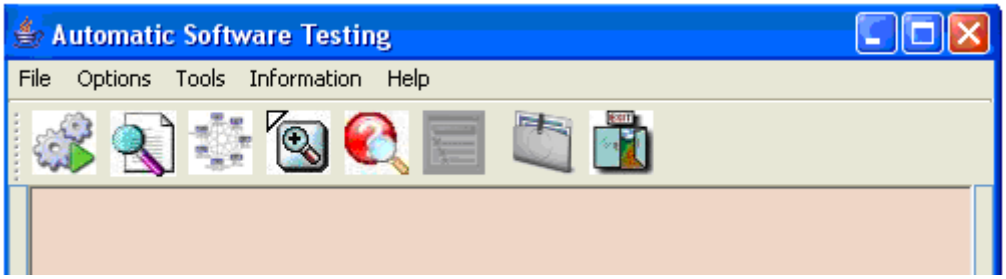
Αρχικά, με την εκκίνηση του εργαλείου, εμφανίζεται η κύρια οθόνη (σχήμα 2), από την οποία ο χρήστης μπορεί να εκτελέσει τις λειτουργίες που του προσφέρει το εργαλείο. Όλες οι επιλογές που του παρέχονται, μπορούν να εκτελεστούν με τρεις διαφορετικούς τρόπους:

- Ο χρήστης πατά το ανάλογο κουμπί από την μπάρα εργαλείων (toolbar)
- Ο χρήστης επιλέγει από το μενού επιλογών (menu bar) την ανάλογη επιλογή
- Ο χρήστης χρησιμοποιεί τον ανάλογο συνδυασμό πλήκτρων



(Σχήμα 2: Κεντρική οθόνη εργαλείου)

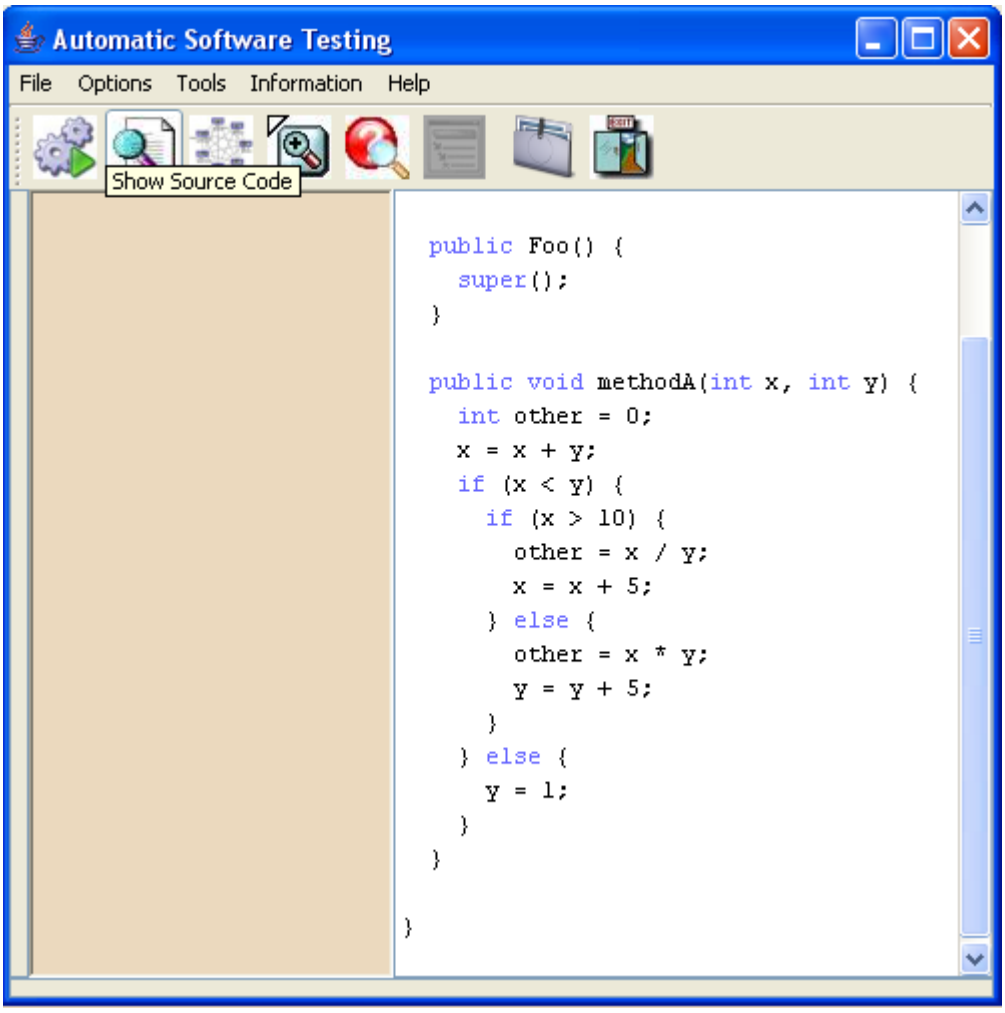
Αρχικά οι λειτουργίες του εργαλείου δεν είναι ενεργές, μέχρι ο χρήστης να επιλέξει το πρόγραμμα που θέλει να αναλύσει. Όταν επιλέξει το πρόγραμμα που θέλει και πατήσει το κουμπί εκτέλεσης στην μπάρα εργαλείων (ή χρησιμοποιήσει κάποιο από τους άλλους τρόπους που αναφέρθηκαν πιο πάνω) τότε και οι υπόλοιπες λειτουργίες ενεργοποιούνται (σχήμα 3), εκτός από την επιλογή για παρουσίαση πληροφοριών κάποιου κόμβου (αφού δεν σχεδιάστηκε ο Γράφος Ροής Ελέγχου ακόμα)



(Σχήμα 3: Μπάρα εργαλείων μετά την επιλογή και ανάλυση ενός προγράμματος)

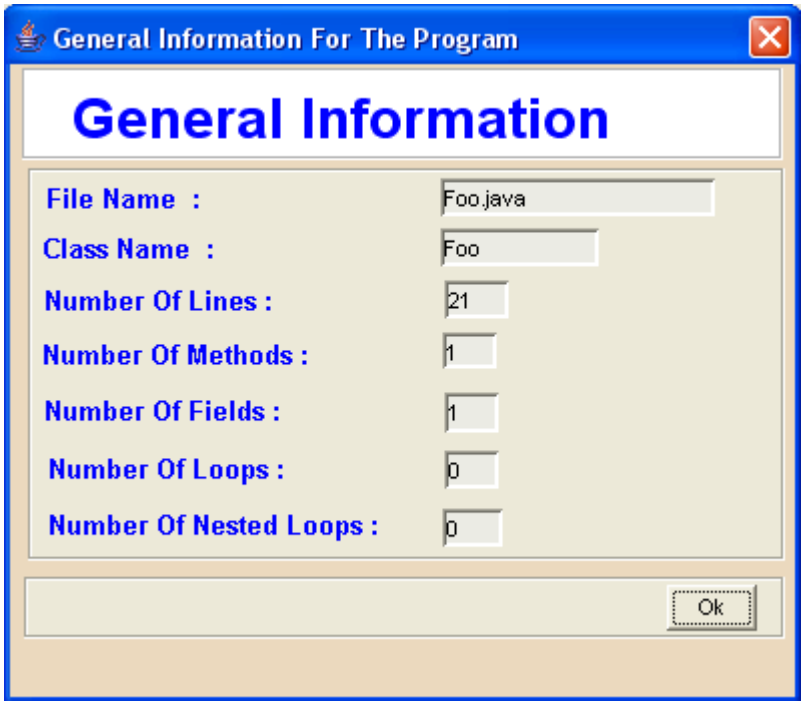
Όταν ενεργοποιηθούν και οι υπόλοιπες λειτουργίες, τότε ο χρήστης:

- Μπορεί να δει τον πηγαίο κώδικα του προγράμματος που επέλεξε



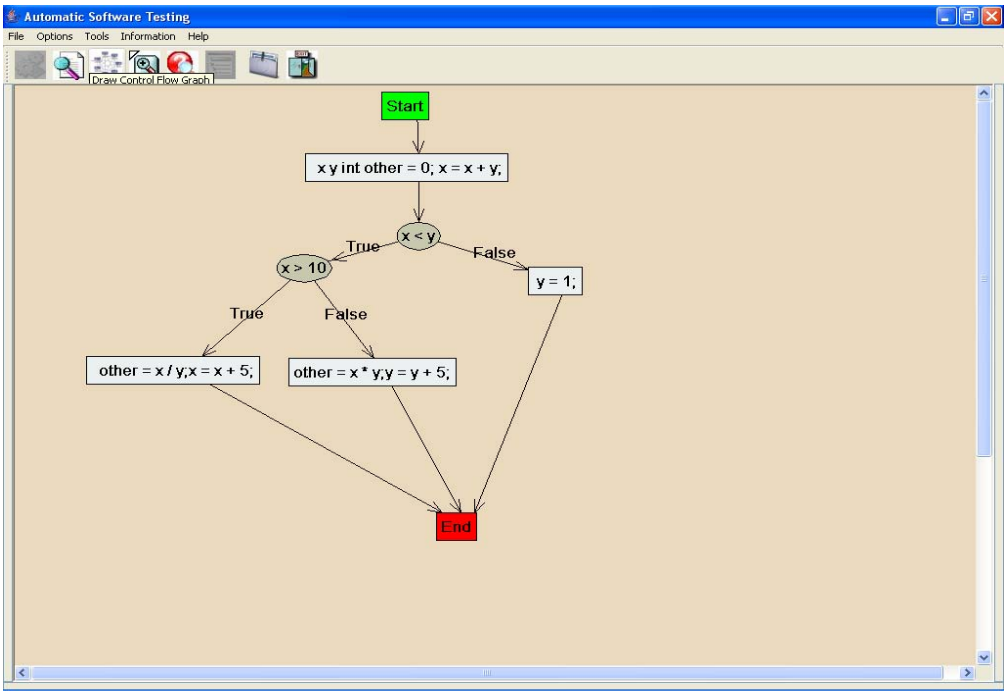
(Σχήμα 4: Παρουσίαση πηγαίου κώδικα)

- Μπορεί να δει διάφορες πληροφορίες για το πρόγραμμα που επέλεξε



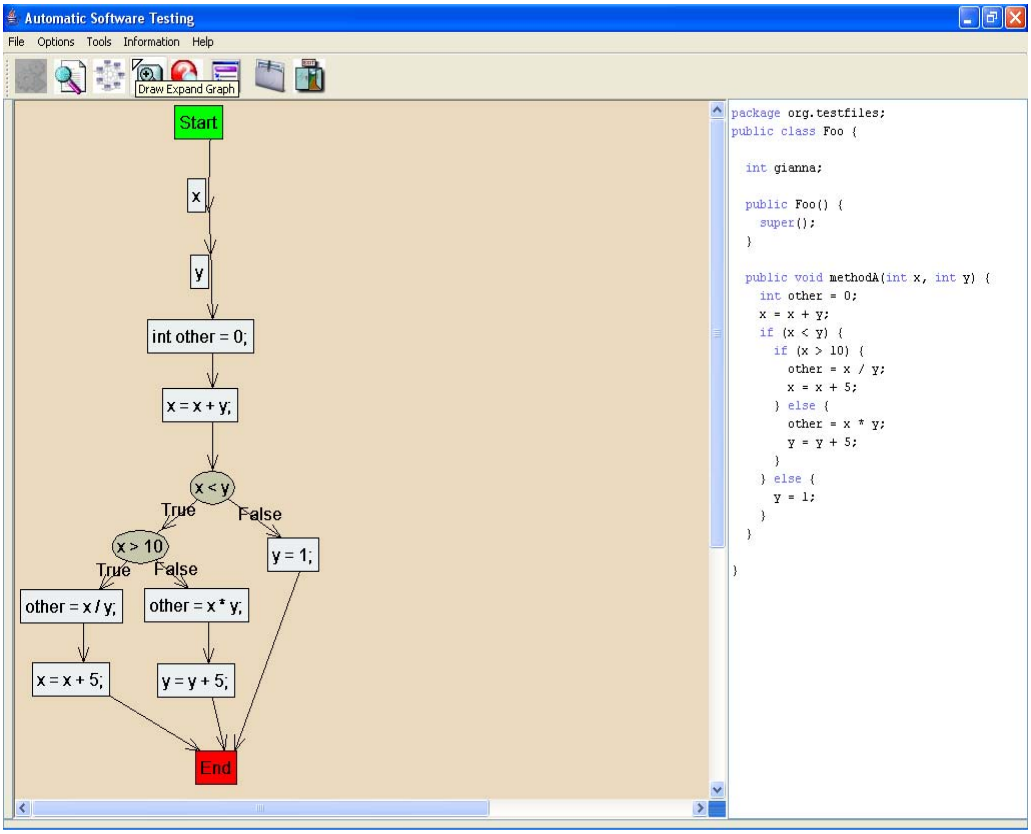
(Σχήμα 5: Γενικές πληροφορίες προγράμματος)

- Μπορεί να δημιουργήσει τον ομαδοποιημένο Γράφο Ροής Ελέγχου του προγράμματος



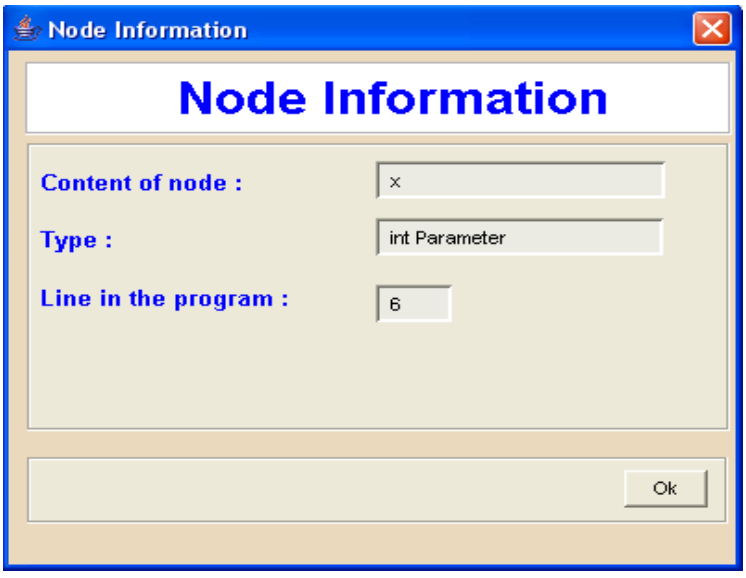
(Σχήμα 6: Παρουσίαση ομαδοποιημένου Γράφου Ροής Ελέγχου)

- Μπορεί να δημιουργήσει τον κανονικό Γράφο Ροής Ελέγχου του προγράμματος



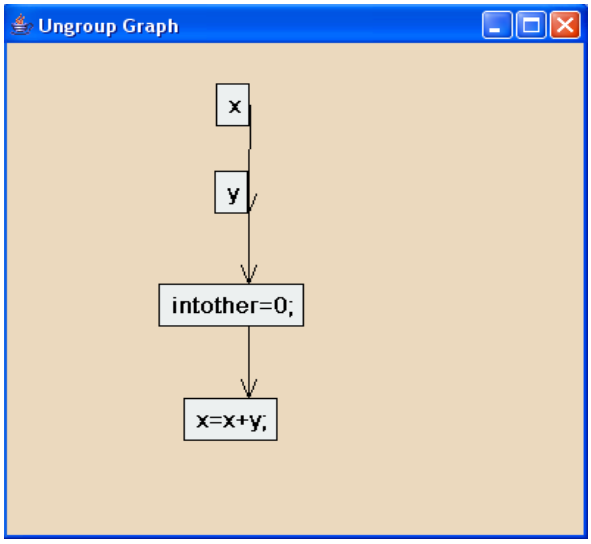
(Σχήμα 7: Παρουσίαση κανονικού Γράφου Ροής Ελέγχου,
με παράλληλη παρουσίαση πηγαίου κώδικα)

- Μπορεί να δει πληροφορίες για κάποιο συγκεκριμένο κόμβο του κανονικού Γράφου Ροής Ελέγχου



(Σχήμα 8: Παρουσίαση πληροφοριών κόμβου κανονικού γράφου)

- Μπορεί να δει τον υπογράφο που αντιστοιχεί στις εντολές ενός κόμβου του ομαδοποιημένου Γράφου Ροής Ελέγχου



(Σχήμα 9: Υπογράφος κόμβου ομαδοποιημένου γράφου)

Όταν ο χρήστης τελειώσει με την ανάλυση κάποιου προγράμματος, τότε μπορεί να αρχικοποιήσει το εργαλείο, επιλέγοντας RESET από την μπάρα εργαλείων και έτσι του δίνεται η δυνατότητα να αναλύσει ένα νέο πρόγραμμα ή να επιλέξει EXIT και να τερματίσει την λειτουργία του εργαλείου.

3. Παρουσίαση δουλειάς κ. Αναστάσιου Σοφοκλέους

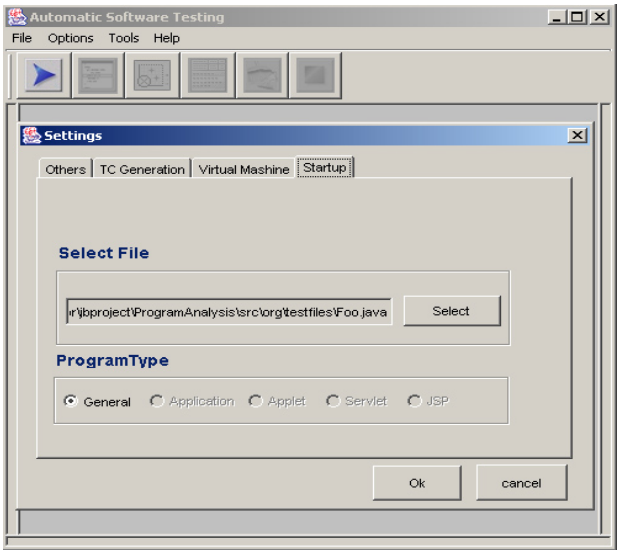
Η δουλειά πάνω στην οποία στηρίζεται η δουλειά της κ. Ιωακείμ, καθώς και η δική μου, αναπτύχθηκε από τον κ. Αναστάσιο Σοφοκλέους.

Η δουλειά του κ. Σοφοκλέους είχε σαν σκοπό την δημιουργία ενός περιβάλλοντος ελέγχου κώδικα λογισμικού με χρήση αλγορίθμων βελτιστοποίησης. Για να επιτευχθεί αυτός ο σκοπός, δημιουργήθηκε ο Γράφος Ροής Ελέγχου (για λίγες μόνο περιπτώσεις) και χρησιμοποιήθηκε γενετικός αλγόριθμος για παραγωγή δεδομένων δοκιμής για το πρόγραμμα.

Το περιβάλλον το οποίο δημιουργήθηκε, αποτελείται από μια γραφική διαπροσωπία, με την οποία ο χρήστης μπορεί να αλληλεπιδράσει με το εργαλείο και να εκτελέσει τις λειτουργίες τις οποίες του προσφέρει.

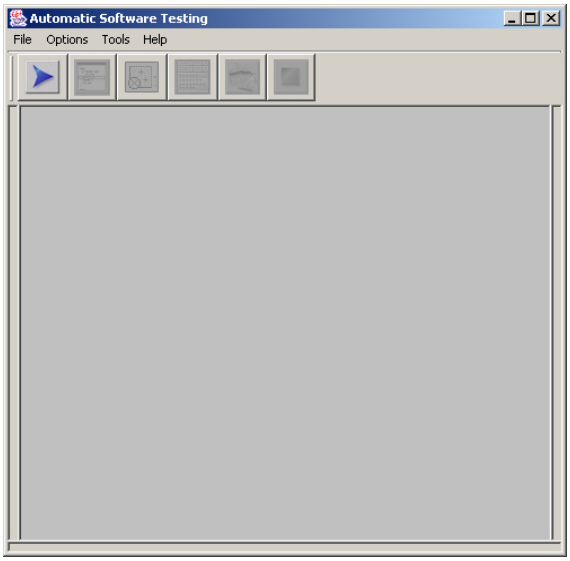
Όταν ο χρήστης τρέξει το εργαλείο, του ζητείται να καθορίσει μερικές βασικές παραμέτρους της εφαρμογής (σχήμα 10):

- Καθορίζει το είδος του προγράμματος προς ανάλυση
- Επιλέγει το πρόγραμμα που θα αναλυθεί
- Καθορίζει το χώρο που θα αποθηκευτούν τα δεδομένα εξόδου



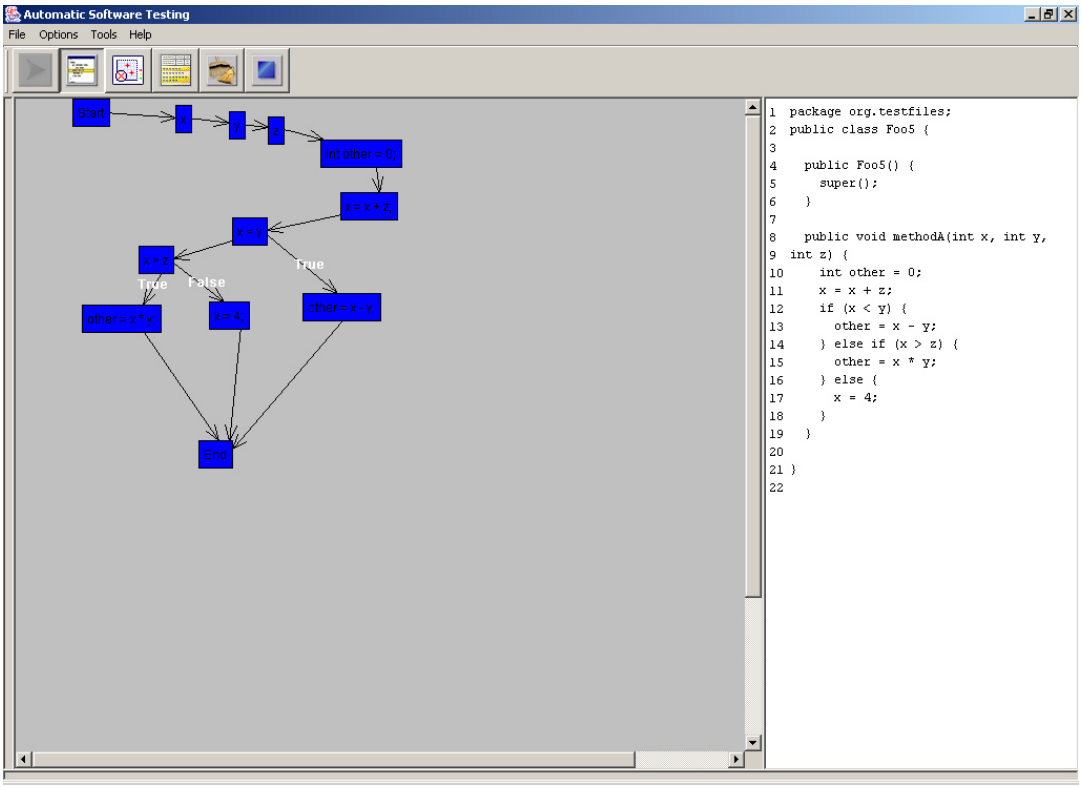
(Σχήμα 10: Οθόνη για καθορισμό παραμέτρων)

Όταν οι παράμετροι καθοριστούν, τότε ο χρήστη μπορεί να εκτελέσει την ανάλυση του προγράμματος μέσω της διεπιφάνειας του εργαλείου, πατώντας το ανάλογο κουμπί στην μπάρα επιλογών της κύριας οθόνης του εργαλείου.



(Σχήμα 11: Κύρια οθόνη εργαλείου)

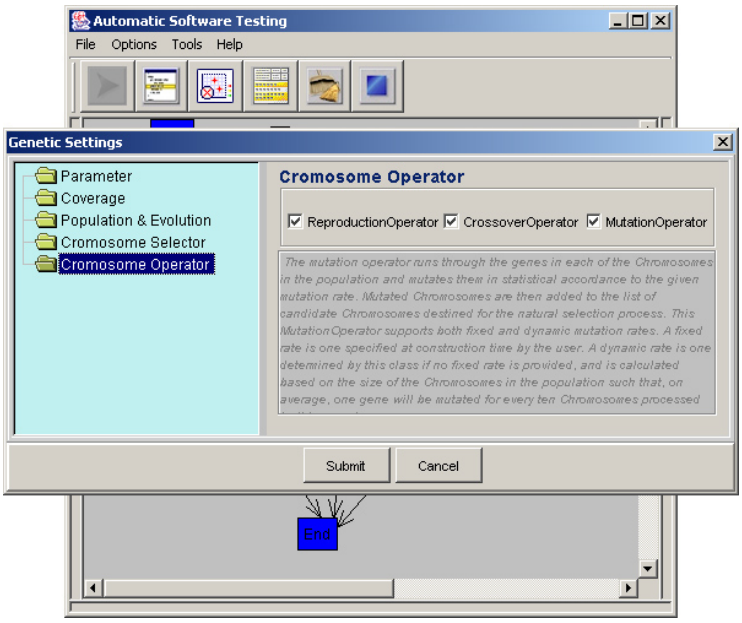
Όταν ο χρήστης πατήσει το κουμπί για την ανάλυση του προγράμματος, τότε μπορεί να δει τον Γράφο Ροής Ελέγχου που δημιουργήθηκε για το πρόγραμμα. Παράλληλα με τον γράφο μπορεί να δει και τον πηγαίο κώδικα του προγράμματος που αναλύθηκε στο δεξί μέρος της οθόνης (σχήμα 12).



(Σχήμα 12: Γράφος Ροής Ελέγχου και πηγαίος κώδικας προγράμματος)

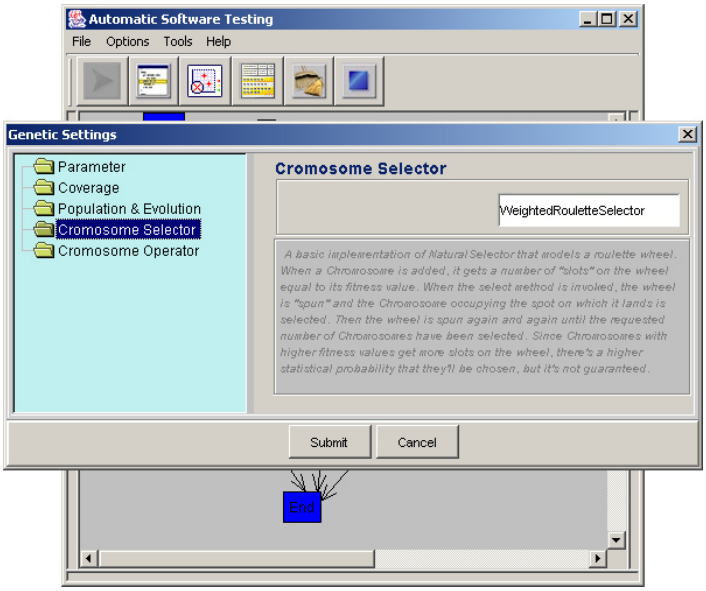
Ακολούθως, ο χρήστης πρέπει να καθορίσει τις παραμέτρους του γενετικού αλγορίθμου, ούτως ώστε να παραχθούν τα δεδομένα ελέγχου. Οι παράμετροι αυτές είναι:

- Chromosome operator: Υπάρχουν τρεις επιλογές που μπορεί να κάνει ο χρήστης (Crossover, Reproduction, Mutation).



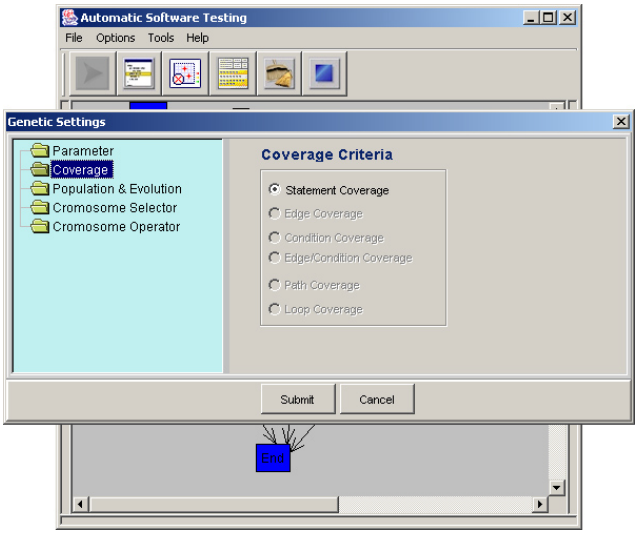
(Σχήμα 12: Επιλογή Chromosome operator)

- Chromosome selector: Υπάρχουν δύο επιλογές για τον χρήστη (Roulette selector, Roulette Selector and Elite Selection)



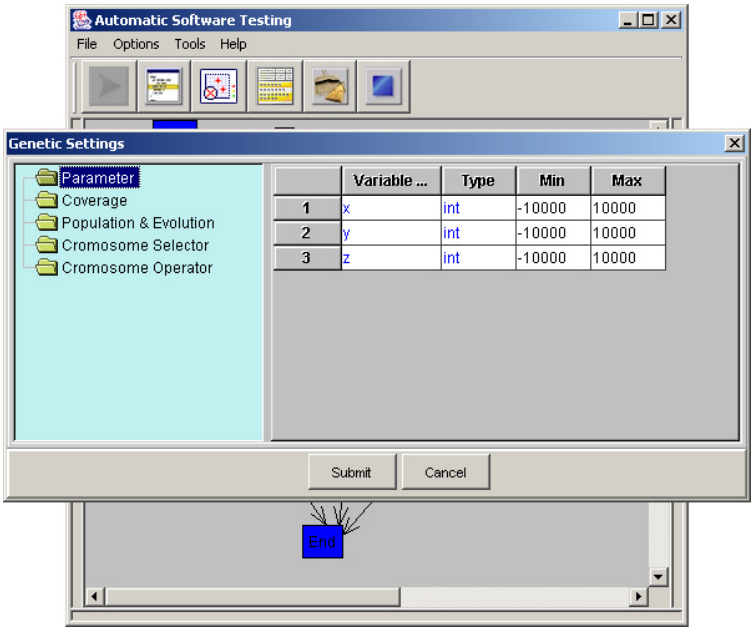
(Σχήμα 13: Επιλογή Chromosome selector)

- Coverage Criteria: Υπάρχουν έξι επιλογές για τον χρήστη (Statement Coverage, Edge Coverage, Condition Coverage, Edge/Condition Coverage, Multiple condition coverage, Path Coverage)



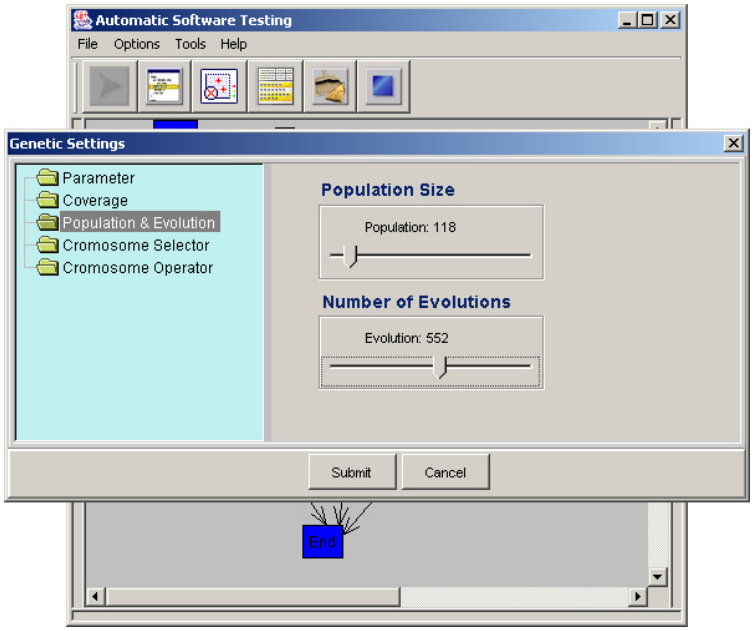
(Σχήμα 14: Επιλογή Coverage Criteria)

- Όρια παραμέτρων: Ο χρήστης μπορεί να καθορίσει το ανώτατο (max) και το κατώτατο (min) όριο των μεταβλητών εισόδου. Με αυτό τον τρόπο, αποφεύγεται η παραγωγή ανεπιθύμητων τιμών.



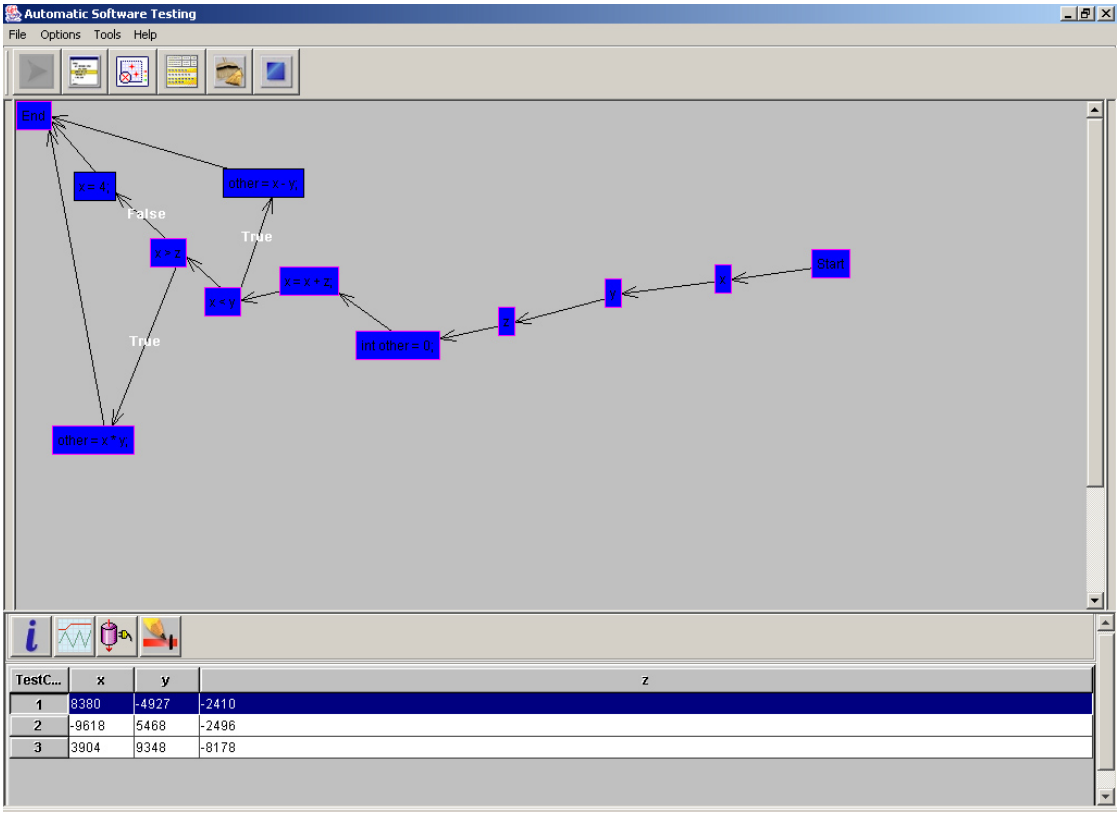
(Σχήμα 15: Καθορισμός ορίων παραμέτρων)

- Μέγεθος πληθυσμού και εξελίξεων: Η παραγωγή των test cases εξαρτάται από το μέγεθος του πληθυσμού και των εξελίξεων, καθώς αυτά τα μεγέθη παίζουν σημαντικό ρόλο στον γενετικό αλγόριθμο που χρησιμοποιεί το εργαλείο.



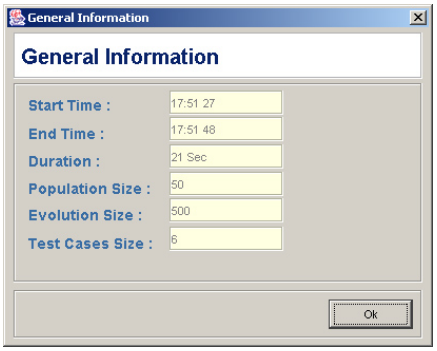
(Σχήμα 16: Καθορισμός μεγέθους πληθυσμών και εξελίξεων)

Όταν ο χρήστης καθορίσει τις παραπάνω παραμέτρους του εργαλείου, γίνεται η εκτέλεση του γενετικού αλγορίθμου για παραγωγή των test cases. Τότε το εργαλείο παρουσιάζει ένα σύνολο με τις πλειάδες των τιμών εισόδου που πρέπει να χρησιμοποιήσει ο χρήστης για τον έλεγχο του προγράμματος (test cases). Ακολούθως επιλέγει μια από αυτές τις πλειάδες και το εργαλείο παρουσιάζει το μονοπάτι που ακολουθείται στον Γράφο Ροής Ελέγχου.
Στο σχήμα 17 φαίνονται στο κάτω μέρος οι πλειάδες με τις τιμές εισόδου και στον γράφο, με χρωματισμένο περίγραμμα, το μονοπάτι το οποίο ακολουθείτε βάσει των επιλεγμένων τιμών εισόδου.



(Σχήμα 17: Παραγωγή test cases και παρουσίαση μονοπατιού στο γράφο)

Επίσης ο χρήστης έχει την δυνατότητα να δει διάφορες πληροφορίες (σχήμα 18) σχετικά με την εκτέλεση του γενετικού αλγορίθμου.



(Σχήμα 18: Πληροφορίες)

Κεφάλαιο 4

Εφαρμογή Αυτόματης Παραγωγής Γράφου Εξαρτήσεων

1. Εισαγωγή

Για την αυτόματη παραγωγή του Γράφου Εξαρτήσεων ενός προγράμματος, πρέπει πρώτα να παραχθεί ο Γράφος Ροής Ελέγχου του προγράμματος, ούτως ώστε να χρησιμοποιηθεί σαν είσοδος στο εργαλείο που αναπτύξαμε. Γι' αυτό τον σκοπό, γίνεται χρήση του Γράφου Ροής Ελέγχου που παράγεται από την εφαρμογή που υλοποίησε η κ. Ιωακείμ.

2. Αλγόριθμος παραγωγής Γράφου Εξαρτήσεων

Στο κεφάλαιο αυτό, θα περιγράψουμε με λίγα λόγια τον αλγόριθμο που χρησιμοποιήσαμε για να δημιουργήσουμε τον Γράφο Εξαρτήσεων κάποιου προγράμματος.

Αρχικά τρέχουμε το εργαλείο, επιλέγουμε το πρόγραμμα που θέλουμε να αναλύσουμε και παράγουμε τον Γράφο Ροής Ελέγχου του (γίνεται χρήση του εργαλείου της κ. Ιωακείμ). Ο γράφος που δημιουργείται, είναι αποθηκεμένος σε ένα αντικείμενο τύπου TGraph, μια δομή δεδομένων όπου μπορούμε να αποθηκέψουμε κάποιο γράφο. Ακολούθως, χρησιμοποιούμε τον γράφο που παράχθηκε σαν είσοδο στο εργαλείο που θα δημιουργήσει τον Γράφο Εξαρτήσεων του προγράμματος που αναλύεται.

Από το αντικείμενο του γράφου που πήραμε, παίρνουμε τις ακμές και τους κόμβους και τις βάζουμε σε δύο Enumerations, για να γνωρίζουμε τον αριθμό τους, ούτως ώστε να μπορούμε, ένα προς ένα, να μεταφέρουμε τους κόμβους σε ένα Vector για να τους επεξεργαστούμε.

```
  // vazw tous komvous tou grafou se emumeration
enumVertices = org.prganalysis.walker.ParseVisitor.graph.enumerateVertices();
  ////////////////////////////////////////////////
  // vazw tous komvou tou grafou se ena vector
  ////////////////////////////////////////////////
  int z= 0;
  while(enumVertices.hasMoreElements()){
    tempObject = enumVertices.nextElement();
    vertexVector.add(z,tempObject);
    ++ z;
  }
  ////////////////////////////////////////////////
```

(Σχήμα 1: Τοποθέτηση κόμβων σε Enumeration και ακολούθως σε Vector)

Όταν έχουμε όλους τους κόμβους του Γράφου Ροής Ελέγχου στον Vector, τους ελέγχουμε ένα προς ένα και ανάλογα με τον τύπο του κάθε κόμβου, τον τοποθετούμε ανάλογα στον γράφο που δημιουργούμε, ο οποίος είναι αποθηκεμένος σε αντικείμενο τύπου TGraph.

Υπάρχουν οκτώ (8) τύποι κόμβων στον γράφο μας:

- Start/End: Στην περίπτωση του κόμβου που περιέχει το Start, ο αρχικός κόμβος του γράφου μας, τοποθετούμε απλώς τον κόμβο στον γράφο, αφού δεν υπάρχει κάποιος προηγούμενος κόμβος.

```
            ////////////////////////
            // START OR END
            ////////////////////////
            if(typeOfVertex == ""){
              // START
              if(tempVertex.getName() == "Start"){

                // vazw ston Dependence graph to start
                graph3.add(tempVertex);
              }
              // END
              if(tempVertex.getName() == "End"){
               // do nothing - den mpainei ston grafo
              }
            }
```

(Σχήμα 3: Περίπτωση κόμβου Start/End)

- Parameter: Σε περίπτωση κόμβου που περιέχει παράμετρο, τοποθετούμε τον κόμβο στον γράφο μας και τον ενώνουμε με τον αρχικό μας κόμβο, αφού οι παράμετροι δεν εξαρτώνται πάνω σε καμία εντολή ή δήλωση του προγράμματος. Επίσης αποθηκεύουμε την θέση στην οποία βρίσκεται ο κόμβος με την παράμετρο μας σε ένα πίνακα, ούτως ώστε να μπορούμε να τοποθετήσουμε αργότερα τις summary edges πάνω στους κόμβους που τις χρησιμοποιούν.

```
// elegxw ti typou einai o komvos
//////////////////////////////////////////////////////////////////////
// Parameter
if(typeOfVertex.endsWith(" Parameter ")){
  System.out.println(" Einai Parameter "); // test

  // vazw to onoma tis parametrou se ena pinaka gia tis parametrous
  name[thesi_parametrou] = tempVertex.getName();

  System.out.println("Parameter Name: " + name[thesi_parametrou]
                     + " AT " + thesi_parametrou);
  ++ thesi_parametrou; // au3anw ton metriti gia tis parametrous

  // vazw ston Dependence graph ton konvo

  graph3.add(tempVertex);
  // vazw akmi meta3i tou start kai tou komvou parametrou
  graph3.addEdge((TVertex)(graph3.vertexAt(0)),tempVertex);

}
//////////////////////////////////////////////////////////////////////
```

(Σχήμα 4: Περίπτωση κόμβου παραμέτρου)

- Variable declaration: Όταν έχουμε κόμβο που περιέχει την δήλωση κάποιας μεταβλητής (variable declaration), τοποθετούμε τον κόμβο στον γράφο και τον ενώνουμε με τον κατάλληλο κόμβο, ανάλογα με την σχέση του με τους προηγούμενους κόμβους. Παράλληλα αποθηκεύουμε την θέση του σε ένα πίνακα, ούτως ώστε να μπορούμε να τοποθετήσουμε αργότερα τις data dependence edges προς τους κόμβους που χρησιμοποιούν την μεταβλητή μας.

```
////////////////////////////////////////////////////////////////////
// Variable Declaration
if(typeOfVertex == " Variable Declaration "){
  System.out.println(" Einai Variable Declaration "); // test

  tempname = tempVertex.getName();
  String tempString = new String();
  String tempString2 = new String();
  StringTokenizer strToken = new StringTokenizer(tempname);
  int helpvar = 0;
  while(strToken.hasMoreElements()){
    ++helpvar;
    tempString = strToken.nextToken();
    System.out.println("TOKEN: " + tempString);
    if(helpvar == 2){
      tempString2 = tempString;
    }
  }

  // vazw to onoma tis metvlitis se ena pinaka gia tis metavlites
  nameMetavlitis[thesi_metavlitis] = tempString2;
  // vazw ton ari8mo tou komvou se ena pinaka
  positionOfVar[thesi_metavlitis] = position;
```

(Σχήμα 5: Ενδεικτικός κώδικας περίπτωσης κόμβου δήλωσης μεταβλητής)

- Expression statement: Σε περίπτωση expression statement, τοποθετούμε τον κόμβο στον γράφο μας και τον ενώνουμε με τον κατάλληλο κόμβο, ανάλογα με την σχέση του με τους προηγούμενους. Ακολούθως ελέγχουμε αν χρησιμοποιεί κάποια παράμετρο ή μεταβλητή και τοποθετούμε, αν χρειάζεται, τις summary edges και τις data dependence edges.

```
//////////////////////////////////////////////////////////////////
// Expression Statement
if(typeOfVertex == " Expression Statement "){
  System.out.println(" Einai Expression Statement ");  // test

  // vazw ston Dependence graph ton konvo
  graph3.add(tempVertex);
  // vazw akmi meta3i tou start kai tou komvou
  if (outoffor == 1) { // an vgainw apo for

    graph3.addEdge((TVertex) (graph3.vertexAt(0)), tempVertex);
    System.out.println(" OUT OF FOR ");
    outoffor = 0;
  }
```

(Σχήμα 6: Ενδεικτικός κώδικας περίπτωσης κόμβου Expression statement)

- Unary Expression: Τοποθετούμε τον κόμβο στον γράφο και, όπως και στις προηγούμενες περιπτώσεις, ενώνουμε τον κόμβο ανάλογα με την σχέση του με τους προηγούμενους. Ακολούθως, αν χρειάζεται, τοποθετούμε summary edges και data dependence edges

```
//////////////////////////////////////////////////////////////////.
// Unary Operation ///////////////////////////////////////////////.
if(typeOfVertex == " Unary Operation "){


  // vazw ston Dependence graph ton konvo
  graph3.add(tempVertex);
  // vazw akmi meta3i tou start kai tou komvou
  if(outoffor == 1){ // an vgainw apo for
    graph3.addEdge((TVertex)(graph3.vertexAt(positionFor)),
                   tempVertex).addElement("False");
  }
  else{
    graph3.addEdge((TVertex)(graph3.vertexAt(0)),tempVertex);
  }
```

(Σχήμα 7: Ενδεικτικός κώδικας περίπτωσης κόμβου Unary Expression)

- Expression statement (While Condition, For Condition, If Condition): Σε περίπτωση που έχουμε μία από αυτές τις περιπτώσεις, αρχικά τοποθετούμε τον κόμβο στον γράφο και τον ενώνουμε ανάλογα με την σχέση του με τους προηγούμενους. Ακολούθως, αλλάζουμε την τιμή της ανάλογης μεταβλητής

σημαίας (whileFlag, forFlag, ifFlag) για να δείξουμε σε ποια κατάσταση προγράμματος θα πάμε (μέσα σε while, for, if)

```
//////////////////////////////////////////////////////
// Expression Statement : If_ Condition ////////////////////
if(typeOfVertex == " Expression Statement : If_ Condition "){

  ifFlag = 1; // eimaste mesa se IF
  ++numberofif; // gia enwsi epomenwn if
  positionIf = position; // thesi if

  // vazw ston Dependence graph ton konvo
  graph3.add(tempVertex);
  //new code
  graph3.addEdge((TVertex) (graph3.vertexAt(0)), tempVertex);
  //////////////////////////////////////////////////////
}
```

(Σχήμα 8: Ενδεικτικός κώδικας περίπτωσης κόμβου Expression statement - If Condition)

Κατά τον έλεγχο του τύπου του κάθε κόμβου, το πρόγραμμα διαχωρίζεται σε τέσσερα (4) μέρη, ανάλογα με την κατάσταση στην οποία βρισκόμαστε σε σχέση με το πρόγραμμα που αναλύουμε.

- Βρισκόμαστε σε κανονική Ροή Προγράμματος
- Βρισκόμαστε μέσα σε IF statement
- Βρισκόμαστε μέσα σε βρόγχο FOR
- Βρισκόμαστε μέσα σε βρόγχο WHILE

Ανάλογα σε ποια από τις προαναφερθείσες καταστάσεις βρισκόμαστε, γίνεται και η κατάλληλη τοποθέτηση των κόμβων στον Γράφο Εξαρτήσεων που δημιουργούμε. Όταν ο κόμβος, στον οποίο βρισκόμαστε, ενσωματωθεί στον γράφο μας, τότε γίνεται συνένωση του με τους ήδη υπάρχοντες κόμβους, πάντα σε σχέση με την κατάσταση στην οποία βρισκόμαστε (σχήμα 9).

```
// vazw akmi meta3i tou komvou  kai tou proigoumenou komvou
/// new code
if(nestedIf == 1){
  if(firstNode == 1){
    graph3.addEdge((TVertex) (graph3.vertexAt(positionIf))
                   ,tempVertex).addElement("True");
    firstNode = 0;
  }
  else{
    graph3.addEdge((TVertex)(graph3.vertexAt(nestedIfPosition))
                   ,tempVertex).addElement("True");
  }
}
else{
  if (nestedFor == 1) {
    if (firstNode == 1) {
      graph3.addEdge((TVertex) (graph3.vertexAt(positionIf))
                     ,tempVertex).addElement("True");
      firstNode = 0;
    }
    else {
  graph3.addEdge((TVertex)(graph3.vertexAt(nestedForPosition))
                 , tempVertex).addElement("True");
    }
  }
  else {
    graph3.addEdge((TVertex) (graph3.vertexAt(positionIf))
                   ,tempVertex).addElement("True");
  }
}
```

(Σχήμα 9: Ενδεικτικός κώδικας συνένωσης νέου κόμβου με προηγούμενους)

Μόλις ο νέος κόμβος μας συνενωθεί με τους προηγούμενους, τοποθετούμε τυχόν summary edges (ακμές που ενώνουν τον κόμβο με τις παραμέτρους που χρησιμοποιεί) ή data dependence edges (ακμές που ενώνουν τον κόμβο με τις μεταβλητές που χρησιμοποιεί).

```
//////////////////////////////////////////////////////
// gia eisagwgi summary edge
tempname = tempVertex.getName();
String tempString = new String();
StringTokenizer strToken = new StringTokenizer(tempname);
int helpvar = 0;
while(strToken.hasMoreElements()){
  tempString = strToken.nextToken();
  for(int i=0;i<name.length;++i){
    if(((helpvar=tempString.compareTo(name[i]))== 0)||
                ((tempString.compareTo(name[i]+";"))== 0)){

      //eisagwgi summary edge
      graph3.addEdge((TVertex)(graph3.vertexAt(i+1)),tempVertex,1);
    }
  }
}
// telos eisagwgis summary edge
//////////////////////////////////////////////////////\
```

(Σχήμα 10: Ενδεικτικός κώδικας για εισαγωγή summary edge)

Αφού γίνει η εισαγωγή των νέων ακμών, προχωρούμε στον επόμενο κόμβο του Γράφου Ροής Ελέγχου, τον οποίο έχουμε αποθηκεμένο στον vector μας, μέχρι να τοποθετήσουμε όλους τους κόμβους στον Γράφο Εξαρτήσεων που δημιουργούμε (εκτός του κόμβου END (Exit Vertex)). Όταν τελειώσει η εισαγωγή των κόμβων, τότε ο γράφος μας είναι έτοιμος και τον παρουσιάζουμε στην οθόνη του εργαλείου.

Κεφάλαιο 5

Παρουσίαση εργαλείου

1. Εισαγωγή

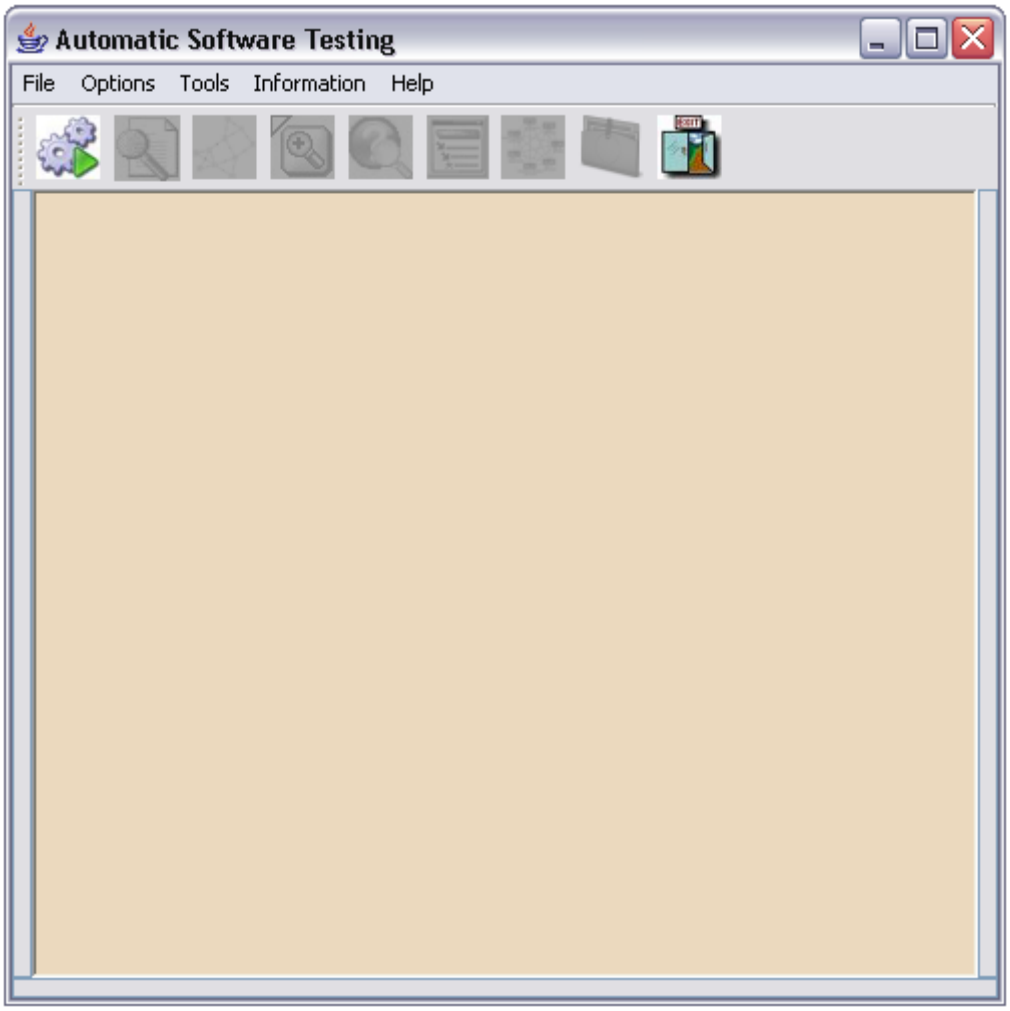
Στο κεφάλαιο αυτό γίνεται, αρχικά, μια λεπτομερής περιγραφή της χρήσης του εργαλείου, με ιδιαίτερη έμφαση στο πως ο χρήστης μπορεί να εκτελέσει κάθε λειτουργία του εργαλείου. Ακολουθεί ένα παράδειγμα ανάλυσης προγράμματος καθώς και μετρήσεις για τον χρόνο εκτέλεσης για την ανάλυση διαφόρων προγραμμάτων.

Ο τρόπος με τον οποίο κάποιος χρήστης μπορεί να χρησιμοποιήσει το εργαλείο είναι απλός, ούτως ώστε η χρήση του να είναι εύκολη για όλο το φάσμα χρηστών, από τον απλό χρήστη, ο οποίος έχει απλώς τις βασικές γνώσεις πληροφορικής μέχρι και τον έμπειρο χρήστη με εξειδικευμένες γνώσεις.

Ακολουθεί μια εκτενής παρουσίαση της λειτουργίας του εργαλείου, μέσω της περιγραφής των διαφόρων οθόνων του.

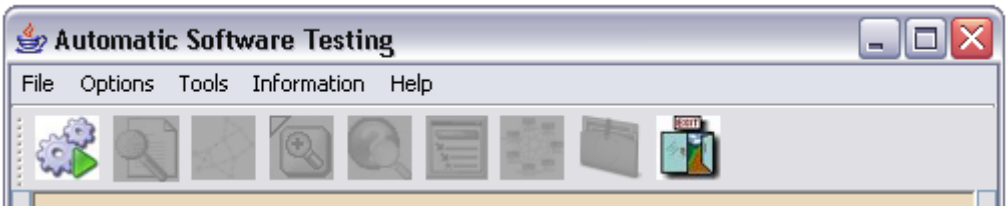
2. Περιγραφή οθόνων – Λειτουργία

Σε αυτό το κομμάτι του κεφαλαίου, γίνεται μια εκτενής περιγραφή των οθόνων του εργαλείου, παρουσιάζοντας παράλληλα και ον τρόπο που ο χρήστης μπορεί να εκτελέσει τις διάφορες λειτουργίες του εργαλείου.



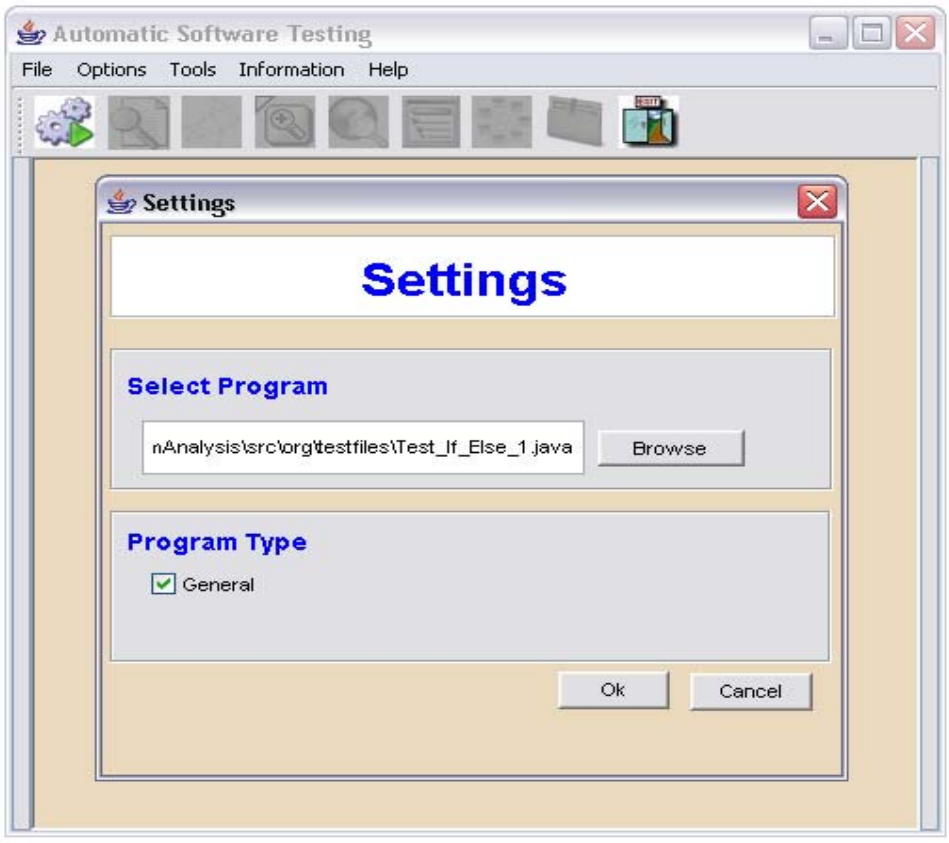
(Σχήμα 1: Κύρια οθόνη εργαλείου)

Αυτή είναι η κύρια οθόνη του εργαλείου. Αποτελείται από μια μπάρα εργαλείων (toolbar) και ένα μενού επιλογών (menu bar) (σχήμα 2), οι οποίες περιλαμβάνουν όλες τις λειτουργίες που παρέχει το εργαλείο, και από μια κενή κεντρική περιοχή, στην οποία θα παρουσιάζονται τα αποτελέσματα της εκτέλεσης του προγράμματος.



(Σχήμα 2: Μπάρα εργαλείων και μενού επιλογών)

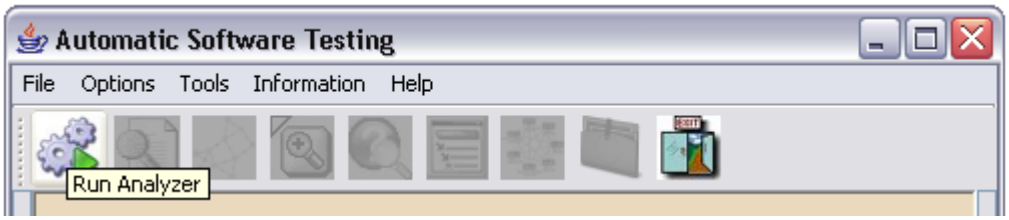
Για να εκτελεστεί κάποια από τις λειτουργίες του προγράμματος, πρέπει ο χρήστης πατά το ανάλογο κουμπί στην μπάρα εργαλείων ή την ανάλογη επιλογή από το μενού. Ακολούθως, θα εμφανιστεί το αποτέλεσμα της λειτουργίας στην οθόνη του χρήστη. Όπως φαίνεται και στο σχήμα 1, στην αρχική κατάσταση της εφαρμογής δύο λειτουργίες είναι ενεργές. Η εκτέλεση της ανάλυσης του προγράμματος και η έξοδος από την εφαρμογή. Για να γίνει η ανάλυση κάποιου προγράμματος, ο χρήστης πρέπει να επιλέξει το πρόγραμμα το οποίο θέλει να αναλύσει. Από το μενού επιλογών, επιλέγει "Options " και ακολούθως "General Settings". Τότε ανοίγει η οθόνη για τις επιλογές (settings), ο χρήστης επιλέγει το πρόγραμμα που θέλει να αναλύσει και πατά το "OK" (σχήμα 3).



(Σχήμα 3: Επιλογή προγράμματος για ανάλυση)

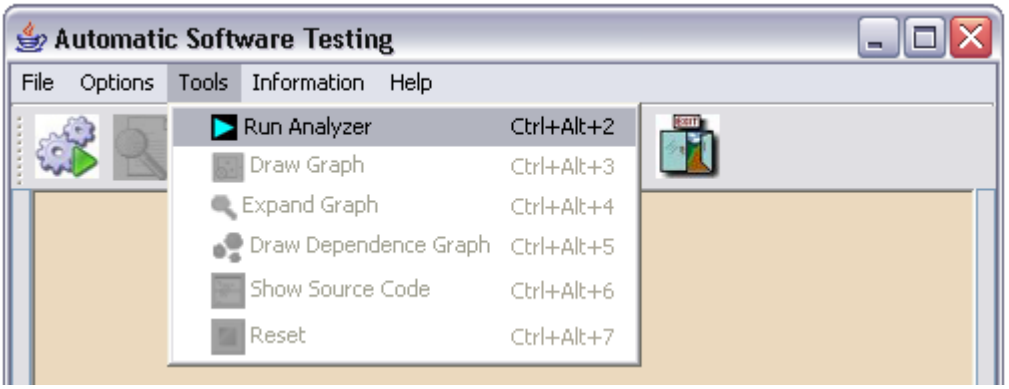
Όταν ο χρήστης επιλέξει το πρόγραμμα που θέλει να αναλύσει, τρέχει το εργαλείο με ένα από τους ακόλουθους τρόπους:

- Πατάει το κουμπί “Run Analyzer” στην μπάρα εργαλείων (1^{ο} κουμπί)



(Σχήμα 4)

- Επιλέγει από το μενού επιλογών την επιλογή “Run Analyzer” από το μενού “Tools”



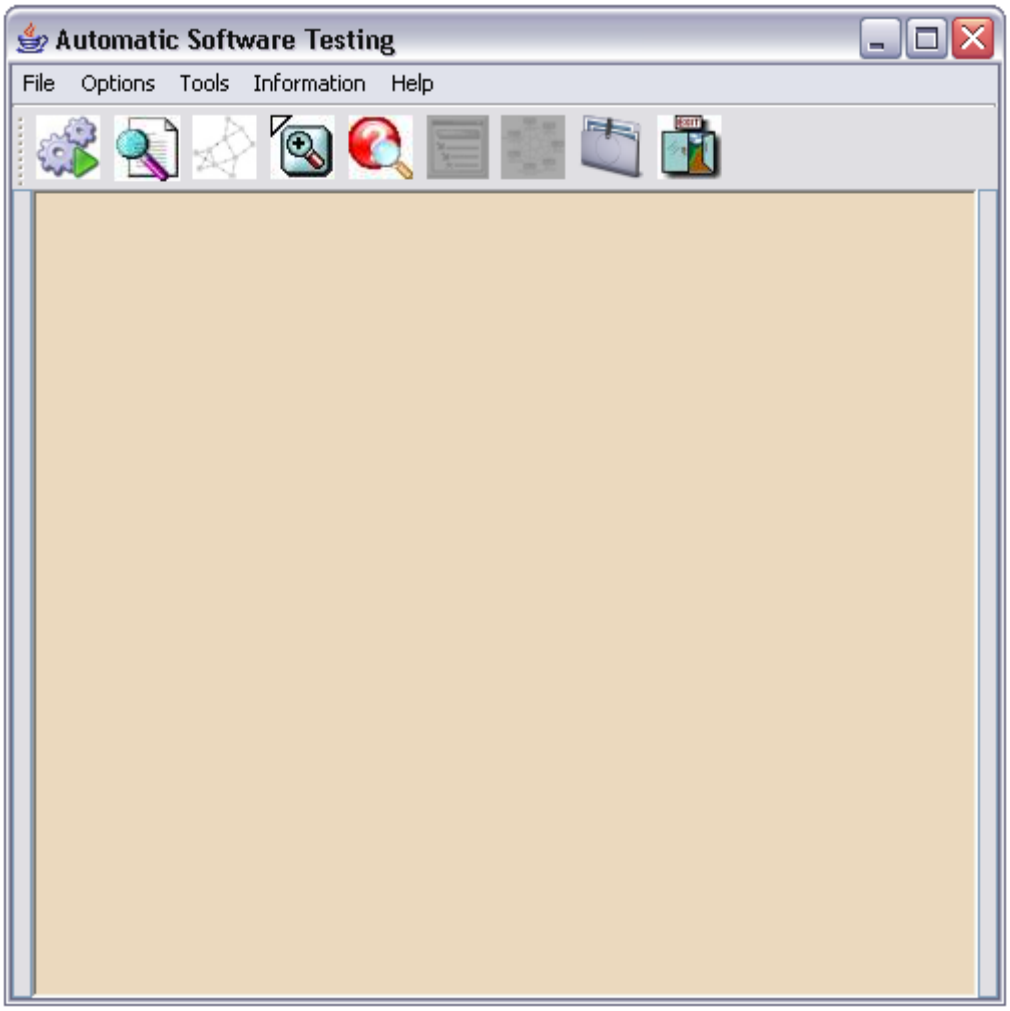
(Σχήμα 5)

- Χρησιμοποιεί τον συνδυασμό των πλήκτρων “CTRL + ALT + 2”

Μόλις ο χρήστης τρέξει τον αναλυτή, εμφανίζεται ένα μήνυμα που λέει ότι η διαδικασία ανάλυσης ολοκληρώθηκε με επιτυχία (σχήμα 6).

(Σχήμα 6: Μήνυμα επιτυχίας ανάλυσης)

Όταν ο χρήστης πατήσει το “OK” στο μήνυμα που εμφανίστηκε, ενεργοποιούνται οι υπόλοιπες λειτουργίες του εργαλείου, εκτός από την επιλογή για εμφάνιση πληροφοριών κάποιου κόμβου και την επιλογή για δημιουργία του Γράφου Εξαρτήσεων (Dependence Graph), έτσι η κύρια οθόνη του εργαλείου εμφανίζεται όπως παρακάτω (σχήμα 7), δίνοντας την δυνατότητα στον χρήστη να χρησιμοποιήσει και τις υπόλοιπες λειτουργίες που του προσφέρει το εργαλείου.



(Σχήμα 7: Κύρια οθόνη μετά την ανάλυση του προγράμματος)

Οι λειτουργίες που ενεργοποιούνται και που ο χρήστης μπορεί να εκτελέσει τώρα από την κύρια οθόνη είναι:

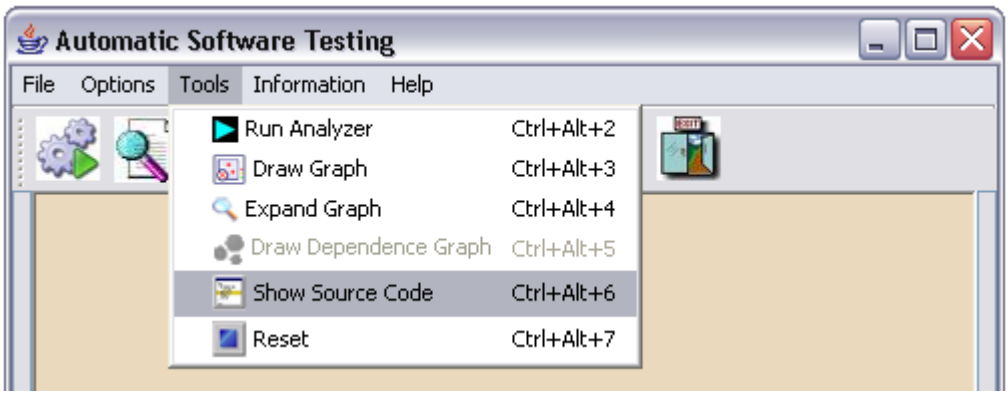
A. Εμφάνιση του κώδικα του προγράμματος που επιλέχθηκε για ανάλυση

Ο χρήστης μπορεί να δει τον κώδικα του επιλεγμένου προγράμματος επιλέγοντας ένα από τους ακόλουθους τρόπους:

1. Μέσω της μπάρας εργαλείων, πατώντας το κουμπί “Show Source Code” ($2^{ο}$ κουμπί)

(Σχήμα 8)

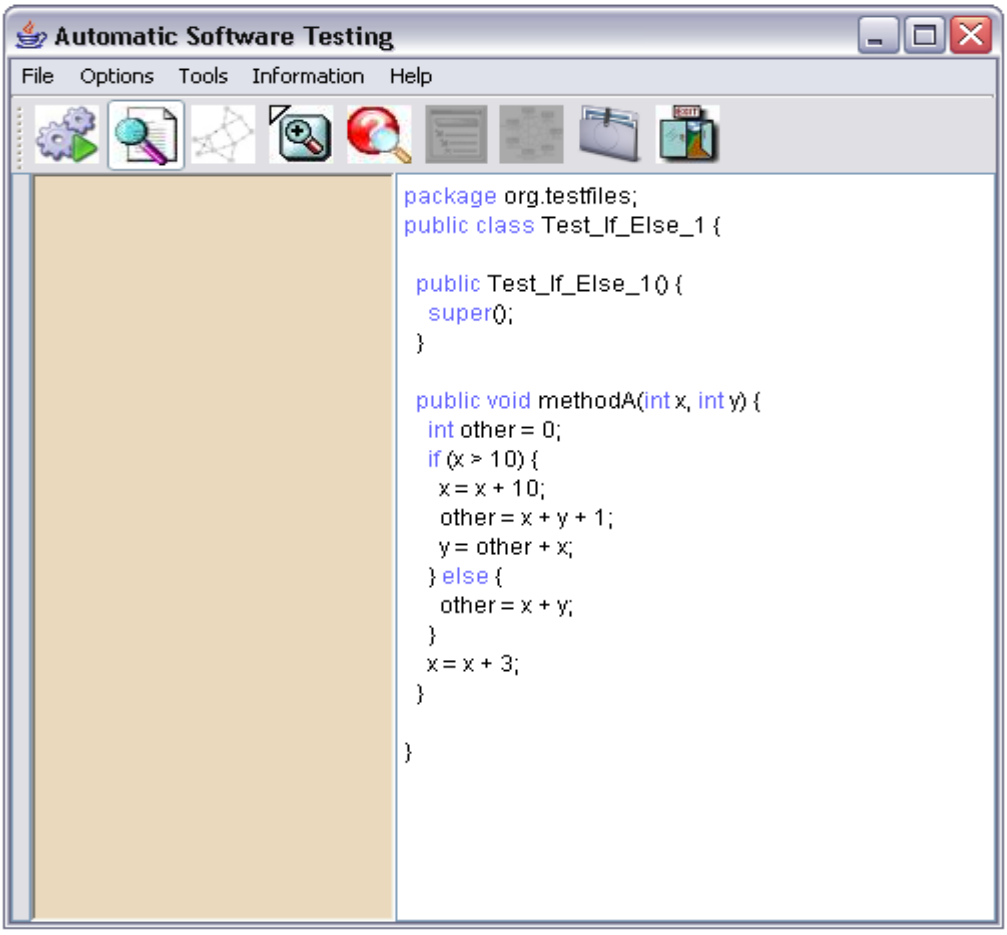
2. Μέσα από το μενού επιλογών, επιλέγει “Show Source Code” από το μενού “Tools”



(Σχήμα 9)

3. Χρησιμοποιεί τον συνδυασμό των πλήκτρων “CTRL + ALT + 6”

Όταν ο χρήστης χρησιμοποιήσει ένα από τους προηγούμενους τρόπους, εμφανίζεται στο δεξιό μισό της οθόνης ο κώδικα του προγράμματος που επιλέχθηκε (σχήμα 10).



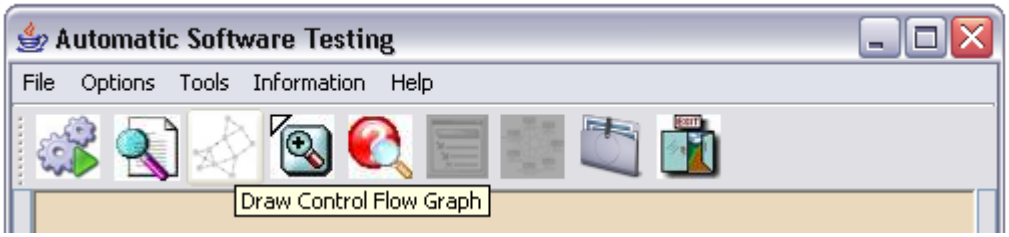
(Σχήμα 10: Εμφάνιση πηγαίου κώδικα προγράμματος)

Για να επανέλθει η οθόνη στην προηγούμενη κατάσταση της, χωρίς το μισό κομμάτι με τον κώδικα του προγράμματος, ο χρήστης χρησιμοποιεί ένα από τους προαναφερθείσες τρόπους και ο κώδικας εξαφανίζεται.

Β. Εμφάνιση ομαδοποιημένου Γράφου Ροής Ελέγχου

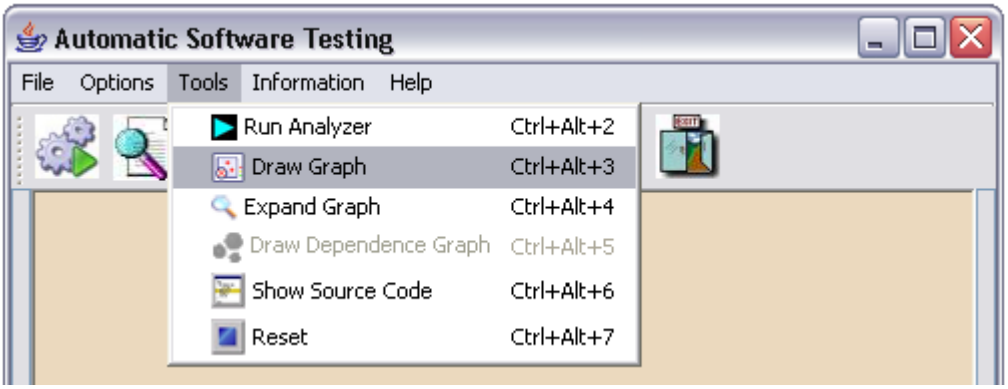
Για να δει ο χρήστης τον ομαδοποιημένο Γράφο Ροής που δημιουργεί το εργαλείο μας, μπορεί να χρησιμοποιήσει ένα από τους τρεις ακόλουθους τρόπους:

1. Στην μπάρα εργαλείων, πατά το κουμπί “Draw Control Flow Graph” ($3^{ο}$ κουμπί)



(Σχήμα 11)

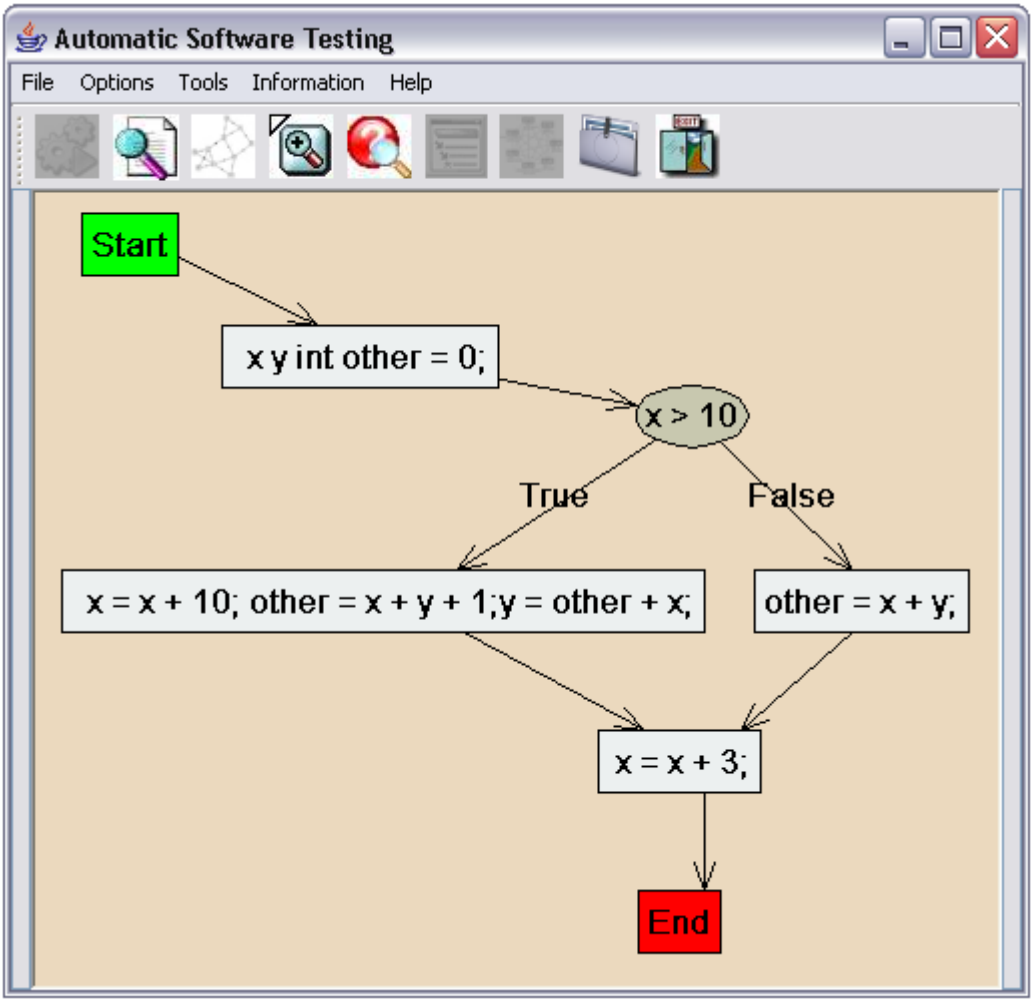
2. Από το μενού επιλογών, επιλέγει “Draw Graph” από το μενού “Tools”



(Σχήμα 12)

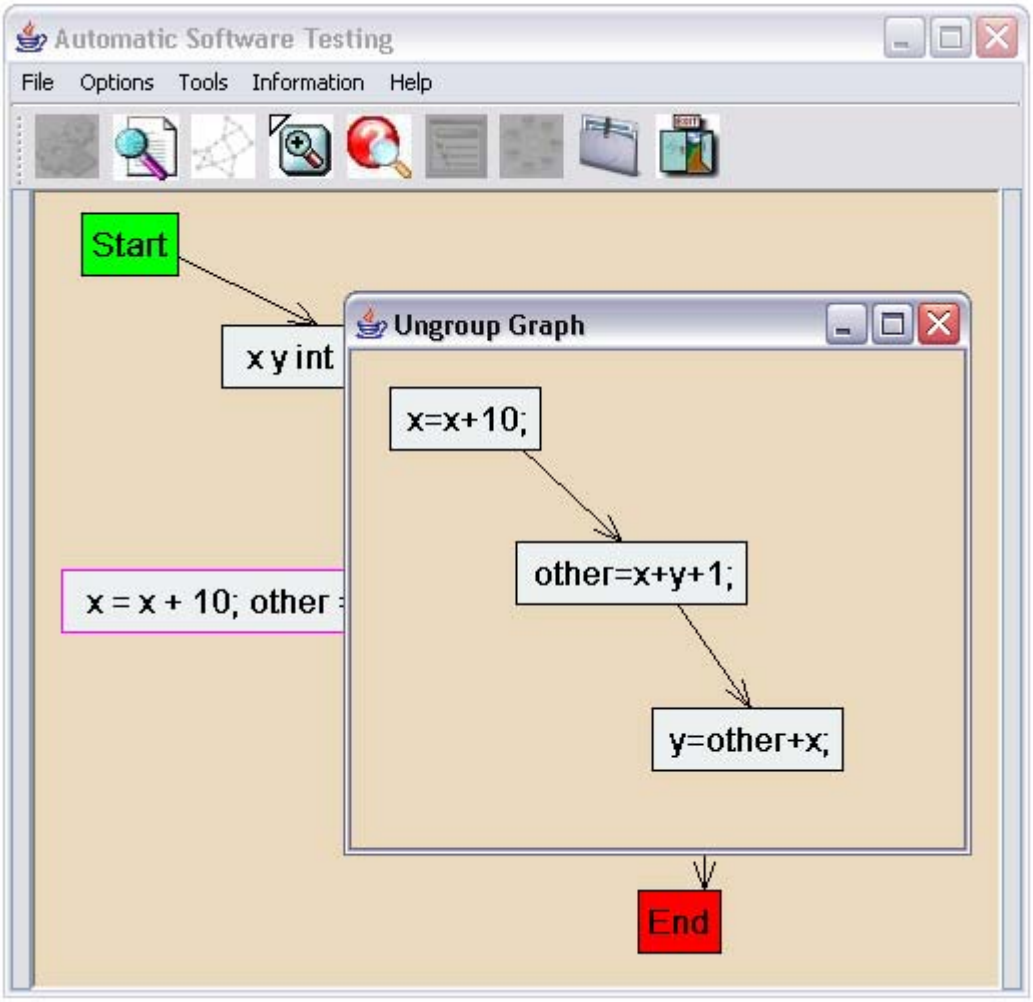
3. Χρησιμοποιεί τον συνδυασμό των πλήκτρων “CTRL + ALT + 3”

Όταν ο χρήστης επιλέξει ένα από τους παραπάνω τρόπους, τότε εμφανίζεται στην οθόνη ο ομαδοποιημένος Γράφος Ροής Ελέγχου που δημιουργήθηκε από το εργαλείο (σχήμα 13).



(Σχήμα 13: Ομαδοποιημένος Γράφος Ροής Ελέγχου)

Στην οθόνη όπου παρουσιάζεται ο ομαδοποιημένος Γράφος Ροής Ελέγχου, ο χρήστης μπορεί να δει λεπτομέρειες για τα περιεχόμενα του κάθε κόμβου (παρουσίαση των περιεχομένων του κόμβου χωρίς να είναι ομαδοποιημένα) χρησιμοποιώντας τον συνδυασμό "CTRL + SHIFT + Left mouse click". Όταν χρησιμοποιήσει τον συνδυασμό αυτό, τότε εμφανίζεται μια οθόνη με τους κόμβους που περιέχονται σε ένα κόμβο του ομαδοποιημένου Γράφου Ροής Ελέγχου (σχήμα 14). Στην εικόνα, ο επιλεγμένος κόμβος φαίνεται με κόκκινο περίγραμμα, σε αντίθεση με τους μη επιλεγμένους κόμβους, οι οποίοι έχουν μαύρο περίγραμμα.

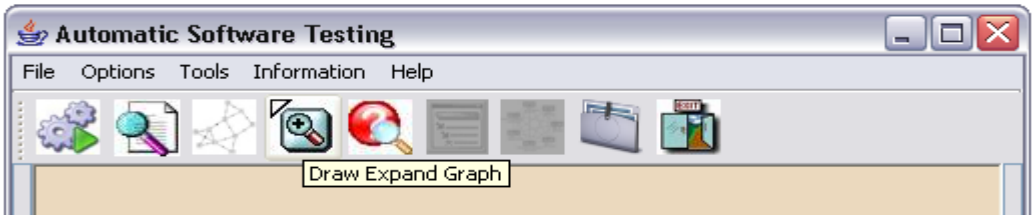


(Σχήμα 14: Υπογράφος επιλεγμένου κόμβου)

C. Εμφάνιση κανονικού Γράφου Ροής Ελέγχου (Expanded Control Flow Graph)

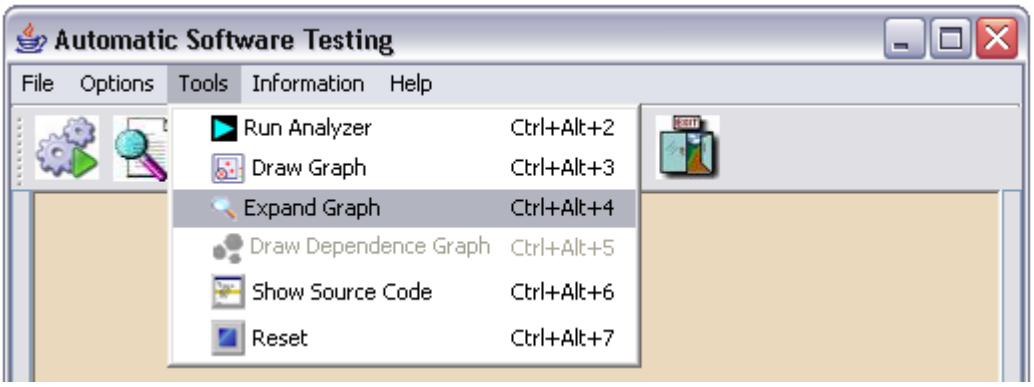
Για να μπορέσει ο χρήστης να δει τον κανονικό Γράφο Ροής Ελέγχου, επιλέγει ένα από τους παρακάτω τρεις τρόπους:

1. Πατάει το κουμπί “Draw Expand Graph” στην μπάρα εργαλείων (4^{ο} κουμπί)



(Σχήμα 15)

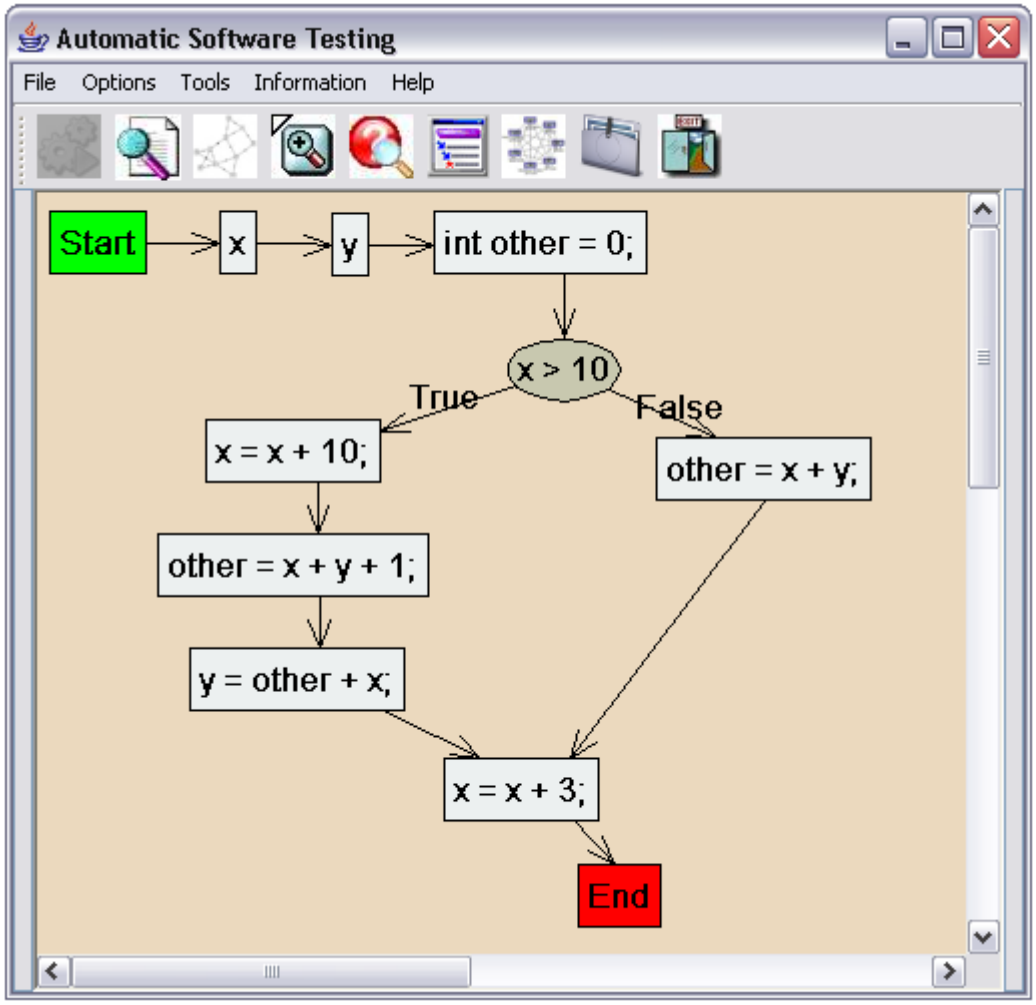
2. Από το μενού επιλογών, επιλέγει “Expand Graph” από το μενού “Tools”



(Σχήμα 16)

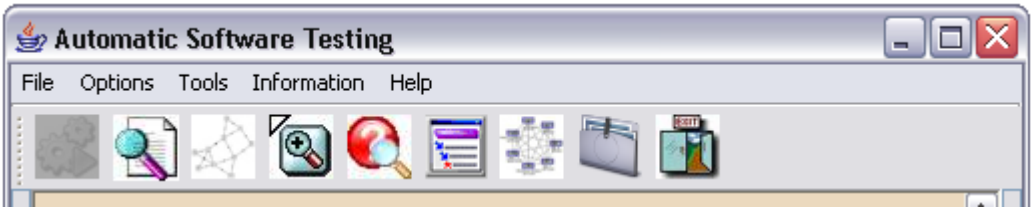
3. Χρησιμοποιεί τον συνδυασμό πλήκτρων “CTRL + ALT + 4”

Ο χρήστης μπορεί να δει τον κανονικό Γράφο Ροής Ελέγχου από την αρχική οθόνη, αμέσως μετά που πατάει το “OK” (σχήμα 6)στην αρχή της εκτέλεσης του προγράμματος (μετά από την επιλογή “Run Analyzer” (σχήμα 4)). Επίσης, μπορεί να τον δει και μετά την παρουσίαση του ομαδοποιημένου Γράφου Ροής Ελέγχου (σχήμα 14), πάντα χρησιμοποιώντας ένα από τους τρεις τρόπους που προαναφέρθηκαν. Όταν χρησιμοποιήσει ένα από τους παραπάνω τρόπους, παρουσιάζεται στην οθόνη ο κανονικός Γράφος Ροής του προγράμματος (σΣχήμα 17).



(Σχήμα 17: Κανονικός Γράφος Ροής Ελέγχου)

Όταν εμφανιστεί ο κανονικός Γράφος Ροής Ελέγχου, στην μπάρα εργαλείων ενεργοποιούνται και οι άλλες δύο λειτουργίες του προγράμματος (σχήμα 18), η παρουσίαση πληροφοριών για κάποιο επιλεγμένο κόμβο (6^{ο} κουμπί) και η δημιουργία του Γράφου Εξαρτήσεων (Dependence Graph) (7^{ο} κουμπί).

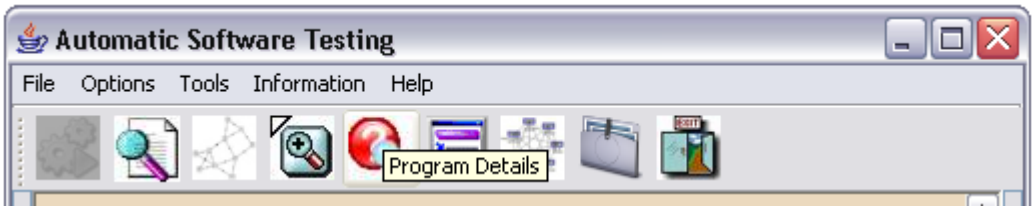


(Σχήμα 18)

D. Εμφάνιση πληροφοριών προγράμματος

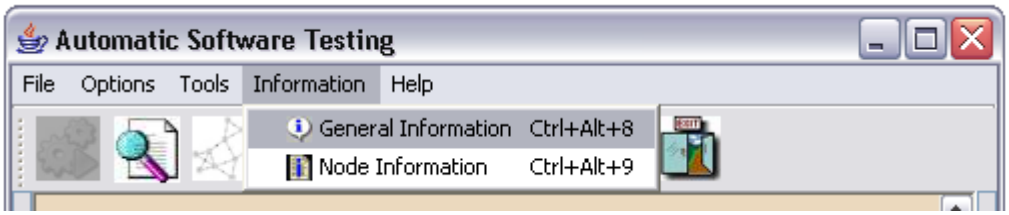
Ο χρήστης μπορεί να δει πληροφορίες για το πρόγραμμα το οποίο επέλεξε για ανάλυση χρησιμοποιώντας ένα από τους ακόλουθους τρόπους:

1. Πατά το κουμπί “Program Details” στην μπάρα εργαλείων (5^{ο} κουμπί)



(Σχήμα 19)

2. Επιλέγει από το μενού “Information”την επιλογή “General Information”

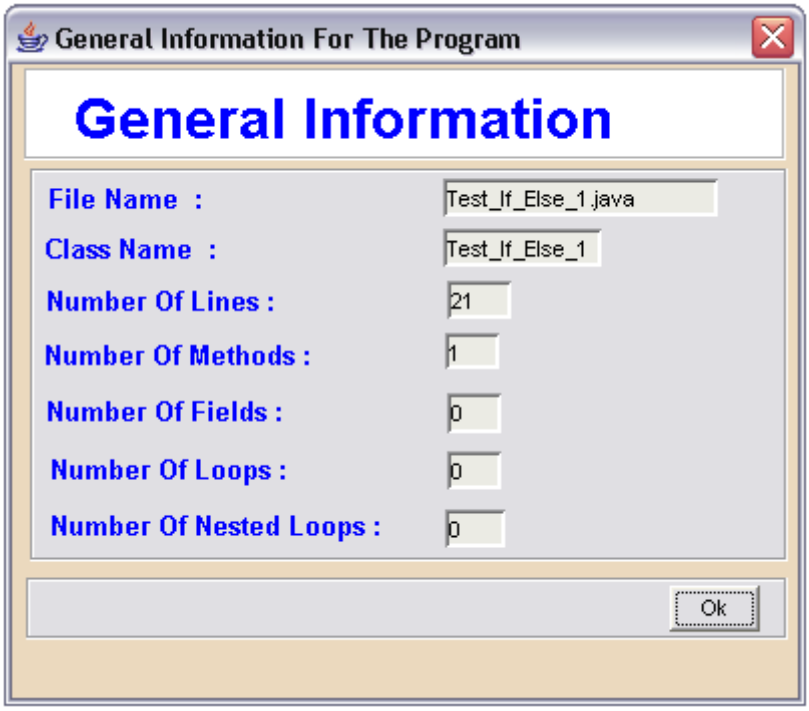


(Σχήμα 20)

3. Χρησιμοποιεί τον συνδυασμό των πλήκτρων “CTRL + ALT + 8”

Με την χρήση ενός από τους τρεις τρόπους, εμφανίζεται μια οθόνη που παρουσιάζει πληροφορίες για το πρόγραμμα το οποίο αναλύθηκε (σχήμα 21). Οι πληροφορίες αυτές είναι:

i. Όνομα αρχείου προγράμματος
ii. Όνομα κλάσης προγράμματος
iii. Αριθμός γραμμών
iv. Αριθμός μεθόδων
v. Αριθμός πεδίων
vi. Αριθμός Βρόγχων και Φωλιασμένων Βρόγχων



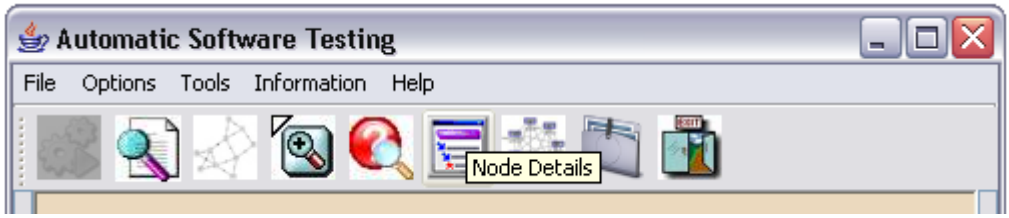
(Σχήμα 21: Πληροφορίες προγράμματος)

Όταν ο χρήστης πατήσει το "OK", η οθόνη με τις πληροφορίες εξαφανίζεται.

Ε. Εμφάνιση πληροφοριών κόμβου

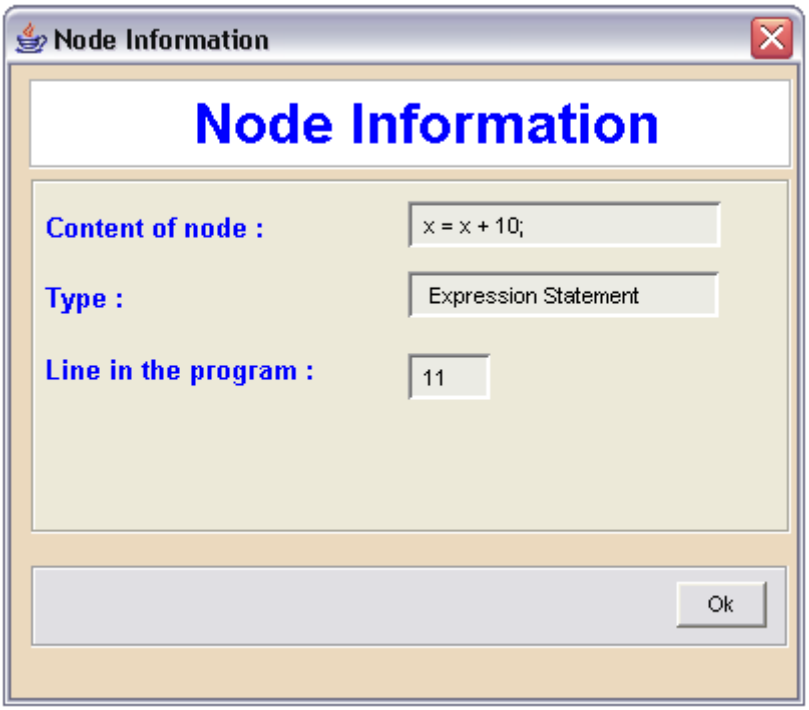
Ο χρήστης μπορεί να δει διάφορες πληροφορίες για κάποιο κόμβο του κανονικού Γράφου Ροής Ελέγχου (σχήμα 17), αλλά και για κάποιο κόμβο του Γράφου Εξαρτήσεων (σχήμα 26).

Για να δει τις πληροφορίες για τον κανονικό Γράφο Ροής Ελέγχου, πρέπει να χρησιμοποιήσει τον συνδυασμό "CTRL + SHIFT + Left Mouse Click" και μόλις πατήσει πάνω σε ένα κόμβο, θα εμφανιστούν οι πληροφορίες για τον κόμβο αυτό. Για να δει τις πληροφορίες για ένα κόμβο του Γράφου Εξαρτήσεων, χρησιμοποιεί τον συνδυασμό "CTRL + SHIFT + Left Mouse Click" και πατά πάνω σε ένα κόμβο για να τον επιλέξει. Ακολούθως πατάει το κουμπί "Node Details" στην μπάρα επιλογών (6^{ο} κουμπί (σχήμα 22))



(Σχήμα 22)

Ακολούθως, εμφανίζεται μια οθόνη με τις πληροφορίες του κόμβου που επιλέχθηκε (σχήμα 23)



(Σχήμα 23: Πληροφορίες επιλεγμένου κόμβου)

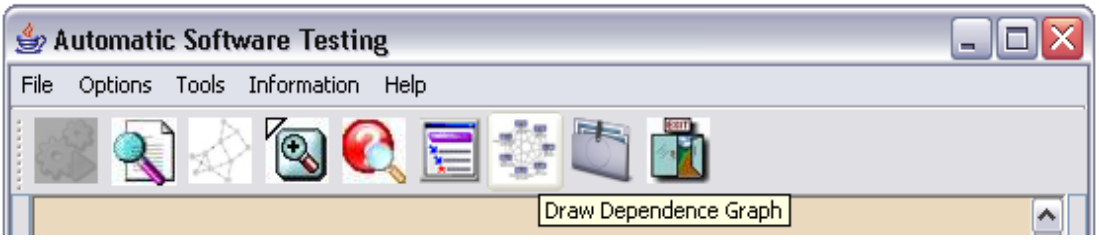
Στην οθόνη με τις πληροφορίες του κόμβου εμφανίζονται:

i. Τα περιεχόμενα του κόμβου
ii. Ο τύπος του κόμβου
iii. Η γραμμή του προγράμματος, στην οποία αναφέρεται ο κόμβος

F. Εμφάνιση Γράφου Εξαρτήσεων (Dependence Graph)

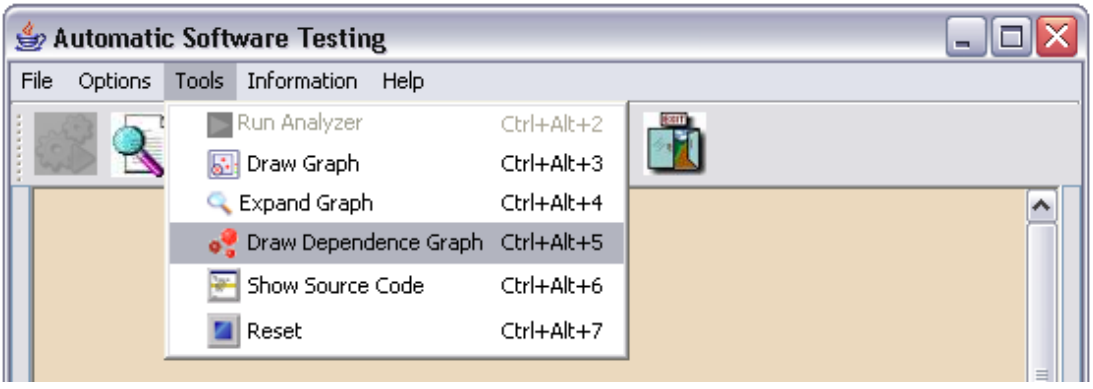
Όταν ο χρήστης δημιουργήσει τον κανονικό Γράφο Ροής Ελέγχου (σχήμα 17), το εργαλείο του δίνει την δυνατότητα να δημιουργήσει τον Γράφο Εξαρτήσεων του επιλεγμένου προγράμματος. Αυτό μπορεί να το κάνει με τους ακόλουθους τρόπους:

1. Στην μπάρα εργαλείων, πατά το κουμπί “Draw Dependence Graph” (7^{ο} κουμπί)



(Σχήμα 24)

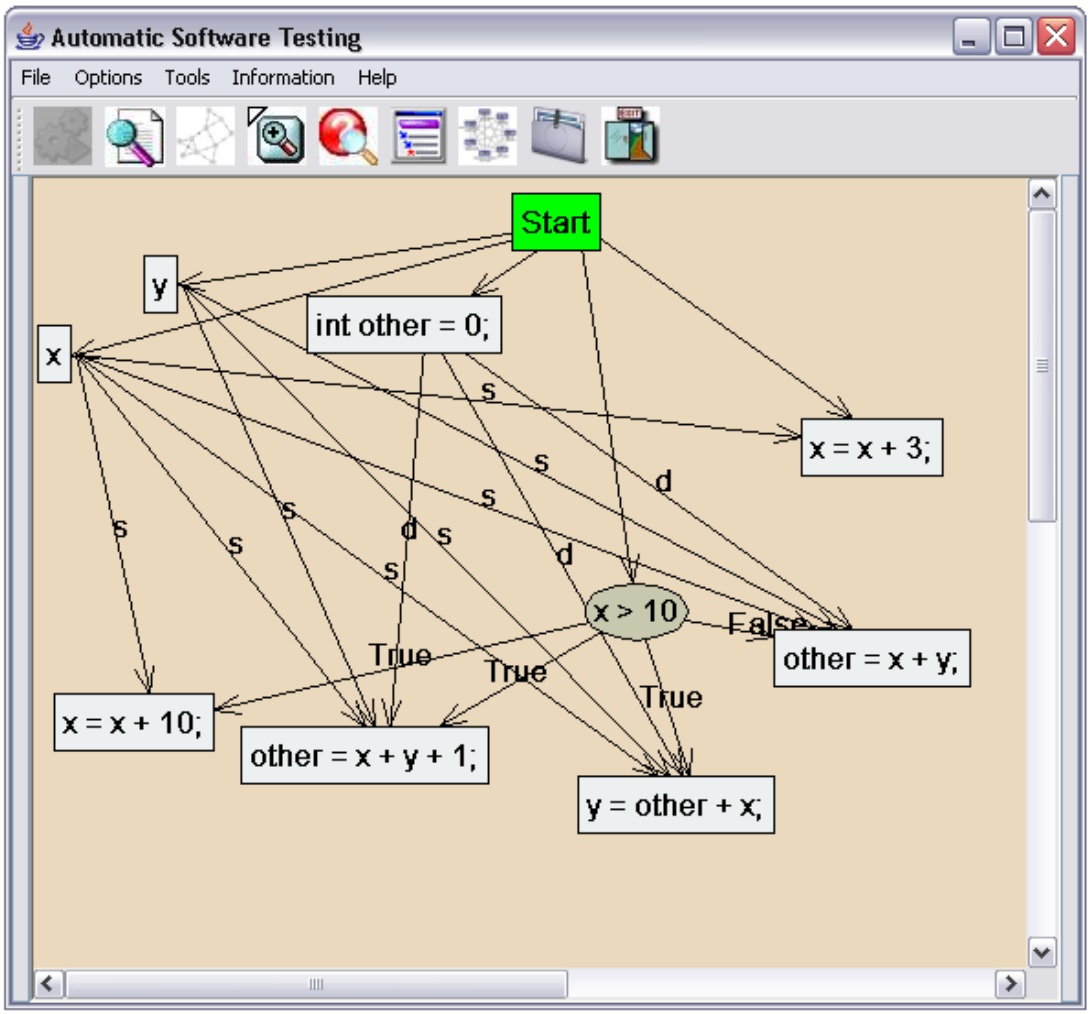
2. Στο μενού επιλογών, επιλέγει “Draw Dependence Graph” από το μενού “Tools”



(Σχήμα 25)

3. Χρησιμοποιεί τον συνδυασμό των πλήκτρων “CTRL + ALT + 5”

Ακολούθως εμφανίζεται στην οθόνη ο Γράφος Εξαρτήσεων (σχήμα 26)



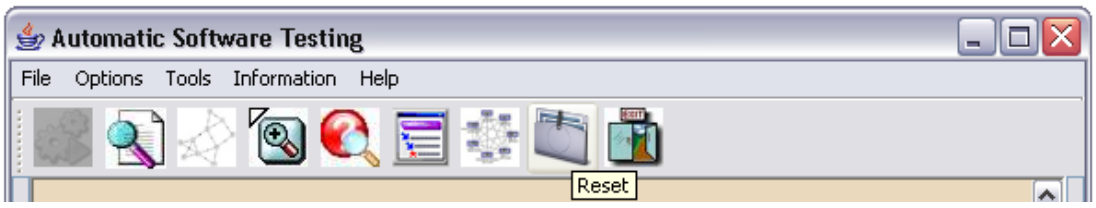
(Σχήμα 26: Γράφος Εξαρτήσεων)

- Παρατήρηση: Παρατηρούμε ότι υπάρχουν τρία (3) είδη ακμών στο Γράφο Εξαρτήσεων.
  - Control Dependence Edges: Δείχνουν την εξάρτηση των κόμβων του γράφου μεταξύ τους
  - Data Dependence Edges: Δείχνουν την εξάρτηση των κόμβων του Γράφου πάνω στις μεταβλητές του προγράμματος (οι ακμές που είναι σημειωμένες με “d”)
  - Summary Edges: Δείχνουν την εξάρτηση των κόμβων του Γράφου πάνω στις παραμέτρους του προγράμματος (οι ακμές που είναι σημειωμένες με “s”)

G. Αρχικοποίηση Εργαλείου (Reset)

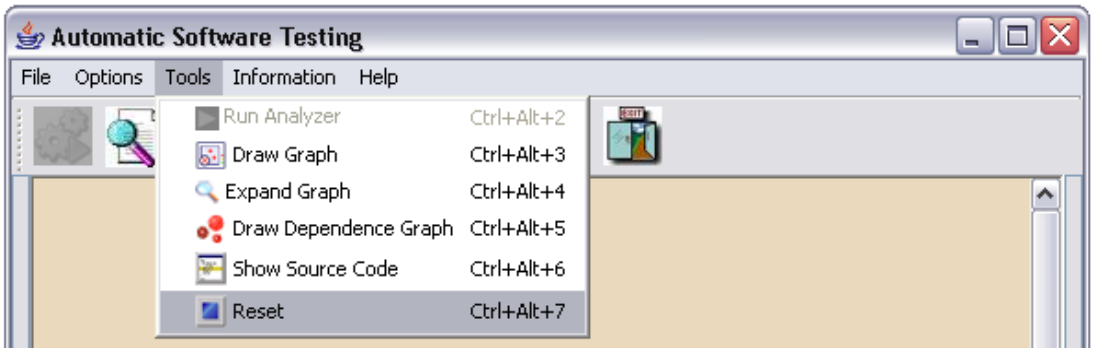
Ο χρήστης μπορεί να αρχικοποιήσει το εργαλείο, ούτως ώστε να αναλύσει ένα νέο πρόγραμμα, χρησιμοποιώντας ένα από τους ακόλουθους τρόπους:

1. Πατάει το κουμπί “Reset” στην μπάρα εργαλείων (8^{ο} κουμπί)



(Σχήμα 27)

2. Επιλέγει στο μενού επιλογών την επιλογή “Reset” από το μενού “Tools”



(Σχήμα 28)

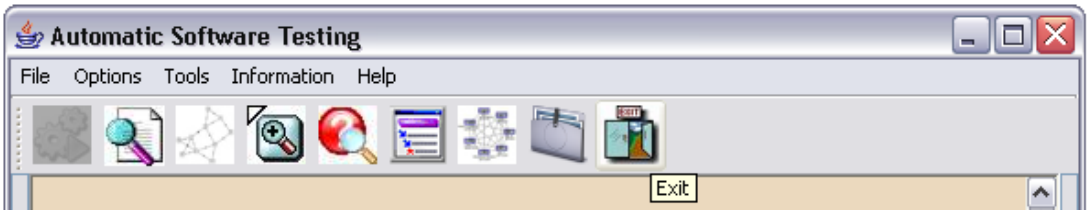
3. Χρησιμοποιεί τον συνδυασμό πλήκτρων “CTRL + ALT + 7”

Όταν ο χρήστης χρησιμοποιήσει ένα από τους ανωτέρω τρόπους, το εργαλείο επιστρέφει στην αρχική κατάσταση (σχήμα 1) και ο χρήστη μπορεί να αναλύσει κάποιο άλλο πρόγραμμα.

Η. Έξοδος (Exit)

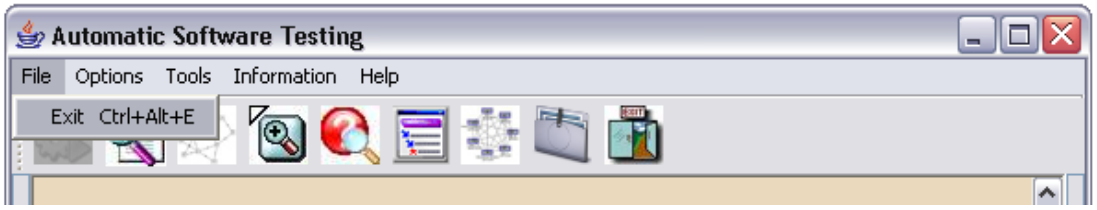
Όταν ο χρήστης θέλει να κλείσει το εργαλείο, μπορεί να το κάνει με τρεις διαφορετικούς τρόπους:

1. Πατάει το κουμπί “Exit” στην μπάρα εργαλείων (9^{ο} κουμπί)



(Σχήμα 29)

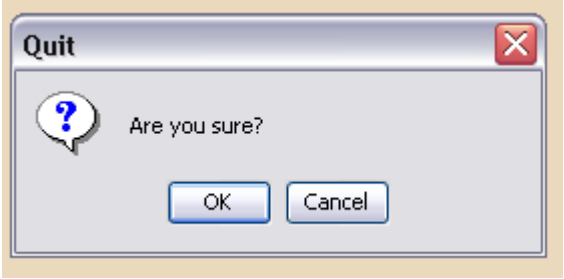
2. Επιλέγει στο μενού επιλογών την επιλογή “Exit” από το μενού “File”



(Σχήμα 30)

3. Χρησιμοποιεί τον συνδυασμό πλήκτρων “CTRL + ALT + E”

Όταν ο χρήστης χρησιμοποιήσει ένα από τους παραπάνω τρόπους, εμφανίζεται μια οθόνη, όπου ζητείται από τον χρήστη να επιβεβαιώσει ότι πράγματι θέλει να κλείσει το πρόγραμμα (σχήμα 31)



(Σχήμα 31: Μήνυμα για επιβεβαίωση εξόδου)

Αν ο χρήστη επιλέξει “OK”, τότε το πρόγραμμα τερματίζεται. Αν επιλέξει “Cancel”, τότε επιστρέφει στην προηγούμενη του κατάσταση και ο χρήστης μπορεί να συνεχίσει την εργασία του.

3. Παράδειγμα Ανάλυσης Προγράμματος

Το εργαλείο αυτό αναλύει προγράμματα, τα οποία είναι γραμμένα στην γλώσσα προγραμματισμού JAVA. Συγκεκριμένα, μπορεί να αναλύσει και να παράξει τον Γράφο Εξαρτήσεων προγραμμάτων που περιέχουν:

- Απλές εντολές (assignments, unary operators, expressions κτλ)
- Μία μέθοδο με παραμέτρους
- If – Else statements (φωλιασμένα και μη)
- For statements (φωλιασμένα και μη)
- While statements (φωλιασμένα και μη)

Δεν παρέχει όμως ανάλυση προγραμμάτων με περισσότερες από μία μεθόδους και προγραμμάτων που περιέχουν πίνακες ή άλλες δομές δεδομένων.

i. Πρόγραμμα για ανάλυση

```
package org.testfiles;
public class Test_Paradeigma {
  public Test_Paradeigma() {
  }
  public void methodA(int x, int y)
  {
    int other = 0;
    if(x>10){
      x = x + 10;
      other = x + y + 1;
      y = other + x;
    }
    else{
      other = x + y;
    }
    x = x+ 3;
    while(y < 11)
    {
      other = x+ y;
      y = y +1 ;
    }
    x = x + 2;
  }
}
```

(Σχήμα 32: Πρόγραμμα για ανάλυση)

Το πρόγραμμα που θα τρέξουμε για παράδειγμα, αποτελείται από:

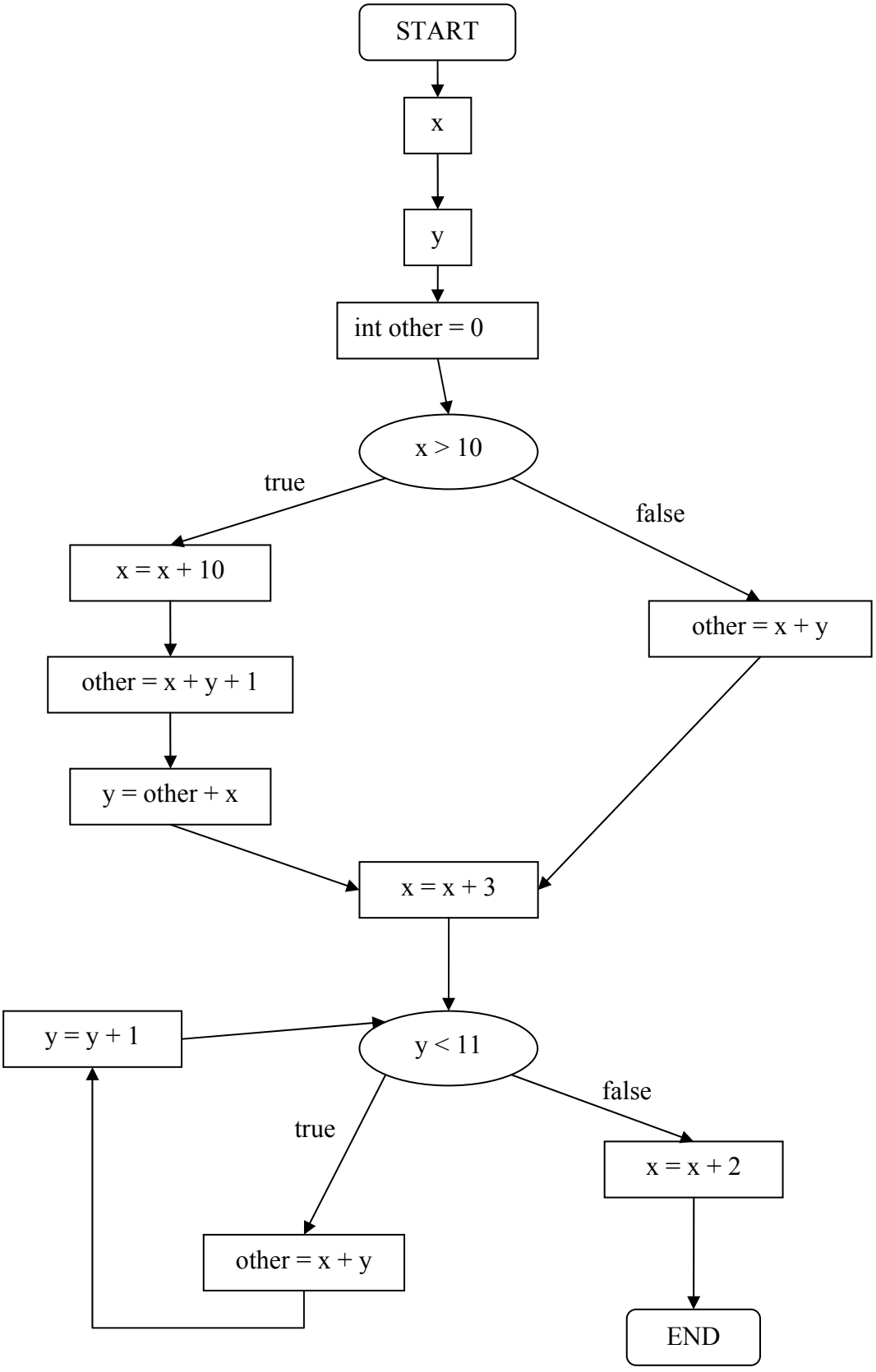
- Μία (1) μέθοδο - methodA
- Δύο (2) παραμέτρους - int x, y
- Μία (1) μεταβλητή - int other
- Ένα (1) If – Else Statement
- Ένα (1) While Statement

Αρχικά θα τρέξουμε το παράδειγμα στο χέρι και ακολούθως στο εργαλείο, ούτως ώστε να δείξουμε ότι το εργαλείο δουλεύει σωστά και οι παραγόμενοι γράφοι είναι οι ίδιοι.

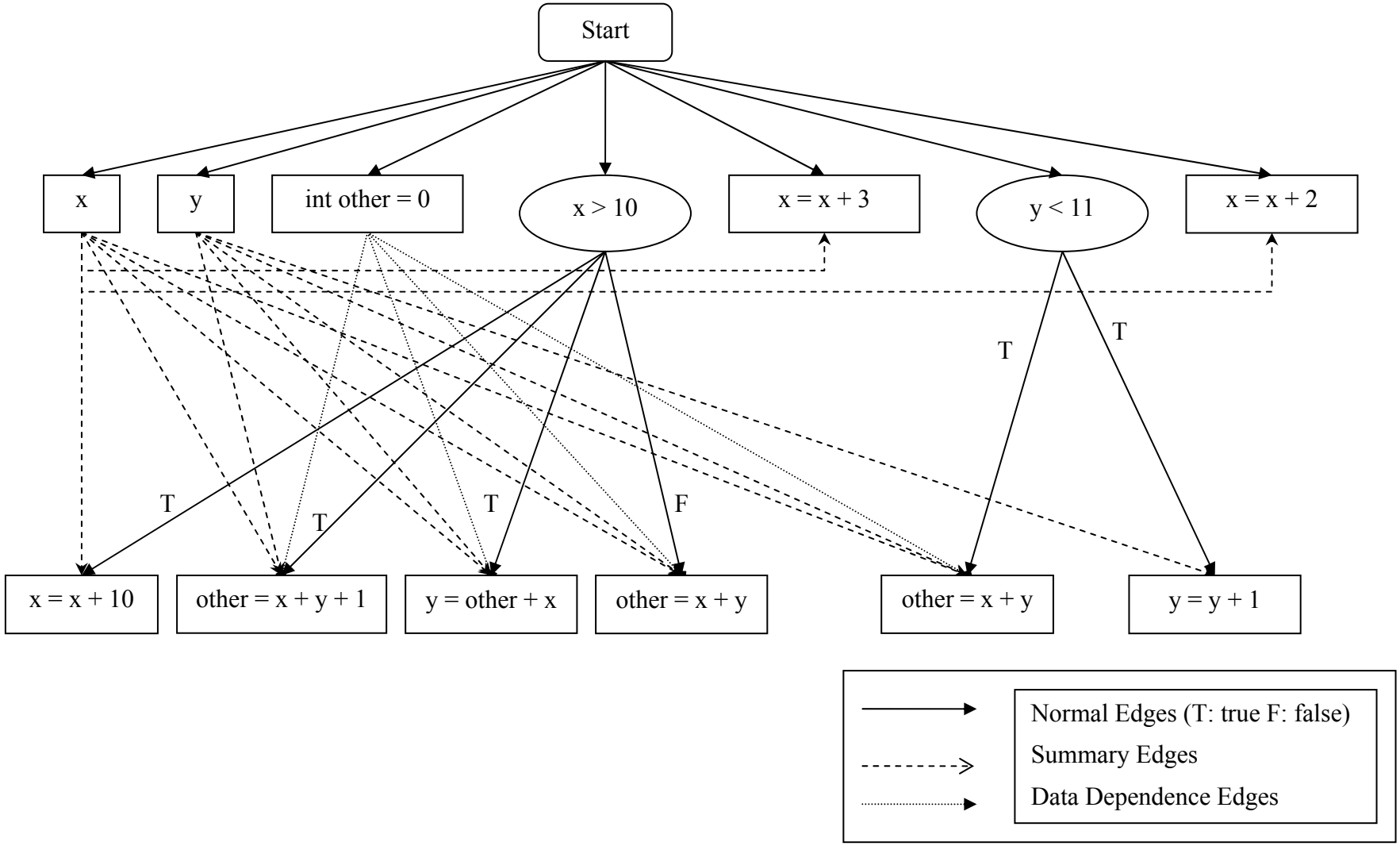
Πρώτα δημιουργούμε τον Γράφο Ροής Ελέγχου του προγράμματος και χρησιμοποιώντας τον Γράφο αυτό, παράγουμε τον Γράφο Εξαρτήσεων του.

ii. Εκτέλεση στο χέρι

a. Παραγωγή Γράφου Ροής Ελέγχου (Control Flow Graph)

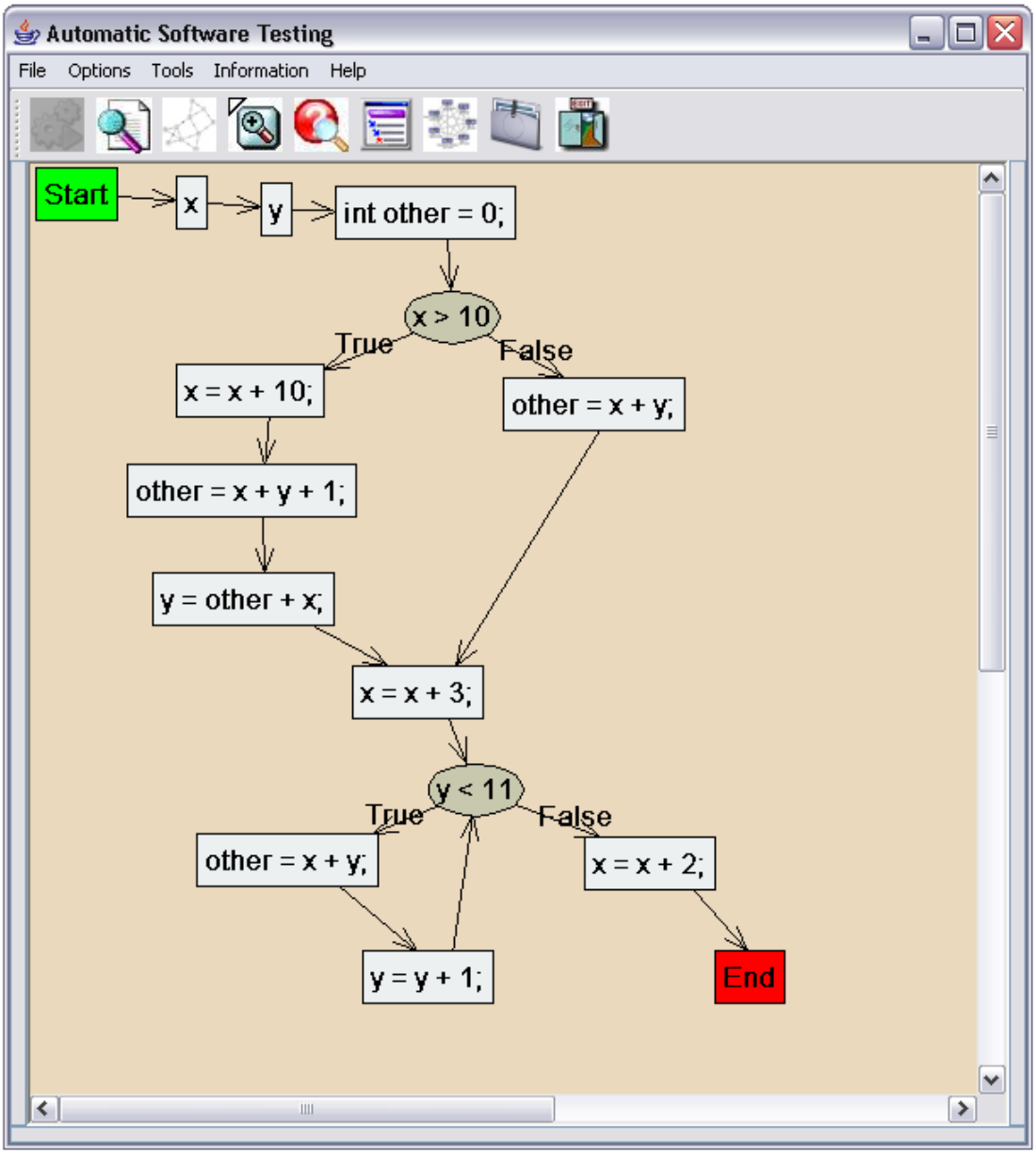


b. Παραγωγή Γράφου Εξαρτήσεων (Dependence Graph)

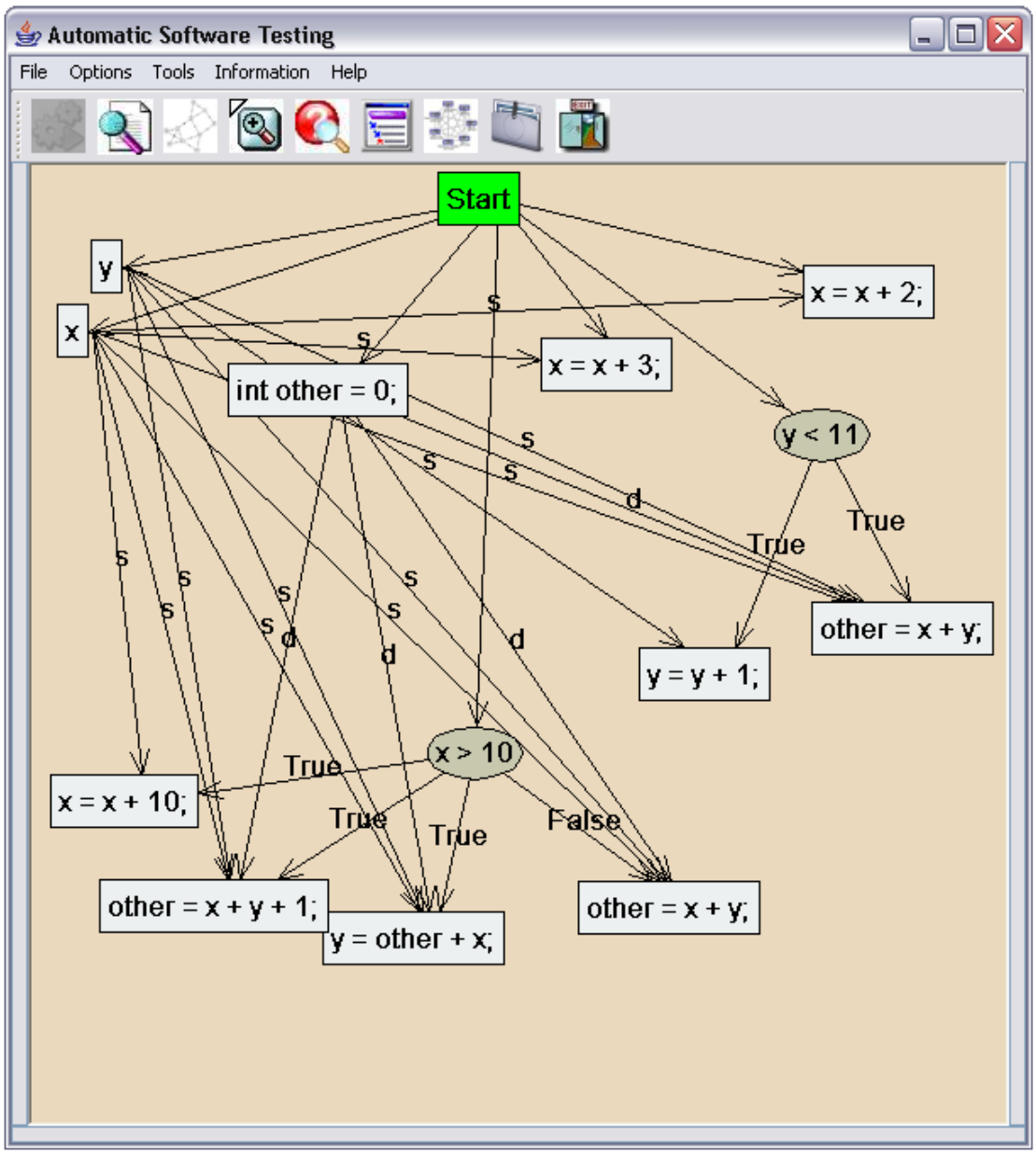


iii. Εκτέλεση στο εργαλείο

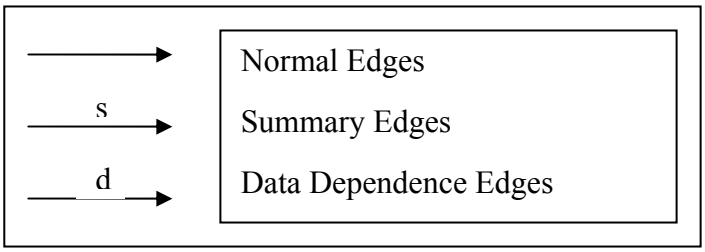
a. Παραγωγή Γράφου Ροής Ελέγχου (Control Flow Graph)



b. Παραγωγή Γράφου Εξαρτήσεων (Dependence Graph)



Υπόμνημα:



iv. Παρατηρήσεις – Συμπεράσματα

Από την εκτέλεση και παραγωγή του Γράφου Εξαρτήσεων στο χέρι, αλλά και στο εργαλείο, βλέπουμε ότι τα αποτελέσματα είναι τα ίδια.

Το εργαλείο δημιούργησε τον Γράφο Εξαρτήσεων του προγράμματος μας ακριβώς όπως και εμείς στο χέρι, άρα μπορεί να παράξει ορθά τον Γράφο Εξαρτήσεων κάποιου προγράμματος.

4. Μετρήσεις

i. Πειραματικό πρόγραμμα για μετρήσεις

Για τις μετρήσεις, χρησιμοποιήθηκε ένα πρόγραμμα 50 γραμμών (πίνακας 1), το οποίο περιείχε διαδοχικά If – Else Statements.
Για τις επόμενες μετρήσεις, χρησιμοποιήθηκε ο ίδιος κώδικας διαδοχικά για σχηματισμό μεγαλύτερων προγραμμάτων.

```
public class Test_0050_lines {
  public Test_0050_lines() {
  }
  public void methodA(int x, int y)
  {
    int other = 0;
    if(x>5){
      x = x + 10;
      x = y - 3;
      y = other + x;
    }
    else{
      other = x + y;
    }
    x = x+ 3;
    if(x>10){
      x = x + 10;
      x = y - 3;
      y = other + x;
    }
    else{
      x = other;
    }
   y = y + 5;
    if(x>15){
      x = x + 10;
      x = y - 3;
      y = other + x;
    }
    else{
      other = x + y;
    }
    x = x+ 3;
    if(x>20){
      x = x + 10;
      x = y - 3;
      y = other + x;
    }
    else{
      x = other;
    }
   y = y + 5;
   if(x>25){
      x = x + 10;
      x = y - 3;
      y = other + x;
    }
    else{
      x = other;
    }
   y = y + 5;
  }
}
```

Πίνακας 1: Πειραματικό πρόγραμμα για μετρήσεις

ii. Χρόνοι εκτέλεσης

Οι χρόνοι των μετρήσεων (πίνακας 2), αφορούν τον χρόνο που χρειάζεται το πρόγραμμα να δημιουργήσει και να σχεδιάσει τον Γράφο Εξαρτήσεων (Dependence Graph), χωρίς να λαμβάνεται υπόψη ο χρόνος που χρειάζεται για να δημιουργηθεί ο Γράφος Ροής Ελέγχου (Control Flow Graph).

Γραμμές	**Κόμβοι**	**Ακμές**	**Χρόνος**		
			Δημιουργίας γράφου	**Σχεδιασμού γράφου**	**Συνολικός**
50	34	60	6,2 sec	0,1 sec	6,3 sec
100	64	157	15,1 sec	5,2 sec	20,3 sec
150	94	234	28,4 sec	16,8 sec	45,2 sec
200	124	311	40,6 sec	33,6 sec	1 min 14,2 sec
250	154	388	58,8 sec	1 min 1,6 sec	2 min 0,4 sec
300	184	465	1 min 23,8 sec	1 min 26,5 sec	2 min 50,3 sec
350	214	542	2 min 0,5 sec	1 min 42.1 sec	3 min 42,6 sec
400	244	619	2 min 26,7 sec	2 min 29,4 sec	4 min 56,1 sec
500	304	773	3 min 45,9 sec	3 min 59,3 sec	7 min 45,2 sec
1000	604	1543	15 min 14,1 sec	14 min 51,7 sec	30 min 5,8 sec

Πίνακας 2: Χρόνοι εκτέλεσης

iii. Παρατηρήσεις

Από τους χρόνους των μετρήσεων (πίνακας 2), παρατηρούμε ότι όσο αυξάνεται το μέγεθος του προγράμματος που αναλύουμε με το εργαλείο μας, ο χρόνος που χρειάζεται για την παραγωγή του Γράφου Εξαρτήσεων (Dependence Graph) είναι αρκετά αυξημένος. Επίσης παρουσιάζεται μια μεγάλη αύξηση και στον χρόνο που χρειάζεται το εργαλείο μας για να σχεδιάσει το γράφο στην επιφάνεια εργασίας. Αυτό σημαίνει ότι για αρκετά μεγάλα προγράμματα, ο χρόνος που θα χρειαζόμαστε για την παραγωγή του Γράφου Εξαρτήσεων θα είναι πολύ μεγάλος. Εν τούτοις, είναι αρκετά πιο γρήγορη η διαδικασία σε σχέση με την παραγωγή του στο χέρι. Έτσι, με το πρόγραμμα αυτό κερδίζουμε αρκετό χρόνο στην διαδικασία για τον έλεγχο ενός προγράμματος.

Κεφάλαιο 6

Συμπεράσματα και προτάσεις για μελλοντική εργασία

1. Εισαγωγή

Η εργασία αυτή είχε σαν σκοπό την αυτόματη παραγωγή του Γράφου Εξαρτήσεων, χρησιμοποιώντας τον Γράφο Ροής Ελέγχου που παράγει το εργαλείο που δημιούργησε η κ. Ιωάννα Ιωακείμ.

Το εργαλείο το οποίο αναπτύχθηκε για αυτό τον σκοπό βασίζεται πάνω στο εργαλείο της κ. Ιωακείμ. Βασικά είναι ενσωματωμένο πάνω σε αυτό, ούτως ώστε η ανάλυση του προγράμματος που επιλέγεται να είναι πιο ολοκληρωμένη και πιο εύκολη στο να επιτευχθεί.

Το εργαλείο προσφέρει ανάλυση προγραμμάτων γραμμένα στη γλώσσα προγραμματισμού JAVA και καλύπτει προγράμματα με τα παρακάτω χαρακτηριστικά:

- If – else statement (φωλιασμένα ή μη),
- Απλές εντολές (assignment, unary operator, expression κτλ)
- For statement (φωλιασμένα ή μη),
- While statement (φωλιασμένα ή μη),
- Μία μέθοδο

- Παραμέτρους στην μέθοδο του προγράμματος,

Προγράμματα που δεν εμπίπτουν σε κάποια από τις παραπάνω περιπτώσεις, το εργαλείο δεν είναι σε θέση να τα αναλύσει. Μερικά χαρακτηριστικά που δεν καλύπτει το εργαλείο είναι:

- Πίνακες και άλλες δομές δεδομένων
- Περισσότερες από μία μέθοδο
- Δημιουργία αντικειμένων

Εκτός από την παραγωγή και την παρουσίαση του Γράφου Εξαρτήσεων ενός προγράμματος, το εργαλείο μας προσφέρει και την δυνατότητα να δούμε πληροφορίες σχετικά με το πρόγραμμα που αναλύουμε, αλλά και επιμέρους πληροφορίες για κάθε κόμβο του γράφου που δημιουργείται, επιλέγοντας τον κόμβο που θέλουμε πάνω στον γράφο.

2. Συμπεράσματα

Η δημιουργία του εργαλείου αυτού και η όλη διαδικασία της παραγωγής του γράφου, μας οδηγεί στην εξαγωγή μερικών συμπερασμάτων:

- Η παραγωγή του Γράφου Εξαρτήσεων γίνεται εύκολα, με απλή χρήση της διεπιφάνειας του εργαλείου, χωρίς την ανάγκη χρήσης χειριού.
- Η γραφική απεικόνιση του Γράφου Εξαρτήσεων βοηθά τον χρήστη να κατανοήσει ευκολότερα τις εξαρτήσεις στο πρόγραμμα που αναλύει και έτσι γίνεται ευκολότερος ο έλεγχος και η συντήρηση του προγράμματος.
- Ενσωματώνοντας το εργαλείο που δημιουργήσαμε πάνω στο εργαλείο της κ. Ιωακείμ, προσφέρουμε στον χρήστη μια πιο ολοκληρωμένη ανάλυση του προγράμματος, δίνοντας του την δυνατότητα να δημιουργήσει και να δει τον Γράφο Ροής Ελέγχου (κανονικό και ομαδοποιημένο) καθώς και τον Γράφο Εξαρτήσεων του προγράμματος που αναλύει.

- Σε περιπτώσεις μεγάλης εκτάσεως προγραμμάτων, παρατηρείται αυξημένος χρόνος για την παραγωγή και την παρουσίαση του Γράφου Εξαρτήσεων. Παρόλα αυτά, είναι σαφώς γρηγορότερη σε σχέση με την δημιουργία στο χέρι.

3. Προτάσεις για μελλοντική εργασία

Μετά την ολοκλήρωση της δημιουργίας του εργαλείου μας και παρόλο που η παραγωγή του Γράφου Εξαρτήσεων μέσω αυτού είναι επιτυχημένη, μπορούν να γίνουν μερικές προσθήκες και βελτιώσεις στην όλη διαδικασία, ούτως ώστε να γίνεται μια πιο πλήρης ανάλυση προγραμμάτων και σε ένα ευρύτερο φάσμα γλωσσών προγραμματισμού:

- Το εργαλείο υποστηρίζει ανάλυση μόνο προγραμμάτων γραμμένα σε JAVA. Μπορεί να γίνει επέκταση του εργαλείου, ούτως ώστε να υποστηρίζει και άλλες γλώσσες προγραμματισμού. Αυτό μπορεί να γίνει με την επιλογή γραμματικής κάποιας άλλης γλώσσας προγραμματισμού. Η αρχιτεκτονική πάνω στην οποία στηρίζεται το εργαλείο, υποστηρίζει την επιλογή κάποιας άλλης γραμματικής, αλλά δεν έχει δοκιμαστεί.
- Τα προγράμματα που υποστηρίζονται, περιέχουν μόνο ένα μικρό φάσμα εντολών. Με μια επέκταση του εργαλείου, μπορεί να γίνει κάλυψη και εντολών οι οποίες δεν υποστηρίζονται, όπως για παράδειγμα packages, constructors, tables και άλλες δομές δεδομένων. Αυτό μπορεί να επιτευχθεί με επέκταση των λειτουργιών του αναλυτή προγραμμάτων που χρησιμοποιείται στο εργαλείο και ειδικότερα του στρώματος walker.
- Το σύστημα να υποστηρίζει και άλλους τύπους προγραμμάτων (π.χ. applets).
- Από τον Γράφο Εξαρτήσεων που δημιουργείται, μπορεί να υλοποιηθεί το slicing του προγράμματος, δηλαδή ο διαχωρισμός του προγράμματος σε διάφορα ανεξάρτητα κομμάτια, με τα οποία θα γίνεται ο έλεγχος του προγράμματος. Αυτό θα προσφέρει περαιτέρω ευκολία στην παραγωγή δεδομένων ελέγχου (test cases) για ευκολότερο έλεγχο του προγράμματος, αφού θα χρειάζονται λιγότερα test cases για τον έλεγχο του.

Βιβλιογραφία

1. Panos E. Livadas, Theodore Johnson, University of Florida, 2000. “An Optimal Algorithm for the Construction of the System Dependence Graph”

2. Richard Johnson, Keshav Pingali, Cornell University, Ithaca, 1993. “Dependence-Based Program Analysis”

3. Mary Jean Harrold, Brian Maloy and Gregg Rothermel, Clemson University, 1993. “Efficient Construction of Program Dependence Graphs”

4. J. A. Bondy and U. S. R. Murty, University of Waterloo, Ontario, Canada, 1986. “Graph Theory With Applications” ISBN: 0-444-19451-7

5. Susan Horwitz and Thomas Reps, University of Wisconsin, 1992. “The Use of Program Dependence Graphs in Software Engineering”

6. Jeanne Ferrante, Karl J. Ottenstein and Joe D. Warren, 1987. “The Program Dependence Graph and Its Use in Optimization”

7. Neil Walkinshaw, Mare Roper, Murray Wood. 2003. “The Java System Dependence Graph”

8. Jianjun Zhao and Martin Rinard, Massachusetts Institute of Technology, 2003. “System Dependence Graph Construction for Aspect - oriented programs”

9. Robbert A. Ballance, Arthur B. Maccabe, The University of New Mexico, 1992. “Program Dependence Graphs for the Rest of Us”

10. Saurabh Sinha, Mary Jean Harrold, Gregg Rothermel, 1999. “System-Dependence-Graph-Based Slicing of Programs with Arbitrary Interprocedural Control Flow”

11. Donglin Liang and Mary Jean Harrold, The Ohio State University, 1998. “Slicing objects using system dependence graph”

12. Aniello Cimitile and Ugo De Carlini, University of Naples, 1991. “Reverse Engineering: Algorithms for Program Graph Production”

13. Thomas Reps, University of Wisconsin, 1994. “On the Sequential Nature of Interprocedural Program-Analysis Problems”

14. GrammaTech Inc., CodeSurfer Technology Overview, 1992. “Dependence Graphs and Program Slicing”

15. Thomas Ball, Susan Horwitz, University of Wisconsin, 1992. “Slicing Programs with Arbitrary Control Flow”

16. Sunil Pilani. “Slicing Using Object-Oriented System Dependence Graphs”

17. Ιωάννα Ιωακείμ, Ιούνιος 2005. “ Δημιουργία και γραφική απεικόνιση/διαχείριση Γράφου Ροής Ελέγχου από κώδικα JAVA”

18. Boris Bokowski, Andre Spiegel, Freie Universitat Berlin, 1998. “Barat – A front-end for Java”

19. Saurabh Sinha, Alessandro Orso and Mary Jean Harrold, Georgia Institute of Technology, 2004. “Automated Support for Development, Maintenance, and Testingin the Presence of Implicit Control Flow”

20. David W. Binkley, Keith Brian Gallagher, 1996. “Program Slicing”