ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΑΝΑΛΥΣΗ ΔΕΔΟΜΕΝΩΝ, ΜΕΤΡΙΚΕΣ ΚΑΙ ΜΕΤΑΒΛΗΤΕΣ ΠΡΟΓΝΩΣΗΣ ΛΟΓΙΣΜΙΚΟΥ

SOFTWARE ANALYTICS, METRICS AND PREDICTORS

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Θανάσης – Σπύρος Κ. Αποστολόπουλος

Επιβλέπων: Κώστας Κοντογιάννης, Καθηγητής ΕΜΠ

Αθήνα, Νοέμβριος 2017

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΑΝΑΛΥΣΗ ΔΕΔΟΜΕΝΩΝ, ΜΕΤΡΙΚΕΣ ΚΑΙ ΜΕΤΑΒΛΗΤΕΣ ΠΡΟΓΝΩΣΗΣ ΛΟΓΙΣΜΙΚΟΥ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Θανάσης – Σπύρος Κ. Αποστολόπουλος

Επιβλέπων: Κώστας Κοντογιάννης, Καθηγητής ΕΜΠ
Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 9η Νοεμβρίου 2017.

Κώστας Κοντογιάννης	Γιώργος Στάμου	Ανδρέας – Γεώργιος Σταφυλοπάτης
Καθηγητής	Αν. Καθηγητής	Καθηγητής

Θανάσης Κ. Αποστολόπουλος

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών ΕΜΠ

ΠΕΡΙΛΗΨΗ

Η εργασία πραγματεύεται τις έννοιες της διαχείρισης και της διασφάλισης ποιότητας του λογισμικού. Γίνεται αναφορά σε μετρικές με τη βοήθεια των οποίων αναλύεται και αξιολογείται η συμπεριφορά των εφαρμογών, ιδιαίτερα στον τομέα της ποιότητας και της παραγωγικότητας του λογισμικού. Στη συνέχεια, γίνεται χρήση της βάσης καταγραφής του ISBSG – Release 11, τα δεδομένα της οποίας αναλύονται με τη βοήθεια του στατιστικού εργαλείου IBM Watson Analytics. Μέσα από τη στατιστική επεξεργασία των αποτελεσμάτων αναζητούνται ισχυρές μεταβλητές πρόγνωσης για την επίδοση των εφαρμογών, με μέτρο αφενός τον αριθμό των σφαλμάτων που παραμένουν στο σύστημα και παραδίδονται στον τελικό χρήστη, αφετέρου την προσπάθεια που απαιτήθηκε για την ανάπτυξη του λογισμικού.

<u>Λέξεις – κλειδιά</u>

Ποιότητα λογισμικού, παραγωγικότητα λογισμικού, μετρική, λειτουργικό μέγεθος, σφάλματα λογισμικού, εμπειρική τεχνολογία λογισμικού, δειγματοληψία, μη-απόκριση, βάση καταγραφής ISBSG

ABSTRACT

This paper discusses quality management and software quality assurance. It focuses on metrics used to rate projects, especially as regards software quality and productivity. The paper makes use of the ISBSG – Release 11 Repository and processes this data with the help of IBM Watson Analytics statistical tool. Statistical analysis of results seeks major predictors on a/ the number of defects remaining in the system and delivered to final users and b/ the effort required for software project development.

Keywords

Software quality, software productivity, metric, functional size, software defect, empirical software technology, sampling, nonresponse, ISBSG repository

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΙΝΑΚΕΣ

ΕΙΚΟΝΕΣ

1 ΕΙΣΑΓΩΓΙΚΑ

«Επιστήμη και θεωρία δεν αρκούν, αδελφέ Ερρίκε, αν δεν μετουσιωθούν σε ισχύ»
Μαργκερίτ Γιουρσενάρ, Η άβυσσος, μτφρ. Ιωάννα Χατζηνικολή

1.1 Αντικείμενο της εργασίας

Είναι δύσκολο να φανταστούμε την επιστήμη του μηχανικού ξέχωρα από την έννοια της μέτρησης και την – συναφή με αυτήν – διαδικασία της πειραματικής επαλήθευσης: ξεκινώντας από μια υπόθεση, πραγματοποιούμε μια σειρά από πειραματικούς ελέγχους πάνω σε δοκιμαστικά δεδομένα εισόδου. Εφόσον επαληθεύσουμε την αρχική υπόθεση, την αποτυπώνουμε σε ένα μαθηματικό μοντέλο, πάνω στο οποίο εφαρμόζουμε στη συνέχεια καινούργια δεδομένα εισόδου και υπολογίζουμε κάθε φορά την απόκριση[1].

Τέτοιες πρακτικές είναι γνώριμες στους ηλεκτρολόγους, μηχανολόγους ή πολιτικούς μηχανικούς. Δεν ισχύει το ίδιο όμως και για τους μηχανικούς της τεχνολογίας λογισμικού. Οι τελευταίοι, υπό το βάρος συχνά ασφυκτικών χρονικών περιθωρίων, δεν έχουν την άνεση να εκτιμήσουν με ακρίβεια την αναγκαία εργασία, τον αριθμό των ατόμων που απαιτείται για την έγκαιρη ολοκλήρωση του έργου, το ρίσκο ή το συνολικό κόστος που συνεπάγεται η επένδυση. Συχνά επίσης θέτουν οι ίδιοι υπερβολικά αισιόδοξους στόχους και υπόσχονται στους εργοδότες ή τους τελικούς αποδέκτες ότι το τελικό προϊόν θα είναι «φιλικό προς το χρήστη, αξιόπιστο, γρήγορο, χωρίς λάθη, ασφαλές» κοκ.

Απέναντι σε αυτή την κατάσταση, η **εμπειρική τεχνολογία λογισμικού** επιδιώκει να καθορίσει αντικειμενικά κριτήρια, αφενός μέσα από την **καθιέρωση μετρικών**, αφετέρου μέσα από την επεξεργασία των μετρήσεων που πραγματοποιούνται πάνω στα έργα λογισμικού. Πρωταρχικό ρόλο παίζει εδώ η σύνθετη έννοια της **ποιότητας λογισμικού**. Η ανάγκη να μετρήσουμε βασικά μεγέθη, τα οποία επηρεάζουν σημαντικά την ποιότητα του λογισμικού μας οδηγεί στην υιοθέτηση ολοκληρωμένων συστημάτων μετρήσεων του λογισμικού για όλες τις φάσεις ανάπτυξής του, με τη βοήθεια κατάλληλων μετρικών. Σε αυτό το πλαίσιο, το **μέγεθος του έργου** είναι προφανώς μια κρίσιμη παράμετρος που επιδρά στην ποιότητα του λογισμικού. Μετά τη μάλλον ανεπαρκή μέτρηση των γραμμών κώδικα καθιερώνεται, από τα τέλη της δεκαετίας του ’70 και ύστερα, η μέτρηση του λειτουργικού μεγέθους. Η μέθοδος αυτή, με τη βοήθεια της **ανάλυσης λειτουργικών σημείων**, έρχεται να καλύψει το κενό και, σε διάφορες παραλλαγές, έχει καταστεί σήμερα ένα κοινά αποδεκτό σύστημα

[1] Για παράδειγμα, αφού επιβεβαιώσουμε μέσα από το πείραμα την ισχύ του νόμου του Ohm, μπορούμε στη συνέχεια να εκτιμήσουμε ποια είναι η απαιτούμενη τάση στα άκρα ενός συγκεκριμένου κυκλώματος με δεδομένη ολική αντίσταση ώστε να πάρουμε την επιθυμητή ένταση ρεύματος.

υπολογισμού του μεγέθους λογισμικού. Με μέτρο το μέγεθος μπορούμε στη συνέχεια να εκτιμήσουμε και άλλες παραμέτρους όπως η προσπάθεια, ο ρυθμός παράδοσης, η πυκνότητα σφαλμάτων κτλ., αναγκαίες για την αξιολόγηση του επιπέδου ποιότητας ενός έργου.

Όπως είναι φυσικό, κεντρική θέση στη συζήτηση γύρω από την ποιότητα κατέχουν τα **σφάλματα λογισμικού**. Αυτά μπορεί να αφορούν τόσο στον πηγαίο κώδικα, όσο και σε άλλες φάσεις του κύκλου ζωής του λογισμικού, όπως λάθη και παρανοήσεις στη φάση συλλογής των απαιτήσεων, ελλειπείς προδιαγραφές κατά τη σχεδίαση, ασάφειες στην τεκμηρίωση κοκ. Έχουν προταθεί διάφορες διαδικασίες, καταρχάς για την αποφυγή και – όπου αυτό δεν είναι αρκετό – τον εντοπισμό και τη διαχείριση των σφαλμάτων. Η αποτελεσματικότητα τέτοιων μεθόδων κρίνεται από το βαθμό στον οποίο ανακαλύπτουν και αντιμετωπίζουν με επιτυχία τα σφάλματα που έχουν ήδη παρεισφρύσει στο λογισμικό.

Όπως αναφέρθηκε ήδη, η εμπειρική τεχνολογία λογισμικού ασχολείται με τις μετρήσεις πάνω σε έργα λογισμικού ώστε να προβεί σε συγκρίσεις των βασικών παραμέτρων τους. Στόχος είναι η εκτίμηση και πρόβλεψη της συμπεριφοράς μιας νέας εφαρμογής, με βάση τα χαρακτηριστικά που απαιτεί χρήστης, σε σύγκριση δε και με άλλες ομοειδείς με αυτήν. Στο πλαίσιο της ανάλυσης δεδομένων της παρούσας έρευνας επιχειρείται η εξαγωγή στατιστικών συμπερασμάτων και εκτιμήσεων με βάση έναν πολύ μεγάλο αριθμό δεδομένων από το **αρχείο καταγραφής ISBSG Repository – Release 11**. Η συζήτηση αφορά κυρίως στον τομέα της **ποιότητας** (με τη στενότερη εδώ έννοια των σφαλμάτων που υπάρχουν στο σύστημα) και της **παραγωγικότητας** (με την έννοια του ρυθμού παράδοσης, δηλ. της προσπάθειας που απαιτείται για την ολοκλήρωση μιας μονάδας μεγέθους). Αναζητώνται στατιστικώς σημαντικά αποτελέσματα για τις περιγραφικές παραμέτρους των έργων λογισμικού, καθώς επίσης και ισχυρές **μεταβλητές πρόβλεψης** (predictors) για τις επιδόσεις καινούργιων εφαρμογών στους δύο παραπάνω τομείς.

1.2 Περίγραμμα της εργασίας

- Το παρόν κεφάλαιο αποτελεί μια γενική εισαγωγή στο θέμα.
- Το Κεφ. 2 πραγματεύεται την έννοια της ποιότητας λογισμικού και την ποικιλία των χαρακτηριστικών της, ανάλογα με τη σκοπιά από την οποία βλέπουν το λογισμικό οι διάφοροι συντελεστές που εμπλέκονται στην ανάπτυξή του. Γίνεται επίσης λόγος για το σχεδιασμό και τη διασφάλιση της ποιότητας λογισμικού ενώ, στο πλαίσιο αυτό, κάνουμε μια σύντομη αναφορά στο πρότυπο *Στόχος – Ερώτημα – Μετρική*.
- Το Κεφ. 3 αναλύει την έννοια της μετρικής ως βασικό εργαλείο των μετρήσεων πάνω στα έργα λογισμικού. Οι μετρικές αυτές μπορούν να αφορούν σε προϊόντα, διαδικασίες και πόρους λογισμικού, ενδεικτικά δε γίνεται αναφορά με δύο λόγια στο πρότυπο CMM ως παράδειγμα ενός συστήματος μετρικών για διαδικασίες. Τέλος, παρατίθενται λίγα στοιχεία για τις διάφορες κλίμακες μετρήσεων και τις διαφοροποιήσεις μεταξύ τους.
- Στο Κεφ. 4 γίνεται λόγος για τη μέτρηση του μεγέθους λογισμικού. Η μέτρηση του μεγέθους δεν είναι απλή υπόθεση, καθώς από την έκταση του κώδικα και μόνο δεν μπορούμε να έχουμε ολοκληρωμένη εικόνα για το σύνολο των γνωρισμάτων (λειτουργικών και μη) της εφαρμογής, όπως τα

εκλαμβάνει και τα προσδοκά ο τελικός χρήστης. Γίνεται διεξοδική αναφορά στον τρόπο υπολογισμού του λειτουργικού μεγέθους λογισμικού, στα πλεονεκτήματα και μειονεκτήματα της μεθόδου αυτής καθώς και στη διαφοροποίησή της από τη μέτρηση των γραμμών κώδικα.

- Το Κεφ. 5 πραγματεύεται τις έννοιες των σφαλμάτων και αστοχιών λογισμικού. Στο πλαίσιο της συζήτησης που αφορά στον εντοπισμό των σφαλμάτων λογισμικού, γίνεται αναφορά στις έννοιες του δυναμικού και της απόδοσης αφαίρεσης σφαλμάτων. Παρατίθενται στοιχεία και περιγράφονται συνοπτικά κάποιες μέθοδοι για την ανίχνευση των σφαλμάτων – τόσο πριν όσο και κατά τη φάση του κυρίως ελέγχου – όπως επίσης και για την αποτροπή τους. Τέλος, αναφερόμαστε εν συντομία στα σφάλματα που ξεφεύγουν από τον έλεγχο και παραμένουν στο λογισμικό κατά τη διάρκεια της λειτουργίας του, όπως επίσης στις τεχνικές με σκοπό τον περιορισμό μιας πιθανής αστοχίας και την ελαχιστοποίηση της ζημιάς που αυτή μπορεί να επιφέρει.
- Το Κεφ. 6 παραθέτει στοιχεία γύρω από την παραγωγικότητα και την ανθρώπινη προσπάθεια κατά την ανάπτυξη εφαρμογών λογισμικού. Ιδιαίτερη μνεία γίνεται στους διάφορους παράγοντες που επηρεάζουν σε σημαντικό βαθμό τα παραπάνω μεγέθη, όπως οι μεθοδολογίες που χρησιμοποιούνται, οι πλατφόρμες ανάπτυξης, εάν πρόκειται για καινούργιο έργο ή για αναβάθμιση παλαιότερου κοκ.
- Το Κεφ. 7 πραγματεύεται έννοιες σχετικές με το πεδίο της εμπειρικής τεχνολογίας λογισμικού. Γίνεται μνεία των μεθόδων που χρησιμοποιούνται γενικά, με μια κάπως εκτενέστερη αναφορά στη θεωρία των δειγματοληπτικών ερευνών και τα διάφορα σφάλματα που σχετίζονται με αυτές. Αναφέρονται επίσης συνοπτικά στοιχεία γύρω από τις ελλείπουσες και ακραίες τιμές, καθώς οι έννοιες αυτές έχουν εφαρμογή στην έρευνα των δεδομένων που επιχειρείται παρακάτω.
- Στο Κεφ. 8 αναλύεται η χρησιμότητα της στατιστικής ανάλυσης και της εξόρυξης δεδομένων στην εκτιμητική πάνω στα έργα λογισμικού. Γίνεται αναφορά, μεταξύ άλλων, σε τεχνικές όπως τα δέντρα αποφάσεων και η πολυμεταβλητή ανάλυση. Επίσης, αναφερόμαστε στην ανάλυση βασικής γραμμής αναφοράς και στην έννοια της συγκριτικής ανάλυσης των επιδόσεων λογισμικού.
- Στο Κεφ. 9 περιγράφεται η βάση καταγραφής δεδομένων ISBSG, που αποτελεί το δείγμα πάνω στο οποίο πραγματοποιήθηκε η παρούσα έρευνα. Δίνονται «δημογραφικά» στοιχεία για τα έργα που καταγράφονται στη βάση και αναλύονται οι ιδιαιτερότητες του δείγματος, οι οποίες υπαγορεύουν και τις μεθόδους που χρησιμοποιήθηκαν για την ανάλυση.
- Στο Κεφ. 10 δίνονται λεπτομέρειες αναφορικά με τον «καθαρισμό» και την προετοιμασία των δεδομένων ώστε να σχηματιστεί το τελικό δείγμα εργασίας.
- Το Κεφ. 11 παρουσιάζει τα αποτελέσματα που προέκυψαν από την ανάλυση των δεδομένων του δείγματος. Κατά την παρουσίαση των αποτελεσμάτων ο αναγνώστης παραπέμπεται στο Παράρτημα, όπου παρουσιάζονται γραφικά τα στατιστικά ευρήματα για την παραγωγικότητα, όπως έχουν κατά κύριο λόγο προκύψει με τη βοήθεια του στατιστικού εργαλείου IBM Watson Analytics. Τα στοιχεία που έχουν συλλεχθεί στη βάση καταγραφής αφορούν ως επί το πλείστον σε έργα μικρού και μεσαίου μεγέθους – είναι ενδεικτικό ότι τα περισσότερα είναι μέχρι 2000 FP, ενώ ελάχιστα ξεπερνούν τα 10000 FP. Για τα έργα αυτά, ο πιο ισχυρός παράγοντας πρόγνωσης (predictor) της παραγωγικότητας φαίνεται να είναι ο συνδυασμός **τύπου γλώσσας** (Γ2Γ, Γ3Γ, Γ4Γ, Γ5Γ ή ApG) και **αρχιτεκτονικής** (Stand-Alone, Multi-Tier, Client-Server κοκ.) – για τα μικρότερα έργα – και η **πλατφόρμα ανάπτυξης** (Mainframe, Mid-Range, Multi-Platform ή PC) – για τις εφαρμογές μεγαλύτερης τάξης. Σε ό,τι αφορά στην ποιότητα του λογισμικού, η πολύ περιορισμένη απόκριση στη βάση καταγραφής δεν επιτρέπει συγκεκριμένα συμπεράσματα. Φαίνεται πάντως μια σαφής διαφοροποίηση ως προς το είδος

εφαρμογής (νέες εφαρμογές ή αναβαθμίσεις παλαιότερων) και μια σαφής προοδευτική μείωση της πυκνότητας σφαλμάτων για μεγαλύτερης τάξης μεγέθη λογισμικού. Εξάλλου, το ποσοστό της αποχής φαίνεται να εξαρτάται από το μέγεθος του έργου και το μέγιστο αριθμό μελών που απασχολήθηκαν στην ομάδα ανάπτυξης του λογισμικού, με τις επιδόσεις να βελτιώνονται για μεγαλύτερες τιμές στους δύο αυτούς τομείς.

- Το Κεφ. 12 συνοψίζει τα συμπεράσματα από την ανάλυση που προηγήθηκε, επισημαίνει τα σημεία που χρειάζονται μεγαλύτερη εξειδίκευση και δίνει κάποιες κατευθύνσεις και προτάσεις για μελλοντική έρευνα σε σχετικούς τομείς.

1.3 Επίλογος

Στη διεθνή βιβλιογραφία γίνεται ολοένα και περισσότερο λόγος για περαιτέρω έρευνα πάνω στην εμπειρική τεχνολογία λογισμικού. Πράγματι, τα τελευταία χρόνια βρίσκονται σε εξέλιξη πολυάριθμες μελέτες με βάση δεδομένα από πειραματικές διαδικασίες και δειγματοληπτικές έρευνες πάνω σε έργα λογισμικού, τα οποία στη συνέχεια αναλύονται διεξοδικά έτσι ώστε να εξαχθούν αποτελέσματα και να προταθούν αξιόπιστα μοντέλα που θα εξηγούν και θα προβλέπουν τη συμπεριφορά ενός υπό ανάπτυξη νέου έργου. Σε αυτό συντείνει και η ολοένα στενότερη συνεργασία ανάμεσα στους μηχανικούς λογισμικού και στους ειδικούς επιστήμονες που εξειδικεύονται στα πεδία της στατιστικής και της εξόρυξης δεδομένων, όπως επίσης και μια ευρεία πλέον γκάμα από εξειδικευμένα εργαλεία στατιστικής ανάλυσης που έχουν τη δυνατότητα να χειρίζονται αποτελεσματικά τεράστιους όγκους δεδομένων.

Κλείνουμε το πρώτο αυτό εισαγωγικό κεφάλαιο με έναν αφορισμό του Barry Boehm, που προτρέπει στο δρόμο που πρέπει να ακολουθήσουμε: «ο χώρος της τεχνολογίας λογισμικού δεν μπορεί να ελπίζει σε έναν Κέπλερ ή έναν Νεύτονα πριν αποκτήσει τη δική του στρατιά από Τύχο Μπράχε[2], οι οποίοι θα επεξεργαστούν εκείνες τις αυστηρά καθορισμένες ακολουθίες παρατηρήσεων από τις οποίες μπορεί τελικά να προκύψει ένα σύνολο από βαθύτερες επιστημονικές γνώσεις». (1)

[2] Τύχο Μπράχε (ή Τύχων Βράχιος, Tyge Brahe, Tycho Brahe): Δανός αστρονόμος, αστρολόγος και αλχημιστής (1546 – 1601). Αναγνώρισε και μελέτησε τον περίφημο καινοφανή αστέρα (nova) που περιέγραψε στο έργο του *Ο νέος αστέρας* (1572). Στον ακριβολόγο αυτόν επιστήμονα – παρατηρητή οφείλονται πολλές αξιόλογες βελτιώσεις στις θεωρίες περί Σελήνης. Συνέταξε λεπτομερή κατάλογο από 777 αστέρες ενώ ήταν ο πρώτος που μελέτησε το φαινόμενο της διάθλασης. Βάσει των χειρόγραφων σημειώσεων που άφησε ο Τύχων, ο μαθητής του και αργότερα μεγάλος αστρονόμος Γιοχάνες Κέπλερ κατάφερε να διατυπώσει τους περίφημους νόμους του για την κίνηση των πλανητών [πηγή: wikipedia]

2 ΠΟΙΟΤΗΤΑ ΛΟΓΙΣΜΙΚΟΥ

2.1 Τι εννοούμε με τον όρο «ποιότητα λογισμικού»;

Στο ερώτημα του τίτλου μπορούν να δοθούν διάφορες απαντήσεις. Η ποιότητα λογισμικού δεν σημαίνει πάντοτε το ίδιο πράγμα για όλους όσους εμπλέκονται στην υλοποίηση και λειτουργία του: μηχανικοί λογισμικού, σχεδιαστές, προγραμματιστές, χρηματοδότες, τελικοί χρήστες, διαχειριστές, μηχανικοί ελέγχου κ. ά. έχουν συχνά διαφορετικές προσεγγίσεις για την ίδια έννοια. Για τον καθένα από αυτούς, «ποιότητα»[3] μπορεί κατά περίπτωση να σημαίνει ευκολία στη χρήση, εκπλήρωση απαιτήσεων, τήρηση προδιαγραφών, απουσία σφαλμάτων, αξιοπιστία, ευχέρεια στην εκμάθηση, σαφήνεια στα συνοδευτικά έγγραφα τεκμηρίωσης, εύκολη συντήρηση και γρήγορη επαναφορά στη λειτουργία, αισθητική ικανοποίηση του χρήστη κοκ. (2)

Στη συνέχεια αναφέρουμε κάποια συναφή με την ποιότητα γνωρίσματα: (3)

- Υψηλό επίπεδο ικανοποίησης του χρήστη, μεταφερσιμότητα, συντηρησιμότητα, ευστάθεια και καταλληλότητα προς χρήση [Barry Boehm]
- Άριστα επίπεδα στην καταλληλότητα προς χρήση, τήρηση των απαιτήσεων, αξιοπιστία και συντηρησιμότητα [Watts Humphrey]
- Καταλληλότητα προς χρήση [Steven Kan]
- Υψηλά επίπεδα ικανοποίησης του χρήστη και χαμηλά επίπεδα σφαλμάτων, κατά κανόνα συσχετισμένα με χαμηλή πολυπλοκότητα [Tom McCabe]
- Χαμηλά επίπεδα σφαλμάτων, συμμόρφωση των λειτουργιών του λογισμικού προς τις ανάγκες του χρήστη και υψηλή αξιοπιστία [John Musa]

Η ποιότητα λογισμικού έχει λοιπόν πολλές διαστάσεις. Τη δεκαετία του 1970, η Hewlett-Packard υιοθέτησε το μοντέλο *FURPS*, από τα αρχικά των ακόλουθων πέντε κριτηρίων:

- *Functionality* (λειτουργικότητα): κατά πόσο το προϊόν διαθέτει τις απαραίτητες λειτουργίες ώστε να διεκπεραιώσει την εργασία που του αναθέτει ο χρήστης.
- *Usabiity* (χρηστικότητα): ευχέρεια στη χρήση, στην εκμάθηση, ευκολία στην αλληλεπίδραση χρήστη – προϊόντος ώστε να έχουμε αυξημένη παραγωγικότητα κτλ.
- *Reliability* (αξιοπιστία): σχετίζεται με τη συχνότητα και σοβαρότητα των βλαβών, καθώς και με τα γνωρίσματα που αφορούν στην ανάνηψη.

[3] Ποιότητα: το σύνολο των χαρακτηριστικών ενός πράγματος, ό,τι το χαρακτηρίζει και το διαφοροποιεί από τα όμοιά του, το σύνολο των θετικών ιδιοτήτων του [από το *Λεξικό της Νέας Ελληνικής Γλώσσας, Μπαμπινιώτης Γ., 2002]*

- *Performance (*απόδοση): σχετίζεται με την ταχύτητα εκτέλεσης των λειτουργιών, τη χρήση πόρων, το χρόνο απόκρισης κοκ.
- *Serviceability* (επίπεδα συντήρησης): αφορά στην τεχνική υποστήριξη, στην ταχύτητα απόκρισης και την ποιότητα των διορθώσεων, στην ευκολία στην εγκατάσταση κτλ. (4)

Κατά μια έννοια, το **πρότυπο ISO/IEC-9126** (ISO, 2001) αποτελεί συνέχεια του μοντέλου *FURPS*. Το πρότυπο αυτό επικαιροποιεί και συμπληρώνει το παλαιότερο πρότυπο ISO του 1991 και δίνει τον πληρέστερο ορισμό της ποιότητας λογισμικού, εξετάζοντάς την από τρεις σκοπιές: (5)

Ποιότητα κατά τη χρήση (quality in use): είναι η ποιότητα από τη σκοπιά του χρήστη. Αποτυπώνει το βαθμό στον οποίο ο χρήστης πετυχαίνει το στόχο του, όταν το προϊόν τίθεται σε λειτουργία σε συγκεκριμένο περιβάλλον.

Εξωτερική (external) ποιότητα: είναι η ποιότητα όταν «τρέχουμε» το λογισμικό, η οποία μετράται και αξιολογείται τυπικά κατά τη φάση του ελέγχου σε ένα προσομοιωμένο περιβάλλον και με προσομοιωμένα δεδομένα.

Εσωτερική (internal) ποιότητα. Μετράται και αξιολογείται ως προς τις αντίστοιχες απαιτήσεις εσωτερικής ποιότητας. Οι λεπτομέρειες της ποιότητας του προϊόντος μπορούν να βελτιωθούν κατά την κωδικοποίηση ή τον έλεγχο, αλλά η εσωτερική ποιότητα του λογισμικού παραμένει αμετάβλητη εκτός και γίνει επανασχεδιασμός της εφαρμογής.

Στο πρότυπο ISO/IEC-9126 (2001a) τίθενται τα ακόλουθα κριτήρια για την *εσωτερική και εξωτερική ποιότητα* (αναφέρονται ονομαστικά κάποια επί μέρους γνωρίσματα για το καθένα από αυτά): (5) (4)

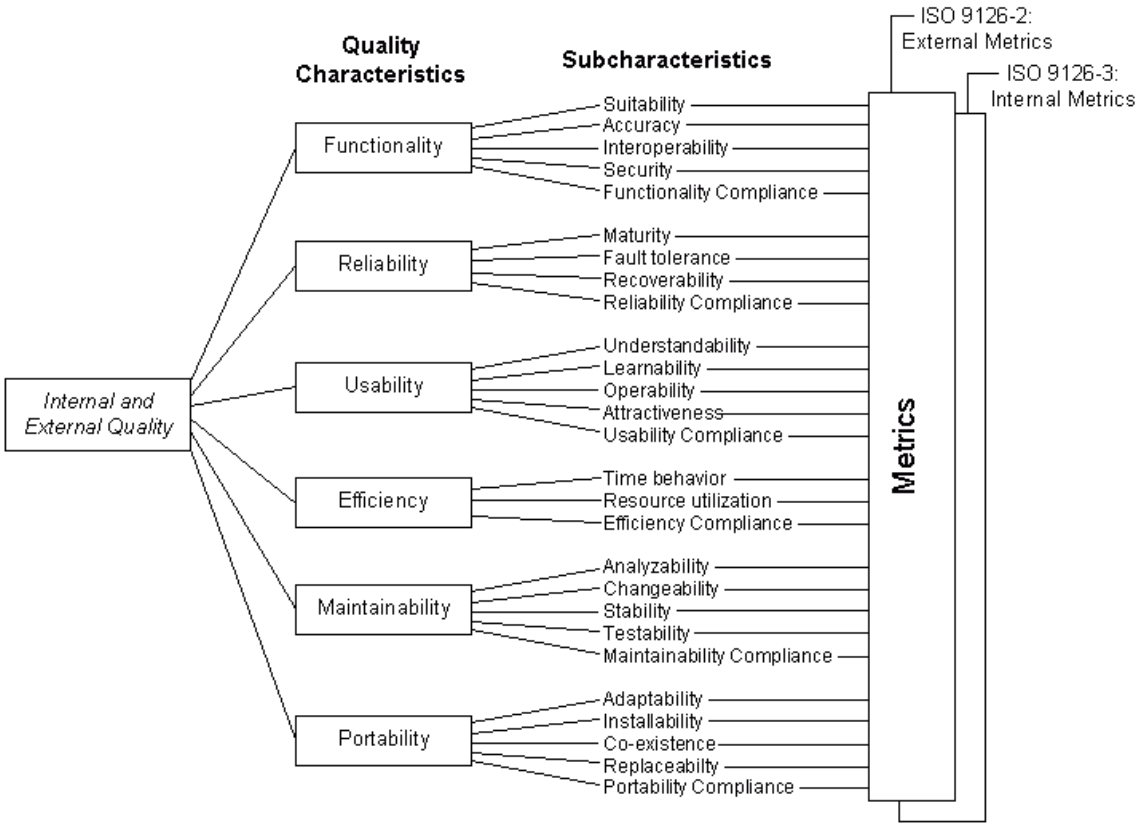
- ***Λειτουργικότητα (functionality)***: καταλληλότητα, ακρίβεια, διαλειτουργικότητα, ακεραιότητα
- ***Αξιοπιστία (reliability)***: ωριμότητα, ανοχή σε σφάλματα, ικανότητα ανάκτησης
- ***Χρηστικότητα (usability)***: ευκολία κατανόησης, ευκολία εκμάθησης, ευκολία χειρισμού
- ***Αποδοτικότητα (efficiency)***: συμπεριφορά ως προς το χρόνο, χρησιμοποίηση πόρων
- ***Συντηρησιμότητα (maintainability)***: αναλυσιμότητα, τροποποιησιμότητα, ευστάθεια, ελεγξιμότητα
- ***Μεταφερσιμότητα (portability)***: προσαρμοστικότητα, δυνατότητα εγκατάστασης, συνύπαρξη, ευκολία αντικατάστασης

Σχηματικά, η παραπάνω κατηγοριοποίηση φαίνεται στην Εικόνα 2-1.

Εξάλλου, το πρότυπο ISO/IEC-9126 (2001a) καθορίζει τα ακόλουθα τέσσερα γνωρίσματα της *ποιότητας κατά τη χρήση*: (5)

- ***Αποτελεσματικότητα (effectiveness)***
- ***Παραγωγικότητα (productivity)***
- ***Ασφάλεια (safety)***
- ***Ικανοποίηση (satisfaction)***

Από τα παραπάνω συνάγεται ότι η «εύκολη» εξομοίωση ανάμεσα στην ποιότητα λογισμικού και τις διαδικασίες αποσφαλμάτωσης είναι τουλάχιστον απλουστευτική. Πέρα από τα σφάλματα, παίζουν εδώ ρόλο και άλλοι παράγοντες (6) όπως



Εικόνα 2-1: Εξωτερική και εσωτερική ποιότητα στο μοντέλο ISO/IEC-9126

- *Καταλληλότητα και χρηστικότητα*. Η συζήτηση αφορά εδώ στην αποτελεσματικότητα (ως προς τις λειτουργικές απαιτήσεις), την απόδοση (ως προς τα μη λειτουργικά χαρακτηριστικά) και την ικανοποίηση του χρήστη
- *Συντηρησιμότητα*. Διακρίνουμε την διορθωτική (επισκευή και ανάνηψη από αστοχίες), προσαρμοστική (τροποποιήσεις ώστε να γίνει δυνατή η προσαρμογή σε νέο περιβάλλον υλικού ή λογισμικού) και προληπτική (αφαίρεση των σφαλμάτων από το προς παράδοση προϊόν πριν τα συναντήσει ο χρήστης)
- *Ασφάλεια*. Ο βαθμός προστασίας απέναντι σε κακόβουλες επιθέσεις και οι προδιαγραφές πρόσβασης στα δεδομένα από άτομα ή άλλα συστήματα ανάλογα με το επίπεδο πιστοποίησης του καθενός από αυτά.
- *Αξιοπιστία*. Ένα μέγεθος που εκφράζεται με την συνάρτηση αξιοπιστίας (ή επιβίωσης)

$$R(t) = 1 - F(t),$$

όπου *F(t)* η σωρευτική πιθανότητα βλάβης δηλ. η πιθανότητα να σημειωθεί βλάβη στο σύστημα μέχρι τη χρονική στιγμή t.

2.2 Διαχείριση ποιότητας

Για να επιτύχουμε ένα επιθυμητό επίπεδο ποιότητας είναι απαραίτητος ο καθορισμός κατευθυντήριων γραμμών και προτύπων. Η αποτελεσματική εφαρμογή διαδικασιών τέτοιων ώστε τα παραγόμενα προϊόντα να βρίσκονται μέσα σε επιθυμητά όρια επιτυγχάνεται με τον **έλεγχο ποιότητας** (quality control), η δε διασφάλιση της ποιότητας γίνεται εφικτή μέσα από έναν ολοκληρωμένο **σχεδιασμό ποιότητας** (quality planning). Ένα πλήρες σχέδιο ποιότητας πρέπει να ξεκινάει παράλληλα με τα αρχικά στάδια της ανάπτυξης, εφαρμόζεται σε κάθε μία φάση ανάπτυξης ξεχωριστά, θέτει τα επιθυμητά χαρακτηριστικά που πρέπει να διαθέτει το τελικό προϊόν και καθορίζει τις μεθόδους ώστε να επιτευχθούν οι στόχοι, τόσο οι επιμέρους ανά φάση όσο και οι καθολικοί. Ένα τέτοιο πλάνο περιλαμβάνει λεπτομερή περιγραφή του προϊόντος, χρονοδιάγραμμα της παραγωγής, καταγραφές των διαδικασιών και προτύπων που θα ακολουθηθούν κατά την ανάπτυξη κοκ. Περιλαμβάνει επίσης ανάλυση και διαχείριση ρίσκου, δηλαδή τη μελέτη των κινδύνων σε συνάρτηση με την πιθανότητα εμφάνισής τους, καθώς και τρόπους αντιμετώπισης ή μετριασμού τους. (7) (8)

Όλα τα παραπάνω συνδιαμορφώνουν την έννοια της διαχείρισης ποιότητας (quality management)[4]. Η **διαχείριση ποιότητας λογισμικού** συνίσταται επομένως στον καθορισμό κατευθυντήριων γραμμών, διαδικασιών και προτύπων που οδηγούν στη δημιουργία λογισμικού καλής ποιότητας, καθώς και στον έλεγχο ότι το σύνολο αυτών των κατευθυντήριων γραμμών ακολουθείται κατά την ανάπτυξη μιας εφαρμογής λογισμικού. (7)

Η ποιότητα λογισμικού είναι μια σύνθετη έννοια με ποικίλα γνωρίσματα, τόσο λειτουργικά όσο και μη λειτουργικά. Η πράξη έχει καταδείξει ότι δεν είναι πάντοτε δυνατή η βελτιστοποίηση ως προς όλα αυτά τα γνωρίσματα ταυτόχρονα. Συχνά παρατηρείται μια ικανοποιητική τιμή σε κάποιο χαρακτηριστικό, η οποία όμως συνοδεύεται από κακή επίδοση σε κάποιο άλλο – για παράδειγμα, μια σημαντική βελτίωση στην ευστάθεια μπορεί να επιφέρει αισθητή πτώση στην απόδοση. Ένας επιτυχημένος σχεδιασμός ποιότητας οφείλει να θέσει τις προτεραιότητες και να προβεί, αν και όπου χρειαστεί, στους αναγκαίους συμβιβασμούς (trade-off) ανάμεσα στις διάφορες αντικρουόμενες επιλογές. (8)

2.3 Διασφάλιση ποιότητας λογισμικού

Οι δραστηριότητες της **διασφάλισης ποιότητας λογισμικού** (software quality assurance) κατατείνουν στην ελαχιστοποίηση του αριθμού των ελαττωμάτων που παραμένουν στο λογισμικό τη στιγμή της κυκλοφορίας του ή της παράδοσής του στους χρήστες. Επιπλέον, οφείλουμε να εξασφαλίσουμε ότι όσα σφάλματα παραμείνουν θα προξενήσουν την ελάχιστη αποδεκτή ζημιά ή διαταραχή στη λειτουργία. (9)

[4] Διοίκηση / διαχείριση (ολικής) ποιότητας: οργανωμένη προσπάθεια διασφάλισης της ποιότητας των παραγόμενων προϊόντων και υπηρεσιών μέσω υιοθέτησης στρατηγικών, συστημάτων, διαδικασιών και διοικητικών εργαλείων που θέτουν την ποιότητα ως προτεραιότητα και ευθύνη όλων των εργαζομένων [*Λεξικό της Νεοελληνικής Γλώσσας*, Ακαδημία Αθηνών]

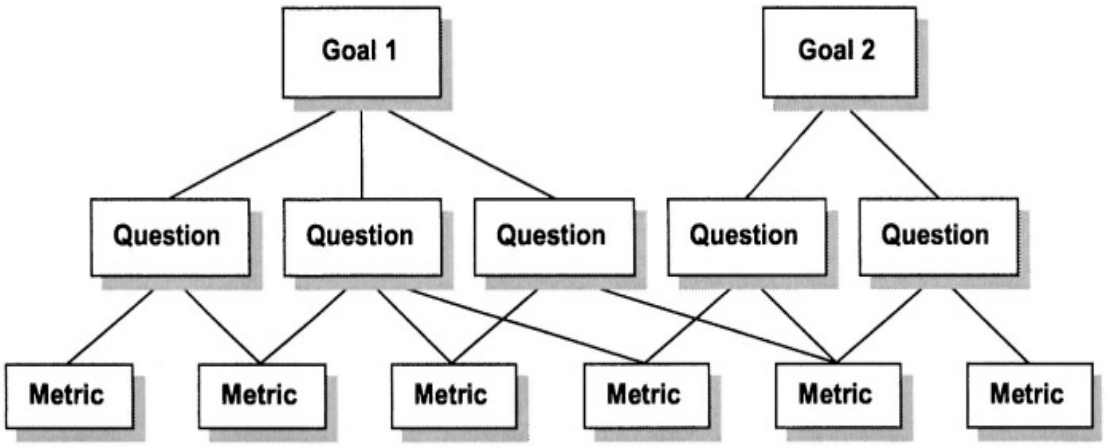
Ένα εφαρμοσμένο πρόγραμμα μετρήσεων λογισμικού πρέπει καταρχήν να απαντάει στα ακόλουθα ερωτήματα: (10)

- Ποιον εξυπηρετεί το πρόγραμμα (δηλ. ποιος είναι ο «πελάτης») και ποιοι είναι οι στόχοι που επιθυμεί να πετύχει σε σχέση με το μετρούμενο προϊόν (product), διαδικασία (process) ή πόρο (resource);
- Ποιές είναι οι μετρικές, βάσει των οποίων θα καθοριστεί η επίτευξη ή μη των στόχων;

Για να αποκτήσουμε μια αίσθηση της μετρικής, η οποία θα αναλυθεί περαιτέρω στη συνέχεια, αξίζει να αναφερθούμε στο πρότυπο **Στόχος – Ερώτημα – Μετρική** (ΣΕΜ, Goal – Question – Metric / GQM), όπως διατυπώθηκε για πρώτη φορά στο (11). Εδώ, η μετρική ορίζεται από την κορυφή προς τα κάτω ως εξής:

1. Όρισε τον *Στόχο* για το προϊόν / τη διαδικασία / τον πόρο.
2. Θέσε τα *Ερωτήματα* που θα καθορίσουν την επίτευξη ή μη του στόχου.
3. Προσδιόρισε τις *Μετρικές* που θα δώσουν ποσοτικοποιημένες απαντήσεις σε κάθε ερώτηση

Με αυτή τη μέθοδο είναι δυνατός ο καθορισμός κατά περίπτωση εξειδικευμένων (customized) μετρικών, ανάλογα με τους επιχειρησιακούς στόχους που τίθενται και τις ερωτήσεις στις οποίες καλούνται αυτές να απαντήσουν.



Εικόνα 2-2: Το πρότυπο Στόχος – Ερώτημα – Μετρική

Όπως είναι φανερό στην Εικόνα 2-2, η απάντηση σε ένα ερώτημα μπορεί να δίνεται από περισσότερες από μια μετρικές. Επίσης, μια μετρική μπορεί να απαντά σε διάφορα ερωτήματα που αναφέρονται στον ίδιον, καμιά φορά ακόμα και σε διαφορετικούς στόχους.

Ένα εφαρμοσμένο πρόγραμμα μετρήσεων επιχειρεί, με τη βοήθεια κατάλληλων μετρικών, να ποσοτικοποιήσει βασικές παραμέτρους ενός έργου λογισμικού, όπως οι ακόλουθες:

- Μέγεθος λογισμικού (software size)

- Προσπάθεια (effort)
- Σφάλματα (defects)
- Κόστος (cost)
- Ικανοποίηση του χρήστη (user satisfaction)

Λόγω του μεγέθους των σύγχρονων εφαρμογών και των ιδιαίτερων απαιτήσεων που προκύπτουν (π.χ. εξειδικευμένες προδιαγραφές για ενσωματωμένα συστήματα κοκ.), καταφεύγουμε πλέον σε ολοκληρωμένα **προγράμματα ποιοτικής αξιολόγησης** (software quality evaluation program). Πρόκειται για δέσμες τυπικών διαδικασιών εκτίμησης (για τα ποιοτικά) και υπολογισμού (για τα ποσοτικά γνωρίσματα) πάνω στα προϊόντα λογισμικού, αλλά και στα τεκμήρια, στις διαδικασίες και στις λειτουργίες που τα συνοδεύουν. Για τα ποσοτικά γνωρίσματα κάνουμε μετρήσεις ώστε να ελέγξουμε κατά πόσο πληρούνται οι στόχοι ως προς κάποια κριτήρια ή παλαιότερα δεδομένα (συγκριτική ανάλυση). Από την άλλη μεριά, η εκτίμηση των ποιοτικών χαρακτηριστικών εξαρτάται (και) από υποκειμενικές κρίσεις. Τέτοια προγράμματα εφαρμόζονται από ειδικές ομάδες που απασχολούνται ως αυτόνομα τμήματα μέσα στην ίδια την επιχείρηση ή, πολύ συχνά, από εξειδικευμένους φορείς εκτός αυτής. Για παράδειγμα, η αξιολόγηση ποιότητας ενός συστήματος ελέγχου αεροσκαφών μπορεί να γίνει από έναν μηχανικό αεροσκαφών, τους μηχανικούς λογισμικού, ένα τμήμα μηχανικών ελέγχου και τέλος μια ομάδα Διασφάλισης Ποιότητας Λογισμικού (software quality assurance, SQA), η οποία αποφαίνεται για την καθολική τήρηση όλων των αρχικών απαιτήσεων και προδιαγραφών. (4)

Η διασφάλιση ποιότητας *των διαδικασιών* κατά την ανάπτυξη έχει πολύ μεγάλη σημασία στην τεχνολογία λογισμικού. Γενικά θεωρείται καίριος παράγοντας ως προς την ποιότητα του τελικού προϊόντος (και κατ' επέκταση την πυκνότητα σε σφάλματα που απομένουν στο λογισμικό που παραδίδεται στο χρήστη).

Η διασφάλιση ποιότητας του ίδιου *του προϊόντος* παίζει επίσης κομβικό ρόλο στην τεχνολογία λογισμικού. Ένας βασικός λόγος είναι ότι η ανίχνευση και διόρθωση σφαλμάτων είναι κατά κανόνα η ακριβότερη διαδικασία τόσο κατά την ανάπτυξη (development) νέου συστήματος όσο και κατά τη συντήρηση (maintenance) ή αναβάθμιση (enhancement) του ήδη εν χρήσει λογισμικού.

Κατά τη διάρκεια ενός πλήρους κύκλου ζωής, περισσότερο από το 50% του συνολικού κόστους οφείλεται στα σφάλματα και τις αστοχίες λογισμικού (3). Η εμπειρία έχει αποδείξει ότι η αφαίρεση των σφαλμάτων γίνεται τόσο δυσκολότερη όσο αργότερα γίνονται αυτά αντιληπτά. Έτσι, είναι σημαντικό να διασφαλίζεται η ποιότητα από τις πρώτες κιόλας φάσεις του κύκλου ζωής του λογισμικού.Επίσης, έχει παρατηρηθεί ότι η βελτίωση στην ποιότητα κατά κανόνα συνοδεύεται από αξιόλογη αύξηση στην παραγωγικότητα. Αυτό το εκ πρώτης όψεως παράδοξο έχει την εξήγησή του εάν λάβουμε υπόψη τον κύκλο ζωής του λογισμικού ως ενιαίο σύνολο και όχι αποσπασματικά. Πράγματι, έχουμε πολλές φορές έργα που εμφανίζουν φαινομενικά μεγάλη παραγωγικότητα στις πρώτες φάσεις, τα οποία όμως μένουν ξαφνικά στάσιμα όταν φτάνουν στη φάση του ελέγχου (όπου «αποκαλύπτεται» ξαφνικά πλήθος συσσωρευμένων σφαλμάτων) και μάλιστα καμιά φορά εγκαταλείπονται ημιτελή στην πορεία. Αντίθετα, έργα που δίνουν εξαρχής μεγάλη έμφαση στις διαδικασίες ποιότητας δεν εμφανίζουν καθυστερήσεις κατά την φάση ενσωμάτωσης (integration), τις διαδοχικές δοκιμασίες ελέγχου (testing) και τις διαδικασίες διασφάλισης ποιότητας (SQA) πριν την παράδοσή τους στον τελικό χρήστη (3).

3 ΜΕΤΡΙΚΕΣ ΛΟΓΙΣΜΙΚΟΥ

3.1 Μετρήσεις, μοντέλα και μετρικές

Οι όροι μέτρηση (measure), δείκτης (indicator) και μετρική (metric) δεν αναφέρονται πάντα στη βιβλιογραφία με μια ενιαία ερμηνεία και συχνά δημιουργείται σύγχυση μεταξύ τους. Ένας από τους πλέον δόκιμους ορισμούς (τον οποίο θα τηρήσουμε στη συνέχεια) είναι ο εξής: (4)

Μέτρηση (measure) είναι ένας αριθμός που αντιστοιχεί σε μια τιμή πάνω σε κάποια κλίμακα. Η διαδικασία της μέτρησης (measurement) αφορά στη διενέργεια υπολογισμών πάνω σε διάφορες μετρικές και στην αναφορά των σχετικών αποτελεσμάτων.

Μετρική είναι ένας συνδυασμός δύο ή περισσότερων μεγεθών ή ιδιοτήτων. Το βασικό χαρακτηριστικό μιας μετρικής είναι ότι η τιμή της χρησιμεύει ως μέτρο της ποιότητας ή της επίτευξης κάποιου στόχου. Οι μετρικές μπορούν να είναι απλές ή σύνθετες. Οι πρώτες καταγράφουν ποσότητες (π.χ. πλήθος σφαλμάτων), μεγέθη (π.χ. μέγεθος ενός έργου λογισμικού), κατηγορίες (π.χ. τύποι εφαρμογών) και διαβαθμίσεις (π.χ. σοβαρότητα μιας αστοχίας). Οι σύνθετες μετρικές υπολογίζονται από δύο ή περισσότερες απλούστερες, συνήθως με μια αριθμητική πράξη: πρόκειται για αναλογίες (π.χ. λάθη ανά χίλιες γραμμές κώδικα), ποσοστά (π.χ. ποσοστό εφαρμογών ιστού επί του συνόλου), αλγεβρικούς γραμμικούς συνδυασμούς και τέλος δείκτες (μεγέθη χωρίς διαστάσεις, κατά κανόνα αθροίσματα κανονικοποιημένα στη συνέχεια ως προς κάποια βάση αναφοράς) (12)

Στην πράξη, **μετρικές λογισμικού** (software metrics) είναι ένας γενικός όρος που περιγράφει μια ολόκληρη γκάμα δραστηριοτήτων σε σχέση με μετρήσεις στην τεχνολογία λογισμικού, από τον απλό υπολογισμό τιμών π.χ. για κάποια γνωρίσματα του κώδικα μέχρι ολοκληρωμένα μοντέλα για την πρόγνωση σε απαιτήσεις πόρων και ποιότητα λογισμικού. Αυτό το φάσμα καλύπτει επίσης την ποσοτικοποιημένη πλευρά της διασφάλισης ποιότητας, όπως για παράδειγμα την καταγραφή και διαχείριση σφαλμάτων κατά την φάση ανάπτυξης και τις διαδικασίες ελέγχου. (13)

Με μια πιο αυστηρή μαθηματική διατύπωση, η πράξη της μέτρησης είναι μια απεικόνιση της εμπειρικής πραγματικότητας σε ένα φορμαλιστικό, σχεσιακό σύμπαν. Με αυτή την έννοια, μέτρηση είναι ένας αριθμός ή σύμβολο που αποδίδεται σε μια οντότητα μέσω αυτής της απεικόνισης και χαρακτηρίζει μια ιδιότητα αυτής της οντότητας. Είναι σημαντικό να τονιστεί ότι η διαδικασία της μέτρησης έχει τότε μόνο νόημα όταν η εμπιερία μας «μεταφράζεται» με συνέπεια σε αφηρημένες αριθμητικές σχέσεις: κάθε φορά που διαπιστώνουμε με την εμπειρία μας ότι, για παράδειγμα, «ο Α είναι ψηλότερος από τον Β», το μετρούμενο ύψος του Α πρέπει πάντα να αντιστοιχεί σε μεγαλύτερη τιμή από αυτήν για τον Β. (14)

Σε κάθε μέτρηση θα έχουμε αναγκαστικά ένα (μεγάλο ή μικρό) σφάλμα, το οποίο αντιστοιχεί στη διαφορά της μετρούμενης τιμής από την πραγματική και οφείλεται στη μετρητική συσκευή, αλλά και το στοχαστικό χαρακτήρα της ίδιας της διαδικασίας της μέτρησης. **Ορθότητα** (accuracy) μέτρησης είναι η τάση της μετρητικής συσκευής να προσεγγίζει την πραγματική τιμή. Από την άλλη πλευρά, η **ακρίβεια** (precision) μέτρησης αναφέρεται στην ικανότητα της συσκευής να περιορίζει την διαφοροποίηση στις παρατηρήσεις κατά τη διενέργεια διαδοχικών μετρήσεων. Η αβεβαιότητα στη μέτρηση προκύπτει ως συνδυαστικό αποτέλεσμα των παραπάνω ατελειών. Η έννοια του **θορύβου** (noise) στη μέτρηση αναφέρεται στο ανεπιθύμητο σήμα, το οποίο παραμορφώνει το πραγματικό. Για παράδειγμα, κατά τη μέτρηση των γραμμών κώδικα σε ένα έργο λογισμικού, οι γραμμές σχολίων μπορούν να θεωρηθούν ως θόρυβος. (15)

Πρέπει να σημειωθεί ότι, σε περίπτωση που ο μέσος όρος μιας μακράς σειράς διαδοχικών μετρήσεων ενός μεγέθους δεν προσεγγίζει ασυμπτωτικά την αληθή τιμή, δεν επιτυγχάνεται απόλυτη ορθότητα στη μέτρηση και προκύπτει **μεροληψία** (bias) η αλλιώς **συστηματικό σφάλμα**. Εξάλλου, η ακρίβεια σχετίζεται άμεσα με τη στατιστική διακύμανση και το **τυπικό σφάλμα** (standard error) της μέτρησης.

Όπως έχει ήδη αναφερθεί προηγούμενα, οι οντότητες πάνω στις οποίες διενεργούμε μετρήσεις στην τεχνολογία λογισμικού ανήκουν κατά βάση σε τρεις κατηγορίες: (14)

- **Διαδικασίες** (processes). Μπορούμε λόγου χάρη να μετρήσουμε τη διάρκεια μιας διαδικασίας, το κόστος της, αν είναι αποτελεσματική ή αποδοτική, όπως επίσης να τη συγκρίνουμε με άλλες εναλλακτικές διαδικασίες με παρόμοια χαρακτηριστικά, με κάποιο πρότυπο ή μια βασική γραμμή αναφοράς (baseline). Ένα τέτοιο σύστημα μετρήσεων αναπτύχθηκε τη δεκαετία του 1990 στο Ινστιτούτο Τεχνολογίας Λογισμικού (Software Engineering Institute – SEI, Carnegie Mellon University) με τα αρχικά **CMM** και στη συνέχεια **CMMI** (Capability Maturity Model Integration). Ο Πίνακας 3-1 δείχνει τη διασύνδεση της βαθμολογίας CMM με τα σφάλματα που παραδίδονται στο χρήστη ανά μονάδα FP (βλ. παράγραφο 4.2): (2)

Επίπεδο SEI CMM	Δυναμικό σφαλμάτων	Αποτελεσματικότητα αφαίρεσης σφαλμάτων	Παραδοθέντα σφάλματα
1	5	85%	0,75
2	4	89%	0,44
3	3	91%	0,27
4	2	93%	0,14
5	1	95%	0,05

Πίνακας 3-1: Επίπεδα CMM και σφάλματα λογισμικού

- **Προϊόντα** (products). Οι σχετικές μετρήσεις δεν πραγματοποιούνται αποκλειστικά πάνω στο τελικό προϊόν που παραδίδεται στο χρήστη, αλλά και σε ενδιάμεσα τεχνουργήματα (artefacts) όπως μοντελοποιήσεις των απαιτήσεων ή της σχεδίασης σε UML, πρωτότυπα (prototypes), υλικό τεκμηρίωσης κτλ.
- **Πόροι** (resources). Αναφερόμαστε εδώ σε μετρήσεις που αφορούν σε ανθρώπινο δυναμικό, υλικό, εργαλεία (υλικού και λογισμικού) και μεθοδολογίες ανάπτυξης.

Μπορούμε να διακρίνουμε επίσης τις μετρήσεις σε:

- *Αντικειμενικές* (όπου διαφορετικές μετρήσεις στο ίδιο αντικείμενο δίνουν ταυτόσημα αποτελέσματα) και *υποκειμενικές* (που εξαρτώνται από το περιβάλλον ή το πρόσωπο το οποίο τις πραγματοποιεί)
- *Ποσοτικές* (που μπορούν να εκφραστούν σε μια κλίμακα) και *ποιοτικές* (για χαρακτηριστικά που δεν μπορούν να αποδοθούν ενιαία με μια τιμή)

Τα μετρούμενα γνωρίσματα μιας οντότητας τέλος μπορούν να είναι είτε:

- *Εσωτερικά*, που για τον υπολογισμό τους αρκεί η ενδελεχής γνώση του προϊόντος, πόρου ή διαδικασίας που μελετάμε και δεν εξαρτώνται από τη συμπεριφορά και την αλληλεπίδραση με εξωτερικούς παράγοντες ή
- *Εξωτερικά*, όταν η μέτρηση οφείλει να λάβει υπόψη της και την αλληλεπίδραση της οντότητας με το περιβάλλον.

Για παράδειγμα, το μέγεθος και η δομή της αρχιτεκτονικής ή η πολυπλοκότητα του κώδικα ανήκουν στην πρώτη κατηγορία. Τα σφάλματα (εκτός αυτών που ανιχνεύονται ήδη από τη στατική ανάλυση), οι αστοχίες κατά τη λειτουργία ή ο χρόνος απόκρισης του συστήματος απαιτούν εκτέλεση του κώδικα σε δυναμικό χρόνο και άρα συγκαταλέγονται στα εξωτερικά χαρακτηριστικά.

Το κλειδί που αξιοποιεί τις μετρήσεις ως ένα εργαλείο της ανάλυσης λογισμικού είναι η υιοθέτηση του μοντέλου. Γενικά, το **μοντέλο** προϋποθέτει εξ ορισμού μια αφαίρεση η οποία παραμερίζει τους επουσιώδεις παράγοντες και επικεντρώνεται μόνο σε εκείνους με την ισχυρότερη επίδραση πάνω στο φαινόμενο που εξετάζεται. Στην τεχνολογία λογισμικού συναντούμε μοντέλα ποιότητας, εκτίμησης κόστους, ικανότητας ωριμότητας (capability maturity model - CMM) κ.ά. Με τη βοήθεια των μοντέλων μπορούμε από τη μια να αξιολογήσουμε υπάρχοντα στατιστικά δεδομένα ως προς κάποιες σημαντικές παραμέτρους, από την άλλη να προβούμε σε στατιστική συμπερασματολογία για δεδομένα που θα προκύψουν στο μέλλον. Για παράδειγμα, προκειμένου για ένα σύστημα με υψηλές απαιτήσεις αξιοπιστίας όπως αυτό του ελέγχου ενός αεροσκάφους, χρειάζεται, από την αρχή ήδη του κύκλου ζωής, να διασφαλιστεί ότι το σύστημα εκπληρώνει τις απαιτούμενες προδιαγραφές. Καταρχήν, η αξιοπιστία ορίζεται με βάση τις αστοχίες του λογισμικού κατά τη φάση της λειτουργίας, είναι όμως προφανώς αδύνατο να περιμένουμε ως τότε για να αξιολογήσουμε το επίπεδό του. Αντίθετα, χρησιμοποιώντας ως παραμέτρους μεγέθη που θεωρούνται ισχυρές *μεταβλητές πρόγνωσης* της αξιοπιστίας, θα πρέπει να κατασκευάσουμε ένα μοντέλο κατάλληλο για το πρόβλημά μας και να

υπολογίσουμε τις εκτιμήτριες αξιοπιστίας βάσει των δεδομένων του συστήματος, όσο αυτό είναι ακόμα σε εξέλιξη.

Τα *εποπτικά μοντέλα* εκφράζουν σχέσεις με τη βοήθεια κειμένων και σχημάτων. Απεικονίζουν πληροφορίες (π.χ. διαγράμματα οντοτήτων – συσχετίσεων), στοιχεία σε αντικειμενοστραφές περιβάλλον (διαγράμματα UML) ή οντολογίες (δηλ. δομημένη γνώση πάνω σε κάποιο πεδίο). Διαθέτουν παρασταστική ικανότητα, παρέχουν όμως μάλλον γενικές κατευθύνσεις και δεν συνιστούν κατά κανόνα ικανή βοήθεια στην περιγραφή των αλληλεπιδράσεων ανάμεσα στις μεταβλητές.

Τα *αλγοριθμικά (ή παραμετρικά) μοντέλα*, από την άλλη πλευρά, είναι κατά κανόνα ισχυρά εργαλεία, καθώς συναρμόζουν τα μετρούμενα μεγέθη σε μαθηματικές σχέσεις – ντετερμινιστικές ή στοχαστικές. Ένα παράδειγμα βλέπουμε στους ακόλουθους γενικούς τύπους, στη λογική αυτών που εισήγαγε ο Barry Boehm με το μοντέλο COCOMO στα τέλη της δεκαετίας του ‘70 (και τις νεώτερες εκδόσεις COCOMOII / 1996 και COCOMO 2000). Εδώ υπολογίζεται η προσπάθεια (Effort E, κατά κανόνα εκφρασμένη σε ανθρωπο-μήνες PM) και η διάρκεια του έργου (Duration D, σε μήνες) σε συνάρτηση με το μέγεθος του κώδικα (Size S, σε χιλιάδες γραμμές πηγαίου κώδικα KSLOC):

$$E = a * S^b \qquad \text{και} \qquad D = c * E^d$$

Οι παράμετροι a, b, c και d εκτιμώνται βάσει ενός μεγάλου αριθμού εμπειρικών δεδομένων με κατάλληλες στατιστικές τεχνικές, όπως δείχνει ο Πίνακας 3-2. (10)

Είδος του έργου λογισμικού	Προσπάθεια	Διάρκεια
Οργανικά (organic)	$E = 2,4 * KLOC^{1,05}$	$D = 2,5 * E^{0,38}$
Ημιαυτόνομα (semi-detached)	$E = 3,0 * KLOC^{1,12}$	$D = 2,5 * E^{0,35}$
Ενσωματωμένα (embedded)	$E = 3,6 * KLOC^{1,2}$	$D = 2,5 * E^{0,32}$

Πίνακας 3-2: Εμπειρικοί τύποι για το μοντέλο COCOMO

3.2 Κλίμακες μετρήσεων

Κάθε μέτρηση προϋποθέτει την αντιστοίχιση των δεδομένων σε ένα πεδίο τιμών, το οποίο αποτελεί την κλίμακα και, ανάλογα με τις ιδιότητές του, απαιτεί ανάλογο χειρισμό κατά την ανάλυση των αποτελεσμάτων. Διακρίνουμε πέντε κατηγορίες, καθεμιά από τις οποίες αποτελεί υποσύνολο της προηγούμενης: (14) (10)

3.2.1 Ονομαστικές κλίμακες.

Στις ονομαστικές (nominal) κλίμακες το πεδίο τιμών αποτελείται από διακριτές τιμές χωρίς κάποια διάταξη. Τα στοιχεία που ανήκουν σε διαφορετικές κλάσεις δεν σχετίζονται μεταξύ τους με οποιοδήποτε τρόπο. Παράδειγμα η κατηγοριοποίηση των λαθών σε σφάλματα προδιαγραφών,

σχεδιασμού ή κωδικοποίησης, ανάλογα με τη φάση κατά την οποία εισήχθησαν στο λογισμικό, χωρίς με αυτό να υπονοείται κάποια διάταξη ή μεγαλύτερη σημασία των μεν από τα δε.

3.2.2 Κλίμακες διάταξης.

Η ονομαστική κλίμακα εμπλουτίζεται με μια σχέση γραμμικής διάταξης (ordinal) ανάμεσα στις κλάσεις. Εδώ έχει νόημα η έκφραση «μεγαλύτερο / μικρότερο από» αλλά δεν νοούνται αλγεβρικές πράξεις ανάμεσα σε δύο τιμές της κλίμακας. Αυτές οι κλίμακες απαντώνται αρκετά συχνά στην τεχνολογία λογισμικού, καθώς αποτυπώνουν διαβαθμίσεις, π.χ. ως προς την σπουδαιότητα στις αστοχίες. Σε αυτή την περίπτωση, δεν μπορεί να γίνει υπολογισμός μέσων τιμών και τυπικών αποκλίσεων και επομένως δεν έχουν εφαρμογή οι συνήθεις παραμετρικές τεχνικές στατιστικής ανάλυσης.

3.2.3 Κλίμακες διαστήματος.

Εδώ γνωρίζουμε επιπλέον το διάστημα (interval) ή την απόσταση ανάμεσα στις τιμές. Στην κατηγορία αυτή με άλλα λόγια έχει μεν νόημα η πρόσθεση δύο τέτοιων τιμών, όχι όμως ο πολλαπλασιασμός ή η αναλογία μεταξύ τους. Για παράδειγμα, η πρόταση «σήμερα είναι δύο φορές πιο ζεστή μέρα από χτες» δεν έχει προφανώς κανένα νόημα[5].

3.2.4 Κλίμακες λόγου.

Μια κλίμακα λόγου (ratio) εμπεριέχει επιπλέον, πέρα από τα προηγούμενα, την πληροφορία για την αναλογία, την έννοια της μοναδιαίας τιμής και το σημείο μηδέν της κλίμακας. Έτσι, έχουν πλέον νόημα ο πολλαπλασιασμός και η διαίρεση μεταξύ δύο τιμών, επίσης δε οι έννοιες της μέσης τιμής και της διασποράς του πληθυσμού. Ως παράδειγμα στην τεχνολογία λογισμικού είναι η μέτρηση του μεγέθους κώδικα με διάφορες μονάδες μέτρησης π.χ. LOC, KLOC, πλήθος χαρακτήρων πολλαπλασιάζοντας με το (μέσο) αριθμό χαρακτήρων ανά γραμμή, Λειτουργικά Σημεία (FP) βάσει μιας (εμπειρικής) αναλογίας μεταξύ LOC και FP κοκ.

3.2.5 Απόλυτες κλίμακες.

Η τελευταία κατηγορία είναι η πιο περιοριστική από όλες. Ως τιμές λογίζονται μόνο το μηδέν και οι θετικοί ακέραιοι. Οι τιμές εδώ έχουν τυπικά το νόημα της απαρίθμησης των εμφανίσεων ενός φαινομένου, όπως π.χ. ο αριθμός αστοχιών κατά τη φάση λειτουργίας του έργου λογισμικού.

[5] Ένα κλασικό παράδειγμα είναι η μέτρηση της θερμοκρασίας με τις κλίμακες Κελσίου C ή Φαρενάιτ F (μέσω του μετασχηματισμού: *F = 9/5 * C + 32*).

4 ΜΕΓΕΘΟΣ ΛΟΓΙΣΜΙΚΟΥ

Η μέτρηση του μεγέθους είναι κεντρικής σημασίας, καθώς αποτελεί βασική παράμετρο για πολλές από τις μετρικές που καθορίζουν την ποιότητα, την παραγωγικότητα και το κόστος του λογισμικού. Οι μετρικές αυτές έχουν ως γνώμονα είτε το *«τεχνικό»* μέγεθος του κώδικα (που χρησιμεύει στη μέτρηση προϊόντων λογισμικού από τη σκοπιά του προγραμματιστή) είτε το *«λειτουργικό»* μέγεθος (για τη μέτρηση προϊόντων λογισμικού από την πλευρά του χρήστη).

4.1 Μετρικές γραμμής κώδικα

Στα πρώτα χρόνια της τεχνολογίας λογισμικού, η μέτρηση του μεγέθους επικράτησε να γίνεται με υπολογισμό των γραμμών κώδικα (LOC-based metrics). Βέβαια, προκύπτει διαφορετικό αποτέλεσμα ανάλογα με το εάν παίρνουμε ως βάση τις «φυσικές» – αυτές που ορίζονται με την αλλαγή γραμμής του δρομέα – ή τις «λογικές» γραμμές (αυτές που συνιστούν τον εκτελέσιμο κώδικα και είναι συχνά περισσότερες από μια στην ίδια φυσική γραμμή). Το αποτέλεσμα διαφοροποιείται ακόμα κατά το εάν εξαιρούνται οι γραμμές δήλωσης δεδομένων, οι γραμμές σχολίων, οι κενές γραμμές κοκ. (10) Αξίζει να σημειωθεί ότι οι εκτελέσιμες γραμμές κώδικα αποτελούν περίπου το 40-45% (ανάλογα με το είδος του έργου) του συνολικού αριθμού ενώ ποσοστό 30-35% συνιστούν οι δηλώσεις δεδομένων, 10% οι κενές γραμμές και 15% τα σχόλια. Στην πράξη συναντούμε συχνά και τον «νεκρό κώδικα», δηλαδή τμήματα του αρχικού κώδικα που έχουν παρακαμφθεί ή αντικατασταθεί από νεώτερο λόγω κάποιας αναβάθμισης ή διόρθωσης σφαλμάτων. Τα τμήματα αυτά μπορεί να έχουν παραμείνει στον τελικό κώδικα είτε για ασφάλεια (σε περίπτωση που η διόρθωση δεν είναι επιτυχής) είτε γιατί το κόστος απόσυρσής τους είναι μεγάλο. Προκειμένου για «κληρονομημένα» έργα (legacy projects) με ηλικία ήδη πάνω από μια δεκαετία, ο νεκρός κώδικας μπορεί να αποτελεί μέχρι και το 15% του συνολικού και επομένως επηρεάζει τη μέτρηση LOC. Σε κάθε περίπτωση, είναι σημαντικό να τηρούνται ενιαίοι κανόνες απαρίθμησης των γραμμών κατά τις συγκρίσεις μεταξύ διαφορετικών έργων. (3)

Από τη δεκαετία του 1970 και μετά, αναπτύχθηκε μια κριτική απέναντι στις μετρικές γραμμής κώδικα. Κάποιες από τις κυριότερες αιτίες είναι οι εξής:

- Η κύρια «κατηγορία» αφορά στην αδυναμία των μετρικών γραμμής κώδικα να ποσοτικοποιήσουν τη λειτουργικότητα του έργου από τη μεριά του τελικού χρήστη, παραμένοντας αποκλειστικά προσκολλημένες στην πλευρά του «προγραμματιστή».
- Καθώς οι μετρικές γραμμών κώδικα αναφέρονται αποκλειστικά στη φάση της κωδικοποίησης, δεν μπορούν να ενσωματώσουν πληροφορίες από άλλες φάσεις του κύκλου ζωής του λογισμικού (συλλογή απαιτήσεων και προδιαγραφών, σχεδιασμός, φάσεις ενσωμάτωσης και ελέγχου κοκ.). Σήμερα όμως είναι κυρίως αυτές οι φάσεις που απορροφούν το μεγαλύτερο μέρος της συνολικής

προσπάθειας και επομένως επηρεάζουν σημαντικά το πραγματικό μέγεθος του έργου. Άλλωστε, με τις γλώσσες προγραμματισμού τελευταίας γενιάς (αντικειμενοστραφείς, εποπτικές κτλ.), μπορούμε να γράφουμε μικρότερο κώδικα και άρα απαιτούνται λιγότεροι «ανθρωπο-μήνες» για την κωδικοποίηση. Ήδη από τη δεκαετία του 1970, η κωδικοποίηση δεν ξεπερνά πλέον το 30% της συνολικής προσπάθειας, η δε αποσφαλμάτωση του κώδικα (debugging) απορροφά ένα επιπλέον ποσοστό 30%. Το υπόλοιπο 40% της συνολικής προσπάθειας αντιστοιχεί στις φάσεις απαιτήσεων, προδιαγραφών, τεκμηρίωσης κ.ά. (3) Υπολογίζεται άλλωστε ότι, για εφαρμογές άνω των 5000 γραμμών πηγαίου κώδικα (αντίστοιχα άνω των 100FP – βλ. παρακάτω), το ποσοστό των σφαλμάτων κωδικοποίησης δεν ξεπερνά το 20% του συνολικού αριθμού σφαλμάτων. (2)

- Ο αριθμός των γραμμών κώδικα εξαρτάται από τη χρησιμοποιούμενη γλώσσα προγραμματισμού. Είναι φανερό ότι ένα δομικό στοιχείο κώδικα με δεδομένη λειτουργικότητα, αν και κατά τεκμήριο αντιπροσωπεύει την ίδια προσπάθεια στην υλοποίηση, παρουσιάζει πολύ μεγάλες αποκλίσεις στον αριθμό γραμμών κώδικα ανάλογα με τη γλώσσα. Πρέπει να σημειωθεί ότι σήμερα υπάρχουν στην αγορά πάρα πολλά πρόσφατα ή παλαιότερα συστήματα γραμμένα σε διάφορες γλώσσες προγραμματισμού (ο αριθμός των οποίων ξεπερνάει τις 700!) και αυτό το γεγονός κάνει εντονότερο το πρόβλημα. Ιδιαίτερα προκειμένου για μη-διαδικασιακές ή αντικειμενοστραφείς γλώσσες, οι μετρικές γραμμής κώδικα μπορούν να οδηγήσουν σε μεγάλες αποκλίσεις και ασυνέπειες – π.χ. στη Visual Basic μεγάλο μέρος της κωδικοποίησης γίνεται περισσότερο μέσα από pull-down μενού και κουμπιά επιλογής και λιγότερο με διαδικασιακό κώδικα.
- Ένα άλλο ζήτημα με τις μετρικές γραμμής κώδικα είναι ότι οι γλώσσες υψηλότερου επιπέδου εμφανίζονται σαν λιγότερο «παραγωγικές». Αυτό οφείλεται στο γεγονός ότι μεγάλο κομμάτι της συνολικής προσπάθειας (σε ανθρωπο-μήνες) για την παραγωγή ενός δομικού στοιχείου του προγράμματος αφορά φάσεις εκτός κωδικοποίησης, π.χ. συλλογή απαιτήσεων, προδιαγραφές, τεκμηρίωση, σχεδίαση κοκ. Αυτό το κομμάτι είναι κατά τεκμήριο σε μεγάλο βαθμό ανεξάρτητο της γλώσσας κωδικοποίησης και επομένως αποτελεί ένα αρκετά υψηλό έμμεσο (overhead) κόστος. Επομένως, ο αντίστοιχος κώδικας εμφανίζεται να έχει τόσο χαμηλότερη παραγωγικότητα όσο λιγότερες γραμμές κώδικα περιέχει, όσο δηλαδή η γλώσσα είναι υψηλότερου επιπέδου και άρα πιο περιεκτική. (3)

Πρέπει πάντως να σημειωθεί ότι οι μετρικές γραμμής κώδικα μπορούν, υπό κάποιες προϋποθέσεις, να είναι μια, αν όχι πάντοτε ακριβής, πάντως αρκετά αξιόπιστη μέθοδος. Κάνοντας μια υπερ-απλουστευτική παρομοίωση, η συνολική έκταση σε τετραγωνικά μέτρα ενός κτιρίου όπως φαίνεται σε μια αρχιτεκτονική κάτοψη είναι καταρχήν ένα χρήσιμο και αξιόπιστο μέγεθος, αν και προφανώς το «πραγματικό» μέγεθος του έργου αναδεικνύεται σε μεγάλο βαθμό από τη λειτουργικότητα της οικοδομής και άλλους παράγοντες που επηρεάζουν την ποιότητα του σχεδίου. Πρέπει επίσης να έχουμε υπόψη μας ότι ο υπολογισμός των γραμμών κώδικα μπορεί να εφαρμοστεί χωρίς ειδικές γνώσεις και επομένως είναι μια χαμηλού κόστους, εύκολη και προσιτή μέθοδος για κάθε ομάδα ανάπτυξης λογισμικού.

4.2 Μέτρηση Λειτουργικού Μεγέθους

Η ολοένα και ισχυρότερη κριτική απέναντι στις μετρικές γραμμής κώδικα οδήγησε στην ανάγκη να καθοριστεί ένας εναλλακτικός τρόπος υπολογισμού του μεγέθους του λογισμικού, με έμφαση κυρίως στο «τι καλείται να κάνει» το σύστημα και όχι στο «πώς το επιτυγχάνει» εσωτερικά. Αυτός ο εναλλακτικός τρόπος μέτρησης είναι γνωστός ως Μέτρηση Λειτουργικού Μεγέθους (Functional Size Measurement). Η πρώτη τέτοια πρόταση προήλθε από τον Alan Albrecht της IBM το 1979 και παραμένει ακόμα και σήμερα η δημοφιλέστερη, μετά από αλεπάλληλες προσθήκες και βελτιώσεις. Πρόκειται για την **Ανάλυση Λειτουργικών Σημείων** (Function Point Analysis), η οποία αναγνωρίζεται ως ένα από τα αποδεκτά πρότυπα ISO και έχει οδηγήσει στη σύσταση του International Function Point Users Group – IFPUG, με πάνω από 1400 μέλη παγκοσμίως. Σήμερα, απαντώνται διάφορες παραλλαγές στη γενική λογική των Function Points (FP) και, πιο συγκεκριμένα, τα παρακάτω μοντέλα με ISO - πιστοποίηση:

IFPUG4.1 (ISO 20926)

MarkII (ISO 20968 –αναπτύχθηκε στην Αγγλία από τον Charles Symons τη δεκαετία του ’80)

NESMA (ISO 24570 – Netherlands Software Measurement Association, πρότυπο αντίστοιχο με τα FP version 4.1)

FISMA (ISO 29881 – Finnish Software Measurement Association) και

COSMIC Full Function Points (ISO 19761 – Common Software Measurement International Consortium)

Οι παραπάνω 5 μέθοδοι μέτρησης του μεγέθους λογισμικού αντιπροσωπεύουν συνολικά γύρω στο 97% του συνόλου των περιπτώσεων σε παγκόσμια κλίμακα. Σπανιότερα απαντώνται τα Feature Points (στην πραγματικότητα πρόκειται για ένα «υπερσύνολο» των Function Points), Object Points, Use Case Points (με συχνή εφαρμογή σε συστήματα αντικειμενοστραφούς λογικής) κ.ά. (3)

Η κεντρική ιδέα πίσω από την Ανάλυση FP και τις άλλες μεθόδους που αναπτύχθηκαν παράλληλα είναι η έννοια του **λειτουργικού μεγέθους λογισμικού**. Αυτό προκύπτει μέσα από την ποσοτικοποίηση των λειτουργικών απαιτήσεων του χρήστη. Σαν τέτοιες μπορούμε να αναφέρουμε τη μεταφορά, τη μετατροπή (π.χ. υπολογισμοί, αριθμητικές και άλλες πράξεις), την αποθήκευση και την ανάκτηση δεδομένων. Από την άλλη μεριά, ως μη λειτουργικές απαιτήσεις λογίζονται π.χ. οι περιορισμοί ποιότητας (αξιοπιστία, αποτελεσματικότητα, μεταφερσιμότητα), οργάνωσης (υλικό, τοποθεσίες κοκ.), περιβάλλοντος (ασφάλεια, δια-λειτουργικότητα) ή εκτέλεσης (γλώσσα εφαρμογής, χρονοδιάγραμμα παράδοσης). Οι τελευταίες αναφέρονται και ως τεχνικές και ποιοτικές απαιτήσεις.

Με λίγα λόγια, το λειτουργικό μέγεθος αντανακλά το τι οφείλει να εκτελέσει το λογισμικό όπως φαίνεται από την σκοπιά του χρήστη, ανεξάρτητα από το πώς είναι αυτό δομημένο εσωτερικά ή τι επίπεδα απόδοσης έχει να επιδείξει. (16)

4.2.1 Μέθοδος υπολογισμού λειτουργικών σημείων

Παρακάτω θα εξετάσουμε τη βασική λογική της μεθόδου υπολογισμού FP, που στηρίζεται στα ακόλουθα σημεία: (10)

1. Η λειτουργικότητα του συστήματος καθορίζεται με βάση 5 παράγοντες (*inputs, outputs, external interface files, internal data files* και *inquiries*)[6]
2. Ο βαθμός δυσκολίας κάθε παράγοντα ορίζεται κατά περίπτωση ως *απλός, μέτριος* ή *σύνθετος* (simple, average, complex)
3. Ο Πίνακας 4-1 δίνει το βαθμό πολυπλοκότητας για κάθε παράγοντα, με γνώμονα τον τύπο και τη δυσκολία:

Παράγοντας	Απλός	Μέτριος	Σύνθετος
Inputs (I)	3	4	6
Outputs (O)	4	5	7
Data files (F)	7	10	15
Interfaces (N)	5	7	10
Inquiries (Q)	3	4	6

Πίνακας 4-1: Βαθμοί πολυπλοκότητας για την ΑΛΣ

Οι βαθμοί πολυπλοκότητας αντιπροσωπεύουν τη σχετική προσπάθεια στην υλοποίηση. Για παράδειγμα, μια μέτριας δυσκολίας διαπροσωπία (interface) υλοποιείται κατά κανόνα δυσκολότερα από ό,τι μια αντίστοιχης δυσκολίας ερώτηση (inquiry), γι' αυτό και το σχετικό βάρος της πρώτης είναι 7 έναντι 4 της δεύτερης, με άλλα λόγια η πρώτη χρειάζεται περίπου 1,75 φορές μεγαλύτερη προσπάθεια από τη δεύτερη. Οι βαθμοί δυσκολίας καθορίζονται βάσει λεπτομερών κριτηρίων που έχουν τεθεί από το International Function Point Users Group (IFPUG).

4. Τα *μη προσαρμοσμένα λειτουργικά σημεία* (unadjusted function points, UFP) υπολογίζονται με άθροιση όλων των παραγόντων πολλαπλασιασμένων με τους αντίστοιχους βαθμούς πολυπλοκότητας
5. Ο συντελεστής προσαρμογής (value adjustment factor - VAF) υπολογίζεται βάσει της πολυπλοκότητας του εν γένει συστήματος

Η προσαρμογή γίνεται με γνώμονα τη δυσκολία στην υλοποίηση του συστήματος. Παρακάτω δίνεται ένας πίνακας Γενικών Χαρακτηριστικών του Συστήματος (ΓΧΣ / General System Characteristics, GSC) με βαθμολόγηση για το καθένα από αυτά από 0 (δηλαδή καμία επίδραση) μέχρι 5 (σοβαρή επίδραση). Τα 14 αυτά χαρακτηριστικά, που ενσωματώνουν κατά κάποιο τρόπο στον τελικό υπολογισμό τις μη λειτουργικές απαιτήσεις και την επίδρασή τους, καταγράφει ο Πίνακας 4-2 :

[6] Στη μέθοδο των feature points συνυπολογίζουμε επιπλέον τον παράγοντα *algorithms*

Ο παράγοντας VAF προκύπτει από την άθροιση των βαθμών επίδρασης των 14 χαρακτηριστικών, π.χ. ένας χαμηλός βαθμός *= 1* για μόλις 7 από τα GSC δίνει *VAF = 7 * 1 = 7*, ενώ βαθμός *= 5* και για τα 14 παραπάνω χαρακτηριστικά, δίνει το μέγιστο αποτέλεσμα για τον *VAF = 14 * 5 = 70*

6. Τα *προσαρμοσμένα λειτουργικά σημεία* (adjusted function points, AFP) υπολογίζονται με βάση τον τύπο: *AFP = UFP * TCF*, όπου *TCF (technical complexity factor) = 0,65 + 0.01 * VAF*. Επομένως, τα UFP πολλαπλασιάζονται επί ένα συντελεστή που κινείται ανάμεσα στο 0,65 και το 1,35 – ενσωματώνει δηλαδή μια απόκλιση +-35% σε σχέση με τα αρχικά UFP, λαμβάνοντας υπόψη τις δυσκολίες (ή την απουσία τους) που εκτιμάται ότι θα αντιμετωπίσει η συγκεκριμένη εφαρμογή κατά την υλοποίηση και ανάπτυξη στο τελικό της περιβάλλον.

data communications, distributed data/processing, performance objectives,
heavily used configuration, transaction rate, online data entry, end-user efficiency,
online update, complex processing, reusability, conversion / installation ease,
operational ease, multiple site use, facilitate change.

Πίνακας 4-2: Γενικά Χαρακτηριστικά Συστήματος για την ΑΛΣ

Παραδείγματος χάρη, ένα πρόγραμμα με 4 απλές εισόδους, ένα απλό αρχείο πληροφορίας και δύο εξόδους μέτριας δυσκολίας δίνει τα ακόλουθα αποτελέσματα:

*Inputs = 4 * 3 = 12*

*Data file = 1 * 7 = 7*

*Outputs = 2 * 5 = 10*

UAF = 12 + 7 + 10 = 29

Αν υποθέσουμε ακόμη ότι, για τη συγκεκριμένη εφαρμογή, όλα τα χαρακτηριστικά GSC έχουν χαμηλό βαθμό δυσκολίας *1* εκτός των πεδίων *complex processing* και *data communication*, που έχουν και τα δύο υψηλό βαθμό *5*, θα έχουμε τελικά:

*VAF = 12 * 1 + 2 * 5 = 22*

*AFP = UFP * (0,65 + 0.01 * VAF) = 29 * (0,65 + 0,22) = 25,23*

4.2.2 Πλεονεκτήματα των λειτουργικών σημείων

Η καθιέρωση της μεθόδου FP είναι αναμφίβολα ένα μεγάλο βήμα σε ό,τι αφορά στα μοντέλα και εργαλεία εκτίμησης παραμέτρων του λογισμικού. Τα Function Points παρουσιάζουν μεγάλα πλεονεκτήματα:

- Είναι ανεξάρτητα της τεχνολογίας και της γλώσσας που χρησιμοποιείται και επομένως μπορούν να χρησιμεύσουν σε συγκρίσεις και εξαγωγή συμπερασμάτων μέσα από ένα πλήθος έργων λογισμικού
- Είναι μια μέθοδος υπολογισμού μέσω των λειτουργικών («τι καλείται να κάνει το σύστημα») και μη λειτουργικών («πώς θα το κάνει») προδιαγραφών, και όχι μέσω του κώδικα. Μπορούν επομένως να εφαρμοστούν ήδη από τις αρχικές φάσεις του κύκλου ζωής του λογισμικού, πολύ πριν την κωδικοποίηση, και άρα παρέχεται τη δυνατότητα έγκαιρης πρόβλεψης του τελικού κόστους, του χρόνου παράδοσης κοκ.
- στηρίζονται σε σαφή πρότυπα και λεπτομερώς καταγεγραμμένους κοινά αποδεκτούς κανόνες και
- είναι αρκούντως ακριβή και αξιόπιστα. Ενδεικτικά αναφέρεται ότι η απόκλιση στον υπολογισμό του μεγέθους λογισμικού σε FP από δύο ανεξάρτητους πιστοποιημένους ελεγκτές δεν υπερβαίνει συνήθως το 10%, αρκεί οι απαιτήσεις του χρήστη να είναι σαφείς και λεπτομερώς προσδιορισμένες (16)

4.2.3 Μειονεκτήματα των λειτουργικών σημείων

Παρά τα σημαντικά πλεονεκτήματά τους, τα FP δεν είναι πλήρως απαλλαγμένα από κάποιες αδυναμίες, ανάμεσα στις οποίες ως κυριότερες μπορούμε να αναφέρουμε τις ακόλουθες:

- Ο υπολογισμός των FP από μη εξειδικευμένο και πιστοποιημένο προσωπικό δεν είναι ευχερής και έτσι προκύπτει αυξημένο κόστος εφαρμογής της μεθόδου (17)
- Η ερμηνεία των βασικών παραγόντων εμπεριέχει ασάφειες – ιδιαίτερα προκειμένου για νέες μεθοδολογίες (agile λογισμικό, εφαρμογές ιστού κ.ά.). Σε τέτοιες περιπτώσεις η καταγραφή ενός στοιχείου (π.χ. Graphical User Interface- GUIs) σε έναν από τους παραπάνω πέντε παράγοντες δε μπορεί να γίνει μονοσήμαντα. Προκειμένου για εφαρμογές ιστού, μπορούμε μεν να κάνουμε τις συνήθεις μετρήσεις που περιγράφονται παραπάνω ως προς το μέγεθος του λογισμικού, δεν διαθέτουμε όμως αντίστοιχες μετρικές για τον υπολογισμό του μεγέθους του περιεχομένου – και επομένως δεν υπάρχει τρόπος να ποσοτικοποιήσουμε την ποιότητα του περιεχομένου ιστού. (3)
- Υπάρχει αυθαιρεσία ως ένα βαθμό στον καθορισμό του βαθμού πολυπλοκότητας για κάθε παράγοντα (βλ. πίνακα ανωτέρω), ομοίως δε στη βαθμολόγηση των ΓΧΣ (από 1 έως 5). (17)
- Η αξιολόγηση των χαρακτηριστικών ΓΧΣ και κατ' επέκταση ο υπολογισμός του αντίστοιχου συντελεστή προσαρμογής ενέχουν ένα στοιχείο υποκειμενικότητας.
- Η μέθοδος των λειτουργικών σημείων αδυνατεί να ανταποκριθεί με την ίδια επιτυχία σε κάποιους τομείς ανάπτυξης λογισμικού, με δεδομένο ότι τα FP αναπτύχθηκαν κατά κύριο λόγο πάνω στο λογισμικό επιχειρησιακών εφαρμογών. Ένας τομέας όπου παρουσιάζεται αυτή η δυσκολία είναι για παράδειγμα το σε πραγματικό χρόνο (real time), πολλαπλών επιπέδων (multi-layered) λογισμικό (π.χ. εφαρμογές τηλεπικοινωνιών, ελέγχου διαδικασιών, λειτουργικά συστήματα κοκ.) (17)

Πρέπει να σημειωθεί ότι η ολοκλήρωση της μεθόδου στο πρώτο στάδιο (αυτό των UFP) και πριν την ενσωμάτωση των ΓΧΣ είναι γρηγορότερη και έχει σημαντικά μικρότερο κόστος. Θα μπορούσε βέβαια εύλογα να αντιτείνει κανείς ότι ο υπολογισμός του μεγέθους λογισμικού με βάση τα AFP

(αντί των UFP) υπερτερεί κατά τεκμήριο σε ακρίβεια. Στη βιβλιογραφία πάντως μπορούμε να συναντήσουμε αναφορές σε έρευνες που δεν καταγράφουν στατιστικά σημαντική βελτίωση στην εκτίμηση πόρων για μοντέλα που λαμβάνουν υπόψη τα ΓΧΣ. (14) (17) Πρέπει εξάλλου να σημειωθεί ότι τα καθιερωμένα και σε κοινή χρήση διεθνή πρότυπα για την Ανάλυση με Function Points (όπως το IFPUG 4.1 κ.ά.) χρησιμοποιούν κατά κανόνα ως βάση υπολογισμού τα UFP και όχι τα AFP.

5 ΣΦΑΛΜΑΤΑ ΛΟΓΙΣΜΙΚΟΥ

5.1 Σφάλματα και αστοχίες

Στην τεχνολογία λογισμικού, γίνεται συχνά αδιακρίτως λόγος για σφάλματα, ελαττώματα, λάθη, αστοχίες κτλ., έννοιες με παραπλήσια σημασία. Σύμφωνα πάντως με το πρότυπο ISO 1044-2009, ο όρος **σφάλμα** (defect) δεν αφορά σε αστοχίες αλλά μόνο σε λάθη που εισάγονται στο σύστημα εξαιτίας κάποιας προβληματικής συμπεριφοράς *κατά την ανάπτυξη του λογισμικού*, π.χ. λάθη στη συλλογή των απαιτήσεων, στη σχεδίαση, στην κωδικοποίηση (τα γνωστά «bugs») κοκ. Από την άλλη μεριά, **αστοχία** (failure) είναι η απομάκρυνση του συστήματος από την αναμενόμενη συμπεριφορά *κατά τη λειτουργία του*. Σχετική με την έννοια της αστοχίας είναι και αυτή της **αξιοπιστίας**, στην ουσία μια στοχαστική μεταβλητή που ορίζεται ως η πιθανότητα R(t) το σύστημα να μην έχει συναντήσει αστοχία ως μια δεδομένη στιγμή κατά τη λειτουργία του σε κάποιο συγκεκριμένο περιβάλλον.

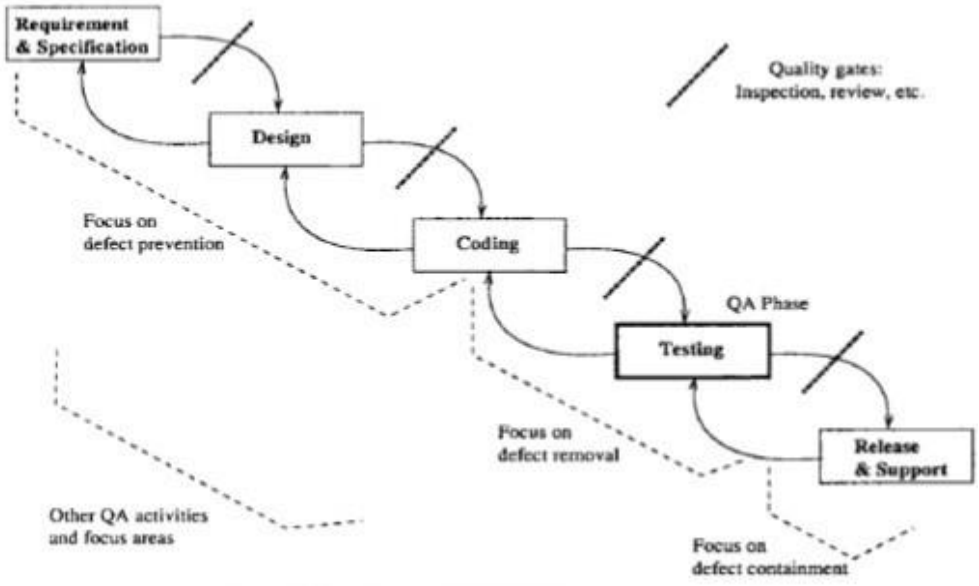
$$R(\tau) = P[\text{system is on at } t=\tau] = 1 - P[\text{system has failed before } t=\tau].$$

Κατά μια έννοια, μπορούμε να θεωρήσουμε τα σφάλματα και τις αστοχίες σαν την εσωτερική και εξωτερική όψη του συστήματος. Τα σφάλματα είναι τα προβλήματα στο λογισμικό όπως τα αντιμετωπίζει ο προγραμματιστής, οι δε αστοχίες είναι τα προβλήματα όπως φαίνονται από την μεριά του χρήστη. (14) Κατ' αντιστοιχία με την παραπάνω διάκριση, διακρίνουμε *μετρικές σφαλμάτων*, οι οποίες αναφέρονται στα ελαττώματα[7] του λογισμικού, και *μετρικές αξιοπιστίας*, που ασχολούνται με τις αστοχίες των πληροφορικών συστημάτων. (10) Μολαταύτα, πρέπει να έχουμε υπόψη μας ότι δεν υφίσταται μια ευθεία συσχέτιση των ελαττωμάτων του λογισμικού με τις αστοχίες που θα συναντήσουμε κατά τη λειτουργία του. Ακόμα και η ακριβέστερη πρόβλεψη της πυκνότητας των σφαλμάτων που παραμένουν στο λογισμικό δεν παρέχει επαρκή πρόγνωση της επιχειρησιακής αξιοπιστίας του συστήματος. Πράγματι, έρευνες (18) καταδεικνύουν ότι το 33% του συνολικού αριθμού σφαλμάτων αφορούν σε ατυχήματα με *μέσο χρόνο ως την αστοχία* (mean time to failure – MTTF) μεγαλύτερο των 5000 ετών. Αντίθετα, το ποσοστό των σφαλμάτων που ευθύνονται για αστοχίες με MTTF λιγότερο από 50 έτη δεν είναι παραπάνω από μόλις 2%.

Στην πράξη μπορούμε να διακρίνουμε κατά βάση τρεις τρόπους χειρισμού των σφαλμάτων, οι οποίοι εφαρμόζονται είτε ξεχωριστά είτε σε συνδυασμό μεταξύ τους: (9)

[7] Εφεξής θα αναφερόμαστε στους όρους «ελάττωμα» και «λάθος» ως συνώνυμα του όρου «σφάλμα».

- **αποτροπή**, δηλαδή πρόληψη κάποιων κατηγοριών σφαλμάτων. Αυτή επιτυγχάνεται είτε μέσω προσεκτικής καταγραφής απαιτήσεων με λεπτομερή τρόπο και τυπικές μεθόδους, είτε μέσω εφαρμογής αυστηρών προτύπων και κατευθυντήριων γραμμών κατά τις πρώτες φάσεις της ανάπτυξης
- **ανίχνευση και αφαίρεση**, δηλαδή διαχείριση των σφαλμάτων αφότου αυτά έχουν εισχωρήσει στο σύστημα. Βασικά αυτό επιτυγχάνεται είτε με *προ-ελεγκτικές* (pre-test) μεθόδους όπως τυπικές επιθεωρήσεις (formal inspections) και στατική ανάλυση (static analysis), είτε με μια αλληλουχία σταδίων *ελέγχου*, και τέλος
- **περιορισμός**, δηλαδή να μην επιτραπεί στις συνέπειες ενός σφάλματος να εξαπλωθούν αλλά να «υποχρεωθούν» να παραμείνουν μέσα στα σύνορα του δομικού στοιχείου το οποίο έχουν ήδη «μολύνει». Με τον τρόπο αυτό επιτυγχάνεται είτε η πρόληψη μιας αστοχίας – που θα οφειλόταν στην ενεργοποίηση κάποιου λανθάνοντος σφάλματος κατά την εκτέλεση – είτε, εάν αυτό δεν είναι εφικτό, ο περιορισμός της ζημιάς που θα επιφέρει η αστοχία αυτή. Συναφής με αυτό είναι η έννοια της **ανοχής σφάλματος** (fault tolerance), η διάρρηξη με άλλα λόγια της σχέσης αιτιότητας ανάμεσα στο σφάλμα και την αστοχία, έτσι ώστε τοπικά σφάλματα να μην οδηγούν σε γενικευμένες αστοχίες.



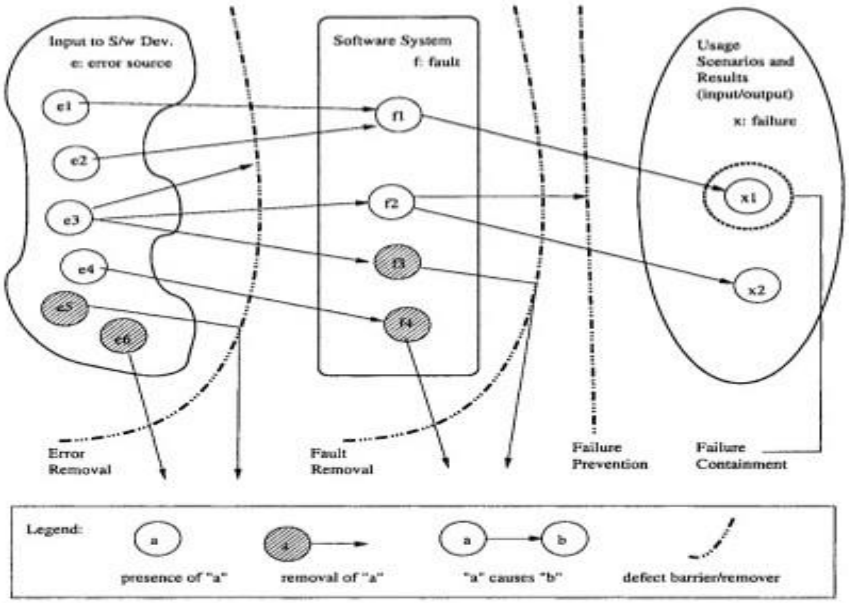
Εικόνα 5-1: Δραστηριότητες διασφάλισης ποιότητας στο μοντέλο καταρράκτη

Η Εικόνα 5-1 δείχνει παραστατικά τις διάφορες δραστηριότητες διασφάλισης ποιότητας (QA) και την εφαρμογή τους στο μοντέλο του *καταρράκτη*. Τα τρία σημεία που χαρακτηρίζουν αυτή την κατανομή των δραστηριοτήτων είναι:

- η κύρια φάση είναι αυτή του ελέγχου
- οι μεταβάσεις από τη μια φάση στην επόμενη – κατά κανόνα επιθεωρήσεις και εκθέσεις – λειτουργούν σαν πύλες που πρέπει να περάσει κανείς με επιτυχία (οι κεκλιμμένες γραμμές του σχήματος). Στην τελευταία τέτοια πύλη αντί για μια ακόμα επιθεώρηση έχουμε τον έλεγχο αποδοχής (acceptance testing)

- όπως φαίνεται στο σχήμα (βλ. τις διακεκομμένες γραμμές), οι πρώιμες φάσεις εστιάζουν στην αποτροπή, η κωδικοποίηση και ο έλεγχος στην αφαίρεση και η φάση της λειτουργικής υποστήριξης στον περιορισμό αντίστοιχα

Στα *επαυξητικά* και *επαναληπτικά* μοντέλα αυτό το σχήμα εφαρμόζεται σε κάθε μια επανάληψη, με την ολοκλήρωση της οποίας το νέο τμήμα που μόλις αναπτύχθηκε πρέπει να ενσωματωθεί στο ήδη υπάρχον. Οι δοκιμασίες ολοκλήρωσης (integration testing) αποκτούν λοιπόν εδώ ιδιαίτερο βάρος, ώστε να εξασφαλιστεί ότι τα διάφορα μέρη συνεργάζονται μεταξύ τους χωρίς πρόβλημα και υλοποιούν σωστά τη ζητούμενη λειτουργικότητα. Εξάλλου, στο *σπειροειδές* μοντέλο η ανάλυση ρίσκου, που αποτελεί συστατικό μέρος της διαδικασίας, μας οδηγεί σε επιλεκτική διασφάλιση ποιότητας, με τα μέρη υψηλότερου ρίσκου να απαιτούν μεγαλύτερη προσοχή από τα υπόλοιπα. (9)



Εικόνα 5-2: Διαχείριση σφαλμάτων και αστοχιών

Η Εικόνα 5-2 παρουσιάζει εποπτικά τη διαχείριση σφαλμάτων και αστοχιών. Βλέπουμε εδώ καταρχάς μια αλληλουχία από φράγματα που παριστάνονται με διάστικτες γραμμές. Με e_i συμβολίζουμε την πηγή λάθους (π.χ. ανθρώπινη λανθασμένη ενέργεια), με f_i το σφάλμα και με x_i την αστοχία. Το φράγμα από το αριστερό στο μεσαίο τμήμα, που αντιπροσωπεύει την αποτροπή σφαλμάτων, δεν επιτρέπει στις πηγές λαθών να εισαγάγουν σφάλματα στο σύστημα. Το φράγμα μεταξύ μεσαίου και δεξιού τμήματος αναπαριστά τις δραστηριότητες αφαίρεσης σφαλμάτων, δηλαδή επιθεωρήσεις και ελέγχους. Η ευθεία γραμμή λίγο δεξιότερα παριστάνει την αποτροπή αστοχιών (π.χ. ανοχή σφάλματος). Το τελευταίο φράγμα, που περικυκλώνει επιλεγμένα στιγμιότυπα αστοχιών (όπως το x_1 στο σχήμα μας), συμβολίζει τις δραστηριότητες περιορισμού μιας αστοχίας. Έτσι, βλέπουμε για παράδειγμα το σφάλμα f_4 που αποσύρεται κατά την επιθεώρηση, το f_3 που εντοπίζεται και αφαιρείται κατά τον έλεγχο (δεύτερο

φράγμα), το f_2 στο οποίο δεν επιτρέπεται να εξελιχτεί σε αστοχία (εμποδίζεται από το τρίτο φράγμα) – αν και μπορεί το ίδιο σφάλμα μέσα σε ένα διαφορετικό δυναμικό περιβάλλον να «ξεπεράσει» το φράγμα και να οδηγήσει τελικά στην αστοχία x_2 – και τέλος την αστοχία x_1 που έχει περιοριστεί με επιτυχία. (9)

5.2 Εντοπισμός και αφαίρεση σφαλμάτων

Κεντρική σημασία στο χειρισμό των σφαλμάτων κατέχει η **δείκτης απόδοσης αφαίρεσης σφαλμάτων** (defect removal efficiency – DRE), που ορίζεται ως ο λόγος των σφαλμάτων που ανιχνεύτηκαν μέχρι την τελική εγκατάσταση του συστήματος δια του συνολικού αριθμού τους, συμπεριλαμβανομένων δηλαδή αυτών που ήρθαν ως ανάδραση (feedback) από τους χρήστες στη διάρκεια των πρώτων 90 ημερών ή, σε κάποιες περιπτώσεις, κατά τον πρώτο χρόνο λειτουργίας του λογισμικού. Μια μέτρια επίδοση για το δείκτη DRE είναι γύρω στο 70%, ενώ οι μεγάλες εταιρίες ανάπτυξης λογισμικού μπορούν να επιτύχουν τιμές έως και 90-95%. Πάντως, σε εξαιρετικά μεγάλα πακέτα του εμπορίου (Commercial-Off-The-Shelf, COTS) μεγέθους άνω των 100.000 FP, κατά τη στιγμή εξόδου του λογισμικού στην αγορά παραμένουν ακόμα κρυμμένα χιλιάδες λανθάνοντα σφάλματα και ο αντίστοιχος δείκτης DRE μπορεί να πέσει κάτω από το 90%. (3)

Ο **πίνακας αφαίρεσης σφαλμάτων** (defect removal matrix) καταγράφει τόσο τη φάση της εισαγωγής ενός σφάλματος όσο και αυτή της ανίχνευσης. Είναι χρήσιμος ως εργαλείο ανάλυσης, επιτρέποντας τον υπολογισμό της αποδοτικότητας άρσης των σφαλμάτων σε κάθε βήμα της διαδικασίας. (10)

Για παράδειγμα, έστω μια εφαρμογή στην οποία έχουμε ανίχνευση σφαλμάτων κατά την επισκόπηση απαιτήσεων, την επισκόπηση σχεδίασης, την φάση του ελέγχου και τέλος κατά τη λειτουργία από τον χρήστη. Ας θεωρήσουμε ακόμη ότι τα σφάλματα εισάγονται κυρίως κατά τις φάσεις των απαιτήσεων, της σχεδίασης και της κωδικοποίησης. Ο Πίνακας 5-1 παρουσιάζει τον αντίστοιχο πίνακα αφαίρεσης σφαλμάτων.

Defect Removal Step	**Defect Injection Phase**			
	Requirements	Design	Coding	**TOTAL**
Requirements Review	13			13
Design Review	2	12		14
Testing	3	5	32	40
Customer Detected	1	3	4	8
TOTAL	19	20	36	75

Πίνακας 5-1: Πίνακας αφαίρεσης σφαλμάτων

Ο αντίστοιχος δείκτης DRE για το παράδειγμά μας είναι

DRE για το σύστημα = Σύνολο πριν την παράδοση / Σύνολο ανιχνευθέντων = (13+14+40) / 75 = 89%.

Εξάλλου, στη φάση σχεδίασης υπήρχαν 26 συνολικά σφάλματα προς ανίχνευση (19 – 13 = 6, που δεν είχαν ανιχνευτεί στο προηγούμενο βήμα, συν 20 που προέκυψαν κατά την ίδια τη φάση σχεδίασης). Από αυτά ανιχνεύτηκαν τα 14 (2 σφάλματα προδιαγραφών + 12 σχεδίασης), άρα:

DRE για τη Φάση Επισκόπησης Σχεδίασης = 14 / 26 = 54%

Ο Πίνακας 5-2 δείχνει τα DRE ανά φάση του κύκλου ζωής του λογισμικού:

Φάση	DRE
Απαιτήσεις	0,68
Σχεδίαση	0,54
Έλεγχος	0,83
Ανιχνευθέντα από το χρήστη	1,00

Πίνακας 5-2: Οι Δείκτες Αφαίρεσης Σφαλμάτων για κάθε φάση

Από τα παραπάνω στοιχεία μπορούμε π.χ. να συναγάγουμε ότι, στη συγκεκριμένη περίπτωση, πρέπει αφενός να επιφέρουμε βελτιώσεις στις πρώτες φάσεις ώστε να μειωθεί ο αριθμός των σφαλμάτων που εισάγονται σε αυτές, αφετέρου να εντείνουμε τη διαδικασία επισκόπησης της σχεδίασης, καθώς η τιμή 54% υπολείπεται της συνήθους τιμής 60-70% που αναμένεται για αυτή τη φάση, με βάση συγκριτικά στοιχεία από άλλα έργα στο παρελθόν.

Ένα άλλο σχετικό με τα σφάλματα μέγεθος που μπορεί να χρησιμεύσει ως δείκτης της ποιότητας του λογισμικού είναι το **δυναμικό σφαλμάτων** (defect potential), που είναι η εκτίμηση του αριθμού λαθών ανά FP για καθεμιά απο τις πέντε κύριες γενεσιουργές αιτίες σφαλμάτων στο λογισμικό:

- Απαιτήσεις (requirements)
- Σχεδίαση (design)
- Κωδικοποίηση (coding)
- Έγγραφα χρήστη (user documents)
- Δευτερογενή λάθη (badfixes), δηλαδή νέα λάθη που παρεισφρύουν ως (ανεπιθύμητο) αποτέλεσμα της διόρθωσης παλαιότερων.

Ο Πίνακας 5-3 δείχνει το δυναμικό και την αντίστοιχη απόδοση αφαίρεσης σφαλμάτων για καθεμιά από τις πέντε παραπάνω «γεννήτριες» λαθών. (3) (18)

Ένας ακόμα δείκτης της ποιότητας λογισμικού είναι ο αριθμός των **σφαλμάτων που παραδίδονται στο χρήστη** (delivered defects), με το διαχωρισμό τους σε *κρίσιμα* (critical – πλήρης αναστολή λειτουργίας),

μεγάλα (significant – πρόβλημα σε κύρια λειτουργία), *μικρά* (minor – δυσχρησία, πρόβλημα σε δευτερεύουσα λειτουργία) και *επιφανειακά* (cosmetic – π.χ. ορθογραφικά λάθη σε μηνύματα). Πρόκειται για λανθάνοντα (latent) λάθη που υπάρχουν στο σύστημα και ενδέχεται κάποια στιγμή να ενεργοποιηθούν από το χρήστη. Είναι αξιοσημείωτο ότι, τη στιγμή παράδοσης του λογισμικού στο χρήστη, έχουν ανιχνευθεί περίπου το 95% των σφαλμάτων κωδικοποίησης, αλλά μόλις το 75% των λαθών στις απαιτήσεις του χρήστη και το 80% των σφαλμάτων σχεδίασης. Ο Πίνακας 5-4 παρουσιάζει μια προσεγγιστική κατανομή των σφαλμάτων λογισμικού, όπως απαντάται στην πράξη. (3)

Πηγή ελαττώματος	Δυναμικό	DRE	Παραμένοντα λάθη ανά FP
Λάθη στις απαιτήσεις	1,00	77%	0,23
Λάθη στη σχεδίαση	1,25	85%	0,19
Λάθη στον κώδικα	1,75	95%	0,09
Λάθη στην τεκμηρίωση	0,60	80%	0,12
Δευτερογενή λάθη	0.40	60%	0.12
Σύνολο	**5,00**	**85%**	**0,75**

Πίνακας 5-3: Δυναμικό και αποδοτικότητα αφαίρεσης σφαλμάτων

Σπουδαιότητα 1	Πλήρης αστοχία (1% των παραδοθέντων σφαλμάτων)
Σπουδαιότητα 2	Αστοχία κύριας λειτουργίας (11% των παραδοθέντων σφαλμάτων)
Σπουδαιότητα 3	Δευτερεύουσα αστοχία (50% των παραδοθέντων σφαλμάτων)
Σπουδαιότητα 4	Επιφανειακό λάθος (38% των παραδοθέντων σφαλμάτων)

Πίνακας 5-4: Κατανομή κατά σπουδαιότητα των σφαλμάτων λογισμικού

Ο Πίνακας 5-4 δείχνει ότι μόλις το 12% από τα εναπομείναντα στο σύστημα σφάλματα είναι υψηλής σπουδαιότητας (κατηγορίας 1 ή 2) και μπορούν να οδηγήσουν σε πλήρη ή βασική αστοχία ώστε τελικά να δημιουργήσουν πρόβλημα. ‘Αλλωστε, έχει παρατηρηθεί ότι ένας χρήστης θα συναντήσει κατά μέσο όρο το 1% του συνολικού αριθμού «εν υπνώσει» (dormant) λαθών υψηλής σπουδαιότητας στο σύστημα. Προφανώς, το ποσοστό αυτό είναι μεγαλύτερο για τους χρήστες των πλέον εξειδικευμένων και ασυνήθιστων λειτουργιών της εφαρμογής, καθώς εκεί αναμένονται κατά προτεραιότητα

περισσότερα τέτοια λάθη. Σε κάθε περίπτωση, μπορεί ένα λογισμικό με μεγάλο αριθμό εν υπνώσει σφαλμάτων να μην παρουσιάσει ποτέ αστοχία, με την προϋπόθεση αυτά τα λανθάνοντα σφάλματα να βρίσκονται σε δομικά στοιχεία του κώδικα που δεν θα ενεργοποιηθούν ποτέ κατά τη λειτουργία του συστήματος με τις συγκεκριμένες τιμές εισόδου που αναμένεται να εισαχθούν από το χρήστη. Από την άλλη μεριά, αρκεί μόλις ένα λανθάνον σφάλμα στο πρόγραμμα εκκίνησης (boot) του συστήματος ώστε να προκληθεί αστοχία και διακοπή της λειτουργίας. (3)

Ο Πίνακας 5-5 δείχνει τον αριθμό των σφαλμάτων ανά FP που παραδίνονται στο χρήστη, ανάλογα με τον τύπο λογισμικού (οι τιμές αφορούν στις καλύτερες επιδόσεις ανά κατηγορία): (3)

	Τάξη μεγέθους λογισμικού (σε FP)				
Τύπος λογισμικού	100	1000	10000	100000	Μέσος όρος
End-user	-	-	-	-	-
Web	0,01	0,07	0,15	-	0,08
MIS	0,05	0,11	0,21	0,35	0,18
US outsource	0,02	0,05	0,13	0,23	0,11
Offshore outsource	0,03	0,08	0,17	0,29	0,14
Commercial	0,06	0,15	0,23	0,34	0,20
Systems	0,03	0,12	0,15	0,22	0,13
Military	0,05	0,15	0,18	0,24	0,15
Average	**0,03**	**0,10**	**0,17**	**0,28**	**0,15**

Πίνακας 5-5: Αριθμός παραδοθέντων σφαλμάτων κατά τύπο λογισμικού

5.3 Προελεγκτικές μέθοδοι εντοπισμού σφαλμάτων

Έχει παρατηρηθεί στην πράξη ότι το χρονικό σημείο όπου πολλά έργα λογισμικού αρχίζουν να δείχνουν σημάδια ότι παρουσιάζουν σοβαρά προβλήματα είναι η φάση του ελέγχου. Όπως αναφέρεται και αλλού, πολλά από τα έργα που εγκαταλείπονται ή έστω παρουσιάζουν μεγάλες καθυστερήσεις εμφανίζουν μια παραπλανητική εικόνα μέχρι να αρχίσει ο έλεγχος, οπότε και εμφανίζεται στην επιφάνεια μια σωρεία κρυμμένων μέχρι τότε λαθών. Ένας σοβαρός λόγος που συνηγορεί υπέρ του προελεγκτικού εντοπισμού και αφαίρεσης σφαλμάτων είναι λοιπόν ο περιορισμός του κόστους και του

συνολικού χρόνου που απαιτείται για την ανάπτυξη του λογισμικού[8]. Πράγματι, η εφαρμογή των προελεγκτικών μεθόδων μπορεί να περιορίσει τη διάρκεια και το κόστος του καθαυτό ελέγχου κατά 30% ή και παραπάνω. Τόσο από οικονομική άποψη όσο και από αυτήν της διασφάλισης ποιότητας, η αποτροπή, ο προελεγκτικός εντοπισμός και ο τυπικός έλεγχος είναι και οι τρεις απολύτως απαραίτητες διεργασίες έτσι ώστε να επιτύχουμε χαμηλό κόστος, σύντομη διάρκεια ανάπτυξης και χαμηλό επιπέδο σφαλμάτων στο λογισμικό που παραδίδεται στο χρήστη.

Οι συνηθέστερες μέθοδοι προελεγκτικού εντοπισμού και αφαίρεσης σφαλμάτων είναι ο προσωπικός επιτραπέζιος εξέλεγχος (personal desk checking), η αυτοματοποιημένη ανάλυση κειμένου, οι συνεδρίες scrum (scrum sessions) , ο προγραμματισμός κατά ζεύγη, οι αξιολογήσεις από ομοτίμους (peer reviews) και από τον πελάτη (client reviews), οι τυπικές επιθεωρήσεις και η στατική ανάλυση. Οι δύο τελευταίες τεχνικές πιστώνονται με τη μεγαλύτερη αποτελεσματικότητα στον εντοπισμό σφαλμάτων πριν την έναρξη του κυρίως ελέγχου. (2)

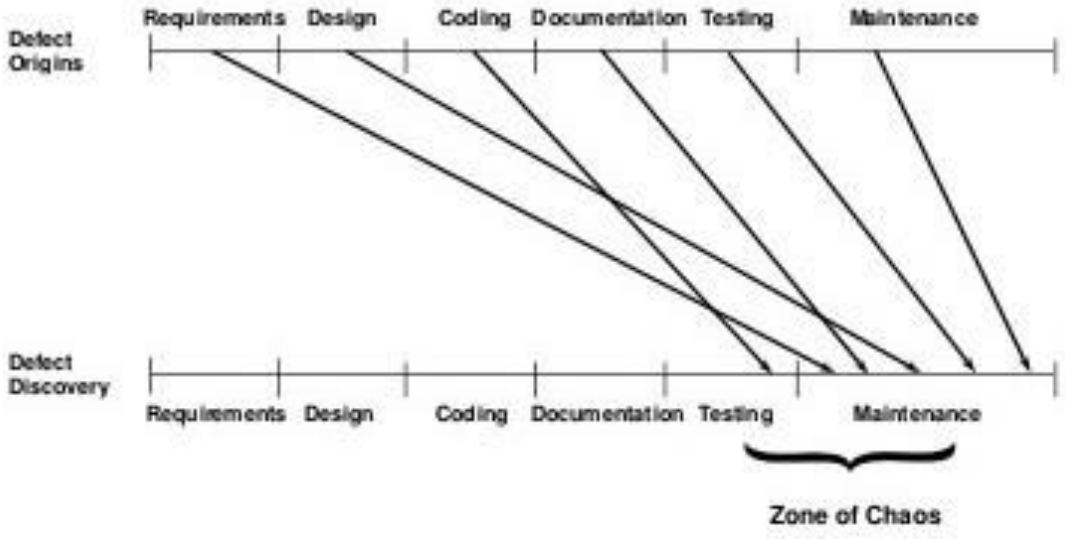
Οι τυπικές επιθεωρήσεις (formal inspections) ακολουθούν κατά κανόνα συγκεκριμένες και τυποποιημένες διαδικασίες. Εφαρμόζονται στο ενδιάμεσο προϊόν κάθε φάσης του κύκλου ζωής (απαιτήσεις, σχεδιασμός, κωδικοποίηση, έλεγχος, τεκμηρίωση κτλ.) και επιτρέπουν έτσι την ανίχνευση ενός σφάλματος όσο το δυνατόν πλησιέστερα στο χρονικό σημείο εισαγωγής του στο λογισμικό. Αποφεύγεται έτσι ο συνωστισμός πολλών λαθών με διαφορετικά χαρακτηριστικά στις υστερότερες φάσεις ελέγχου και συντήρησης (testing & maintenance), οπότε και θα υπήρχε ο κίνδυνος χαοτικών καταστάσεων, ακόμα και κατάρρευσης. Άλλωστε, οι επιθεωρήσεις πιστώνονται με τη μεγαλύτερη επίδοση στον δείκτη DRE από οποιοδήποτε άλλο τρόπο αφαίρεσης σφαλμάτων, 85% κατά μέσο όρο. Ας σημειωθεί εδώ ότι οι τεχνικές αυτές δεν έχουν την ίδια αποτελεσματικότητα προκειμένου για έργα μικρότερου μεγέθους (κάτω από 500 FP) ή για ευέλικτες (agile) μεθοδολογίες.

Η Εικόνα 5-3 δείχνει παραστατικά τη μεγάλη χρονική απόσταση που υφίσταται ανάμεσα στην εισαγωγή και τον εντοπισμό των σφαλμάτων, όταν κατά την ανάπτυξη εφαρμόζεται μόνη η διαδικασία του ελέγχου. Τα βέλη των γραμμών δείχνουν όλα προς την ίδια περιοχή στα δεξιά του σχήματος και τα σφάλματα συσσωρεύονται στις δύο τελευταίες φάσεις του κύκλου ζωής. Είναι προφανώς προτιμότερο οι γραμμές να είναι κατά το δυνατόν κάθετες στη χρονική ροή. Πράγματι, η Εικόνα 5-4 δείχνει το συγκριτικά μικρότερο χρονικό διάστημα που μεσολαβεί ανάμεσα στην εισαγωγή και τον εντοπισμό χάρη στις επιθεωρήσεις που εφαρμόζονται σε κάθε φάση ξεχωριστά. (2) (3)

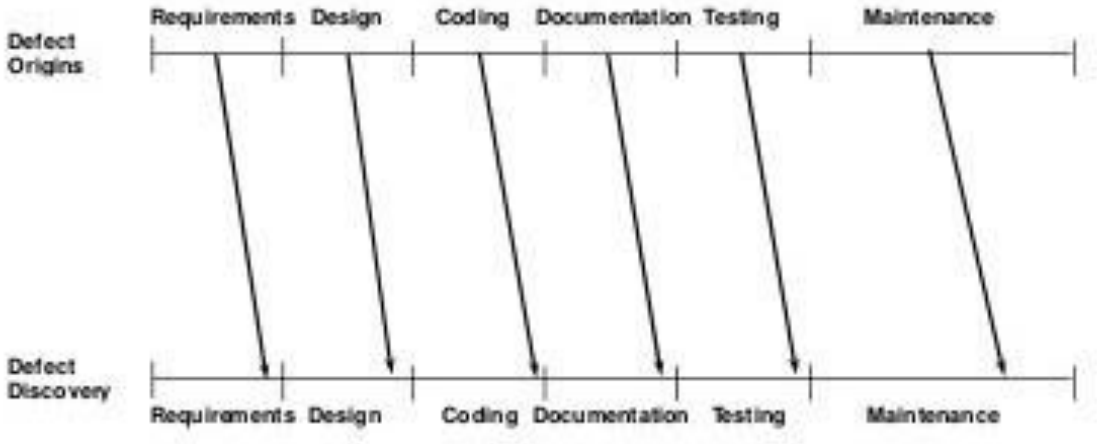
- *Η αυτοματοποιημένη στατική ανάλυση του πηγαίου κώδικα* (automated static analysis of source code) εφαρμόζεται τόσο στα τεχνουργήματα της φάσης σχεδιασμού όσο και στον ίδιο τον κώδικα και υποστηρίζεται από πολλά εργαλεία με πολύ καλές επιδόσεις στη γρήγορη ανεύρεση σφαλμάτων. Η αποτελεσματικότητα και αυτής της μεθόδου στην ανεύρεση και αφαίρεση λαθών φτάνει το 85% κατά μέσο όρο. Λόγω του χαμηλού κόστους και της υψηλής επίδοσής τους, οι εφαρμογές στατικής ανάλυσης έχουν καταστεί το κατεξοχήν εργαλείο ποιότητας για λογισμικό ανοικτού κώδικα (open source software). Βέβαια, από τις εκατοντάδες γλωσσών προγραμματισμού τα εργαλεία αυτά δεν

[8] Με δεδομένη τη σημασία των προελεγκτικών μ εθόδων εντοπισμού σφαλμάτων, είναι αξιοσημείωτη η έλλειψη σχετικής βιβλιογραφίας σε βαθμό ανάλογο αυτής που αφορά στον έλεγχο λογισμικού. Από μια γρήγορη έρευνα στο διαδίκτυο προκύπτει μια αναλογία 1:20 ανάμεσα στις προελεγκτικές και ελεγκτικές μεθόδους αντίστοιχα (2)

υποστηρίζουν παρά ένα μικρό υποσύνολο, όχι παραπάνω από 50 γλώσσες συνολικά. Επίσης, τα εργαλεία αυτά τείνουν να είναι πιο αργά και λιγότερο αποδοτικά όταν έχουν να αντιμετωπίσουν εφαρμογές που έχουν υλοποιηθεί με έναν συνδυασμό διαφορετικών γλωσσών (π.χ. COBOL και SQL για παλαιότερα ή Java και HTML για πιο πρόσφατα έργα). Ένα ακόμα σχετικό ζήτημα είναι οι «ψευδο-επιτυχίες» (false positives): το εργαλείο της στατικής ανάλυσης εκλαμβάνει ενίοτε ως σφάλματα διάφορα στοιχεία του προγραμματισμού, τα οποία στην πραγματικότητα δεν είναι τέτοια. Παλιότερα, το ποσοστό των ψευδο-επιτυχιών μπορούσε να ξεπεράσει το 10%. Σήμερα πάντως, με τα βελτιωμένα εργαλεία στατικής ανάλυσης, το ποσοστό αυτό κινείται μεταξύ 3% και 7%. (2)



Εικόνα 5-3: Εντοπισμός σφαλμάτων χωρίς προ-ελεγκτικές μεθόδους



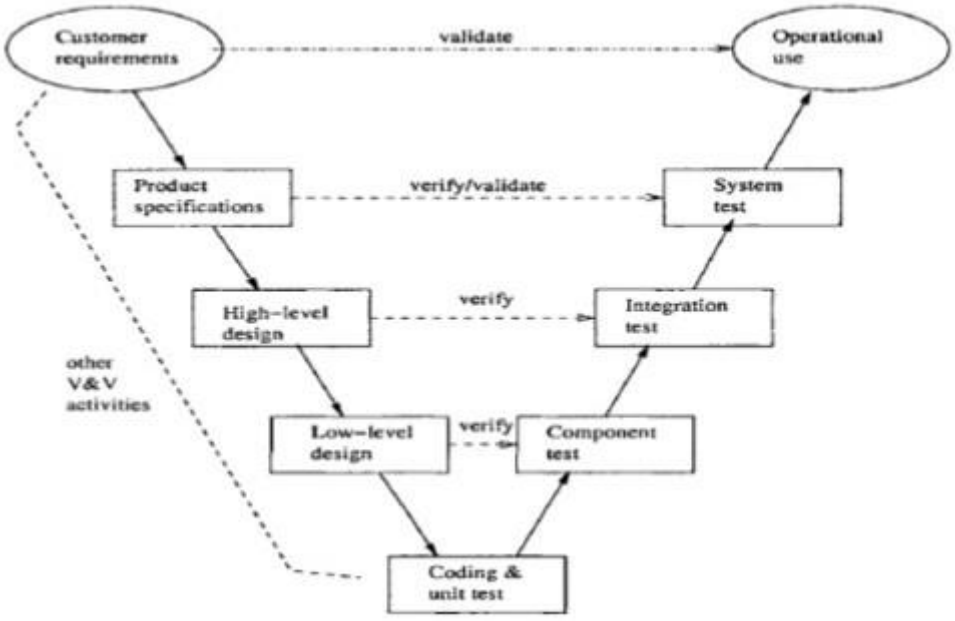
Εικόνα 5-4: Εντοπισμός σφαλμάτων με τυπικές επιθεωρήσεις

5.4 Έλεγχος λογισμικού

Ο έλεγχος είναι μια κεντρική διαδικασία για τη διασφάλιση της ποιότητας λογισμικού. Το πεδίο είναι ευρύ, η σχετική βιβλιογραφία εκτενής και μια αναλυτική παράθεση ξεφεύγει από τα πλαίσια της παρούσας μελέτης. Στα επόμενα θα περιοριστούμε λοιπόν σε λίγα βασικά σημεία.

5.4.1 Επαλήθευση και επικύρωση

Οι δραστηριότητες διασφάλισης ποιότητας που εξασφαλίζουν ότι εκτελούνται οι σωστές λειτουργίες ("do the right job"), δηλαδή αυτές που αναμένονται από τον χρήστη και έχουν καταγραφεί στο στάδιο των απαιτήσεων, συναπαρτίζουν τις δραστηριότητες της **επικύρωσης** (validation). Εξάλλου, εκείνες που μερμνούν για την ορθή και αξιόπιστη επίδοση αυτών των λειτουργιών ("do the job right") συναποτελούν τις δραστηριότητες της **επαλήθευσης** (verification). Οι τελευταίες ελέγχουν το συμβατό ενός συστήματος λογισμικού ως προς τις προδιαγραφές του. Η επικύρωση αφορά λοιπόν άμεσα στους χρήστες και στις απαιτήσεις τους ενώ η επαλήθευση στις εσωτερικές προδιαγραφές του προϊόντος.



Εικόνα 5-5: Επαλήθευση και Επικύρωση (V-Model)

Η Εικόνα 5-5 παρουσιάζει τις δύο παραπάνω έννοιες στο πλαίσιο του **μοντέλου V** (V-model). Πρόκειται για μια επέκταση του μοντέλου του καταρράκτη με ορθή φορά για την ανάπτυξη του λογισμικού και αντίστροφη φορά για τη φάση του ελέγχου. Εδώ η εμπλοκή του χρήστη περιορίζεται στην αρχή και το τέλος της πορείας, άρα στο ανώτερο τμήμα του σχήματος, όπου τον κύριο ρόλο έχει η επικύρωση. Στο κατώτερο τμήμα το βάρος πέφτει στην επαλήθευση και στη συμμόρφωση προς τις εσωτερικές προδιαγραφές, κανόνες και κριτήρια, χωρίς επαφή με το χρήστη. Σε άλλα μοντέλα ανάπτυξης, όπως το επαυξητικό, το εξελικτικό ή το σπειροειδές υπάρχει κάποιος βαθμός συμμετοχής του χρήστη σε κάθε μέρος ή επανάληψη, επομένως ο ρόλος της επικύρωσης είναι αυξημένος. (9)

5.4.2 Δοκιμασίες ελέγχου

Μια βασική κατηγοριοποίηση των διαφόρων δοκιμασιών είναι αυτή ανάμεσα στον δομικό (structural) και τον λειτουργικό (functional) έλεγχο. Κατά τον **δομικό έλεγχο** (ή έλεγχο του «διαφανούς κουτιού», white-box testing / WBT) μπορεί κανείς να δει στο εσωτερικό και να διακρίνει τις μονάδες (εντολές, δομές δεδομένων κοκ.) και τις σχέσεις αναμεταξύ τους. Ο **λειτουργικός έλεγχος** (ή έλεγχος του «μαύρου κουτιού», black-box testing / BBT) εστιάζει στην εξωτερική συμπεριφορά του συστήματος και των συστατικών στοιχείων του και βλέπει το αντικείμενο που ελέγχει από μακριά, χωρίς πρόσβαση στο εσωτερικό του. Ο πρώτος αφορά κατά κύριο λόγο σε μικρά στοιχεία λογισμικού και στις πρώτες υποφάσεις του ελέγχου, αντίθετα με τον δεύτερο που εφαρμόζεται σε μεγαλύτερες οντότητες (ολόκληρα συστήματα ή τμήματά τους) και στα υστερότερα στάδια. Έτσι, οι δοκιμασίες του διαφανούς κουτιού έχουν να κάνουν περισσότερο με την επαλήθευση και αντίστοιχα του μαύρου κουτιού με την επικύρωση. (9)

Όπως φαίνεται και στην Εικόνα 5-5, οι δοκιμασίες ελέγχου είναι διαφόρων ειδών και εφαρμόζονται διαδοχικά από κάτω προς τα πάνω στο δεξί σκέλος του μοντέλου V:

- *έλεγχος μονάδας* (τυπικά WBT) που επικεντρώνει στις λεπτομέρειες της υλοποίησης και γίνεται παράλληλα με την κωδικοποίηση
- *έλεγχος συστατικών στοιχείων* (περισσότερο BBT) π.χ. για αντικείμενα και κλάσεις στον αντικειμενοστραφή προγραμματισμό
- *έλεγχος συνένωσης ή ενσωμάτωσης* (κυρίως WBT) που δοκιμάζει τα διάφορα συστατικά στοιχεία και τη συνεργασία τους με έμφαση στη διαπροσωπεία και τις αλληλεπιδράσεις μεταξύ τους
- *έλεγχος συστήματος* (τυπικά BBT) για τις συνολικές λειτουργίες του συστήματος στο επιχειρησιακό περιβάλλον του τελικού χρήστη.

Μια εικόνα που απαντάται στην πράξη ως προς τα σφάλματα και το στάδιο ελέγχου στο οποίο αυτά ανιχνεύονται δίνει ο Πίνακας 5-6. (19)

Παράλληλα με τον τελευταίο στη σειρά έλεγχο συστήματος – και προς το τέλος του – πραγματοποιείται επίσης ο *έλεγχος αποδοχής* (acceptance testing) ώστε να διαπιστωθεί εάν το προϊόν

είναι έτοιμο για κυκλοφορία στην αγορά. (9) Έχουμε ακόμα τον *έλεγχο ανάδρασης* (regression testing), που μπορεί να εφαρμοστεί σε κάθε στάδιο της φάσης ελέγχου, όταν η προσθήκη ενός νέου γνωρίσματος μπορεί εν δυνάμει να επηρεάσει μια ήδη υπάρχουσα λειτουργία του λογισμικού. Χρειάζεται τότε να πάμε προς τα πίσω και να επαναλάβουμε κάποιες δοκιμασίες, ώστε να επιβεβαιώσουμε ότι δεν έχει προκύψει κάποιο σφάλμα στα προηγούμενα ως ανεπιθύμητη συνέπεια αυτής της προσθήκης. (2) Αναφέρουμε επίσης τον *βήτα-έλεγχο* (beta testing) που λαμβάνει χώρα κατά το αρχικό στάδιο της λειτουργικής χρήσης ενός λογισμικού – συνήθως μεγάλου μεγέθους – οπότε αυτό διατίθεται με ελεγχόμενο τρόπο σε ένα περιορισμένο αριθμό χρηστών, ώστε να καταδειχτούν πιθανά σφάλματα και αστοχίες πριν την ευρεία κυκλοφορία. (9)

	Σφάλματα κώδικα & λογικού σχεδιασμού	Σχεδιαστικά σφάλματα
Έλεγχος μονάδας	65%	0%
Έλεγχος δομικού στοιχείου	30%	60%
Έλεγχος συστήματος	3%	35%
Σύνολο	**98%**	**95%**

Πίνακας 5-6: Προσεγγιστική εκτίμηση για τα στάδια ελέγχου

Τέλος, ανάμεσα στα πολλά εξειδικευμένα είδη δοκιμασιών, ας αναφέρουμε απλώς τον *έλεγχο έντασης και όγκου* (stress & volume testing, για φορτίο αιχμής για σύντομο διάστημα και φορτίο πλήρους ισχύος για παρατεταμένο διάστημα αντίστοιχα), τον *καταμερισμό ισοδυναμίας*, τον *έλεγχο συνοριακών τιμών*, τους *γράφους αιτίας – αποτελέσματος*, τον *έλεγχο απόδοσης* (performance testing), τον *έλεγχο ανάνηψης* (recovery testing), τον *έλεγχο αξιοπιστίας* (reliability testing), τον *έλεγχο εισβολής* (penetration testing), τον *έλεγχο «καθαρού δωματίου»* (clean room testing) κ.ά. (2) (19)

5.5 Αποτροπή σφαλμάτων

Στην ενότητα αυτή θα αναφερθούμε σε μια ακόμη έννοια που αφορά στην ποιότητα του λογισμικού, την **αποτροπή σφαλμάτων** (defect prevention). Αυτή περιλαμβάνει τις μεθόδους οι οποίες εφαρμόζονται κατά τις πρώτες κυρίως φάσεις ανάπτυξης και εμποδίζουν την εισαγωγή σφαλμάτων στο λογισμικό. Τέτοιες είναι ενδεικτικά οι παρακάτω: (2)

- *Joint Application Design (JAD):* Με τη μέθοδο αυτή σχηματίζονται ομάδες από 4-8 τελικούς χρήστες μαζί με ισάριθμους προγραμματιστές με σκοπό την εξαντλητική καταγραφή των απαιτήσεων χωρίς παραλείψεις και παρανοήσεις που οδηγούν σε σφάλματα στη συνέχεια. Για μεγάλου μεγέθους εφαρμογές – της τάξης των 10.000 FP – η εφαρμογή αυτής της μεθόδου στην πράξη έχει οδηγήσει σε επίπεδα δυναμικού σφαλμάτων απαιτήσεων κάτω από 0,25 ανά FP. Μια εξέλιξη αυτής της τεχνικής,

που έχει εφαρμογή σε agile μεθοδολογίες, είναι αυτή ενός «ενσωματωμένου» (embedded) χρήστη (ενίοτε δε και περισσότερων), ο οποίος αποτελεί μέλος της ομάδας εξαρχής και καθ' όλη τη διάρκεια της ανάπτυξης του λογισμικού.

- *Πρωτοτυποποίηση (prototyping)*: Πρόκειται, όπως είναι γνωστό, για μια από τις βασικές μεθοδολογίες ανάπτυξης λογισμικού. Συνήθως δεν εφαρμόζεται σε μικρού μεγέθους έργα (καθώς αυτή θα αποτελούσε περιττό επιπλέον κόπο), ούτε όμως σε ιδιαίτερα μεγάλο μέγεθος (της τάξης των 10.000 FP και άνω, καθώς καθίσταται μάλλον απαγορευτική από το ίδιο το μέγεθος του πρωτοτύπου – που εκτιμάται περίπου στο 1/10 του συνολικού). Τα πρωτότυπα κατασκευάζονται λοιπόν σε μεσαίου μεγέθους έργα (500 – 10.000 FP) και διακρίνονται κυρίως σε *πρόσκαιρα* (disposable), που απορρίπτονται πριν την ανάπτυξη του κυρίως λογισμικού, και σε *εξελικτικά* (evolutionary), που μεταλλάσσονται και εμπλουτίζονται σταδιακά ώσπου να πάρουν την τελική μορφή τους. Τα πρωτότυπα αναπτύσσονται βιαστικά και γι' αυτό τείνουν να εμφανίζουν υψηλά επίπεδα δυναμικού σφαλμάτων (άνω του 1,25 ανά FP) και χαμηλό δείκτη αφαίρεσης DRE (κάτω από 80%). Παρόλα αυτά, έχει αποδειχτεί ότι προλαμβάνουν έναν ικανό αριθμό από σοβαρά λάθη πριν την εμφάνισή τους στην κύρια εφαρμογή. Εξάλλου, καθώς τα πρόσκαιρα (κυρίως) πρωτότυπα απορρίπτονται, τα σφάλματα που περιέχουν εξαφανίζονται με τη σειρά τους. Γενικά, τα έργα που εντάσσουν τεχνικές πρωτοτυποποίησης κατά την ανάπτυξη θεωρείται ότι φτάνουν σε λειτουργική σταθερότητα νωρίτερα από τα υπόλοιπα.
- *Quality Function Deployment (QFD)*: Συνίσταται σε μια εξαιρετικά λεπτομερή, δομημένη και τεκμηριωμένη μέθοδο στη διάρκεια της οποίας χρήστες, πελάτες και προγραμματιστές εργάζονται από κοινού στην εξαντλητική απαρίθμηση των απαιτήσεων, όπως επίσης και στον ακριβή καθορισμό της απόκρισης ώστε να πληρούνται οι προϋποθέσεις μία προς μία. Αποτέλεσμα είναι τα πολύ χαμηλά επίπεδα νέων απαιτήσεων στην πορεία, αλλά και τα χαμηλά επίπεδα σφαλμάτων απαιτήσεων (κάτω του 0,2/FP), σχεδίασης (0,5/FP) και κώδικα (1,1/FP).

Αναφέρουμε εδώ κάποιες επιπλέον μεθόδους πρόληψης: *Six Sigma for Software, Επισκοπήσεις SQA, Scrum Sessions, PSP/TSP, CMMI, Αυτοματοποιημένη ανάλυση ρίσκου* κτλ. Τέλος, τεχνικές όπως η *τυπική επιθεώρηση* και η *στατική ανάλυση*, αν και κατά βάση οι δύο ισχυρότερες προ-ελεγκτικές (pre-test) μέθοδοι άρσης σφαλμάτων με μέσες τιμές DRE η καθεμιά τους περί το 85%, μπορούν παράλληλα να καταστούν πολύ αποδοτικές και στην ίδια την αποφυγή των σφαλμάτων.

Με τα σημερινά δεδομένα (όπως δείχνει και ο Πίνακας 3-1), το δυναμικό σφαλμάτων λογισμικού προσεγγίζει τα 5,0 λάθη ανά FP, τα λάθη ανιχνεύονται σωρευτικά σε ποσοστό περίπου 85% και ο αριθμός αυτών που παραμένουν τελικά στο λογισμικό και παραδίδονται στο χρήστη είναι περίπου 0,75 λάθη ανά FP. Ενσωματώνοντας συστηματικά και σε μεγαλύτερο βαθμό τις πλέον προηγμένες μεθόδους αποτροπής και αφαίρεσης, είναι εφικτή η μείωση του δυναμικού σφαλμάτων σε 2,5 ανά FP – τιμή που μπορεί να μειωθεί περαιτέρω έως και σε 1,5 λάθη ανά FP με τη χρήση πιστοποιημένου επαναχρησιμοποιήσιμου λογισμικού (certified reuse). Μπορούμε τότε να αναμένουμε μια τιμή κοντά στα 0,025 λάθη ανά FP για τα σφάλματα που παραμένουν στο λογισμικό και παραδίδονται στο χρήστη. Παράλληλα, ο κατάλληλος συνδυασμός προ-ελεγκτικών μεθόδων (αυτοματοποιημένη στατική ανάλυση και τυπικές επιθεωρήσεις για τη σχεδίαση και τον κώδικα) μαζί με αλεπάλληλους σωρευτικούς ελέγχους και ακόμα τη συμβολή μιας ειδικευμένης ομάδας τυπικής διασφάλισης ποιότητας μπορεί να επιφέρει τιμές του δείκτη DRE μέχρι και 99%. (2)

5.6 Ανοχή σφάλματος και περιορισμός αστοχίας

Η πλήρης απουσία ελαττωμάτων στο λογισμικό τη στιγμή που αυτό παραδίδεται στο χρήστη δεν είναι εφικτή. Θα υπάρχουν πάντα λανθάνοντα σφάλματα που έχουν παραμείνει στο σύστημα, ακόμα και μετά τους πλέον εξαντλητικούς ελέγχους. Από την εμπειρία μάλιστα, πρέπει να είμαστε ικανοποιημένοι όταν ο αριθμός τους δεν υπερβαίνει τα 2 σφάλματα ανά KLOC, ενώ μπορούμε να θεωρούμε πρακτικά ασφαλές ένα λογισμικό με δυναμικό 0,5 – 1,5 σφάλματα ανά KLOC. (10)

Ιδιαίτερα για συστήματα λογισμικού για τα οποία οι επιπτώσεις μιας αστοχίας είναι σημαντικές, όπως το λογισμικό που «τρέχει» τις υποδομές των τηλεπικοινωνιών παγκοσμίως, οικονομικές και κρίσιμες βάσεις δεδομένων για μεγάλες εταιρίες ή πραγματικού χρόνου λογισμικά υπο-συστήματα ελέγχου σε ιατρικά, πυρηνικά και άλλα ενσωματωμένα συστήματα, τα χαμηλά επίπεδα ελαττωμάτων και κινδύνου αστοχιών μπορεί να μην είναι αρκετά. Σε τέτοιες περιπτώσεις, η τακτική που ακολουθείται είναι είτε η ανοχή σε σφάλματα είτε ο περιορισμός των αστοχιών. Η **ανοχή σφάλματος** παρακάμπτει τα τοπικά λάθη, εμποδίζοντας παράλληλα την επέκτασή τους με στόχο να προλάβει γενικευμένες αστοχίες. Ο **περιορισμός αστοχίας** εφαρμόζει τα κατάλληλα μέτρα εγκλωβισμού μετά από κάποια τέτοια γενικευμένη αστοχία. Η πρώτη εστιάζει στη διατήρηση του συστήματος σε λειτουργική κατάσταση παρά τα δυναμικά προβλήματα που μπορεί να συναντήσουμε. Ο δεύτερος επικεντρώνει στο να κρατήσει το σύστημα ασφαλές ή να ελαχιστοποιήσει τη ζημιά χωρίς να ενδιαφέρεται εάν το σύστημα παραμένει ή όχι λειτουργικό.

Σε ό,τι αφορά στην ανοχή σε σφάλματα, συναντούμε ακόμα τις *κόπιες* (duplication) και τα *αντίγραφα ασφαλείας* (backup). Οι κόπιες είναι πολλαπλά προγράμματα που τρέχουν παράλληλα, κάτι ανάλογο με τις περισσότερες από μια μηχανές αεροπλάνου σε ταυτόχρονη λειτουργία. Τα αντίγραφα ασφαλείας προϋποθέτουν ένα κύριο πρόγραμμα σε δυναμική εκτέλεση και την ενεργοποίηση των αντιγράφων μόνο εάν προκύψει πρόβλημα σε αυτό.

Γενικά, οι τεχνικές περιορισμού των αστοχιών περιλαμβάνουν είτε την αποφυγή του ατυχήματος πριν αυτό συμβεί ή αλλιώς την ελαχιστοποίηση των συνεπειών. Στην πρώτη περίπτωση επιδιώκουμε να περιορίσουμε το εύρος και τον αντίκτυπο μιας αστοχίας ώστε να μην οδηγήσει σε ατύχημα. Στη δεύτερη, κύριος στόχος είναι ο έλεγχος της ζημιάς – όσο αυτό είναι δυνατό – και η διαχείριση του ρίσκου σε συνάρτηση και με την εκτίμηση του κινδύνου εμφάνισης του ατυχήματος. (9)

6 ΠΑΡΑΓΩΓΙΚΟΤΗΤΑ ΛΟΓΙΣΜΙΚΟΥ

6.1 Μετρήσεις της παραγωγικότητας λογισμικού

Για τον υπολογισμό της παραγωγικότητας, ορίζουμε το **ρυθμό παράδοσης (project delivery rate)**, δηλ. τη σωρευτική ανθρώπινη προσπάθεια (effort) κανονικοποιημένη ως προς το μέγεθος του λογισμικού, σε ώρες ανά function point [hours/FP]. Με ανάλογο τρόπο, μπορούμε να ορίσουμε μια αντίστοιχη μετρική για το **κόστος ανά function point**. Ορίζουμε ακόμη την **ταχύτητα παράδοσης (speed of delivery)** ως τον αριθμό των function points που παράγει ένα άτομο σε ένα μήνα απασχόλησής του στο έργο.

Ο Πίνακας 6-1 παρουσιάζει την κατανομή της επιμέρους προσπάθειας (σε ποσοστά επί της συνολικής), ανάλογα με την τάξη μεγέθους του έργου: (3)

	Διαχείριση	Έγγραφα	Αφαίρεση σφαλμάτων	Κωδικοποίηση	Σύνολο
1FP	5,00%	5,00%	20,00%	70,00%	100,00%
10FP	10,00%	7,00%	18,00%	65,00%	100,00%
100FP	12,00%	15,00%	23,00%	50,00%	100,00%
1000FP	15,00%	25,00%	30,00%	30,00%	100,00%
10000FP	16,00%	31,00%	35,00%	18,00%	100,00%
100000FP	17,00%	35,00%	36,00%	12,00%	100,00%
Μέσος όρος	**12,50%**	**19,67%**	**27,00%**	**40,83%**	**100,00%**

Πίνακας 6-1: Κατανομή της προσπάθειας ανά δραστηριότητα

Είναι φανερό ότι η *κωδικοποίηση* μπορεί να αποτελεί μεν την κύρια δραστηριότητα για έργα μεγέθους έως 1000 FP, η κατάσταση όμως μεταβάλλεται δραστικά όταν μιλάμε για μεγαλύτερα έργα, όπου η *τεκμηρίωση* και ο *χειρισμός των σφαλμάτων* απορροφούν το ίδιο ή και μεγαλύτερο τμήμα της συνολικής προσπάθειας.

Στην πράξη παρατηρούνται μεγάλες διαφοροποιήσεις στην παραγωγικότητα, ανάλογα με το είδος της εφαρμογής (καινούργια έργα, αναβαθμίσεις, συντηρήσεις κτλ.), το μέγεθος του λογισμικού, το πεδίο εφαρμογής, τα εργαλεία και μεθόδους ανάπτυξης και άλλους παράγοντες. Για παράδειγμα, ο

Πίνακας 6-2 δίνει ένα μέτρο της παραγωγικότητας (σε FP/μήνα) ανάλογα με την πλατφόρμα της εφαρμογής: (10)

Client Server	17
Main Frame	13
Web	25
E-business Web	15
Vendor Packages	18
Data Warehouse	9

Πίνακας 6-2: Μέσος όρος παραγωγής λογισμικού

Ενδεικτικά, παρατίθενται ακόμη ο Πίνακας 6-3 και ο Πίνακας 6-4 με στοιχεία για τη μέση παραγωγικότητα (σε FP ανά ανθρωπο-μήνα / χώρα προέλευσης των έργων: Η.Π.Α.) για διάφορες τάξεις μεγέθους της εφαρμογής. (3)

Σε αυτό το σημείο είναι σκόπιμο να σημειωθεί ότι, όσον αφορά σε έργα αναβάθμισης (enhancement), υπεισέρχονται και άλλοι παράγοντες που επηρεάζουν την παραγωγικότητα, όπως η πολυπλοκότητα του βασικού κληρονομημένου (legacy) συστήματος, η πολυπλοκότητα του νέου κώδικα που προστίθεται ή αυτού που μετατρέπεται, διαγράφεται κοκ. Άλλωστε, τα μικρά έργα αναβάθμισης έχουν κατά κανόνα χαμηλή παραγωγικότητα. Αιτία είναι το υψηλό έμμεσο κόστος (overhead), κάτι που αυξάνει κατά πολύ το μεταβλητό κόστος ανά FP και που οφείλεται σε εκ νέου έλεγχο ανάδρασης και μετάφραση (re-compiling) του κληρονομημένου συστήματος, στην εξαρχής συγγραφή των εγχειριδίων προδιαγραφών και χρήστη κτλ.

Τα νέα εργαλεία και μεθοδολογίες λογισμικού έχουν κατά κανόνα θετική επίδραση στην παραγωγικότητα σε μεσο- και μακροπρόθεσμο επίπεδο (όπως δείχνει ο Πίνακας 6-3) για τις επιδόσεις με την χρήση μεθοδολογιών agile και XP, CMM, TSP/PSP κτλ.). Το δυναμικό σφαλμάτων μειώνεται και υπάρχει τελικά αξιόλογη απόδοση στην επένδυση. Ένας ανασταλτικός παράγοντας στην εφαρμογή

τους μπορεί να είναι η βραχυπρόθεσμη μείωση της παραγωγικότητας που οφείλεται στην καμπύλη εκμάθησης του νέου εργαλείου από το προσωπικό. Για παράδειγμα, αν και για αντικειμενοστραφή έργα η παραγωγικότητα φτάνει περίπου τα 10-12 FP / ανθρωπο-μήνα έναντι μόλις 4-6 FP για διαδικασιακές εφαρμογές, η καμπύλη εκμάθησης είναι τόσο απότομα ανοδική που η βραχυπρόθεσμη παραγωγικότητα καταλήγει αρνητική και οδηγεί το 50% των καινούργιων έργων να εγκαταλείψει πρόωρα τις νέες πρακτικές και να επανέλθει στις παλιές μεθόδους. (3)

Μεθοδολογία	100	1000	10000	100000	Μέσος όρος
Agile	37,00	25,00	7,50	-	23,17
CMM1	12,00	7,20	4,50	1,70	6,35
CMM3	21,00	10,20	6,30	4,80	10,58
CMM5	23,00	12,60	8,90	6,50	12,75
Extreme (XP)	35,00	23,60	8,60	-	22,40
Iterative	19,00	9,90	6,70	4,20	9,25
Object-oriented	20,00	16,70	7,70	5,30	12,43
Six-sigma	17,50	11,00	8,20	6,30	10,75
TSP/PSP	16,00	18,00	9,20	6,85	12,51
Waterfall	8,80	6,50	4,20	1,80	5,33
Average	**20,93**	**14,07**	**7,14**	**4,16**	**11,58**

Πίνακας 6-3: Μέση παραγωγικότητα κατά μεθοδολογία

Είδος λογισμικού	100	1000	10000	100000	Μέσος όρος
End-user	52,60	-	-	-	52,60
Web	47,30	25,60	12,00	-	28,30
MIS	21,00	16,40	4,65	2,70	11,19
US outsource	23,00	17,20	4,90	3,20	12,08
Offshore outsource	19,00	15,80	4,70	3,00	10,63
Commercial	11,00	9,30	4,60	4,00	7,23
Systems	9,00	6,90	5,10	4,15	6,29

Military	5,60	4,80	3,80	2,10	4,08
Average	**23,56**	**13,71**	**5,68**	**3,19**	**11,54**

Πίνακας 6-4: Μέση παραγωγικότητα κατά είδος λογισμικού

6.2 Έρευνα πάνω στη βάση καταγραφής ISBSG

Στο σημείο αυτό αξίζει να αναφερθούμε εν συντομία στα ευρήματα μιας εργασίας που, ως προς την παραγωγικότητα, κινείται στη γραμμή που θα ακολουθήσει και η παρούσα έρευνα στη συνέχεια. Πράγματι, στο (16) γίνεται αναφορά σε παράγοντες που επηρεάζουν την παραγωγικότητα του λογισμικού, βάσει των εμπειρικών δεδομένων που έχουν συγκεντρωθεί στη βάση καταγραφής ISBSG (βλ. Κεφ. 8 και εξής). Πιο συγκεκριμένα, η *γλώσσα προγραμματισμού* φαίνεται να είναι ένας ισχυρός τέτοιος παράγοντας. Εξάλλου, η γλώσσα που χρησιμοποιείται σε μια εφαρμογή σχετίζεται με την επιλογή της πλατφόρμας ανάπτυξης (Mainframe, Mid-Range, Multi-platform, PC). Πράγματι, συχνά μια γλώσσα επιλέγεται σε συνδυασμό με μια και μόνη πλατφόρμα, εκτός από την Java που αντιπροσωπεύεται ισομερώς σε όλες τις πλατφόρμες. Γενικά πάντως, οι γλώσσες Γ4Γ τείνουν να έχουν καλύτερους ρυθμούς παράδοσης απότις Γ3Γ. Ένας άλλος παράγοντας που αναφέρεται στο (16) είναι ο *μέγιστος αριθμός μελών στην ομάδα*. Εδώ παρατηρούμε ότι έργα με ομάδες μεγαλύτερες των 5 ατόμων τείνουν να έχουν χειρότερες επιδόσεις στην παραγωγικότητα. Σε ό,τι αφορά στην *πλατφόρμα ανάπτυξης*, τα έργα mainframe παρουσιάζουν ένα μεγάλο εύρος τιμών στο ρυθμό παράδοσης, ενώ τα έργα που έχουν αναπτυχθεί σε PC έχουν μικρότερη διασπορά και γενικά καλύτερη παραγωγικότητα. Οι εφαρμογές mid-range βρίσκονται κάπου στη μέση ανάμεσα στις δύο άλλες κατηγορίες. Τέλος, οι εφαρμογές multi-platform έχουν τιμές παραπλήσιες με αυτές των PC. Αναφορικά με το *είδος της εφαρμογής*, παρατηρείται ότι οι νέες εφαρμογές εμφανίζουν ρυθμούς παραγωγής 8-12 ώρες ανά FP, ενώ οι αναβαθμίσεις (enhancements) αντίστοιχα 12-16 ώρες ανά FP. Η διαφορά αυτή μπορεί όμως να μην αφορά άμεσα στο είδος του έργου, αλλά να οφείλεται για παράδειγμα στο ότι η πλειοψηφία των δεύτερων έχουν αναπτυχθεί σε mainframe, ενώ οι (πιο πρόσφατες) πρώτες σε πλατφόρμες PC (που, όπως αναφέρθηκε προηγούμενα, είναι πιο «παραγωγικές»). Στο πεδίο της *αρχιτεκτονικής*, οι εφαρμογές multi-tier έχουν τις καλύτερες επιδόσεις και οι client-server τις χειρότερες με τις εφαρμογές stand-alone να βρίσκονται στη μέση. Και πάλι όμως, τα αποτελέσματα μπορεί να σχετίζονται με άλλους παράγοντες, όπως για παράδειγμα ότι ο μέγιστος αριθμός μελών στην ομάδα είναι μεγαλύτερος για τις client-server, ενώ αισθητά μικρότερος κατά μέσο όρο για τις multi-tier εφαρμογές.

7 ΕΜΠΕΙΡΙΚΗ ΤΕΧΝΟΛΟΓΙΑ ΛΟΓΙΣΜΙΚΟΥ

7.1 Γενικά για την εμπειρική τεχνολογία λογισμικού

Ο χώρος της τεχνολογίας λογισμικού εμφορείται από στερεότυπες και ενίοτε ατεκμηρίωτες αντιλήψεις γύρω από την αποτελεσματικότητα πρακτικών που τυχαίνει να είναι σε ευρεία χρήση. Συχνά έχουμε την τάση να υιοθετούμε τεχνολογικές καινοτομίες χωρίς να έχουμε προηγουμένως προχωρήσει σε ενδελεχείς μετρήσεις που να ποσοτικοποιούν την διαφορά στις επιδόσεις ανάμεσα στη νέα και τις παλιότερες μεθόδους. Ξεκινώντας κατά τη δεκαετία του ’70, με πιο συστηματικό δε τρόπο από τη δεκαετία του ’80 και αργότερα, μέσα από τις εργασίες των Victor Basili, Dieter Rombach, Norman Fenton, Barbara Kitchenham, Shari L. Pfleeger, Claes Wohlin, Elaine Weyuker, Norman Schneiderwind κ.ά. προβάλλει το αίτημα για τυπικές μεθόδους και ένα αυστηρό μαθηματικό υπόβαθρο, γνωρίσματα στα οποία αρχικά οι μηχανικοί λογισμικού δεν έδιναν την απαιτούμενη προσοχή.

Η αναλυτική ερευνητική μέθοδος, η οποία αρχικά προτείνει μια θεωρία και στη συνέχεια επιχειρεί να την επιβεβαιώσει μέσω των παρατηρήσεων, μπορεί να είναι κατάλληλη για τους πιο τυπικούς τομείς της επιστήμης των υπολογιστών (όπως την ηλεκτρομαγνητική θεωρία και τους αλγορίθμους), δεν ταιριάζει όμως στο χώρο της τεχνολογίας λογισμικού, καθώς αυτή εξαρτάται σε μεγάλο βαθμό από τον ανθρώπινο παράγοντα. Δεν μπορούμε λοιπόν να αναμένουμε εδώ κανόνες και νόμους με καθολική θεωρητική ισχύ, όπως π.χ. στη φυσική. Είναι προτιμότερο να προτείνουμε και να αξιολογούμε μοντέλα με βάση τις παρατηρήσεις μας μέσα από εμπειρικές ερευνητικές μεθόδους, όπως αυτές που παραδοσιακά έχουν χρησιμοποιηθεί στις κοινωνικές επιστήμες, επίσης εξαρτημένες από την ανθρώπινη συμπεριφορά. (20)

Αυτή είναι η τακτική της **εμπειρικής[9] τεχνολογίας λογισμικού**. Οι έρευνες που πραγματοποιούνται σε αυτόν τον τομέα διερευνούν, περιγράφουν, ερμηνεύουν και προβλέπουν φυσικά, κοινωνικά ή γνωστικά φαινόμενα της τεχνολογίας λογισμικού μέσα από αποδεικτικά στοιχεία που βασίζονται σε παρατηρήσεις ή στην καταγεγραμμένη εμπειρία. (21) Το βάρος δίνεται στη συλλογή ενός ικανού αριθμού «αντιπροσωπευτικών» δεδομένων και την ανάλυσή τους με σκοπό την εξαγωγή εμπειρικών νόμων που να μπορούν να προβλέψουν με σχετική ασφάλεια την απόκριση (response) σε καινούργια δεδομένα εισόδου.

Σημείο αναφοράς για την έρευνα πάνω στα ζητήματα της εμπειρικής τεχνολογίας λογισμικού είναι το International Symposium on Empirical Software Engineering & Measurement – ESEM, ένα διεθνές

[9] Εμπειρικός: που προέρχεται από την άμεση (μέσω των αισθήσεων) αντίληψη, όχι από διαλογισμό, σκέψη, φαντασία / Εμπειρική απόδειξη: αυτή που στηρίζεται στην παρατήρηση και στο πείραμα (Λεξικό της κοινής νεοελληνικής, Ινστιτούτο Νεοελληνικών Σπουδών, Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης, 1998)

φόρουμ από ερευνητές, μηχανικούς και μέλη της ακαδημαϊκής κοινότητας, το οποίο ενθαρρύνει την ανταλλαγή απόψεων και προάγει τη μοντελοποίηση φαινομένων στην τεχνολογία λογισμικού από τη σκοπιά της εμπειρικής έρευνας.

Προχωρώντας ένα βήμα πέρα από την απλή περιγραφή της πραγματικότητας, με τη βοήθεια της περιγραφικής στατιστικής μπορούμε πολλές φορές να προτείνουμε ένα καινούργιο μοντέλο. Πράγματι, τα εμπειρικά δεδομένα μπορούν να δώσουν μια σαφέστερη εικόνα, αν όχι του «τι» πραγματικά συμβαίνει, πάντως του «γιατί» συμβαίνει (αν και χρειάζεται ιδιαίτερη προσοχή στην ερμηνεία της στατιστικής συσχέτισης με όρους αιτιότητας). (22) Τα μοντέλα αυτά, στηριγμένα σε ένα μεγάλο όγκο εμπειρικών δεδομένων, επιτρέπουν μια σχετικά επιτυχή πρόβλεψη της μελλοντικής συμπεριφοράς, είτε μέσα από στατιστικές τεχνικές όπως:

- διερευνητική ανάλυση δεδομένων (exploratory data analysis)
- πολυμεταβλητή ανάλυση (multivariate analysis)
- γραμμική και λογιστική παλινδρόμηση (linear & logistic regression)
- ανάλυση διασποράς (ANOVA)
- ανάλυση σε κύριες συνιστώσες (principal component analysis)

είτε με εναλλακτικές πρακτικές όπως:

- μπεϋζιανά και νευρωνικά δίκτυα
- γραμμική διαχωριστική ανάλυση (linear discriminant analysis)
- τεχνικές ασαφούς λογικής (fuzzy logic)
- μεθόδους εξόρυξης δεδομένων (data mining): δέντρα ταξινόμησης και παλινδρόμησης (CART, CHAID κοκ.), ανάλυση σε ομάδες (cluster analysis) και δενδρογράμματα κοκ.

Συχνά καταφεύγουμε σε ένα συνδυασμό των πιο πάνω μεθόδων ώστε να αναλύσουμε από διαφορετικές πλευρές το συγκεκριμένο πρόβλημα. Κατά το σχεδιασμό της έρευνας, ο μηχανικός καλείται να επιλέξει την προσφορότερη μέθοδο εκμεταλλευόμενος τα δυνατά σημεία της και μετριάζοντας κατά το δυνατόν τις αδυναμίες της. Η εγκυρότητα (validity) της μεθόδου είναι τόσο μεγαλύτερη όσο αυτές οι αδυναμίες έχουν αντιμετωπιστεί σε ικανό βαθμό. (23)

Οι συνηθέστερες μέθοδοι στην εμπειρική τεχνολογία λογισμικού είναι τα ελεγχόμενα πειράματα (controlled experiments), οι μελέτες περιπτώσεων (case studies) και οι δειγματοληπτικές έρευνες (survey research).

7.2 Πειράματα και μελέτες περίπτωσης

Σε ένα πείραμα, επιχειρούμε να ελέγξουμε την βασιμότητα μιας υπόθεσης μέσα από το χειρισμό μιας ή περισσότερων ανεξάρτητων μεταβλητών (παράγοντες, factors, predictors) ώστε να μετρήσουμε την επίδρασή τους σε μια εξαρτημένη μεταβλητή απόκρισης (response variable). Ο έλεγχος άλλων ανεξάρτητων μεταβλητών είναι πρωταρχικής σημασίας για την εγκυρότητα των αποτελεσμάτων και επιτυγχάνεται βασικά με την εξουδετέρωση (blocking) της επίδρασής τους.

Μια μελέτη περίπτωσης από την άλλη πλευρά χρησιμοποιεί σκοπούμενη (purposive) και όχι τυχαία δειγματοληψία. Σκοπός είναι η μελέτη μιας επιλεγμένης περίπτωσης, πιθανώς κρίσιμης ή ακραίας, ώστε να εξάγουμε συμπεράσματα για αυτήν την ειδική κατάσταση, μέσα στο πραγματικό της περιβάλλον. Οι μελέτες περίπτωσης ενδείκνυνται στις περιπτώσεις κατά τις οποίες δεν είναι σαφές το σύνορο ανάμεσα στο φαινόμενο καθαυτό και το περιβάλλον του. Τα αποτελέσματα μιας μελέτης περίπτωσης είναι a priori ανοιχτά σε διαφορετικές ερμηνείες και συχνά μεροληπτικά (researcher biased), απαιτείται λοιπόν ένα σαφές πλαίσιο για την επιλογή της συγκεκριμένης περίπτωσης και τη συλλογή δεδομένων. Συχνά πάντως, μια κατάλληλα σχεδιασμένη μελέτη μιας πραγματικής περίπτωσης όπου η θεωρία αποδεικνύεται λανθασμένη μπορεί να είναι πολύ πιο πειστική από πολλά αποτυχημένα πειράματα «εργαστηρίου». (20)

7.3 Δειγματοληπτικές έρευνες

7.3.1 Μέθοδοι δειγματοληψίας

Πολύ συχνά είναι αδύνατο να καταγράψουμε όλα ανεξαιρέτως τα υποκείμενα ενός πληθυσμού για διάφορους λόγους. Σε αυτή την περίπτωση, ο ερευνητής καταφεύγει στην επιλογή ενός περιορισμένου αντιπροσωπευτικού δείγματος από έναν πολύ μεγαλύτερο αρχικό πληθυσμό και χρησιμοποιεί τεχνικές ανάλυσης δεδομένων που επιτρέπουν τη γενίκευση από τους δειγματικούς στατιστικούς δείκτες (sample statistics) στις παραμέτρους του πληθυσμού (population parameters).

Βασικό μέλημα εδώ είναι από τη μια η εξαγωγή *αμερόληπτων εκτιμητριών*, από την άλλη ο *περιορισμός του τυπικού σφάλματος* του δείγματος. Οι δύο αυτοί στόχοι, της **ορθότητας** (accuracy) και της **ακρίβειας** (precision) αντίστοιχα, κινούνται συχνά σε αντίθετη κατεύθυνση και ο ερευνητής καλείται να επιλέξει έναν ικανοποιητικό συμβιβασμό (trade-off) ανάμεσά τους, ανάλογα με τις προδιαγραφές του προβλήματος που μελετά. Καταρχήν, ένα ιδιαίτερα μικρό τυπικό σφάλμα είναι πολύ χρήσιμο για την ακριβή πρόβλεψη της απόκρισης από νέα δεδομένα εισόδου και «συγχωρεί» ένα μικρό ποσό μεροληψίας (bias) στις προβλεπόμενες τιμές, ειδικά αν εργαζόμαστε πάνω σε *συνεπείς εκτιμήτριες* (consistent statistics), οι οποίες προσεγγίζουν ασυμπτωτικά τις παραμέτρους του πληθυσμού όσο μεγαλύτερο είναι το δείγμα μας. Βέβαια, ένας περιοριστικός παράγοντας του πλήθους του δείγματος είναι συνήθως το κόστος της δειγματοληψίας, πολύ συχνά δε και η πλήρης αδυναμία να έχουμε στη διάθεσή μας στοιχεία για τον συνολικό πληθυσμό, όπως π.χ. για *όλα* τα έργα λογισμικού που ολοκληρώθηκαν τον τελευταίο χρόνο σε ολόκληρο τον κόσμο.

Οι μέθοδοι δειγματοληψίας υποδιαιρούνται βασικά σε πιθανοτικές και μη πιθανοτικές. Σε μια *πιθανοτική* μέθοδο, κάθε μονάδα πληθυσμού (population unit) έχει μια εξαρχής γνωστή, μη μηδενική πιθανότητα να επιλεγεί στο δείγμα. Μια τέτοια μέθοδος είναι η *απλή τυχαία δειγματοληψία* (simple random sampling - SRS), κατά την οποία κάθε μονάδα του πληθυσμού έχει την *ίδια* πιθανότητα επιλογής της στο δείγμα. Με αυτή τη μέθοδο είναι δυνατή η αμερόληπτη εκτίμηση των βασικών παραμέτρων του πληθυσμού. Άλλες πιθανοτικές μέθοδοι είναι η *συστηματική δειγματοληψία* και η *δειγματοληψία κατά ομάδες* (cluster sampling) σε ένα ή περισσότερα στάδια. Ως μη πιθανοτικές δειγματοληψίες αναφέρουμε τις δειγματοληψίες *ευκολίας* (convenience), με *προκαθορισμένα ποσοστά* (quota), *χιονοστιβάδας* (snowball) κ.ά. (24)

Σε ό,τι αφορά στις πιθανοτικές δειγματοληψίες, ιδιαίτερο ενδιαφέρον για τη συνέχεια παρουσιάζει η *στρωματοποιημένη δειγματοληψία* (stratified sampling). Εδώ θεωρούμε εξαρχής μια διαμέριση του πληθυσμού σε στρώματα αποτελούμενα από μονάδες του πληθυσμού σχετικά ομοειδείς ως προς το παρατηρούμενο χαρακτηριστικό. Κατά βάση, επιδιώκουμε η διασπορά *εντός* του κάθε στρώματος να είναι μικρή σε σύγκριση με αυτήν *ανάμεσα* στα στρώματα, ενώ παράλληλα οι μέσες τιμές των στρωμάτων να απέχουν κατά το δυνατόν περισσότερο μεταξύ τους. Μπορεί να δειχθεί ότι η στρωματοποίηση (stratification) όχι μόνο οδηγεί και πάλι σε αμερόληπτες εκτιμήτριες αλλά ταυτόχρονα, με μια ικανοποιητική διαμέριση με τα παραπάνω χαρακτηριστικά, μπορεί να δώσει και σημαντικά μικρότερο τυπικό σφάλμα σε σχέση με αυτό ενός ενιαίου απλού τυχαίου δείγματος. Στην περίπτωση της στρωματοποιημένης δειγματοληψίας, μπορούμε να επιλέξουμε είτε *αναλογικό καταμερισμό* στο δείγμα (διατηρώντας τις αναλογίες πληθικότητας μεταξύ των στρωμάτων, άρα τελικά κρατώντας την *ίδια* πιθανότητα επιλογής για την κάθε μονάδα πληθυσμού) είτε *βέλτιστο καταμερισμό*, όπως π.χ. στην περίπτωση του καταμερισμού Neyman. Στην τελευταία αυτή περίπτωση, ο αριθμός των αντιπροσώπων κάθε στρώματος στο δείγμα είναι ανάλογος τόσο του πλήθους όσο και της τυπικής απόκλισης των μονάδων του στρώματος αυτού. (24)

7.3.2 Σφάλματα δειγματοληψίας

Όσο καλό και αν είναι το δειγματοληπτικό σχέδιο, οι δείκτες που μας παρέχει το δείγμα δεν θα ταιριάζουν απόλυτα με τις παραμέτρους του πληθυσμού. Το συνολικό σφάλμα μιας δειγματοληπτικής έρευνας μπορεί να διαχωριστεί κατά βάση σε τέσσερις συνιστώσες, οι οποίες λειτουργούν αθροιστικά:

Συνολικό_Σφάλμα = Σφάλμα_Μη_Κάλυψης + Δειγματικό_Σφάλμα + Σφάλμα_Μη_Απόκρισης + Σφάλμα_Μέτρησης

Εκτός από το *δειγματικό σφάλμα*, που αφορά στο σφάλμα που έχουμε λόγω της χρήσης μόνο ενός μικρού τυχαίου δείγματος και όχι ολόκληρου του πληθυσμού, τα άλλα τρία συναποτελούν το *μη δειγματικό σφάλμα* και μπορούν να προκύψουν ακόμα και στην περίπτωση μιας απογραφής ολόκληρου του πληθυσμού. Πιο συγκεκριμένα, καθώς ο αρχικός πληθυσμός δεν ταυτίζεται κατά κανόνα με το δειγματοληπτικό πλαίσιο, μπορεί να προκύψει **σφάλμα μη-κάλυψης**, λόγω υπο-κάλυψης ή υπερ-κάλυψης. Στην περίπτωση τώρα κατά την οποία τα υποκείμενα που έχουν επιλεγεί στο δείγμα δεν έχουν απαντήσει στις ερωτήσεις, είτε καθ' ολοκληρίαν είτε μερικώς, έχουμε **σφάλμα μη-απόκρισης**. Τέλος, λανθασμένες και ανακριβείς απαντήσεις ευθύνονται για το **σφάλμα μέτρησης**. Αυτά τα σφάλματα οδηγούν σε ένα μεγαλύτερο ή μικρότερο ποσό μεροληψίας, όταν οι στατιστικοί δείκτες επηρεάζονται από το μέγεθος του σφάλματος προς τη μια ή την άλλη κατεύθυνση. (25)

7.3.3 Κάλυψη και ελλείπουσες τιμές

Κάλυψη είναι το ποσοστό του αρχικού πληθυσμού που περιέχεται στο δειγματοληπτικό πλαίσιο. Το **ποσό μεροληψίας λόγω μη-κάλυψης** ορίζεται ως η διαφορά $B_F = Y_F - Y_P$, όπου Y_F και Y_P οι μέσες τιμές για το δειγματοληπτικό πλαίσιο (frame) και τον αρχικό πληθυσμό (population) αντίστοιχα. Μπορεί να δειχτεί εύκολα ότι το ποσό μεροληψίας λόγω κάλυψης είναι:

$$B(y_F) = \gamma_{NF}(Y_F - Y_{NF}),$$

όπου $\gamma_{NF} = 1 - N_F / N$ είναι το κλάσμα μη-κάλυψης δηλ. το ποσοστό του αρχικού πληθυσμού ο οποίος δεν είναι παρών στο δειγματοληπτικό πλαίσιο και Y_{NF} η μέση τιμή των εκτός πλαισίου μονάδων του πληθυσμού. (26) Η μεροληψία λόγω μη-κάλυψης είναι ένας υπαρκτός κίνδυνος κατά τη δειγματοληψία και μπορεί να φτάσει σε υψηλά επίπεδα[10].

Κατ' αναλογία με τα παραπάνω για τη μη-κάλυψη, το **ποσό μεροληψίας λόγω μη-απόκρισης** είναι:

$$B(y_R) = P_m (Y_R - Y_{NR}),$$

όπου P_m είναι το ποσοστό της αποχής και Y_R, Y_{NR} οι μέσες τιμές στον πληθυσμό για τους αποκριθέντες και μη. Εδώ φαίνεται ότι ο πλέον αποτελεσματικός τρόπος να έχουμε χαμηλά επίπεδα μεροληψίας λόγω μη-απόκρισης είναι να φροντίσουμε να έχουμε υψηλό ποσοστό απαντήσεων, καθώς οι τιμές Y_R και Y_M (ιδιαίτερα η δεύτερη) δεν είναι κατά κανόνα γνωστές.

Ο Donald Rubin (27) (28) , εισήγαγε πρώτος την κατηγοριοποίηση που έχει επικρατήσει στη βιβλιογραφία με τον όρο ***μηχανισμός ελλειπουσών τιμών*** (missing data mechanism).

Έστω Y_{obs} οι παρατηρούμενες τιμές στο δείγμα, Y_{mis} οι ελλείπουσες, R οι τιμές απόκρισης και $Prob (R \mid Y_{obs}, Y_{mis}, \vartheta)$ η κατανομή πιθανότητας της τυχαίας μεταβλητής R με δεδομένο ένα στατιστικό μοντέλο με παράμετρο ϑ και απόκριση Y. Θα λέμε ότι δεδομένα λείπουν **κατά τυχαίο τρόπο** (missing at random, MAR) όταν ισχύει

$$Prob (R \mid Y_{obs}, Y_{mis}, \vartheta) = Prob (R \mid Y_{obs}, \vartheta),$$

δηλαδή όταν η κατανομή των αποκρίσεων εξαρτάται μόνο από τις παρατηρούμενες τιμές, όχι όμως και από αυτές που λείπουν.

Η ισχυρότερη συνθήκη

$$Prob (R \mid Y_{obs}, Y_{mis}, \vartheta) = Prob (R \mid \vartheta),$$

δηλαδή όταν οι αποκρίσεις εξαρτώνται μόνο από το στατιστικό μοντέλο και καθόλου από τις τιμές των μεταβλητών, παρατηρούμενες ή μη, αντιστοιχεί στην περίπτωση έλλειψης **κατά εντελώς τυχαίο τρόπο** (missing completely at random, MCAR). Στη περίπτωση αυτή, οι ελλείψεις μπορούν να αγνοηθούν εντελώς κατά την ανάλυση, καθώς έχουν προκύψει με τρόπο που διαισθητικά εννοούμε ως «τυχαίο». Κάτι τέτοιο πάντως είναι μάλλον σπάνιο, ενώ μάλλον συχνότερα απαντάται η πρώτη περίπτωση. Μπορούμε όμως και τότε να διαχωρίσουμε τα δεδομένα σε κλάσεις (ανάλογες με τα στρώματα στη δειγματοληψία) με παρόμοια πεδία Y_{obs} και, για κάθε τέτοια κλάση, να χρησιμοποιήσουμε ως βάση τις

[10] Αναφέρεται – ως ακραίο αλλά χαρακτηριστικό παράδειγμα – μια έρευνα των Βρετανών κατά το Β' παγκόσμιο πόλεμο κατά την οποία εξετάζονταν τα χτυπημένα αεροσκάφη που γύριζαν από αερομαχίες ώστε να διαπιστωθούν τα πιο ευαίσθητα σημεία στην άτρακτο και να ενισχυθούν κατάλληλα. Η έρευνα πήρε τελείως άλλη τροπή τη στιγμή που κάποιος παρατήρησε εύλογα ότι τα πιο ευαίσθητα σημεία ήταν ακριβώς αυτά που *δεν* παρατηρούνταν στις υπό εξέταση ατράκτους, καθώς τα αεροπλάνα που είχαν χτυπηθεί εκεί δεν είχαν καν καταφέρει να επιστρέψουν στη βάση.

καταγεγραμμένες τιμές για να εκτιμήσουμε τις ελλείπουσες. Έτσι μπορούμε και εδώ να θεωρήσουμε ότι η μη-απάντηση τελικά *μπορεί να αγνοηθεί* (ignorable nonresponse) με τον κατάλληλο χειρισμό.

Στο άλλο άκρο συναντούμε την περίπτωση ελλειπουσών τιμών **κατά μη τυχαίο τρόπο** (missing not at random, MNAR), όταν το ενδεχόμενο να μην έχει καταγραφεί μια τιμή εξαρτάται όχι μόνο από άλλες καταγεγραμμένες τιμές αλλά και από αυτήν ακριβώς την τιμή που λείπει[11]. Σε μια τέτοια περίπτωση, η μη-απάντηση *δε μπορεί να αγνοηθεί* (non-ingorable nonresponse), καθώς δεν μπορεί κανείς να εκτιμήσει τις τιμές που λείπουν με βάση τα καταγεγραμμένα δεδομένα της έρευνας και μόνο, άρα πρέπει να προβεί σε ad hoc παραδοχές πάνω στο συγκεκριμένο πλαίσιο της δειγματοληψίας και το μοντέλο που περιγράφει το πρόβλημα. (29)

Μπορεί κανείς κατά βάση να διακρίνει τέσσερα κύρια είδη ελλείψεων στα δεδομένα. Καταρχάς έχουμε την ολική μη-απάντηση (ή **μη-απάντηση μονάδας** – unit nonresponse), όταν έχουμε πλήρη έλλειψη στοιχείων για μια μονάδα του πληθυσμού η οποία έχει επιλεγεί στο δείγμα. Η αντιστάθμιση για τέτοιες ελλείπουσες τιμές γίνεται συνήθως μέσα από αναπροσαρμογές βαρών στην εκτιμήτρια συνάρτηση, δίνεται δηλαδή μεγαλύτερο βάρος στις τιμές απόκρισης αυτών που απάντησαν ώστε να εκπροσωπήσουν και αυτούς που αρνήθηκαν. Ένα δεύτερο είδος ελλειπουσών τιμών προκύπτει από τη **μη-απάντηση στοιχείου** (item nonresponse), όταν μια περίπτωση στο δείγμα δεν έχει δώσει απαντήσεις παρά για κάποια μόνο από τα ερωτώμενα στοιχεία της έρευνας. Ο συνήθης τρόπος αντιμετώπισης τέτοιων ελλείψεων είναι ο καταλογισμός (ή τεκμαρτή εκτίμηση – imputation). Ένα τρίτο είδος είναι η **μη-κάλυψη**, που αναφέρθηκε προηγούμενα, όταν κάποιες μονάδες βρίσκονται εκτός του πλαισίου δειγματοληψίας και επομένως η πιθανότητα επιλογής τους στο δείγμα είναι μηδενική. Τέλος, υπάρχει η **μερική μη-απάντηση**, όπου ο ερωτώμενος σταματάει να δίνει απαντήσεις από κάποιο σημείο και μετά. (31)

Οι τεχνικές **επαναπροσδιορισμού** (reweighting) και **ρύθμισης βαρών** (weight calibration), με τις οποίες αντιμετωπίζεται κατά κανόνα η μη-απάντηση μονάδας, είναι στην ουσία μια γενίκευση των βαρών που προβλέπει η θεωρία σχετικά με τις εκτιμήτριες Horvitz-Thompson για τη στρωματοποιημένη δειγματοληψία (30). Στην περίπτωση του **καταλογισμού τιμής** (imputation) έχουμε διάφορες επιλογές. Μπορούμε για παράδειγμα να πάρουμε την μέση τιμή κάθε κλάσης και να την τοποθετήσουμε στις κενές θέσεις – έχουμε τότε *καταλογισμό μέσης τιμής* (mean value imputation). Πρόκειται για μια γρήγορη μέθοδο, αρκετά ικανοποιητική στην εκτίμηση μέσων τιμών ενός πληθυσμού που όμως διαστρεβλώνει την πραγματική κατανομή εμφανίζοντας πλασματικά μεγάλο αριθμό όμοιων τιμών και επομένως υπο-εκτιμώντας το τυπικό σφάλμα. Μια τεχνική με συχνά καλύτερα αποτελέσματα είναι ο καταλογισμός *hot-deck*, όπου διατρέχουμε τις μονάδες στη σειρά συμπληρώνοντας μια ελλείπουσα τιμή με την υπάρχουσα που συναντήσαμε τελευταία. Ένα είδος hot-deck είναι ο καταλογισμός του *πλησιέστερου γείτονα* (nearest neighbor imputation) και το *ταίριασμα κατά απόσταση* (distance function matching), όπου γίνεται χρήση μιας μετρικής αποστάσεων για να βρεθούν οι «κοντινότεροι» δότες με βάση μεταβλητές που είναι παρατηρήσιμες για όλες τις περιπτώσεις. Τέλος, ο καταλογισμός

[11] Χαρακτηριστικό παράδειγμα είναι τα στοιχεία για το προσωπικό εισόδημα που δίνονται σε μια έρευνα, καθώς έχει παρατηρηθεί ότι οι ερωτώμενοι με υψηλό εισόδημα αποφεύγουν κατά κανόνα να απαντήσουν σε μεγαλύτερο ποσοστό από το μέσο όρο, υπάρχει δηλαδή ισχυρή συσχέτιση ανάμεσα στη μεταβλητή «εισόδημα» και την πιθανότητα απόκρισης του ατόμου.

μέσω παλινδρόμησης (regression imputation) χρησιμοποιεί πολλαπλή γραμμική παλινδρόμηση της ελλιπούς μεταβλητής πάνω σε άλλες πλήρως παρατηρούμενες μεταβλητές. (32) (33) Ας σημειωθεί τέλος ότι τόσο η ρύθμιση των βαρών όσο και ο καταλογισμός τιμής, σε όλες τις παραπάνω εκδοχές, αποσκοπούν κατά βάση στη μείωση του ποσού μεροληψίας μιας εκτιμήτριας.

Πολλές φορές επιλέγουμε να αποσύρουμε κάποιες από τις ελλειπείς περιπτώσεις προτού προχωρήσουμε στον υπολογισμό των παραμέτρων. Η **διαγραφή από τη λίστα** (ή πλήρης διαγραφή περίπτωσης, listwise ή casewise deletion) αφαιρεί από το δείγμα όλες τις περιπτώσεις στις οποίες λείπει έστω και ένα στοιχείο. Αν βέβαια οι περιπτώσεις αυτές δεν αποτελούν ένα εντελώς τυχαίο δείγμα του συνολικού πληθυσμού, προκύπτει μεροληψία. Από την άλλη μεριά, η **διαγραφή σε επίπεδο ζεύγους** (pairwise deletion) απορρίπτει τις ελλειμματικές περιπτώσεις για κάθε μια ξεχωριστά από τις μεταβλητές σε συνδυασμό με τη μεταβλητή απόκρισης. Αυτό μπορεί να είναι πολύ χρήσιμο σε έρευνες με πολλές μεταβλητές, άλλες με περισσότερες και άλλες με λιγότερες ελλείψεις. Έτσι παραμένουν στο δείγμα περισσότερες παρατηρήσεις για κάθε ζεύγος μιας ανεξάρτητης μεταβλητής και της απόκρισης, με το μειονέκτημα όμως ότι μπορεί ο πίνακας συνδιακύμανσης να μην είναι πλέον θετικά ορισμένος και επομένος ακατάλληλος για περαιτέρω ανάλυση, π.χ. για πολυμεταβλητή παλινδρόμηση. (29)

Κατά την ανάλυση πάνω στη βάση καταγραφής ISBSG που ακολουθεί θα συναντήσουμε το πρόβλημα της αποχής. Εκεί θα δοθούν κάποια παραπάνω στοιχεία για το φαινόμενο αυτό (βλ. Ενότητα 11.2.2).

7.4 Ακραίες τιμές

7.4.1 Γενικά

Ο εντοπισμός των ακραίων τιμών σε ένα σύνολο δεδομένων, ιδιαίτερα σε ένα πολυ-διάστατο δείγμα όπως αυτά που προκύπτουν κατά κανόνα στην εμπειρική τεχνολογία λογισμικού, είναι ένα πρώτο βασικό βήμα πριν την επεξεργασία που λαμβάνει χώρα στη συνέχεια. Ως **ακραία τιμή** (outlier) λογίζεται μια παρατήρηση που παρεκκλίνει τόσο πολύ από τις υπόλοιπες ώστε να εγείρει την υποψία ότι προήλθε από ένα διαφορετικό μηχανισμό, με άλλα λόγια πρόκειται για ένα στιγμιότυπο που είναι ασυνεπές ως προς το υπόλοιπο σύνολο των δεδομένων. (34)

Δεν είναι πάντοτε όλες οι ασυνήθιστες τιμές ενός δείγματος ακραίες – κάποιες από αυτές θα μπορούσαν να είναι απλώς «θόρυβος». Στην πραγματικότητα, δεν είναι τόσο η απόσταση μιας τιμής αυτή που προσδίδει υποχρεωτικά το χαρακτηρισμό της «ακραίας» – επαφίεται μάλλον στον ίδιο τον αναλυτή να καθορίσει τη διαχωριστική γραμμή ανάμεσα στον κύριο όγκο των δεδομένων και τις ακραίες τιμές. Προφανώς, οι ανώμαλες τιμές αναμένεται να είναι ακραίες σε μεγαλύτερο βαθμό από αυτές που θα αποδεχτούμε ως θόρυβο. Στην *μη-επιβλεπόμενη* (unsupervised) περίπτωση, κατά την οποία δεν είναι διαθέσιμα προηγούμενα παραδείγματα ακραίων τιμών, ο θόρυβος αποτελεί το σημασιολογικό σύνορο ανάμεσα στα υγιή δεδομένα και τις ανωμαλίες – είναι δηλαδή μια ασθενής μορφή ακραίας τιμής, η οποία δεν πληροί τα κριτήρια που θέτει ο αναλυτής. Ο καλύτερος τρόπος πάντως να διαχωριστούν οι ανωμαλίες από το θόρυβο είναι εκείνες με βάση παλαιότερα παραδείγματα ακραίων τιμών (ή καμιά φορά επίσης τη γνώση ειδικών πάνω στο συγκεκριμένο τομέα

που μελετάμε), γι' αυτό και οι *επιβλεπόμενες* μέθοδοι εντοπισμού είναι κατά κανόνα πολύ πιο αποτελεσματικές. (35)

Κατά τη διαδικασία του εντοπισμού ακραίων τιμών, είναι δυνατό να προκύψει αρχικά μια ακραία τιμή και, μετά την αφαίρεσή της, να εντοπιστεί με τη σειρά της ως ακραία μια δεύτερη τιμή, η οποία ως τότε έδειχνε φυσιολογική ***(masking effect)***. Μπορεί επίσης να παρατηρηθεί και το αντίθετο, δηλαδή μια τιμή να εμφανίζεται ως ακραία μόνο υπό την παρουσία μιας άλλης ακραίας τιμής, μόλις δε η δεύτερη αφαιρεθεί από το δείγμα η πρώτη αυτομάτως να καταστεί φυσιολογική ***(swamping effect)***. (34) Και τα δύο αυτά φαινόμενα είναι ενδεικτικά των δυσκολιών που μπορεί να προκύψουν κατά τον εντοπισμό ακραίων τιμών και γι' αυτό ο καθαρισμός και προετοιμασία του δείγματος χρειάζεται να γίνεται με ιδιαίτερη προσοχή.

Οι μέθοδοι εντοπισμού ακραίων τιμών μπορεί να είναι είτε **παραμετρικές** (στατιστικές) είτε **μη παραμετρικές**. Στις τελευταίες συγκαταλέγονται μεταξύ άλλων τεχνικές ομαδοποίησης (clustering) και μέθοδοι με βάση την απόσταση (distance-based). (34) Οι παραμετρικές μέθοδοι εξάλλου υιοθετούν ένα μοντέλο για τα δεδομένα και στη συνέχεια βαθμολογούν κάθε στιγμιότυπο ως προς την απόκλισή του από το μοντέλο αυτό. Μια συχνή επιλογή μοντέλου είναι η **κανονική κατανομή** $N(\mu, \sigma^2)$, οπότε το πρόβλημα του εντοπισμού μεταφράζεται στην εύρεση των παρατηρήσεων που βρίσκονται στην περιοχή:

$$out(k, \mu, \sigma^2) = \{ x : |x - \mu| > k\sigma \},$$ με τυπική τιμή της παραμέτρου $k = 5$.

Γενικά, χρειάζεται προσοχή ώστε να μην απορριφθούν βιαστικά κάποια στιγμιότυπα ως ακραία λόγω μη «συμμόρφωσής» τους προς το επιλεγμένο μοντέλο, έτσι που τα εναπομείναντα δεδομένα να μην επαρκούν ώστε να μας καθοδηγήσουν στην απόρριψη του εσφαλμένου μοντέλου, όπως θα έπρεπε (35). Σε δεδομένα όπως αυτά που προέρχονται από μετρήσεις λογισμικού κάτι τέτοιο μπορεί να συμβεί, καθώς οι στατιστικές κατανομές απέχουν συχνά σημαντικά από το κανονικό μοντέλο. Μια εναλλακτική μέθοδος εντοπισμού ακραίων τιμών στηρίζεται στο **διατεταρτημοριακό εύρος τιμών** (inter-quartile range, IQR), δηλαδή το τμήμα ανάμεσα στις τιμές που αντιστοιχούν στο 25% και το 75% του συνόλου των τιμών (δηλ. την περιοχή του δεύτερου και του τρίτου τεταρτημορίου). Η μέθοδος αυτή, που ταιριάζει καλύτερα σε κάποιες μη συμμετρικές κατανομές[12], χαρακτηρίζει ως ακραία μια τιμή x_{out} για την οποία ισχύει μια από τις δύο ανισότητες:

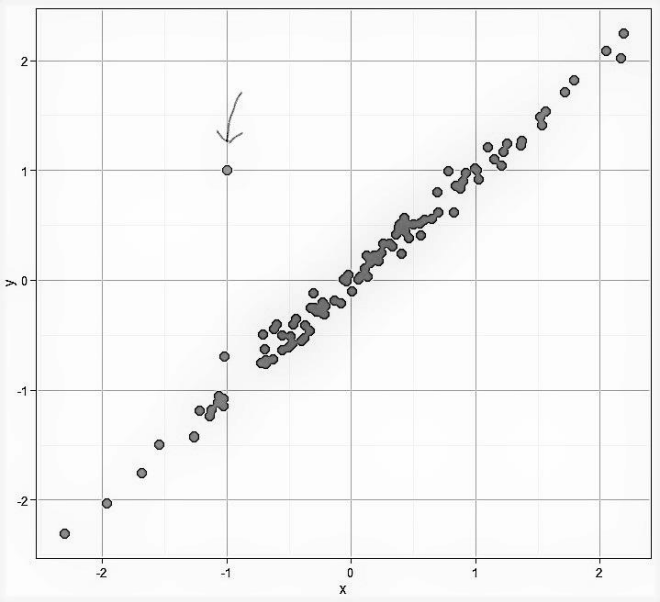
$$x_{out} > x_{0.75} + 1.5 * IQR \qquad \text{ή} \qquad x_{out} < x_{0.25} - 1.5 * IQR$$

7.4.2 Ακραίες τιμές σε πολυδιάστατα δεδομένα

Η παρουσία ακραίων τιμών είναι πιο πιθανή όσο αυξάνει ο αριθμός των στιγμιοτύπων και των μεταβλητών. Ένα βασικό πρόβλημα είναι ότι συχνά συμβαίνει να μην είναι δυνατός ο εντοπισμός τους αν θεωρήσουμε την κάθε μια μεταβλητή ανεξάρτητα. Αυτές οι ακραίες τιμές δεν εντοπίζονται παρά μόνο όταν ληφθούν υπόψη οι *αλληλεπιδράσεις* μεταξύ των ανεξάρτητων μεταβλητών.

[12] Θα γίνει χρήση αυτής της μεθόδου πιο κάτω κατά την ανάλυση των δεδομένων του ISBSG Repository ως προς την παραγωγικότητα των έργων λογισμικού.

Όπως φαίνεται και στην Εικόνα 7-1, η τιμή που δείχνεται με το βέλος είναι προφανώς ακραία, αλλά αν θεωρηθεί μεμονωμένα για την κάθε μια ανεξάρτητη μεταβλητή, φαίνεται να βρίσκεται κοντά στο κέντρο της μονοδιάστατης κατανομής και επομένως δεν εντοπίζεται. (34)



Εικόνα 7-1: Διδιάστατος χώρος με μια ακραία παρατήρηση

Από την άλλη μεριά, η ύπαρξη πολλών διαστάσεων επιτείνει από τη μια το φαινόμενο του θορύβου και κάνει από την άλλη όλα τα σημεία να φαίνονται ότι ισαπέχουν λίγο πολύ μέσα στον πολυ-διάστατο χώρο. Σε τέτοιες περιπτώσεις, οι ακραίες τιμές αναδύονται καλύτερα στην επιφάνεια αν τις εξετάσουμε σε ένα υποσύνολο του αρχικού χώρου που ορίζεται μόνο από τα χαρακτηριστικά με άμεση συνάφεια. Αν και η αναζήτηση τέτοιων υποχώρων καθίσταται δυσχερής υπολογιστικά για μεγάλο πλήθος διαστάσεων, μπορεί συχνά να δώσει σημαντικά αποτελέσματα στον εντοπισμό ακραίων τιμών. (35)

7.5 Εγκυρότητα στις μετρήσεις λογισμικού

Η ποιότητα μιας εμπειρικής έρευνας συναρτάται άμεσα με τα «διαπιστευτήρια» εγκυρότητας που διαθέτει, βασικά ως προς τους ακόλουθους τέσσερις τομείς: (14)

1. **Εγκυρότητα στο συμπέρασμα:** ένα συμπέρασμα είναι έγκυρο με την προϋπόθεση ότι έχει εξαχθεί μέσα από στατιστικούς ελέγχους στο επιθυμητό επίπεδο σημαντικότητας. Αυτό μπορεί να είναι διαβλητό εάν χρησιμοποιηθούν ακατάλληλες για την περίπτωση στατιστικές μέθοδοι, εάν το δείγμα είναι υπερβολικά μικρό ή εάν ερευνώνται σχέσεις ανάμεσα σε πολλές μεταβλητές, οπότε αυξάνεται η πιθανότητα να προκύψουν ψευδείς συσχετίσεις.
2. **Εγκυρότητα στην κατασκευή**: οι μετρήσεις που πραγματοποιούνται πρέπει να αναφέρονται σε μεγέθη που είναι σχετικά με το ζητούμενο της έρευνας. Για παράδειγμα, ο υπολογισμός των

σφαλμάτων ως μέτρο της ποιότητας λογισμικού καταλήγει σε άλλα συμπεράσματα ανάλογα εάν μιλάμε για τα σφάλματα στη φάση ελέγχου ή αυτά μετά την παράδοση.

3. **Εσωτερική εγκυρότητα**: αφορά στη σχέση αιτίου / αιτιατού μεταξύ ανεξάρτητων και εξαρτημένων μεταβλητών. Όπως είναι γνωστό, οι στατιστικές συσχετίσεις δεν αντιστοιχούν πάντα σε σχέσεις αιτιότητας, καθώς θα μπορούσαν να οφείλονται σε τρίτες, κρυμμένες μεταβλητές. Χρειάζεται επομένως μια a priori θεωρητική εξήγηση ως προς το εάν και πώς μια ανεξάρτητη μεταβλητή Α επηρεάζει πράγματι την εξαρτημένη μεταβλητή Β. Πρέπει επίσης να ληφθεί υπόψη η σωστή χρονική αλληλουχία, διαφορετικά θα μπορούσε το ίδιο καλά να είναι η Β αυτή που προκαλεί τη μεταβολή της Α. Ένας ακόμη παράγοντας που απειλεί την εσωτερική εγκυρότητα είναι το ζήτημα των ελλειπουσών τιμών και η μεροληψία που επιφέρει.
4. **Εξωτερική εγκυρότητα**: αφορά στη δυνατότητα μεταφοράς των ευρημάτων από το στενό περιβάλλον της έρευνας στο γενικότερο περιβάλλον που αντιμετωπίζεται στην πράξη. Συναφής είναι και η δυνατότητα επανάληψης ενός πειράματος ή μιας μελέτης από διαφορετικές ερευνητικές ομάδες στις ίδιες συνθήκες ώστε να δούμε εάν καταλήγουν σε ανάλογα αποτελέσματα.

Ειδικότερα, στις εμπειρικές μελέτες και συγκριτικές αναλύσεις (benchmarking) πάνω στην τεχνολογία λογισμικού πρέπει κανείς να λαμβάνει υπόψη του κάποια ζητήματα που ανακύπτουν συχνά, όπως για παράδειγμα (3)

- *Διαφοροποιήσεις στον καθορισμό του μεγέθους λογισμικού*. Οι μετρικές FP, όταν αυτές υπολογίζονται από εξειδικευμένο και αναγνωρισμένο προσωπικό, είναι ιδιαίτερα ακριβείς με διακύμανση γύρω στο 3%. Από την άλλη μεριά, οι εμπειρικές μέθοδοι μετατροπής του μεγέθους από LOC σε FP μπορεί να έχουν διακυμάνσεις της τάξης του 25%, καμιά φορά δε έως και 100% (3). Καθώς όμως αυτές εφαρμόζονται εξ ανάγκης σε μεγάλο αριθμό «κληρονομημένων» (legacy) συστημάτων, δημιουργούν ζήτημα ως προς την ακρίβεια και εγκυρότητα στον υπολογισμό του μετρούμενου μεγέθους.
- *Διαφοροποιήσεις στον υπολογισμό της συνολικής προσπάθειας*. Ένα σημαντικό θέμα που ανακύπτει αναφορικά με τα στοιχεία από βάσεις καταγραφής δεδομένων, όπως το ISBSG Repository, είναι η διαρροή (leakage) κατά τον προσδιορισμό της εργασίας που απαιτεί η υλοποίηση του λογισμικού. Επιμέρους στοιχεία που αφορούν σε ώρες εργασίας, ανθρωπο-μήνες κτλ. συχνά ξεφεύγουν από τον υπολογισμό σε ποσοστό από 30% έως και 70%. Αυτό μπορεί να οφείλεται σε ελλιπή καταμέτρηση της προσπάθειας για φάσεις εκτός της κωδικοποίησης και ελέγχου, ανακριβή καταγραφή των ωρών κανονικής εργασίας, υπερωριών, διακοπών και αργιών (ιδιαίτερα για εφαρμογές ιστού) κοκ. Επομένως, η παραγωγικότητα που καταγράφεται τείνει να εμφανίζει πλασματικά υψηλότερες τιμές. Ενδεικτικά, για έργα μεγαλύτερα των 1000 FP, ρυθμοί παράδοσης άνω των 10 FP / ανθρωπο-μήνα πρέπει να αντιμετωπίζονται με επιφύλαξη, όπως επίσης ρυθμοί άνω των 15 FP / ανθρωπο-μήνα για ίδιου μεγέθους εφαρμογές που έχουν αναπτυχθεί με μεθοδολογίες agile.

8 ΕΚΤΙΜΗΣΗ ΠΑΡΑΜΕΤΡΩΝ ΛΟΓΙΣΜΙΚΟΥ

8.1 Γενικά

Στην πράξη συναντούμε δεκάδες μεθοδολογίες, εργαλεία και μοντέλα εκτίμησης λογισμικού. Κατά κύριο λόγο παράγουν ως έξοδο μιαν εκτίμηση για τα βασικά μεγέθη ενός έργου λογισμικού, όπως είναι π.χ. ο συνολικός χρόνος για την ανάπτυξη του λογισμικού, η ανθρώπινη προσπάθεια που θα απαιτηθεί κοκ. Ως παραμέτρους εισόδου δέχονται μια μεγάλη ποικιλία χαρακτηριστικών όπως το μέγεθος, την πολυπλοκότητα, τις μεθοδολογίες ανάπτυξης κτλ., τις οποίες συχνά συνδυάζουν σε κάποιο αναλυτικό τύπο. Όπως έχει ήδη αναφερθεί, η ευρεία χρήση της Ανάλυσης Λειτουργικών Σημείων έχει δώσει τη δυνατότητα της πρόγνωσης της πυκνότητας ελαττωμάτων με βάση μετρικές που μπορούν να εξαχθούν ήδη στο στάδιο των προδιαγραφών. Αυτό, σε συνδυασμό με την εδραιωμένη πεποίθηση ότι τα FP υπερτερούν των LOC (βλ. και παρ. 4.2.2 ανωτέρω), έχει μετατοπίσει το βάρος των εμπειρικών ερευνών προς την πλευρά των μετρικών πάνω στο λειτουργικό μέγεθος. (18)

Ένας διαδεδομένος τρόπος εκτίμησης είναι μέσα από *συγκριτική ανάλυση δεδομένων (*θα δοθούν περισσότερα στοιχεία στην Ενότητα 8.3 παρακάτω). Μια δεύτερη μέθοδος είναι η *εκτίμηση κατ' αναλογίαν*: η ιδέα είναι να αναζητηθούν ολοκληρωμένα έργα παρόμοια με το προκείμενο και να γίνει χρήση της εμπειρίας και της πληροφορίας από αυτά. Η ομοιότητα ή αναλογία υπολογίζεται με βάση την ευκλείδεια απόσταση σε έναν ν-διάστατο χώρο, όπου κάθε μια από τις ν διαστάσεις αντιστοιχεί σε ένα χαρακτηριστικό. Ένα τυπικό όριο είναι το +-25%: αν τα δύο έργα διαφέρουν κατά περισσότερο από 25%, πρέπει να βρεθεί κοντινότερο ανάλογο ή να εφαρμοστεί άλλη μέθοδος. (10)

Μια άλλη διαδεδομένη μέθοδος είναι η *εκτίμηση με βάση το λειτουργικό μέγεθος*. Η διάρκεια ενός έργου σε ημερολογιακούς μήνες, για παράδειγμα, μπορεί να εκτιμηθεί από τον ακόλουθο εμπειρικό τύπο:

$$Schedule = Size^{d} \qquad \textbf{(8-1)}$$

όπου *Size* είναι το μέγεθος του έργου σε *FP*. Ο εκθέτης d παίρνει τις ακόλουθες τιμές ανάλογα με τον τύπο της εφαρμογής.

Τύπος Project	Προτεινόμενο d
Military	0,45
Average	0,4
Simple	0,32

Large Web	0,32
Small Web	0,5

Ο αριθμός των ατόμων που απασχολούνται στην εφαρμογή σχετίζεται με το λειτουργικό μέγεθος του έργου μέσα από τους ακόλουθους τύπους:

Staff = Size/150 (για καινούργια έργα) και

Staff = Size/750 (για έργα συντήρησης) ***(8-2)***

Σημειώνεται ότι στον αριθμό αυτό περιλαμβάνεται το σύνολο του προσωπικού που απασχολείται στο έργο (μηχανικοί λογισμικού, προγραμματιστές, ομάδες ελέγχου, ομάδα διασφάλισης ποιότητας, διαχειριστές, γραμματειακή υποστήριξη κοκ.)

Η συνολική προσπάθεια (σε ανθρωπο-μήνες) που καταβάλλεται ως την ολοκλήρωση του έργου προκύπτει ως το γινόμενο των δύο εκτιμήσεων που δίνονται από τις *(8-1)* και *(8-2)*:

*Effort = Staff * Schedule* ***(8-3)***

Σημ.: το μοντέλο έχει ενσωματώσει μια πρόβλεψη ως προς την αύξηση των προδιαγραφών χρήστη (user requirements) κατά 1-5% από τη φάση σχεδίασης μέχρι και τη φάση ελέγχου, ανάλογα με τη μεθοδολογία που ακολουθείται. Κατά μέσο όρο η αύξηση αυτή εκτιμάται σε 2%. (10)

Τέλος, υπάρχουν τα *αλγοριθμικά μοντέλα,* που προκύπτουν μέσα από εμπειρικές έρευνες με χρήση της ανάλυσης παλινδρόμησης[13]. Στο περιβάλλον της Ανάλυσης Λειτουργικών Σημείων έχουν προταθεί μια σειρά τέτοια μοντέλα, όπως τα εξής:

- *Albrecht – Gaffney* — $Effort = 13,39 + 0,0545 * FP$
- *Kemerer* — $Effort = 60,62 + 7,728 * (10^{-8}) * FP^3$
- *Matson – Barret – Meltichamp* — $Effort = 585,7+ 15,12 * FP$

Είναι φανερό ότι το κάθε μοντέλο οδηγεί σε διαφορετικό αποτέλεσμα[14]. Αυτό συμβαίνει, καθώς τα μοντέλα αυτά έχουν προκύψει από την στατιστική επεξεργασία ενός συγκεκριμένου συνόλου εμπειρικών δεδομένων και σε διαφορετικά περιβάλλοντα. Γι' αυτό και η εφαρμογή τέτοιων μεθόδων εκτίμησης πρέπει να γίνεται λαμβάνοντας υπόψη τα ιδιαίτερα χαρακτηριστικά του λογισμικού που εξετάζεται και επιλέγοντας το «κοντινότερο» μοντέλο κατά περίπτωση. (10)

8.2 Διερευνητική ανάλυση και εξόρυξη δεδομένων

Τα δεδομένα που προκύπτουν από τις μετρήσεις λογισμικού είναι κατά κανόνα κατηγορικής ή διατακτικής φύσης (π.χ. οι διάφορες πλατφόρμες εφαρμογής, η σπουδαιότητα των σφαλμάτων κοκ.).

[13] Το πλέον διαδεδομένο αλγοριθμικό μοντέλο, το οποίο κάνει χρήση των μετρικών γραμμής κώδικα (*KLOC*), είναι το *COCOMO* (βλ. Κεφ. 3.1)

[14] Είναι αξιοσημείωτο ότι τα δύο από αυτά προβλέπουν παραγωγικότητα σταθερή, ανεξάρτητη του μεγέθους.

Εξάλλου, ακόμα και αν πρόκειται για συνεχείς μεταβλητές, πολλές φορές διακριτοποιούμε τις μετρήσεις τοποθετώντας τις σε κλάσεις ανά κατηγορία τιμών (binning). Στην πράξη, τα δεδομένα μας είναι λοιπόν συχνά *διακριτά και μη αρνητικά*, οι δε κατανομές τους (ακόμα και για σχετικά μεγάλα δείγματα) δεν προσομοιάζουν στην κανονική κατανομή αλλά είναι συνήθως στρεβλές (skewed) προς τα αριστερά. Επίσης εμφανίζουν κατά κανόνα κάποιες ιδιαίτερα μεγάλες τιμές με αποτέλεσμα οι μέσες τιμές (means) να είναι σημαντικά μεγαλύτερες από τις διαμέσους (medians), γεγονός που επιτείνεται και από την παρουσία ακραίων τιμών (outliers). Όμως, μεγάλο μέρος της «κλασικής» στατιστικής συμπερασματολογίας ασχολείται με τις παραμέτρους κεντρικής τάσης και διασποράς *συνεχών* μεταβλητών. Εκ των πραγμάτων λοιπόν, η εμπειρική τεχνολογία λογισμικού καλείται να στραφεί σε πιο εξειδικευμένα εργαλεία της στατιστικής ανάλυσης, όπως π.χ. η λογιστική (logistic) και η ευσταθής παλινδρόμηση (robust correlation) με τους συντελεστές συσχέτισης Spearman και Kendall και γενικά μη-παραμετρικοί έλεγχοι (χ^2, Wilcoxon, Mann-Whitney και Kruskal-Wallis) (14)

Πέρα από αυτά, ο ίδιος ο τρόπος με τον οποίο συλλέγουμε δεδομένα όπως επίσης το μέγεθος, η ποικιλία και το περιεχόμενό τους έχουν αλλάξει ριζικά τα τελευταία χρόνια. Σήμερα, η χρήση ισχυρών υπολογιστών και εργαλείων μας δίνει τη δυνατότητα αυτοματοποιημένων συστημάτων συλλογής που επιτρέπουν τεράστιους όγκους δεδομένων, χωρίς να έχουμε όμως οι ίδιοι δυνατότητα ελέγχου πάνω στο σχεδιασμό ή τον τρόπο συλλογής. Από την κλασική σκοπιά της στατιστικής συμπερασματολογίας, το γεγονός αυτό εγείρει ζητήματα ως προς την ποιότητα και την αποδοχή των συλλεγόμενων στοιχείων. (36) Από την άλλη μεριά, μέσα σε αυτό τον όγκο δεδομένων μπορεί να κρύβεται εν δυνάμει χρήσιμη πληροφορία, η οποία δύσκολα καθίσταται ευδιάκριτη και εκμεταλλεύσιμη[15].

Από τη μεριά της στατιστικής, αυτό το κενό επιχειρεί να καλύψει η **διερευνητική ανάλυση δεδομένων** (exploratory data analysis, EDA). Η ανάλογη τάση από την πλευρά της επιστήμης της πληροφορικής οδήγησε στην **ανακάλυψη γνώσης στις βάσεις δεδομένων** (knowledge discovery in databases, KDD), ενώ στο σημείο σύγκλισης των δύο αυτών τάσεων βρίσκεται η **εξόρυξη δεδομένων** (data mining, DM), δηλαδή η ανάλυση (συχνά πολύ μεγάλου μεγέθους) δεδομένων από παρατηρήσεις (observational data) αναζητώντας μη αναμενόμενες σχέσεις και συνοψίζοντας τα δεδομένα με νέους τρόπους, κατανοητούς και χρήσιμους για τον χρήστη της πληροφορίας. Οι σχέσεις και τα συνοπτικά αποτελέσματα που παράγονται αναφέρονται συνήθως ως **μοντέλα** (models) ή **μοτίβα** (patterns), ανάλογα με το αν έχουν εφαρμογή σε καθολικό ή τοπικό επίπεδο αντίστοιχα. Οι γενικεύσεις που παρέχει η εξόρυξη δεδομένων μπορεί να μην είναι εφικτές μέσα από τυπικές στατιστικές προσεγγίσεις γιατί συχνά τα δεδομένα μας δεν αποτελούν τυχαία δείγματα με την κλασική στατιστική έννοια, αλλά μάλλον δείγματα ευκαιρίας ή ευκολίας. (37)

Ο κεντρικός στόχος είναι λοιπόν κατά βάση η ανακάλυψη άγνωστων και μη αναμενόμενων δομών ή μοτίβων στα δεδομένα μας[16]. Τα μοτίβα αυτά πρέπει να έχουν νόημα, με την έννοια ότι πρέπει να οδηγούν σε κάποιο πλεονέκτημα, συνήθως οικονομικής και επιχειρησιακής φύσης, επιτρέποντάς μας να κάνουμε προγνώσεις για νέα δεδομένα που θα προκύψουν στο μέλλον. (38)

[15] Τα δεδομένα από το ISBSG Repository, τα οποία θα αναλύσουμε στη συνέχεια, αποτελούν ένα χαρακτηριστικό παράδειγμα.

[16] Ο χώρος αυτός έχει μεγάλη συνάφεια με εκείνον της μη-επιβλεπόμενης μηχανικής μάθησης (un-supervised machine learning) και οι τεχνικές είναι σε μεγάλο βαθμό παρεμφερείς ή και ταυτόσημες.

Η συνεργασία ανάμεσα στους ειδικούς της στατιστικής ανάλυσης και σε αυτούς της εξόρυξης δεδομένων δεν ήταν πάντοτε χωρίς προβλήματα και ανταγωνισμούς[17]. Στην πραγματικότητα, δεν υφίσταται μια καθαρή διαχωριστική γραμμή ανάμεσα στις δύο έννοιες, αλλά μάλλον ένα συνεχές – και μάλιστα πολυμεταβλητό – πεδίο τεχνικών ανάλυσης δεδομένων. Θα μπορούσε ίσως κανείς να κάνει μια διάκριση ως προς την έμφαση: η στατιστική τείνει να ασχολείται περισσότερο με τον έλεγχο υποθέσεων, ενώ η εξόρυξη δεδομένων προσπαθεί μάλλον να διαμορφώσει μια διαδικασία γενίκευσης διαμέσου της αναζήτησης ανάμεσα σε διάφορες υποθέσεις. Πάντως, αν και ιστορικά οι δύο πλευρές ξεκίνησαν από διαφορετικές αφετηρίες, οι δύο αντιλήψεις έχουν πλέον συγκλίνει σημαντικά και οι τεχνικές της εξόρυξης δεδομένων ενσωματώνουν πια μεγάλο μέρος της στατιστικής σκέψης. (38)

Στη συνέχεια περιγράφονται πολύ περιληπτικά κάποιες από τις πιο συνηθισμένες τεχνικές της εξόρυξης δεδομένων.

8.2.1 Δέντρα αποφάσεων

Πρόκειται για μη-παραμετρικές μεθόδους υψηλής υπολογιστικής έντασης που επιτρέπουν την εξακρίβωση των συνδυασμών τιμών των ανεξάρτητων μεταβλητών που οδηγούν στις βέλτιστες τιμές της απόκρισης. Τα δέντρα ταξινόμησης (για κατηγορικές) και παλινδρόμησης (για συνεχείς μεταβλητές) είναι ένα βασικό εργαλείο της *προγνωστικής μοντελοποίησης*, που επιτρέπει την πρόβλεψη της άγνωστης τιμής μιας μεταβλητής από τις γνωστές τιμές άλλων μεταβλητών. (37) Τα στιγμιότυπα ταξινομούνται καθοδηγώντας τα από τη ρίζα του δέντρου προς τα κάτω μέχρι κάποιο φύλλο, ανάλογα με το αποτέλεσμα των ελέγχων που συναντούμε σε κάθε κόμβο στην πορεία.

Τα δέντρα αποφάσεων ενδείκνυνται σε σύνολα δεδομένων με πολλά στιγμιότυπα και πολλές μεταβλητές και είναι ανθεκτικά (robust) στην ύπαρξη ακραίων τιμών. (39) Πρέπει να τονιστεί εδώ ότι η επαγωγή ενός βέλτιστου δέντρου από ένα συγκεκριμένο σύνολο δεδομένων είναι μια πολύ δύσκολη διαδικασία. Έχει αποδειχτεί ότι η εύρεση ενός ελάχιστου δέντρου αποφάσεων συνεπούς προς το σύνολο εκμάθησης (training set) είναι NP-hard πρόβλημα, ενώ η κατασκευή ενός ελάχιστου δυαδικού δέντρου (ως προς τον αναμενόμενο αριθμό ελέγχων που απαιτεί η ταξινόμηση ενός άγνωστου στιγμιοτύπου) είναι πρόβλημα NP-complete. (40)

Ανάμεσα σε μια μεγάλη ποικιλία τέτοιων εργαλείων, μπορούν να αναφερθούν τα δέντρα ταξινόμησης και παλινδρόμησης CART (Breiman και συν., 1984), το ID3 (Quinlan, 1979), ο απόγονός του C4.5 (Quinlan, 1993) και το CHAID (Chi-squared Automatic Interaction Detector / Kass, 1980).

8.2.2 Πολυμεταβλητή παλινδρόμηση

Η πολυμεταβλητή γραμμική και λογιστική παλινδρόμηση (multivariate linear & logistic regression) μελετά τη συσχέτιση ανάμεσα σε μια εξαρτημένη μεταβλητή απόκρισης (συνεχή και κατηγορική αντίστοιχα) και περισσότερες της μιας ανεξάρτητες μεταβλητές (ή μεταβλητές ελέγχου). Εδώ χρειάζεται προσοχή καθώς μια από τις παραδοχές της γραμμικής παλινδρόμησης είναι η ανεξαρτησία μεταξύ των μεταβλητών ελέγχου. Λόγω όμως του μεγάλου πλήθους των τελευταίων σε δεδομένα που προέρχονται

[17] Η έκρηξη και κάποιες υπερβολές της αγοράς γύρω από την εξόρυξη δεδομένων έκαναν παλιότερα κάποιους, με αρκετή στ' αλήθεια δόση κυνισμού, να την χαρακτηρίσουν αφοριστικά ως «στατιστική συν μάρκετινγκ»...

από εμπειρικές μετρήσεις λογισμικού, κάποιες από αυτές τις ανεξάρτητες μεταβλητές ενδέχεται να είναι οι ίδιες ισχυρά συσχετισμένες μεταξύ τους – θετικά ή αρνητικά. Επομένως, αν χρησιμοποιηθούν ταυτόχρονα σε μια εξίσωση πρόγνωσης, θα επιφέρουν ένα είδος υπερ-εκτίμησης, καθώς κατά μια έννοια θα έχουμε κάνει «διπλό» υπολογισμό, φαινόμενο που είναι γνωστό ως **πολυσυγγραμικότητα** (collinearity). Μπορεί λόγου χάρη να προκύψουν ενίοτε αναπάντεχα πρόσημα σε κάποιους συντελεστές της εξίσωσης παλινδρόμησης, όπως π.χ. αρνητικό πρόσημο για μια από τις «συγγραμμικές» μεταβλητές, ενώ για την καθεμιά από αυτές υφίσταται εμφανής θετική συσχέτιση με τη μεταβλητή απόκρισης. (14) (18)

8.2.3 Ανάλυση πολυμεταβλητών δεδομένων

Από μια πλειάδα σχετικών τεχνικών, τόσο της στατιστικής θεωρίας όσο και της εξόρυξης δεδομένων, αναφέρουμε απλώς την *ανάλυση πρωτευόντων παραγόντων* (principal component analysis). Με μέθοδο αυτή επιδιώκουμε τη μετατροπή των αρχικών μεταβλητών σε νέες, κατά το δυνατόν ανεξάρτητες (ορθογώνιες) μεταξύ τους, από τις οποίες κρατάμε αυτές μόνο που είναι υπεύθυνες για το μέγιστο κομμάτι της συνολικής διακύμανσης, ενώ αγνοούμε εκείνες που αφορούν σε πολύ μικρό ποσοστό (τυπικά <=5%) της διακύμανσης αυτής[18].

Άλλες σχετικές τεχνικές είναι η *ανάλυση κατά ομάδες* (cluster analysis) και η *ανάλυση διακρίνουσας* (discriminant analysis), όπου τα δεδομένα χωρίζονται σε ομάδες ανάλογα με το βαθμό ομοιότητας στα χαρακτηριστικά τους (ορίζοντας με κάποιο τρόπο την «απόσταση» αναμεταξύ τους), προσδοκώντας ότι «κοντινά» δεδομένα θα εμφανίζουν παρόμοια συμπεριφορά ή όπου δημιουργούμε κλάσεις με βάση τα υπάρχοντα δεδομένα και στη συνέχεια τοποθετούμε σε αυτές νέα δεδομένα ανάλογα με τις τιμές των μεταβλητών τους. (14)

8.3 Συγκριτική αξιολόγηση επιδόσεων λογισμικού

Με τη **συγκριτική αξιολόγηση επιδόσεων λογισμικού** (software benchmarking) επιχειρείται η σύγκριση των επιδόσεων του έργου που εξετάζουμε με άλλα αντίστοιχα. Η σύγκριση αφορά σε μετρήσεις και στα ποσοτικοποιημένα αποτελέσματά τους πάνω σε βασικά χαρακτηριστικά του λογισμικού (όπως π.χ. η *ποιότητα* ή η *παραγωγικότητα*). Ο στόχος μπορεί να είναι να επιδιώξουμε κάποια βελτίωση στις διαδικασίες, να επιτύχουμε καλύτερα οικονομικά ή ποιοτικά αποτελέσματα, να διαφοροποιήσουμε την επιχείρηση, το προϊόν ή τις υπηρεσίες μας στο πλαίσιο του ανταγωνισμού κοκ. (10)

Συχνά οι μεγάλες εταιρίες ανάπτυξης λογισμικού διαθέτουν ειδικό εσωτερικό (in-house) τμήμα συγκριτικής αξιολόγησης για τα προϊόντα τους. Άλλοτε πάλι, καθώς η σύγκριση αφορά σε δύο (και συχνότερα σε πολύ περισσότερες) επιχειρήσεις, το αντίστοιχο έργο αναλαμβάνει μια ανεξάρτητη εταιρία συμβούλων διοίκησης επιχειρήσεων.

[18] Στο (18) αναφέρεται ενδεικτικά ότι ένα σύνολο 38 μετρικών , προερχόμενων από 1000 περίπου δομικά στοιχεία λογισμικών, μπόρεσε να περιοριστεί σε μόλις έξι ορθογώνιες μεταξύ τους διαστάσεις, υπεύθυνες για το 90% της συνολικής μεταβλητότητας.

Η αξιολόγηση του λογισμικού μπορεί να γίνει με τους εξής τρόπους: ανάλυση των χαρακτηριστικών, πειράματα, μελέτες περίπτωσης και δειγματοληπτικές έρευνες. Ο πρώτος και απλούστερος βαθμολογεί διαφορετικά προϊόντα βάσει των χαρακτηριστικών τους ώστε να επιλέξει το εργαλείο ή τη μέθοδο που θα χρησιμοποιηθεί. Οι άλλοι τρεις τρόποι, που έχουν αναπτυχθεί προηγούμενα, διαφοροποιούνται ως προς την κλίμακα. Τα πειράματα απαιτούν μεγάλο βαθμό ελέγχου και γι' αυτό τείνουν να είναι περιορισμένης έκτασης – μπορούμε να τα θεωρήσουμε ως «έρευνα της μικρής κλίμακας». Οι μελέτες περίπτωσης εξετάζουν κατά κανόνα μια μόνο τυπική περίπτωση, παρά όλες ταυτόχρονα – πρόκειται για «έρευνα της τυπικής κλίμακας». Οι δειγματοληψίες αφορούν σε έρευνες πάνω στο τι συμβαίνει ευρέως σε μεγάλα σύνολα έργων – μιλάμε εδώ για «έρευνα της μεγάλης κλίμακας». (41)

Η συγκριτική αξιολόγηση λογισμικού κατά περίπτωση αναφέρεται σε:

- *Ανάλυση κόστους και πόρων*, όπου συλλέγονται συνολικά στοιχεία για τα ετήσια έσοδα και έξοδα της επιχείρησης από όλες τις δραστηριότητές της και συγκρίνονται με το μέσο όρο άλλων επιχειρήσεων παρόμοιας δυναμικότητας. Σε επιχειρήσεις με πολλές έδρες, η σύγκριση αφορά συχνά σε διαφορετικά τμήματα της ίδιας της επιχείρησης.
- *Ανάλυση παραγωγικότητας και ποιότητας,* αφενός για ένα έργο συνολικά, αφετέρου στις επιμέρους δραστηριότητες κατά τη διάρκεια του κύκλου ζωής του λογισμικού. Η συλλογή στοιχείων (προσπάθεια, διάρκεια, προσωπικό, πυκνότητα σφαλμάτων, συμμόρφωση με πρότυπα διαδικασιών κοκ.) γίνεται πάνω σε ένα συγκεκριμένο έργο ή μια ομάδα από ομοειδή έργα. Τα δεδομένα συλλέγονται είτε από απόσταση (remote) με την αποστολή ερωτηματολογίου που συμπληρώνεται από τους υπεύθυνους του έργου είτε επί τόπου (on-site) μέσω συνεντεύξεων με το προσωπικό. Η πρώτη περίπτωση παρέχει τη δυνατότητα εξαγωγής συμπερασμάτων μέσα από τη συλλογή και ανάλυση πολύ μεγάλου πλήθους δεδομένων[19]. Η δεύτερη έχει κατά κανόνα μεγαλύτερο κόστος αλλά υπερέχει σε ακρίβεια, καθώς μπορεί να επιλύσει επί τόπου ζητήματα εγκυρότητας που προκύπτουν στην πορεία. (3)

Συχνά σε συγκριτικές έρευνες γίνεται χρήση της **βασικής γραμμής αναφοράς** (baseline), η οποία περιγράφει με κάποιο μετρήσιμο τρόπο το σύνηθες ή τυπικό αποτέλεσμα σε μια κατηγορία. Αν η απόσταση των αποτελεσμάτων μας από τη γραμμή αναφοράς είναι σημαντική, χρειάζεται ίσως να εξετάσουμε την κατάσταση από κοντά. Κάποιες φορές μπορεί να υπάρχουν επαρκείς λόγοι που να δικαιολογούν αυτή τη διαφοροποίηση, αλλά συχνά αυτή είναι μια προειδοποίηση για απαραίτητες ρυθμίσεις και διορθωτικές ενέργειες. (41)

Η ανάλυση βασικής γραμμής αναφοράς (baseline analysis) ασχολείται με θέματα όπως: (3)

- *Χαρακτηριστικά του έργου:* εάν το έργο είναι καινούργιο ή συντήρηση παλιότερου, εάν είναι πρωτότυπο, μεμονωμένο (stand-alone), τμήμα μιας μεγαλύτερης εφαρμογής κοκ.
- *Προσωπικό και διαχείριση*: βαθμός εξειδίκευσης, ελλείψεις σε προσωπικό, διοικητική οργάνωση της επιχείρησης
- *Μεθοδολογίες*: ευρύ φάσμα εργαλείων και διαδικασιών αναφορικά με τη μέθοδο συλλογής των απαιτήσεων, τον καθορισμό των προδιαγραφών, τη σχεδίαση, την τεκμηρίωση κοκ.

[19] Αυτή είναι η περίπτωση του ISBSG Repository, που χρησιμοποιείται στη συνέχεια της παρούσας έρευνας.

- *Γλώσσες προγραμματισμού*: ανάμεσα σε 700 περίπου διαθέσιμες γλώσσες όλων των γενεών, η ανάλυση αναδεικνύει τις καλύτερες επιλογές για τη συγκεκριμένη επιχείρηση.
- *Αποσφαλμάτωση και διασφάλιση ποιότητας*: πλήρες εύρος μεθόδων και τεχνικών, όπως επισκοπήσεις, επιθεωρήσεις, όλα τα είδη δοκιμασιών ελέγχου κτλ.
- *Επαναχρησιμοποίηση λογισμικού*: τροποποίηση και εκ νέου χρήση παλαιότερου λογισμικού, όπως επίσης και επιλογή στη χρήση ή μη ολοκληρωμένων πακέτων λογισμικού από την αγορά (off-the-shelf). Η επαναχρησιμοποίηση υψηλού επιπέδου λογισμικού έχει πολύ μεγάλη απόδοση επένδυσης. Από την άλλη μεριά, η χρήση παλαιότερου λογισμικού χαμηλού επιπέδου μπορεί να επισύρει πολλά λανθάνοντα σφάλματα και να οδηγήσει τελικά σε απόδοση χειρότερη από κάθε άλλη σχετική τεχνολογία. Γενικά, πρόκειται για μια αναπτυσσόμενη τεχνολογία με καλές προοπτικές, αν και απαιτείται περαιτέρω έρευνα μέχρι να καταστεί αρκετά ασφαλής και απαλλαγμένη από κινδύνους.
- *Συντήρηση και αναβάθμιση λογισμικού*: ο τομέας αυτός αντιπροσωπεύει πλέον το 50% τη συνολικής προσπάθειας. Αναλύονται στοιχεία γύρω από την πολυπλοκότητα του υπάρχοντος κώδικα, την ανανέωση λογισμικού, τη σκοπιμότητα της απόσυρσης δομικών στοιχείων που είναι ύποπτα για λάθη (error-prone). Ας σημειωθεί ότι οι δυσκολίες που συναντάμε κατά την ασφαλή ενημέρωση «γηρασμένου» λογισμικού μπορούν να οδηγήσουν σε δυναμικό σφαλμάτων τρεις φορές υψηλότερο του κανονικού και σε μέση αποτελεσματικότητα αφαίρεσης σφαλμάτων κατά 5-10% χαμηλότερη από αυτήν που αναμένεται για καινούργιο λογισμικό. (3)

9 ΔΕΔΟΜΕΝΑ ΤΗΣ ΕΡΕΥΝΑΣ

9.1 Η βάση καταγραφής δεδομένων ISBSG

Η στατιστική ανάλυση που αποτελεί το αντικείμενο της παρούσας εργασίας έγινε πάνω στη βάση δεδομένων του ISBSG. Το ISBSG (International Software Benchmarking Standards Group) είναι ένας μη κερδοσκοπικός οργανισμός που ιδρύθηκε το 1997 από μια ομάδα εθνικών οργανισμών μέτρησης λογισμικού με σκοπό την προώθηση της χρήσης δεδομένων από τη βιομηχανία της πληροφορικής τεχνολογίας για τη βελτίωση των διαδικασιών και προϊόντων λογισμικού. Τα δεδομένα προκύπτουν ως απαντήσεις σε λεπτομερή ερωτηματολόγια που αποστέλλονται από τους οργανισμούς κάθε χώρας σε ένα μεγάλο πλήθος εταιριών ανάπτυξης λογισμικού. Στη συνέχεια τα συλλεγόμενα στοιχεία τοποθετούνται σε μια αποθήκη δεδομένων (data warehouse), τη **βάση καταγραφής ανάπτυξης και αναβάθμισης** (development and enhancement repository).

Χώρα προέλευσης	Έργα	Ποσοστό
ΗΠΑ	1548	30,7%
Ιαπωνία	841	16,7%
Αυστραλία	801	15,9%
Φιλανδία	515	10,2%
Ολλανδία	389	7,7%
Ινδία	280	5,5%
Καναδάς	252	5,0%
Δανία	131	2,6%
Βραζιλία	87	1,7%
Ηνωμένο Βασίλειο	82	1,6%
Κίνα	65	1,3%
Άλλη	58	1,1%
Σύνολο	5049	

Πίνακας 9-1: ISBSG Repository – Release 11 / Χώρες προέλευσης

Μέχρι σήμερα έχουν συλλεγεί στοιχεία για περισσότερα από 6700 έργα λογισμικού από 32 χώρες, με ρυθμό αύξησης περίπου 500 καινούργια έργα κάθε χρόνο. [πηγή: isbsg.org]. Η έκδοση που χρησιμοποιήθηκε στην παρούσα ανάλυση είναι αυτή του Ιουνίου 2009 - Release 11. (42)

Τα συνολικά 5049 υποβληθέντα έργα προήλθαν από 24 χώρες (Πίνακας 9-1) και το 70% από αυτά είχε ηλικία μικρότερη από 9 έτη.

Ο Πίνακας 9-2 καταγράφει τις διάφορες **πλατφόρμες ανάπτυξης** του έργου που εκπροσωπούνται στη βάση:

Development platform	Projects	% percent
Mainframe	1604	39.1%
Mid-range	418	10.2%
Personal computer	750	18.3%
Multi-platform	1336	32.5%

Πίνακας 9-2: ISBSG Repository – Release 11 / Πλατφόρμες ανάπτυξης

Τα έργα multi-platform εμφανίζονται ολοένα και πιο συχνά και εκτιμάται ότι είναι πιθανό σύντομα να ξεπεράσουν σε αριθμό τα αντίστοιχα mainframe.

Στο επίπεδο της **αρχιτεκτονικής** (Πίνακας 9-3), διακρίνονται δύο μεγάλες κατηγορίες, *πελάτη – εξυπηρετητή* και *πολλαπλών επιπέδων* (και οι δύο σε διάφορες παραλλαγές) όπως επίσης και η ενιαία, μεμονωμένη αρχιτεκτονική που επίσης απαντάται σε σημαντικό ποσοστό.

Architecture	Projects	% percent
Stand-alone	971	39.8%
Client Server	972	39.9%
Multi-tier	108	4.4%
Multi-tier / Client server	276	11.3%
Multi-tier with web interface	111	4.6%

Πίνακας 9-3: ISBSG Repository – Release 11 / Αρχιτεκτονικές

Αθροιστικά, 20% των έργων έχουν αρχιτεκτονική πολλαπλών επιπέδων, ενώ 51% έχουν σχεδιαστεί στη λογική πελάτη – εξυπηρετητή.

Αναφορικά με τη **πρωτεύουσα γλώσσα προγραμματισμού** της εφαρμογής (Πίνακας 9-4), έχουν καταγραφεί πάνω από 100 διαφορετικές γλώσσες, κάτι που κάνει πολύ δύσκολη τη σύγκριση έργων μεταξύ τους.

Type of programming language	Projects	% percent
2GL	19	0.4%
3GL (COBOL, Java, Visual Basic, PL/1, C++, C, SQL, C#, Scripting, PL/SQL, Javascript, Smalltalk, Ada, Pascal, άλλη)	2893	63,5%
4GL (ORACLE, .Net, NATURAL, ACCESS, ASP, PowerBuilder, ABAP, Lotus Notes, EASYTRIEVE, Delphi, άλλη)	1507	53,1%
Application generator (COOL:GEN, TELON, HPS, άλλη)	136	3.0%

Πίνακας 9-4: ISBSG Repository – Release 11 / Τύποι γλώσσας

9.2 Στατιστικά χαρακτηριστικά του δείγματος

Για την στατιστική ανάλυση που ακολουθεί χρησιμοποιήθηκε η βάση καταγραφής ISBSG – Release 11, με σκοπό να εξαχθούν στατιστικά στοιχεία σχετικά με

Α. την παραγωγικότητα (με μέτρο το ρυθμό παράδοσης PDR) και

Β. την ποιότητα (με μέτρο την πυκνότητα σφαλμάτων DD) των έργων λογισμικού.

Η βάση καταγραφής ISBSG είναι στην ουσία μια δειγματοληπτική έρευνα πάνω σε όλα τα έργα λογισμικού που έχουν ολοκληρωθεί σε παγκόσμια βάση τα τελευταία χρόνια. Το δείγμα απαρτίζουν εγγραφές πάνω σε έργα που έχουν κατατεθεί από έναν μεγάλο αριθμό εθνικών οργανισμών μετρήσεων (metrics organizations) και ανεξάρτητων επιχειρήσεων λογισμικού. Ένα βασικό χαρακτηριστικό της βάσης ISBSG είναι το γεγονός ότι η συμπλήρωση των στοιχείων πραγματοποιείται σε πολύ μεγάλη κλίμακα εδώ και αρκετά χρόνια, πάντοτε δε σε εθελοντική βάση.

Αναμφίβολα, το μεγάλο προτέρημα αυτής της βάσης είναι ο εξαιρετικά μεγάλος αριθμός εγγραφών – αν και αυτό δεν είναι από μόνο του αρκετό να δώσει αξιόπιστα στατιστικά συμπεράσματα. Στην πραγματικότητα, υπάρχουν αρκετοί λόγοι για τους οποίους η βάση *δεν* πληροί τις προϋποθέσεις ενός «σωστού» πιθανοτικού δείγματος ούτε μπορεί να θεωρηθεί a priori αντιπροσωπευτική του συνολικού πληθυσμού. Καταρχήν, υπάρχουν σημαντικά ζητήματα στην κάλυψη του αρχικού πληθυσμού. Στη βάση συμμετέχουν έργα λογισμικού προερχόμενα από έναν περιορισμένο αριθμό χωρών, αν και είναι αλήθεια ότι αυτές συγκαταλέγονται ανάμεσα στις ισχυρότερες στο χώρο της παραγωγής νέου

λογισμικού. Ακόμη, αναμένεται παραγωγικότητα υψηλότερη του μέσου όρου, καθώς οι συμμετέχοντες στην έρευνα αφενός ανήκουν κατά κανόνα στο πιο «ώριμο» κομμάτι των επιχειρήσεων ανάπτυξης λογισμικού, αφετέρου τείνουν να υποβάλλουν στοιχεία επιλέγοντας τα καλύτερα από τα προϊόντα τους. Επίσης, λόγω του εθελοντικού χαρακτήρα των απαντήσεων, δεν έχουμε καθόλου πληροφορίες για αυτούς που *δεν* έστειλαν συμπληρωμένο το ερωτηματολόγιο (ολική μη-απόκριση, βλ. πιο πάνω), ενώ παράλληλα παρατηρούνται πολλά κενά πεδία (μη-απόκριση στοιχείου), σε βαθμό που να υφίσταται σοβαρός κίνδυνος μεροληπτικότητας. Αναμένονται τέλος και κάποια σφάλματα μέτρησης, αν και έχει γίνει προσπάθεια να περιοριστούν μέσα από διασταυρώσεις στοιχείων. Έχουμε να κάνουμε λοιπόν με ένα δείγμα όπου οι δειγματοληπτικές μονάδες επιλέγονται από τον πληθυσμό με κριτήριο την προσβασιμότητα και όχι την τυχαιότητα ή την επιδίωξη της αντιπροσωπευτικότητας του πληθυσμού. Πρόκειται ουσιαστικά για ένα *δείγμα ευκολίας* (convenience ή accessibility sample), πράγμα που μπορεί να εγείρει ζήτημα *μη δειγματικού σφάλματος*. Κατά συνέπεια, δεν είναι καταρχήν εφικτό να εξαχθούν αποτελέσματα ώστε στη συνέχεια να γενικευτούν στον πληθυσμό, εξαιτίας της μη αντιπροσωπευτικότητας του δείγματος. (24)

Κάτι τέτοιο μπορεί να γίνει μόνο με επιφύλαξη και θεωρώντας ορισμένες παραδοχές σαν δεδομένες. Μπορούμε π.χ. να κάνουμε τη (βάσιμη σε κάποιο βαθμό) υπόθεση ότι η έλλειψη γεωγραφικής κάλυψης δεν επηρεάζει αισθητά το αποτέλεσμα, καθώς κατά κανόνα αφορά σε χώρες με περιορισμένη παραγωγή, ενώ στο δείγμα συγκαταλέγονται οι ισχυρότερες χώρες στον τομέα της ανάπτυξης λογισμικού. Αναφορικά με την μη-απόκριση σε διάφορα πεδία μιας εγγραφής μπορούμε να θεωρήσουμε ότι ισχύει σε κάποιο βαθμό η παραδοχή της αγνοήσιμης αποχής – για παράδειγμα, μπορούμε να θεωρήσουμε ότι η καταγραφή ή μη μιας τιμής για κάποια εγγραφή στη βάση δεν εξαρτάται από αυτή την ίδια την (μη καταγεγραμμένη) τιμή του συγκεκριμένου πεδίου. Προφανώς, η ελλείπουσα αυτή τιμή μπορεί να εξαρτάται από τα γενικά χαρακτηριστικά του έργου, όπως αυτά αποτυπώνονται σε άλλα (καταγεγραμμένα) πεδία της εγγραφής. Με παρόμοιες υποθέσεις μπορούμε λοιπόν να προχωρήσουμε στη στατιστική ανάλυση των δεδομένων, θεωρώντας απλώς την παρούσα έρευνα σαν ένα πιλοτικό, προκαταρκτικό στάδιο τα αποτελέσματα του οποίου δεν αποτελούν παρά αρχικές ενδείξεις για τη συμπεριφορά και την πρόγνωση λογισμικού. Οι ενδείξεις αυτές δίνουν κάποιες αρχικές κατευθύνσεις, κρατάμε όμως την επιφύλαξη ότι αυτές θα πρέπει στη συνέχεια να επαληθευτούν από νεώτερες εμπειρικές έρευνες, πειράματα και μελέτες περίπτωσης.

Στο πλαίσιο της στατιστικής ανάλυσης που ακολουθεί, το αρχικό δείγμα περιορίστηκε καταρχήν αποκλειστικά και μόνο στις εγγραφές που διαθέτουν

α/ βαθμολογία A ή B στην ποιότητα δεδομένων. Τα αντίστοιχα δεδομένα θεωρούνται ως απολύτως ασφαλή ή έστω με μικρές επιφυλάξεις ως προς την ακεραιότητά τους. Αποκλείονται οι κατηγορίες C και D, καθώς δεν είναι δυνατή η διασφάλιση της ακεραιότητας ή θεωρούνται σε μεγάλο βαθμό αναξιόπιστα.

β/ UFP-βαθμολογία A ή B. Στις αντίστοιχες εγγραφές η μέτρηση των FP είτε θεωρείται απολύτως ασφαλής είτε απλώς αληθοφανής, ακόμα και αν στη δεύτερη αυτή περίπτωση δεν υπάρχει αναλυτική καταγραφή στοιχείων. Αποκλείονται και εδώ οι κατηγορίες C και D, όπου δεν διατίθενται αναλυτικά

στοιχεία ώστε να είναι δυνατός ο υπολογισμός των FP ή οι σχετικές καταγραφές στερούνται αξιοπιστίας.

Στη συνέχεια, το δείγμα περιορίστηκε περαιτέρω στις εγγραφές που αφορούν σε μετρήσεις FP με τη μέθοδο IFPUG. Ο λόγος είναι ότι οι διάφορες μέθοδοι (IFPUG, COSMIC, Mark II, FiSMA, NESMA κοκ.) εμφανίζουν μεγάλες αποκλίσεις στη μέτρηση παρόμοιων έργων. Ο περιορισμός του δείγματος στις IFPUG-εγγραφές επιτρέπει τη σύγκριση ομοειδών πραγμάτων – άλλωστε, όπως έχει ήδη αναφερθεί , οι εγγραφές αυτές αποτελούν τη συντριπτική πλειοψηφία στη βάση καταγραφής και το μέγεθος του δείγματος παραμένει έτσι πολύ μεγάλο. Επιλέχτηκε επίσης ο υπολογισμός του μεγέθους να γίνεται με βάση τα UFP και όχι τα AFP.

Για την ανάλυση της **παραγωγικότητας**, έγινε χρήση του πεδίου «Κανονικοποιημένος Ρυθμός Παράδοσης» (Normalized Productivity Delivery Rate). Το πεδίο αυτό είναι ο λόγος της «Κανονικοποιημένης Προσπάθειας» (Normalized Work Effort) προς το μέγεθος του έργου σε UFP[20]. *Κανονικοποιημένη Προσπάθεια* είναι η συνολική προσπάθεια στη διάρκεια ενός πλήρους κύκλου ζωής. Σε περίπτωση που η βάση καταγράφει στοιχεία μόνο για κάποιες φάσεις, η μετρική υπολογίζεται συνεκτιμώντας κατ' αναλογία και τις υπόλοιπες. Το τι ακριβώς συμπεριλαμβάνει το σύνολο της προσπάθειας αυτής ποικίλει, επειδή τα έργα που συμμετέχουν στη βάση έχουν καταγράψει τη συνολική προσπάθεια σε διάφορα επίπεδα. Και πάλι με γνώμονα τη συγκρισιμότητα των δεδομένων, η παραγωγικότητα θα αφορά πάντοτε στη *Στάθμη 1* (Level 1) του Κανονικοποιημένου Ρυθμού Παράδοσης και της αντίστοιχης προσπάθειας, δηλαδή αυτής που οφείλεται αποκλειστικά στην ομάδα ανάπτυξης του λογισμικού (software development team). Γενικά, η καταγραφή του ISBSG προβλέπει άλλες τρεις στάθμες, συνυπολογίζοντας διαδοχικά τις προσπάθειες της ομάδας υποστήριξης, των λειτουργιών στον υπολογιστή και του τελικού χρήστη. Από αναλύσεις παλαιότερων βάσεων δεδομένων του ISBSG προκύπτει ότι η προσπάθεια στη στάθμη 2 είναι αυξημένη κατά 10-12% σε σχέση με τη στάθμη 1, η στάθμη 3 προσθέτει 1-2% παραπάνω, η δε στάθμη 4 φτάνει αθροιστικά το 20-25% επιπλέον σε σχέση με τη στάθμη 1 (16). Τέλος, ας σημειωθεί ότι προτιμήθηκε η χρήση της *Normalized Level 1 Productivity Delivery Rate* (χάριν συντομίας, στο εξής PDR) έναντι του πεδίου Pre2002 Productivity Delivery Rate, που κάνει χρήση των AFP.

Σε ό,τι αφορά στην ανάλυση της **ποιότητας**[21], χρησιμοποιήθηκε το πεδίο *Πυκνότητα Σφαλμάτων* (Defect Density, στο εξής για συντομία DD), που υπολογίζεται με τον τύπο *Total Defects Delivered * 1000 / Functional Size*. Το πεδίο αυτό είναι ένας δείκτης της ποιότητας του λογισμικού και ορίζεται ως ο αριθμός σφαλμάτων ανά 1000 UFP παραδοθέντος λογισμικού. Το πεδίο *Total Defects Delivered* είναι το άθροισμα τριών άλλων πεδίων (Minor, Major & Extreme Defects) και αφορά σε σφάλματα που ανιχνεύτηκαν μέσα *στον πρώτο μήνα χρήσης* της εφαρμογής.

[20] Ας σημειωθεί ότι, με βάση αυτόν τον ορισμό που έχει καταγραφεί στο ISBSG και τηρούμε εφεξής, καλύτερη παραγωγικότητα θεωρείται αυτή που αντιστοιχεί στον *ελάχιστο* ρυθμό παράδοσης, δηλ. στην *μικρότερη* προσπάθεια ανά FP, και όχι στην υψηλότερη τιμή PDR (όπως θα μπορούσε εκ πρώτης όψεως εύλογα να υποθέσει κανείς). Παρά τη σύγχυση που μπορεί να δημιουργηθεί από το αντίθετο «πρόσημο» σε σχέση με τους ορισμούς που δόθηκαν προηγούμενα, προτιμήθηκε να διατηρηθεί εφεξής ο ορισμός του ISBSG ώστε τα ευρήματα της έρευνας να είναι συνεπή με τα δεδομένα της βάσης καταγραφής.

[21] Όπως αναφέρθηκε και προηγούμενα, η «ποιότητα λογισμικού» αντιμετωπίζεται εφεξής με τη στενότερη έννοια του αριθμού των σφαλμάτων που παραδίδονται στο χρήστη.

Από την πληθώρα πεδίων (συνολικά πάνω από 100) τα οποία καταγράφονται στη βάση ISBSG, μεγαλύτερη συνάφεια με την ανάλυση που ακολουθεί παρουσιάζουν κυρίως τα εξής: (16) (43)

- **Είδος εφαρμογής** (type of development): new development, enhancement, re-development.
- **Αρχιτεκτονική** (architecture): Stand-alone, Multi-tier, Client Server, Mulit-tier with web interface.
- **Εύρος δραστηριοτήτων έργου** (project activity scope). Αναφέρεται στις φάσεις που έχουν καταγραφεί σαν μέρος της συνολικής προσπάθειας από τις ακόλουθες έξι που αντιστοιχούν στον πλήρη κύκλο ζωής: planning – specify – design – build – test – implement.
- **Εφαρμογή ιστού** (web development). Ένας δείκτης ανάλογα με το εάν από τις πληροφορίες για το έργο προκύπτει ότι αυτό είναι εφαρμογή ιστού
- **Πλατφόρμα ανάπτυξης** (development platform). Αναφέρεται στο κύριο περιβάλλον ανάπτυξης της εφαρμογής (όπως καθορίζεται από το λειτουργικό σύστημα που χρησιμοποιήθηκε): PC, Mid-Range, Mainframe και Multi-platform.
- **Τύπος γλώσσας προγραμματισμού** (π.χ. 3GL, 4GL, Application Generator κοκ.) και πρωτεύουσα γλώσσα (π.χ. Java, C++,PL/1,COBOL κοκ.)
- **Μέσος αριθμός μελών στην ομάδα** (Average Team Size). Ο μέσος αριθμός ατόμων που απασχολήθηκαν στο έργο.
- **Μέγιστος αριθμός μελών στην ομάδα** (Max Team Size). Ο μέγιστος αριθμός ατόμων που απασχολήθηκαν σε κάποια στιγμή (σημείο αιχμής – peak value) στο έργο.

Σημειωτέον ότι τα δύο τελευταία πεδία καταγράφουν μόνο τη *Στάθμη 1* (Level 1), δηλαδή ό,τι αφορά στην ομάδα ανάπτυξης της εφαρμογής.

10 ΕΠΕΞΕΡΓΑΣΙΑ ΤΗΣ ΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ

10.1 Το στατιστικό εργαλείο IBM Watson Analytics

Ένα μεγάλο κομμάτι της ανάλυσης των δεδομένων του ISBSG Repository πραγματοποιήθηκε με τη βοήθεια του IBM Watson Analytics, ενός εργαλείου το οποίο επιτρέπει την επεξεργασία μεγάλου όγκου δεδομένων και την εξαγωγή εποπτικών αποτελεσμάτων *περιγραφικής στατιστικής* (descriptive statistics) και *προγνωστικής ανάλυσης* (predictive analytics). Μια ενδιαφέρουσα ιδιότητά του είναι ότι, από το πλήθος των συσχετίσεων που εξερευνά με αυτοματοποιημένο τρόπο, κάνοντας χρήση πολλών από τις τεχνικές στις οποίες έχουμε αναφερθεί περιληπτικά στα προηγούμενα, επιλέγει από μόνο του με τη σειρά αυτές με την μεγαλύτερη σημασία και τις οπτικοποιεί μέσα από ένα συνδυασμό γραφημάτων και απεικονίσεων, έτσι ώστε να αναδεικνύονται οι κρυμμένες αλληλεπιδράσεις και τάσεις ανάμεσα σε μεταβλητές πρόβλεψης και την απόκριση. Όπως έχει αναφερθεί παραπάνω, ως απόκριση για τη δική μας περίπτωση επιλέχτηκε κατά πρώτον ο **ρυθμός παράδοσης** (PDR) και στη συνέχεια η **πυκνότητα σφαλμάτων** (DD).

10.2 Προετοιμασία των δεδομένων

Όπως αναφέρθηκε προηγούμενα, το αρχικό δείγμα περιορίστηκε καταρχάς στις περιπτώσεις που είχαν αφενός αξιολόγηση A ή B ως προς την ποιότητα των δεδομένων και τη διαδικασία της μέτρησης FP, αφετέρου είχαν μετρηθεί με τη μέθοδο IFPUG. Οι περιπτώσεις που πληρούσαν τις παραπάνω προϋποθέσεις ήταν 2560.

Στο πεδίο *Είδος Εφαρμογής* κρατήσαμε μόνο τις περιπτώσεις που αφορούσαν σε νέες εφαρμογές και σε αναβαθμίσεις, αποκλείοντας τις εκ νέου αναπτύξεις (re-developments). Στη συνέχεια, επιλέχτηκε ένα υποσύνολο των μεταβλητών ελέγχου και συγκεκριμένα τα πεδία *Αρχιτεκτονική, Πλατφόρμα Ανάπτυξης, Τύπος Γλώσσας Προγραμματισμού* και *Μέγεθος*, ως αυτά που κατά τεκμήριο θεωρούνται ότι έχουν την μεγαλύτερη επιρροή στον ρυθμό παράδοσης PDR. Για αυτές τις μεταβλητές και όσον αφορά στην αποχή, κρατήθηκαν μόνο οι συμπληρωμένες τιμές, εφαρμόστηκε δηλαδή κατ' ουσίαν διαγραφή περίπτωσης μόνο πάνω στο υποσύνολο αυτών των μεταβλητών (βλ. ενότητα 7.3.3). Με τις διαγραφές αυτές, ο αριθμός των περιπτώσεων που συναποτελούν το *τελικό δείγμα εργασίας* διαμορφώθηκε σε 1227.

Πρέπει να σημειωθεί ότι από το τελικό δείγμα έχουν αφαιρεθεί κάποιες από τις μεταβλητές με καταφανώς ισχυρότατη συσχέτιση με άλλες, οι οποίες παρέμειναν ως βασικές (παραδείγματος χάρη, δεν παρέμειναν τα πεδία *Summary Work Effort, Normalized PDR* ή *Adjusted Function Points*, ώς άμεσα συσχετιζόμενα με το πεδίο *Normalized Level 1 PDR* ή το *Functional Size*) – αποφεύγουμε έτσι φαινόμενα πολυ-συγγραμικότητας, όπως αυτά που περιγράφονται στην ενότητα 8.2.2.

10.3 Στρωματοποίηση των δεδομένων

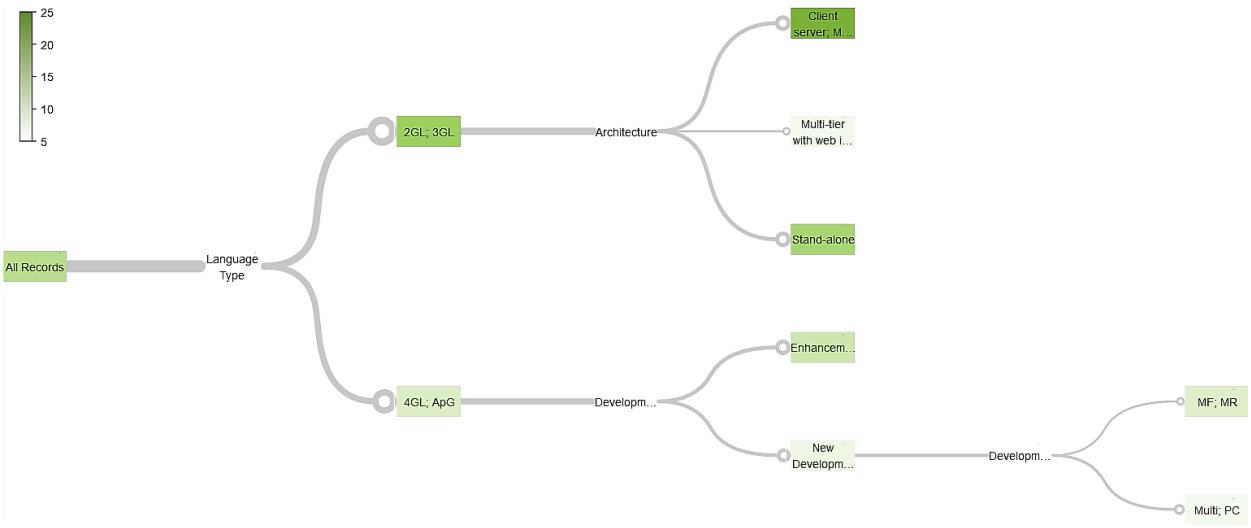
Για την πιο λεπτομερειακή ανάλυση του δείγματος,προχωρήσαμε στη στρωματοποίησή του σε *L* στρώματα με (συνεχή) βοηθητική μεταβλητή το *μέγεθος*. Ως πλήθος των στρωμάτων, επιλέχτηκε αυθαίρετα ο αριθμός *L = 5*. Σα γενική αρχή, ο καθορισμός των ορίων των στρωμάτων αυτών πρέπει να οδηγεί στη μεγαλύτερη δυνατή αξιοπιστία των στατιστικών εκτιμήσεων που προκύπτουν. Είναι λοιπόν σκόπιμο η επιλογή να γίνει κατά τρόπο που να ελαχιστοποιεί τη μεταβλητότητα της εκτίμησης που προκύπτει κατά το βέλτιστο καταμερισμό (π.χ. Neyman allocation). Η επιλογή για τα όρια αυτά ήταν ανάμεσα σε 3 μεθόδους, αυτήν του *ίσου εύρους* (equal range), αυτήν των *εκατοστημορίων* (quantile) και τέλος αυτήν της *αθροιστικής τετραγωνικής ρίζας των συχνοτήτων* (cumulative root frequency method, rootfreq / Dalenius & Hodges]. Η πρώτη χωρίζει το δείγμα σε στρώματα με ίσο εύρος τιμών. Για δείγματα με έντονη ασυμμετρία (όπως στην περίπτωσή μας), αυτή η μέθοδος οδηγεί συχνά σε κάποια στρώματα με μεγάλες πληθικότητες, όταν οι αντίστοιχες περιοχές έχουν μεγάλη συγκέντρωση τιμών. Η δεύτερη μέθοδος κατανέμει τις παρατηρήσεις σε ισοπληθικές κατηγορίες, αλλά μερικές φορές με πολύ διαφορετικές διακυμάνσεις ανάμεσα στα στρώματα. Η **μέθοδος της αθροιστικής τετραγωνικής ρίζας συχνοτήτων** είναι μια προσπάθεια συγκερασμού των δύο προηγούμενων μεθόδων. Λεπτομέρειες σχετικές με την υλοποίηση της μεθόδου αναφέρονται στα (44) και (24).
Η εφαρμογή της μεθόδου *rootfreq* στο υπό μελέτη δείγμα από το ISBSG Repository έδωσε πέντε στρώματα με τα παρακάτω διαστήματα τιμών ως προς το μέγεθος (σε UFP):

A. 3 – 150 UFP 492 περιπτώσεις
B. 151 – 350 UFP 354 περιπτώσεις
C. 351 – 850 UFP 235 περιπτώσεις
D. 851 – 1800 UFP 089 περιπτώσεις
E. >1800 UFP 057 περιπτώσεις

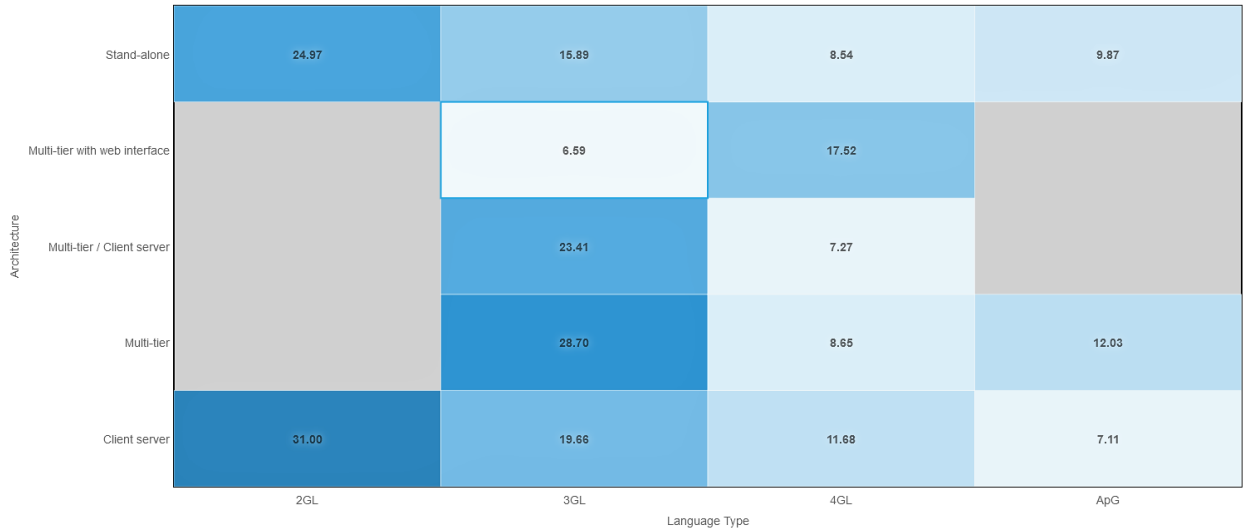
11 ΑΠΟΤΕΛΕΣΜΑΤΑ

11.1 Ευρήματα για την παραγωγικότητα έργων λογισμικού

Με τη βοήθεια του IBM Watson Analytics, διατυπώσαμε καταρχάς ένα ερώτημα πρόγνωσης (prediction) της εξαρτημένης μεταβλητής PDR πάνω στα δεδομένα του ενιαίου δείγματος και λάβαμε τα ακόλουθα γραφήματα (βλ. Εικόνα 11-1και Εικόνα 11-2)



Εικόνα 11-1 Δέντρο απόφασης για την PDR



Εικόνα 11-2: PDR (μέσες τιμές) για δύο μεταβλητές πρόγνωσης

Από τα κριτήρια διαχωρισμού στους κόμβους του δέντρου απόφασης[22] (βλ. Εικόνα 11-1) διακρίνονται οι βασικές μεταβλητές πρόγνωσης (predictors) για την PDR:

- ✓ **Τύπος Γλώσσας**
- ✓ **Αρχιτεκτονική**
- ✓ **Είδος Εφαρμογής**
- ✓ **Πλατφόρμα Ανάπτυξης**

Στις τέσσερις ως άνω αυτές μεταβλητές συνυπολογίζουμε μία ακόμα, η οποία είναι ούτως ή άλλως ένας καθοριστικός παράγοντας για τις μετρήσεις σε έργα λογισμικού, το

- ✓ **(λειτουργικό) Μέγεθος**.

Από αυτές οι δύο πρώτες είναι οι ισχυρότερες, όπως φαίνεται στην Εικόνα 11-2, η οποία προκύπτει όταν η αναζήτηση μεταβλητών πρόγνωσης περιορίζεται σε μόλις δύο. Η *ισχύς πρόγνωσης* [23] (predictive strength) υπολογίζεται σε μόλις 11%.

Σημειώνεται ότι τα παραπάνω ευρήματα βρίσκονται σε συμφωνία με σχετικές αναφορές στη βιβλιογραφία (16) ως προς τους κύριους παράγοντες που επηρεάζουν την παραγωγικότητα του λογισμικού (βλ. Κεφ. 6).[24]

[22]Η μεταβλητή PDR είναι συνεχής, οπότε το IBM Watson Anaytics σχηματίζει ένα δέντρο παλινδρόμησης με την τεχνική CHAID (CHAID regression tree)

[23] Προκειμένου για συνεχή εξαρτημένη μεταβλητή, έχουμε: predictive strength = 1 – relative error. Εδώ, σχετικό σφάλμα (σε όχι αυστηρή διατύπωση) είναι: [διασπορά των τιμών y_i σε σχέση με το μοντέλο] / [διασπορά των y_i σε σχέση με τη μέση τιμή τους]. Όταν το σχετικό σφάλμα είναι μικρότερο της μονάδας, με άλλα λόγια όταν το μοντέλο «προβλέπει» σε κάποιο βαθμό τη διασπορά της απόκρισης, η ισχύς πρόβλεψης είναι μεγαλύτερη του μηδενός και το μοντέλο βελτιώνει την αρχική πρόβλεψη της μέσης τιμής.

Στη συνέχεια εφαρμόζουμε ερωτήματα πρόγνωσης για το κάθε ένα από τα 5 στρώματα A, B, C, D και E ξεχωριστά. Τα σχετικά γραφήματα βρίσκονται στο Παράρτημα Α.

Στρώμα Α (έργα μέχρι 150 FP)

Από την Εικόνα 12-1 και την Εικόνα 12-2 παίρνουμε τις δύο ισχυρότερες μεταβλητές πρόγνωσης για το PDR των «μικρών» έργων: **Αρχιτεκτονική & Τύπος Γλώσσας** (Γ2Γ, Γ3Γ, Γ4Γ, ApG). Η ισχύς πρόγνωσης είναι 19%, δηλ. βελτιωμένη δηλαδή ως προς την αντίστοιχη για το ενιαίο δείγμα (βλ. Εικόνα 11-2). Αναγράφονται επίσης οι μέσες τιμές (τιμές πρόγνωσης) για τον κάθε συνδυασμό τιμών των δύο μεταβλητών. Αξιοσημείωτη είναι εδώ η μεγάλη αύξηση στη μέση τιμή PDR για τα έργα με *Αρχιτεκτονική Πολλαπλών Επιπέδων με Διαπροσωπεία Ιστού*, με φορά αντίστροφη από αυτήν για κάθε άλλη κατηγορία (βλ.Εικόνα 12-2): η μέση τιμή ανέρχεται από *6,16* (για 55 έργα με γλώσσα Γ3Γ – διάμεσος ίση με *5,8*) σε *19,24* (για 18 έργα με γλώσσα Γ4Γ – διάμεσος ίση με *18,9*).

Το IBM Watson Analytics υπολογίζει με τη μέθοδο ANOVA ότι $F = 7.69$ με *p-τιμή* < 0.001, επομένως η αλληλεπίδραση των δύο μεταβλητών είναι στατιστικώς σημαντική. Άρα, απορρίπτουμε τη μηδενική υπόθεση και συνάγουμε ότι η μέση τιμή της PDR εξαρτάται από το *συνδυασμό* Αρχιτεκτονικής και Τύπου Γλώσσας. Η *ισχύς συσχέτισης*[25] (association strength) εκτιμάται από το IBM Watson Analytics ως *μέτρια* (moderate).

Το δέντρο παλινδρόμησης (βλ.Εικόνα 12-3) δείχνει και πάλι τις δύο αυτές μεταβλητές και την επίδρασή τους στην πρόγνωση της PDR. Στην Εικόνα 12-4 βλέπουμε την επίδραση αποκλειστικά της Αρχιτεκτονικής στην μεταβλητή PDR. Εδώ φαίνονται καθαρά οι μέσες τιμές PDR για την κάθε μια αρχιτεκτονική ξεχωριστά (μικρές βούλες) και οι αποκλίσεις από τη μέση τιμή PDR για το Στρώμα Α ενιαίο (κάθετη γραμμή). Η επίδραση της Αρχιτεκτονικής είναι και πάλι στατιστικώς σημαντική: το $F = 9.19$, η *p-τιμή* < 0.001 και επομένως η υπόθεση ότι η Αρχιτεκτονική *δεν* επηρεάζει τη μέση τιμή της PDR πρέπει να απορριφθεί. Πάντως, η *συνδυαστική* επίδραση της Αρχτεκτονικής με τον Τύπο Γλώσσας είναι ισχυρότερη.

Για την περίπτωση που επισημάνθηκε πιο πάνω, δηλ. τις εφαρμογές *Multi-tier with web interface* και την «περίεργη» συμπεριφορά τους ανάλογα με τονΤύπο Γλώσσας που χρησιμοποιούν, προχωρήσαμε στη μελέτη των αποτελεσμάτων που λάβαμε παραπάνω ως προς τη στατιστική τους σημαντικότητα. Λόγω της προφανούς μη-κανονικότητας της κατανομής και του μικρού μεγέθους των δειγμάτων (ιδιαίτερα αυτού που αφορά σε γλώσσες Γ4Γ) χρειάστηκε να καταφύγουμε σε μη-παραμετρικό έλεγχο και, καθώς τα δύο δείγματα 3GL & 4GL είναι ανεξάρτητα μεταξύ τους, ως πλέον ενδεδειγμένη κρίθηκε

[24] Ένας ακόμη παράγοντας που αναφέρεται στο (16) είναι ο μέγιστος αριθμός μελών στην ομάδα. Τα στοιχεία της βάσης καταγραφής όμως παρουσιάζουν πολύ μεγάλα ποσοστά μη-απόκρισης και, για το λόγο αυτό, δεν εξετάζεται η συμπεριφορά του ως προς την παραγωγικότητα στην παρούσα έρευνα.

[25] Η στατιστική σημαντικότητα δεν ταυτίζεται με την πρακτική σημασία ενός αποτελέσματος. Όσο μάλιστα η ποσότητα των δεδομένων μεγαλώνει, η περίπτωση να βρεθούν ούτως ή άλλως στατιστικώς σημαντικά αποτελέσματα ανάμεσα στις εγγραφές γίνεται πιο πιθανή. Το **μέγεθος του αποτελέσματος** (effect size, η^2) δείχνει κατά πόσο ένα (στατιστικώς σημαντικό) αποτέλεσμα έχει πρακτική σημασία, δεν εξαρτάται από το μέγεθος του δείγματος και καθορίζει την **ισχύ της συσχέτισης**.

η **δοκιμασία Mann – Whitney**. Τα δύο δείγματα συνδυάστηκαν αρχικά σε ένα ενιαίο δείγμα (κατά αύξουσα τάξη ως προς PDR) και στη συνέχεια υπολογίστηκε ο *στατιστικός δείκτης ελέγχου U* μέσω της προσέγγισής του προς την κανονικοποιημένη κανονική κατανομή (45). Βρέθηκε ότι *U = -5,52*. Αυτό το αποτέλεσμα δίνει p-τιμή μικρότερη από 0,0003, που είναι αυτή που θα αντιστοιχούσε σε τιμή Z = -4,00 στην κανονική κατανομή. Είναι εμφανές ότι η p-τιμή που βρήκαμε είναι κατά πολύ μικρότερη όχι μόνο από 0,05 αλλά και από 0,01 (που είναι τα συνήθη επίπεδα σημαντικότητας), επομένως απορρίπτουμε τη μηδενική υπόθεση και συνάγουμε ότι η διαφοροποίηση στις κεντρικές τιμές PDR ανάμεσα στις δύο ομάδες (Γ3Γ & Γ4Γ) είναι στατιστικώς σημαντική. Το γεγονός αυτό της χειροτέρευσης της επίδοσης στην παραγωγικότητα για τις εφαρμογές *Multi-tier with web interface* όταν κινούμαστε από γλώσσες Γ3Γ σε γλώσσες Γ4Γ χρειάζεται προφανώς να μελετηθεί από πιο κοντά με άλλες διαδικασίες, πιθανώς με πειράματα ειδικά σχεδιασμένα για αυτές τις συνθήκες, ή καλύτερα με μια μελέτη περίπτωσης.

Στρώμα Β (από 151 έως 350 FP)

Η Εικόνα 12-5 μας δίνει και πάλι τις εξής δύο ισχυρότερες μεταβλητές πρόγνωσης για το PDR: **Αρχιτεκτονική & Τύπος Γλώσσας**. Η *ισχύς πρόγνωσης είναι 28%,* αισθητά μεγαλύτερη της αντίστοιχης για το ενιαίο δείγμα (βλ. Εικόνα 11-2). Όπως και προηγούμενα, φαίνονται οι τιμές πρόγνωσης για τον κάθε συνδυασμό τιμών των δύο μεταβλητών. Η «ποιότητα» της πρόγνωσης είναι εδώ πολύ καλύτερη από πριν, δηλαδή η διακύμανση που παραμένει *εντός* του κάθε τέτοιου συνδυασμού τιμών είναι αυτή τη φορά μικρότερη. Η μέθοδος ANOVA υπολογίζει το *F = 5.69* με *p-τιμή < 0.001* και επομένως η υπόθεση ότι η PDR είναι ανεξάρτητη από τον *συνδυασμό* Αρχιτεκτονικής και Τύπου Γλώσσας απορρίπτεται [$\eta^2 = 0.06$, μέτρια συσχέτιση]

Ακολουθούν η Εικόνα 12-6 και η Εικόνα 12-7, που παριστάνουν την σχέση της PDR με τη συνδυασμένη επίδραση των μεταβλητών

- ✓ **Τύπος Γλώσσας & Είδος Εφαρμογής** (ισχύς πρόγνωσης 15%, *F = 0.78, p-τιμή = 0.46*)
- ✓ **Τύπος Γλώσσας & Πλατφόρμα Ανάπτυξης** (ισχύς πρόγνωσης 17%, *F = 0.93, p-τιμή = 0.46*)

Όπως φαίνεται, και στις δύο περιπτώσεις το αποτέλεσμα *δεν* είναι στατιστικώς σημαντικό (στο επίπεδο α=0.05).

Η Εικόνα 12-8 παρουσιάζει την επίδραση της Αρχιτεκτονικής πάνω στην PDR. Η υπόθεση ότι η Αρχιτεκτονική *δεν* έχει επίδραση στην PDR απορρίπτεται (*F = 5.25, p-τιμή < 0.001*). Αντίστοιχα, η Εικόνα 12-9 δείχνει τη σχέση Τύπου Γλώσσας και PDR. Το αποτέλεσμα είναι και πάλι στατιστικώς σημαντικό (*F = 19.41, p-τιμή < 0.001*), άρα απορρίπτουμε την υπόθεση της ανεξαρτησίας μεταξύ των δύο[26]. Πάντως, ο συνδυασμός Αρχιτεκτονικής και Τύπου Γλώσσας ισχυροποιεί και εδώ την επίδραση στην πρόγνωση μέσης τιμής της PDR σε σχέση με τη μεμονωμένη επίδραση του κάθε παράγοντα ξεχωριστά.

Στρώμα C (από 351 έως 850 FP)

[26] Όπως και προηγούμενα, η κάθετη γραμμή στις δύο εικόνες δείχνει τη μέση τιμή για το ενιαίο στρώμα. Παράλληλα, φαίνονται οι μέσες τιμές PDR για κάθε τιμή της μεταβλητής στον κάθετο άξονα και οι αντίστοιχες αποκλίσεις.

Η Εικόνα 12-10 δείχνει την επίδραση των μεταβλητών *Αρχιτεκτονική & Τύπος Γλώσσας* στο PDR. Η *ισχύς πρόγνωσης* είναι *18%*. Με τη μέθοδο ANOVA υπολογίζονται και πάλι τα *F = 1.91* και *p-τιμή = 0.09.* Σε αυτή την περίπτωση το αποτέλεσμα *δεν* είναι στατιστικώς σημαντικό. Επομένως, τα συγκεκριμένα δεδομένα δεν παρέχουν επαρκή στοιχεία ώστε να απορρίψουμε τη μηδενική υπόθεση της ανεξαρτησίας ανάμεσα στην PDR και τον *συνδυασμό* Αρχιτεκτονικής και Τύπου Γλώσσας.

Η Εικόνα 12-11 παριστάνει το δέντρο παλινδρόμησης (CHAID) για την μεταβλητή PDR. Φαίνεται εδώ ως μοναδική μεταβλητή πρόγνωσης ο Τύπος Γλώσσας.

Στη συνέχεια παρουσιάζονται η Εικόνα 12-12 και η Εικόνα 12-13 που αναπαριστούν την σχέση της PDR με τις μεταβλητές

- ✓ **Τύπος Γλώσσας** (*F = 11.58, p-τιμή < 0.001, η^2 = 0.09*)
- ✓ **Αρχιτεκτονική** (*F = 2.54, p-τιμή = 0.04, η^2 = 0.04*)

Από τα παραπάνω συνάγεται ότι στην περίπτωση του Τύπου Γλώσσας έχουμε στατιστικώς σημαντικά αποτελέσματα, άρα συνάγουμε ότι η PDR είναι εξαρτημένη από την τιμή της αντίστοιχης μεταβλητής ελέγχου. Για την Αρχιτεκτονική έχουμε επίσης σημαντικό αποτέλεσμα στο επίπεδο α=0.05, σχετικά κοντά όμως στην κρίσιμη τιμή, ενώ στο επίπεδο αυξημένης σημαντικότητας α=0.01 η μηδενική υπόθεση δεν απορρίπτεται. Βλέπουμε λοιπόν ότι υπάρχει καταρχήν ισχυρή συσχέτιση της PDR με τον **Τύπο Γλώσσας**, κάτι που φάνηκε επίσης στην Εικόνα 12-11. Η Αρχιτεκτονική φαίνεται να έχει εδώ μια δευτερεύουσα, όχι τόσο ευκρινή συσχέτιση με την παραγωγικότητα. Τέλος, ο συνδυασμός των δύο δε φαίνεται να ισχυροποιεί την εξάρτηση αυτή.

<u>Στρώμα D (από 851 έως 1800 FP)</u>

Η Εικόνα 12-14 και η Εικόνα 12-15 παριστάνουν την συνδυασμένη επίδραση πάνω στο PDR των μεταβλητών:

- ✓ *Πλατφόρμα Ανάπτυξης & Τύπος Γλώσσας* (*ισχύς πρόγνωσης: 24%, F = 1.50, p-τιμή = 0.21*)
- ✓ *Πλατφόρμα Ανάπτυξης & Αρχιτεκτονική* (*ισχύς πρόγνωσης: 27%, F = 2.03, p-τιμή = 0.14*)

Τα αποτελέσματα δεν είναι στατιστικώς σημαντικά, οπότε με τα συγκεκριμένα δεδομένα παραμένουμε στην αρχική (μηδενική) υπόθεση ότι οι μέσες τιμές PDR δεν διαφοροποιούνται για κάθε ένα από τους παραπάνω δύο συνδυασμούς μεταβλητών ελέγχου. Οι σχετικά υψηλές τιμές της ισχύος πρόγνωσης δεν έχουν αντίκτυπο στο αποτέλεσμα καθώς προφανώς το πλήθος των μονάδων δεν είναι αρκετά μεγάλο και το στατιστικό αποτέλεσμα είναι τελικά άδηλο.

Τα επόμενα δύο γραφήματα (Εικόνα 12-16 και Εικόνα 12-17) παρουσιάζουν τη σχέση της PDR με την **Πλατφόρμα Ανάπτυξης**. Φαίνεται, σε αναλογία με τα προηγούμενα, η απόκλιση της μέσης τιμής της PDR ανά πλατφόρμα ανάπτυξης σε σχέση με την ενιαία μέση τιμή για όλο το Στρώμα D. Το αποτέλεσμα είναι αυτή τη φορά στατιστικώς σημαντικό, η υπόθεση ότι η μέση τιμή της PDR *δεν* εξαρτάται από την πλατφόρμα ανάπτυξης απορρίπτεται (*F = 4.61, p-τιμή < 0.01*) και επομένως συνάγουμε ότι ο ισχυρότερος παράγοντας πρόγνωσης για τα έργα μεγέθους από 851 μέχρι 1800 FP είναι η Πλατφόρμα Ανάπτυξης.

Στρώμα Ε (έργα μεγαλύτερα των 1800 FP)

Στο στρώμα που περιέχει τα έργα με μέγεθος μεγαλύτερο των 1800 UFP (από τα οποία έχουμε αποσύρει έναν αριθμό ακραίων τιμών της PDR, βλ. Κεφ. 10.2), το IBM Watson Analytics δεν δίνει καμία πρόγνωση για την μεταβλητή PDR. Αυτό οφείλεται προφανώς στη μεγάλη διακύμανση των περιπτώσεων με βάση το μέγεθος (υπάρχει π.χ. μια περίπτωση με μέγεθος 14656 UFP χωρίς άλλες στην ίδια περιοχή) και επομένως στη μεγάλη στρεβλότητα της σχετικής κατανομής.

Αξίζει να σημειωθεί ότι, εάν διατηρήσουμε τις ακραίες τιμές, η κατάσταση αλλάζει ριζικά, καθώς τότε παραμένουν δύο περιπτώσεις με μέση τιμή PDR 42.20 (*βλ. βέλος*, Πλατφόρμα = PC & Αρχιτεκτονική = Multi-tier). Η επίδραση των μεταβλητών *Αρχιτεκτονική & Πλατφόρμα Ανάπτυξης* στο PDR παίρνει τη μορφή που φαίνεται στην Εικόνα 12-18, με *ισχύ πρόγνωσης 43%*. Το αποτέλεσμα είναι στατιστικώς σημαντικό (*F = 5.99* και *p-τιμή < 0.01).* Η αλληλεπίδραση αξιολογείται μάλιστα ως ισχυρή (strong) με μέγεθος αποτελέσματος $\eta^2 = 0.22$.

11.2 Ευρήματα για την ποιότητα έργων λογισμικού

11.2.1 Η έννοια της ποιότητας στη βάση καταγραφής ISBSG

Μετά την ανάλυση των δεδομένων σε ό,τι αφορά στην παραγωγικότητα των έργων λογισμικού, όπως αυτή χαρακτηρίζεται από το ρυθμό παράδοσης PDR, η προσοχή μας στρέφεται στην ανάλυση της ποιότητας. Όπως έχει διεξοδικά περιγραφεί στο Κεφ.2, η ποιότητα λογισμικού είναι γενικά μια σύνθετη έννοια – για την ανάλυση όμως που ακολουθεί, ο όρος χρησιμοποιείται με μια πολύ στενότερη σημασία και συγκεκριμένα αυτή που αφορά στα σφάλματα. Κάτω από αυτό το πρίσμα, μέτρο της ποιότητας ενός έργου λογισμικού θεωρείται ο αριθμός των σφαλμάτων που ανιχνεύονται μέσα στον πρώτο μήνα από την στιγμή της παράδοσης της εφαρμογής στον τελικό χρήστη. Στη βάση δεδομένων του ISBSG βρίσκουμε καταγραφές για τα σφάλματα αυτά, τα οποία κατανέμονται σε τρεις κατηγορίες: Δευτερεύοντα (Minor), Πρωτεύοντα (Major) και Ακραία (Extreme). Παράλληλα, για κάθε έργο για το οποίο διατίθενται στοιχεία στη βάση, μπορούμε να αναζητήσουμε την **πυκνότητα σφαλμάτων** (defect density – DD), την οποία ορίζουμε ως το πλήθος των σφαλμάτων ανά 1000 FP. Έχουμε επομένως:

DD = [Συνολικό Πλήθος Σφαλμάτων] * 1000 / [Λειτουργικό Μέγεθος σε FP].

11.2.2 Το φαινόμενο της μη-απόκρισης

Καταρχάς, ξεκινώντας από τη βάση του ISBSG, σχηματίσαμε ένα δείγμα με προδιαγραφές ανάλογες αυτών της παραγράφου 10.2. Ακόμα και με μια γρήγορη πρώτη ματιά μπορούμε να διακρίνουμε ότι το πρόβλημα της μη-απόκρισης είναι εδώ ιδιαίτερα έντονο. Πράγματι, σε σύνολο 2814 εγγραφών που αφορούν σε έργα για τα οποία υπάρχει καταγραφή κάποιων δεδομένων (όχι πάντοτε σε όλα τα πεδία), μόλις 563 εγγραφές έχουν συμπληρωμένο το πεδίο της πυκνότητας σφαλμάτων. Ανάμεσα σε διάφορες αιτίες, το φαινόμενο μπορεί να οφείλεται είτε σε αμέλεια ή άρνηση των ερωτώμενων να παράσχουν την απαιτούμενη πληροφορία είτε ακόμη σε αδιαφορία ή αντικειμενική αδυναμία τους να

υπολογίσουν το ζητούμενο μέγεθος – π.χ. εάν δε διαθέτουν κάποιο μηχανισμό ανάδρασης (feedback) από τους χρήστες ώστε να συλλέξουν τις πληροφορίες σχετικά με τα λάθη που έχουν προκύψει κατά τη λειτουργία του προϊόντος.

Όπως αναφέραμε ήδη στο Κεφ. 7.3.3 παραπάνω, σε περιπτώσεις με σημαντικά επίπεδα μη-απόκρισης είναι πολύ πιθανό οι αποκριθέντες να μην αποτελούν τυπικό υπο-δείγμα του συνολικού και επομένως είναι ενδεχόμενο οι τιμές κεντρικής τάσης που προκύπτουν από αυτούς να διαφέρουν σημαντικά από τις αντίστοιχες για το υπο-δείγμα που αντιστοιχεί στη μη-απόκριση. Εξάλλου, έχουμε να αντιμετωπίσουμε μια σημαντική αύξηση της διακύμανσης στις εκτιμήσεις μας, που οφείλεται σε μια «ντε φάκτο» ελάττωση του μεγέθους του δείγματος, αφού οι υπολογισμοί μπορούν να γίνουν αποκλειστικά με βάση τα δεδομένα που προέρχονται από τους αποκριθέντες. (30)

Τα παραπάνω στοιχεία από τη βάση καταγραφής δίνουν ένα ποσοστό μη-απόκρισης (nonresponse rate) ίσο με 79,99%, το οποίο ξεπερνάει κατά πολύ ένα μέγιστο επιτρεπτό επίπεδο της τάξης του 30-40%. Με αυτές τις συνθήκες, κάθε προσπάθεια εξαγωγής εκτιμητριών τιμών ή προγνώσεων είναι παρακινδυνευμένη. Άλλωστε, το IBM Watson Analytics με είσοδο το ενιαίο αυτό δείγμα δεν δίνει καν αποτέλεσμα, λόγω ακριβώς των χαρακτηρτιστικών αυτών του δείγματος.Στη συνέχεια θα επιχειρήσουμε να μελετήσουμε από πιο κοντά τα χαρακτηριστικά της μη-απόκρισης που συναντάμε στο δείγμα.

Γενικά, ένα τυπικό μη-ντετερμινιστικό μοντέλο για τη μη-απόκριση αποδίδει σε κάθε στοιχείο k του δείγματος μια **πιθανότητα απόκρισης** (response probability) ϑ_k, βάσει της οποίας το στοιχείο αποκρίνεται ή όχι, αφού προηγουμένως αυτό έχει επιλεγεί μέσα από τον πληθυσμό και πάρει τη θέση του στο δείγμα. Εάν επομένως *s* είναι το δείγμα μας, *r* το υπο-δείγμα των αποκριθέντων και $o = s - r$ αυτό των μη-αποκριθέντων, έχουμε: $\vartheta_k = Prob\ [k \epsilon r \mid k \epsilon s]$. Καταρχήν, η πιθανότητα απόκρισης για κάποιο στοιχείο παραμένει άγνωστη ακόμα και μετά την επιλογή του δείγματος, αφού από την ύπαρξη της απάντησης ενός στοιχείου δεν είμαστε σε θέση να συναγάγουμε πόσο πιθανό ήταν αυτό να συμβεί[27]. Το ποσό μεροληψίας λόγω της μη-απόκρισης είναι τότε κατά προσέγγιση: (46) (47)

$$B\ (y_{average,resp}) = E[y_{average,resp}] - Y_{average,\pi\lambda\eta\vartheta} = C_{\vartheta,Y} / \vartheta_{average} = R_{\vartheta,Y} S_{\vartheta} S_Y / \vartheta_{average} \qquad 11\text{-}1$$

όπου

- $y_{average,resp}$: η μέση τιμή της μεταβλητής (η DD στην περίπτωσή μας) που λαμβάνουμε από τους αποκριθέντες και μόνο.
- $Y_{average,\pi\lambda\eta\theta}$ και Y_i: η πραγματική μέση τιμή της DD στον πληθυσμό και η τιμή της για την i-μονάδα του πληθυσμού αντίστοιχα
- $C_{\theta,Y}$ και $R_{\theta,Y}$: η συνδιακύμανση και ο συντελεστής συσχέτισης ανάμεσα στις τιμές της DD και τις αντίστοιχες πιθανότητες απόκρισης (στον πληθυσμό)

[27] Αυτό αντιδιαστέλλει την παράμετρο ϑ_k με την πιθανότητα επιλογής π_k του στοιχείου στο δείγμα, η οποία είναι ορισμένη ήδη κατά το στάδιο του σχεδιασμού της δειγματοληψίας. Επομένως, το βάρος σχεδιασμού d_k (design weight) κάθε στοιχείου του δείγματος είναι εξαρχής γνωστό – προκειμένου για την εκτιμήτρια Horvitz-Thompson, πρόκειται για τον αντίστροφο της πιθανότητας επιλογής του στο δείγμα – ενώ το επιπλέον βάρος w_k που αποκτά μια αποκριθείσα μονάδα στο δείγμα δε μπορεί να υπολογιστεί ακριβώς, παρά μόνο να εκτιμηθεί, έτσι ώστε να γίνει δυνατή η προσαρμογή.

- S_θ και S_Y: οι τυπικές αποκλίσεις των κατανομών θ και Y (στον πληθυσμό) και
- $\theta_{average}$ και θ_i: η μέση τιμή της πιθανότητας απόκρισης (στον πληθυσμό) και η τιμή της για την i-μονάδα του πληθυσμού αντίστοιχα

Από τη σχέση 11-1– αν και αυτή καθαυτή δεν έχει πρακτική εφαρμογή, καθώς οι τιμές στον πληθυσμό και οι πιθανότητες απόκρισης είναι άγνωστες – διακρίνουμε τα βασικά γνωρίσματα της μεροληψίας που προκύπτει από την αποχή. Βλέπουμε λοιπόν ότι

- η εκτιμήτρια θα είναι (σχεδόν) αμερόληπτη εάν η μεταβλητή που μελετάμε και η συμπεριφορά της απόκρισης είναι μεταξύ τους ασυσχέτιστες ή (ακόμα καλύτερα) ανεξάρτητες. Αντίστροφα, όσο πιο ισχυρή είναι αυτή η συσχέτιση, τόσο μεγαλύτερο θα είναι το ποσό της μεροληψίας.
- δεν προκύπτει μεροληψία εάν όλες οι πιθανότητες απόκρισης είναι ίσες, οπότε $S_\theta = 0$.
- ομοίως, δεν έχουμε μεροληψία εάν $S_Y = 0$, δηλ. εάν η μεταβλητή DD έχει την ίδια τιμή για ολόκληρο τον πληθυσμό.
- το ποσό μεροληψίας είναι αντιστρόφως ανάλογο της μέσης πιθανότητας απόκρισης στις μονάδες του πληθυσμού, η οποία μπορεί να εκτιμηθεί αμερόληπτα από το ποσοστό απόκρισης στο δείγμα. Όσο περισσότεροι ερωτώμενοι συνεργάζονται και δίνουν απαντήσεις, τόσο η μεροληψία περιορίζεται.
- η μεροληψία *δεν* τείνει στο μηδέν όσο αυξάνουμε το μέγεθος του δείγματος.

Εάν ο ενιαίος πληθυσμός δεν διαθέτει ήδη τα παραπάνω χαρακτηριστικά (κάτι που προφανώς είναι σχεδόν σίγουρο ότι θα συμβεί στην πράξη), είναι επόμενο να επιδιώξουμε μια διαμέριση τέτοια ώστε οι παραπάνω συνθήκες να ισχύουν στο μεγαλύτερο δυνατό βαθμό για την κάθε κλάση ξεχωριστά.

Το κλειδί για μια επιτυχημένη προσαρμογή λόγω μη-απόκρισης (nonresponse adjustment) βρίσκεται στη χρήση «ισχυρής» **βοηθητικής πληροφορίας** (auxiliary information), με στόχο τη μείωση της μεροληψίας αλλά παράλληλα και της διακύμανσης, οι οποίες οφείλονται στη μη-απόκριση. Για να επιτύχουμε την καλύτερη δυνατή τέτοια μείωση αναζητούμε ένα «βέλτιστο» διάνυσμα βοηθητικών μεταβλητών $\boldsymbol{x}_k$, το οποίο επιδιώκουμε να κινείται κατά το δυνατόν στις ακόλουθες κατευθύνσεις: (30) (46)

- να επεξηγεί την **επίδραση απόκρισης** (το «βάρος») φ_k (που ορίζεται ως ο αντίστροφος της πιθανότητας ϑ_k) για την κάθε μονάδα στον πληθυσμό (11-2)
- να επεξηγεί τη διαφοροποίηση στη μεταβλητή που μελετάμε (11-3)

Παράλληλα, το διάνυσμα βοηθητικών μεταβλητών είναι σκόπιμο να

- να οριοθετεί τις σπουδαιότερες περιοχές (domains) του πληθυσμού (11-4)

Η βοηθητική αυτή πληροφορία συχνά συνίσταται σε ένα συνδυασμό από μεταβλητές (στήλες) της βάσης, ο οποίος οδηγεί σε μια διαμέριση του συνόλου των διαθέσιμων εγγραφών σε ένα σύνολο

αλληλοαποκλειόμενων κατηγοριών[28]. Οι κλάσεις που προκύπτουν απαρτίζουν τα *κελιά προσαρμογής* (adjustment cells).[29]

Γενικά, δεν υφίσταται κάποιος κλειστός τύπος υπολογισμού για το ποσό μεροληψίας. Προσεγγιστικά μπορούμε να χρησιμοποιήσουμε τον πρώτο όρο του σχετικού αναπτύγματος Taylor, για τον οποίο παίρνουμε την ακόλουθη σχέση:

$$nearbias\ (Y_{estim.}) = (\Sigma_{\pi\lambda\eta\vartheta}\, \boldsymbol{x_k})' \,(\boldsymbol{B_{\pi\lambda\eta\vartheta,\,\vartheta}} - \boldsymbol{B_{\pi\lambda\eta\vartheta}}) \qquad (11\text{-}5)$$

Το διάνυσμα $\boldsymbol{x_k}$ αποτελεί το σύνολο των μεταβλητών πάνω στις οποίες προσπαθούμε να πετύχουμε τη γραμμική παλινδρόμηση της μεταβλητής y. Λόγω όμως της μη-απόκρισης δεν έχουμε στη διάθεσή μας ολόκληρο τον πληθυσμό ώστε να εκτιμήσουμε τον πραγματικό συντελεστή παλινδρόμησης $\boldsymbol{B_{\pi\lambda\eta\vartheta}}$, παρά μόνο τους αποκριθέντες. Έτσι προκύπτει ο «αλλοιωμένος» συντελεστής παλινδρόμησης $\boldsymbol{B_{\pi\lambda\eta\vartheta,\vartheta}}$ του οποίου η τιμή έχει επηρεαστεί από τις πιθανότητες ϑ_k, το δε *nearbias* είναι ανάλογο της διαφοράς των δύο συντελεστών. Η επιλογή του διανύσματος βοηθητικών μεταβλητών έχει κομβική σημασία στο μέγεθος του ποσού μεροληψίας που θα παραμείνει τελικά. Ενδεικτικά, μπορεί να δειχτεί (48) ότι το *nearbias* μηδενίζεται στην (ιδεατή) περίπτωση όπου έχουμε επιλέξει το διάνυσμα κατά τρόπον ώστε να επιτυγχάνεται απόλυτη γραμμική σχέση μεταξύ φ_k και $\boldsymbol{x_k}$, δηλαδή όταν μπορούμε να βρούμε ένα σταθερό διάνυσμα $\boldsymbol{\lambda}$ τέτοιο ώστε: $\varphi_k = \boldsymbol{\lambda' x_k}$ για κάθε μονάδα k του πληθυσμού.

Μπορεί τέλος να δειχτεί ότι, αν και οι πιθανότητες ϑ_k – άρα και τα $cv_{\vartheta U;p}$ – είναι άγνωστες ποσότητες, μια καλή κατηγοριοποίηση μπορεί να επιτευχθεί μέσα από τη μεγιστοποίηση της ποσότητας:

$$\Sigma_P\, [N_p\, (\vartheta_{average,p} - \vartheta_{\text{average}})^2],$$

όπου τα $\vartheta_{average,p}$ και $\vartheta_{\text{average}}$ είναι οι μέσες τιμές των πιθανοτήτων απόκρισης υπολογιζόμενες πάνω σε όλα τα στοιχεία *του πληθυσμού*, αντίστοιχα για την κατηγορία *p* και για τον πληθυσμό συνολικά. Ας σημειωθεί ότι, στη θέση των άγνωστων αυτών μέσων τιμών, μπορούμε να χρησιμοποιήσουμε τις εκτιμήτριες τιμές που προκύπτουν από τα ποσοστά μη απόκρισης *του* δείγματος, αφενός για την κάθε κατηγορία και αφετέρου για το ενιαίο δείγμα. (30)

11.2.3 Ανάλυση του δείγματος ως προς την ποιότητα

Με βάση την παραπάνω θεωρητική προσέγγιση αναλύσαμε το δείγμα της βάσης ISBSG.

Ομαδοποιώντας τα δεδομένα κατά το *Είδος Εφαρμογής*, ο Πίνακας 11-1 μας δίνει την *ανάλυση απόκρισης*, δηλαδή τα ποσοστά μη-απόκρισης για κάθε υποκατηγορία:

[28] Η προσθήκη στο διάνυσμα μιας αριθμητικής βοηθητικής μεταβλητής (π.χ. το μέγεθος) οδηγεί σε προσαρμογή με εκτιμήτριες λόγου (ratio) ή/και παλινδρόμησης (regression).

[29] Προκειμένου για την πρώτη από τις παραπάνω κατευθύνσεις, τα κελιά προσαρμογής αναφέρονται και ως *ομάδες ομογενούς απόκρισης* (Response Homogeneity Groups – RHG).

Είδος Εφαρμογής	*Ερωτηθέντες*	*Αποκριθέντες*	*Ποσοστό Μη-Απόκρισης*
Αναβαθμίσεις	1769	342	**0.81**
Νέες εφαρμογές	1045	221	**0.79**

Πίνακας 11-1 Ανάλυση απόκρισης κατά Είδος Εφαρμογής

Ο Πίνακας 11-2 εξάλλου παρουσιάζει τα αντίστοιχα περιγραφικά στατιστικά δεδομένα για την πυκνότητα σφαλμάτων.

	Αναβαθμίσεις	*Νέες Εφαρμογές*
Mean	20.15284	**30.79817**
Standard Error	2.010399	**4.809324**
Median	7.7	**6.7**
Mode	0	0
Standard Deviation	36.79634	**71.1715**
Sample Variance	1353.971	5065.383
Kurtosis	23.02178	12.72007
Skewness	4.125271	3.561265
Range	310.6	411.3
Minimum	0	0
Maximum	310.6	**411.3**
Sum	6751.2	6744.8
Count	335	219[30]

Πίνακας 11-2 Ανάλυση εξαρτ. μεταβλητής κατά Είδος Εφαρμογής

Σημείωση: Στον παραπάνω πίνακα με τα περιγραφικά στοιχεία (όπως και στους αντίστοιχους που ακολουθούν) ο υπολογισμός κεντρικών τάσεων έχει γίνει αποκλειστικά πάνω στις μονάδες που έχουν δώσει απάντηση στο πεδίο DD. Υιοθετούμε την παραδοχή ότι η πιθανότητα απόκρισης είναι ενιαία για όλες τις μονάδες του πληθυσμού που εντάσσονται στην κάθε υπο-κατηγορία και επομένως η αναγωγή στον πληθυσμό από τα στοιχεία του δείγματος μπορεί να γίνει με ευθεία επέκταση (direct expansion), χωρίς δηλαδή κάποια αναπροσαρμογή βαρών λόγω μη-απόκρισης.

Από τους πίνακες αυτούς παρατηρούμε ότι η ανάλυση απόκρισης δεν δίνει σημαντική διαφορά ως προς τις δύο υπο-κατηγορίες. Μπορούμε να δεχτούμε ότι αυτό αποτελεί αυξημένη ένδειξη απουσίας μεροληψίας λόγω μη-απόκρισης (έχουμε δηλ. πιθανώς μια περίπτωση MCAR και αγνοήσιμης μη-απόκρισης ως προς τα δύο είδη εφαρμογής) και επομένως τα αποτελέσματα της ανάλυσης της μεταβλητής DD μπορούν να γίνουν βάσιμα αποδεκτά για τον πληθυσμό μας παρά τα υψηλά ποσοστά

[30] *Κατά τον υπολογισμό των περιγραφικών στατιστικών στοιχείων στην παραπάνω, όπως και στις αντίστοιχες περιπτώσεις που ακολουθούν, έχουν αφαιρεθεί από το δείγμα οι ακραίες τιμές, συγκεκριμένα αυτές που βρέθηκαν να απέχουν από την μέση τιμή περισσότερο από 5 φορές επί την τυπική απόκλιση[30]. Κάποιες φορές, η διαδικασία επαναλήφθηκε περισσότερες από μία φορές καθώς προέκυπταν νέες ακραίες τιμές (masking effect, βλ. 7.4.1 ανωτέρω).*

μη-απόκρισης. (49) Με αυτή την οπτική, διακρίνουμε καταρχήν αξιοσημείωτα υψηλότερη μέση τιμή για τις *Νέες Εφαρμογές* από αυτήν για τις *Αναβαθμίσεις*. Αυτό έχει μια λογική εξήγηση καθότι μια αναβάθμιση, ιδιαίτερα σε σχετικά μικρή κλίμακα, δεν επισύρει κατά κανόνα τόσα σφάλματα όσα μια καινούργια εφαρμογή που αποδίδεται για πρώτη φορά στους τελικούς χρήστες. Πάντως η διαφοροποίηση αυτή στις μέσες τιμές πρέπει να γίνει δεκτή με επιφύλαξη, λόγω του μεγάλου τυπικού σφάλματος που συνοδεύει την μέση τιμή για τις νέες εφαρμογές. Αυτό οδηγεί σε ευρύ διάστημα εμπιστοσύνης, το οποίο υπερκαλύπτει το αντίστοιχο για την κατηγορία των αναβαθμίσεων, βρισκόμαστε δηλαδή μέσα στα όρια του στατιστικού σφάλματος. Άλλωστε, οι κατανομές για τις δύο υπο-κατηγορίες διαφέρουν κατά πολύ ως προς τα χαρακτηριστικά τους. Παρά την υψηλότερη μέση τιμή, η διάμεσος για τις νέες εφαρμογές είναι *χαμηλότερη* από αυτήν για τις αναβαθμίσεις – κάτι που υποδηλώνει μεγάλη συσσώρευση νέων εφαρμογών με σχετικά χαμηλές τιμές πυκνότητας σφαλμάτων. Από την άλλη, οι κατά πολύ υψηλότερες τυπική απόκλιση και μέγιστη τιμή υποδεικνύουν μεγάλη διακύμανση και την ύπαρξη κάποιων εφαρμογών με ιδιαίτερα υψηλό αριθμό σφαλμάτων ανά FP.

Στη συνέχεια παρατίθενται οι αντίστοιχοι πίνακες για την διαμέριση κατά Πλατφόρμα Ανάπτυξης (Πίνακας 11-3 και Πίνακας 11-4).

Μπορεί να παρατηρήσει κανείς αμέσως (Πίνακας 11-4) την αφύσικα πολύ χαμηλή μέση τιμή πυκνότητας σφαλμάτων για τα έργα που έχουν αναπτυχθεί σε πλατφόρμα προσωπικού υπολογιστή (κατηγορία *PC*), παράλληλα με το ελαφρά υψηλότερο ποσοστό απόκρισης. Αυτό πιθανώς οφείλεται στην αυξημένη αδυναμία του μηχανικού λογισμικού αυτής της κατηγορίας να συλλέξει πληροφορίες για τα σφάλματα από ικανό αριθμό χρηστών, οπότε πιθανώς κάποια σφάλματα δεν καταγράφονται.

Πλατφόρμα Ανάπτυξης	***Ερωτηθέντες***	***Αποκριθέντες***	***Ποσοστό Μη-Απόκρισης***
Mainframe	986	219	**0.78**
Medium Range	255	87	**0.75**
Multi-platform	548	149	**0.73**
PC	307	92	**0.70**
Χωρίς στοιχεία	718	66	**0.91**

Πίνακας 11-3 Ανάλυση απόκρισης κατά Πλατφόρμα Ανάπτυξης

Αξιοσημείωτη είναι επίσης εδώ η μεγάλη κύρτωση για τα έργα της πρώτης κατηγορίας (mainframe), όπως και η πολύ μικρή τιμή της διαμέσου για την κατηγορία αυτή σε σχέση με τις επόμενες δύο (medium range και multi-platform).

	Mainframe	*Medium Range*	*Multi-platform*	*PC*
Mean	16.72594	16.05278	18.57959	**6.514607**
Standard Error	2.07092	3.29091	1.655299	1.140608
Median	2.8	9.3	13.1	0
Mode	0	0	0	0
Standard Deviation	30.15306	19.74546	20.06943	10.76048
Sample Variance	909.2069	389.8831	402.782	115.7879
Kurtosis	10.9589	5.136417	0.949924	2.208948
Skewness	3.035439	2.149593	1.251912	1.76622
Range	181.8	86.8	82.6	39.9
Minimum	0	0	0	0
Maximum	181.8	86.8	82.6	39.9
Sum	3545.9	577.9	2731.2	579.8
Count	212	36	147	89

Πίνακας 11-4 Ανάλυση εξαρτ. μεταβλητής κατά Πλατφόρμα Ανάπτυξης

Ακολουθούν τα στοιχεία για την ομαδοποίηση ως προς την Αρχιτεκτονική (Πίνακας 11-5, Πίνακας 11-6).

Παρατηρούμε εδώ τα ακόλουθα:

1. Η κατηγορία *Multi-Tier / Client Server* παρουσιάζει από τη μια ιδιαίτερα μεγάλο ποσοστό αποχής, αισθητά υψηλότερο από το ποσοστό για το ενιαίο δείγμα –από την άλλη αφύσικα υψηλή μέση τιμή και διάμεσο, αν και σε πολύ μικρό πλήθος δείγματος. Εδώ φαίνεται να αντιμετωπίζουμε ιδιαίτερα μεγάλο ποσό μεροληψίας λόγω μη-απόκρισης – έχουμε δηλαδή μια περίπτωση MNAR (βλ. Κεφ. 7.3.3), για την οποία οποιοδήποτε αποτέλεσμα ενέχει μεγάλο βαθμό αυθαιρεσίας.
2. Αν και με αισθητά χαμηλότερο ποσοστό μή-απόκρισης από αυτό του ενιαίου δείγματος, οι κατηγορίες *Multi-Tier with Web Interface* και *Multi-Tier* αντιστοιχούν επίσης σε χαμηλό πλήθος μονάδων. Έτσι, οι μέσες τιμές και διάμεσοι που καταγράφονται για τη μεταβλητή DD, ιδιαίτερα υψηλές για την πρώτη και χαμηλες για την δεύτερη υπο-κατηγορία, πρέπει επίσης να αντιμετωπίζονται με επιφύλαξη.
3. Οι κατηγορίες *Client-Server* και *Stand Alone* είναι οι πολυπληθέστερες και επιπροσθέτως παρουσιάζουν σχετικά καλή συμπεριφορά ως προς την απόκριση (αν και πάντοτε σε υψηλά επίπεδα). Παρατηρούμε εδώ ότι οι μέσες τιμές πυκνότητας σφαλμάτων κινούνται στην περιοχή [16.0, 20.0] που έχουμε συναντήσει και σε προηγούμενες υπο-κατηγορίες πιο πάνω.

Αρχιτεκτονική	*Ερωτηθέντες*	*Αποκριθέντες*	*Πσοστό Μη-Απόκρισης*
Multi-Tier / Client-Server	244	22	**0.91**
Client-Server	471	127	**0.73**
Multi-Tier with web interface	102	29	**0.71**
Stand-Alone	643	202	**0.68**
Multi-Tier	57	20	**0.65**

Πίνακας 11-5 Ανάλυση απόκρισης κατά Αρχιτεκτονική

	Multi-Tier / Client Server	*Client-Server*	*Multi-Tier with Web interface*	*Stand Alone*	*Multi-Tier*
Mean	**31.65909**	18.27063	**27.07241**	17.91856	**6.64**
Standard Error	5.01835	2.7284	6.461378	2.183291	2.647366
Median	**29.45**	10.5	16.5	5.95	0
Mode	0	0	0	0	0
Standard Deviation	23.53815	30.62621	34.79559	30.40973	11.83938
Sample Variance	554.0444	937.9648	1210.733	924.7518	140.1709
Kurtosis	-0.66066	39.30349	8.640648	11.11365	3.52522
Skewness	0.431067	5.408265	2.530242	3.05684	2.088134
Range	80	271.5	166.7	181.8	40.3
Minimum	0	0	0	0	0
Maximum	80	271.5	166.7	181.8	40.3
Sum	696.5	2302.1	785.1	3476.2	132.8
Count	**22**	126	**29**	194	**20**

Πίνακας 11-6 Ανάλυση εξαρτ. μεταβλητής κατά Αρχιτεκτονική

Στη συνέχεια (Πίνακας 11-7 και Πίνακας 11-8) βλέπουμε τα στοιχεία για την κατηγοριοποίηση κατά Τύπο Γλώσσας Προγραμματισμού.

Εδώ παρατηρούμε τα ακόλουθα:

1. Προκειμένου για έργα που έχουν υλοποιηθεί με γλώσσες δεύτερης και τρίτης γενιάς (κατηγορίες *Γ2Γ* και *Γ3Γ)*, παρατηρούμε ποσοστό μη-απόκρισης πολύ κοντά σε αυτό για το ενιαίο δείγμα. Όπως και για την διαμέριση σε νέες εφαρμογές και αναβαθμίσεις προηγούμενα, μπορούμε να θεωρήσουμε ότι δεν προκύπτει μεροληψία λόγω αποχής και ότι τα έργα 4GL συνοδεύονται από αισθητά υψηλότερη πυκνότητα σφαλμάτων κατά μέσο όρο, όπως και από μεγαλύτερη διακύμανση στις τιμές τους (αν και περίπου τα μισά από αυτά της κατηγορίας 2&3GL). Έτσι, για την κατηγορία Γ2&3Γ η πυκνότητα σφαλμάτων DD είναι μέσα στην περιοχή [16.0, 20.0] που έχουμε αναφέρει πιο πάνω, για τα έργα Γ4Γ όμως η μέση τιμή DD ανέρχεται σε

22.85. Πρέπει πάντως να σημειωθεί ότι οι διάμεσοι για τις δύο αυτές κατηγορίες είναι παραπλήσιες.

2. Για την κατηγορία *Γ5Γ & ApG* υπάρχει και πάλι μεγάλο πρόβλημα στα διαθέσιμα στοιχεία. Είναι χαρακτηριστικό ότι, παρά τα συγκριτικά πολύ καλά ποσοστά απόκρισης, από τα 36 έργα (μετά την απομάκρυνση κάποιων ακραίων τιμών) που έχουν καταγεγραμμένο αριθμό σφαλμάτων, κανένα δεν αφορά σε εφαρμογή που έχε υλοποιηθεί με γλώσσα πέμπτης γενιάς, αλλά όλα ανήκουν στην κατηγορία ApG. Η μέση τιμή είναι εξώφθαλμα υψηλή, αλλά τα επίπεδα τυπικού σφάλματος δεν επιτρέπουν ασφαλή συμπεράσματα.

Γλώσσα Προγραμματισμού	***Ερωτηθέντες***	***Αποκριθέντες***	***Ποσοστό Μη-Απόκρισης***
2GL, 3GL	1543	305	**0.80**
4GL	793	175	**0.78**
5GL, ApG	66	46	**0.45**

Πίνακας 11-7 Ανάλυση απόκρισης κατά Τύπο Γλώσσας Προγρ/μού

	2GL & 3GL	*4GL*	*5GL & ApG*
Mean	**17.22007**	**22.84412**	70.86944
Standard Error	1.743118	3.167963	25.27636
Median	**7.2**	**7.4**	5.5
Mode	0	0	0
Standard Deviation	30.14133	41.30519	151.6582
Sample Variance	908.4995	1706.118	23000.2
Kurtosis	23.9139	13.69931	5.760051
Skewness	4.100868	3.400742	2.519501
Range	271.5	272.7	593.6
Minimum	0	0	0
Maximum	271.5	272.7	593.6
Sum	5148.8	3883.5	2551.3
Count	299	170	**36**

Πίνακας 11-8 Ανάλυση εξαρτ. μεταβλητής κατά Τύπο Γλώσσας Προγρ/μού

Στη συνέχεια, αναλύουμε τη συμπεριφορά ως προς την απόκριση και ως προς την εξαρτημένη μεταβλητή DD για τα ακόλουθα δύο αριθμητικά πεδία, στα οποία έχουμε εφαρμόσει διαχωρισμό κατά κατηγορίες τιμών (binning):

1. Μέγιστος Αριθμός μελών στην ομάδα (Max Team Size)[31]
2. Λειτουργικό μέγεθος έργου (μόνο για εφαρμογές για τις οποίες το πεδίο δεν είναι κενό)

[31] Προτιμήθηκε το πεδίο «Μέγιστος Αριθμός Μελών στην Ομάδα» από το «Μέσο Αριθμό Μελών», που θα μπορούσε με μια πρώτη ματιά να είναι το ίδιο ή και περισσότερο ενδεικτικό για την ανάλυσή μας. Αν και υπάρχει καταγεγραμμένη πληροφορία και για τα δύο αυτά πεδία στη βάση ISBSG, το πρώτο παρουσιάζει καλύτερα ποσοστά απόκρισης

Τα αποτελέσματα για την πρώτη περίπτωση φαίνονται παρακάτω (Πίνακας 11-9, Πίνακας 11-10):

Maximum Team Size (MTS)	***Ερωτηθέντες***	***Αποκριθέντες***	***Ποσοστό Μη-Απόκρισης***
MTS <5	409	112	**0.73**
5 <= MTS <10	399	158	**0.60**
MTS>= 10	285	124	**0.56**
Σύνολο εγγραφών με *μη κενό* το πεδίο MTS	1093	394	**0.64**

Πίνακας 11-9 Ανάλυση απόκρισης ως προς Μέγιστο Πλήθος Ομάδας

Βλέπουμε με μια πρώτη ματιά το αισθητά χαμηλότερο ποσοστό μη-απόκρισης για τα έργα που έχουν συμπληρωμένο το πεδίο που αφορά στο μέγιστο αριθμό μελών στην ομάδα σε σχέση με το ενιαίο δείγμα. Παρατηρούμε επίσης ένα σημαντικά μεγαλύτερο ποσοστό αποχής στα έργα με μέγιστο πλήθος στην ομάδα έως 5 άτομα σε σχέση με αυτά στα οποία έχει απασχοληθεί μεγαλύτερος αριθμός.

	MTS<5	*5<=MTS<10*	*MTS>=10 (1)*	*MTS>=10 (2)*
Mean	20.51468	17.4271	18.55164	14.19
Standard Error	4.009689	2.26888	3.613042	1.931897
Median	6	8	6.35	5.8
Mode	0	0	0	0
Standard Deviation	41.86238	28.24733	39.90735	21.16287
Sample Variance	1752.459	797.9119	1592.597	447.867
Kurtosis	10.40909	18.16025	29.66333	3.531107
Skewness	3.221186	3.694564	4.950391	1.972515
Range	222.6	202.7	289	94.3
Minimum	0	0	0	0
Maximum	222.6	202.7	289	94.3
Sum	2236.1	2701.2	2263.3	1702.8
Count	109	155	122	120

Πίνακας 11-10 Ανάλυση εξαρτ. μεταβλητής ως προς Μέγιστο Πλήθος Ομάδας

Τα αποτελέσματα που καταγράφει ο Πίνακας 11-10 δείχνουν διαφοροποιήσεις ανά υπο-κατηγορία, αν και προς άλλη κατεύθυνση ανάλογα εάν λαμβάνουμε υπόψη τις μέσες τιμές (με τα αντίστοιχα τυπικά σφάλματα) ή τις διαμέσους. Για τη δεύτερη περίπτωση, λόγου χάρη, φαίνεται μια άνοδος στην πυκνότητα σφαλμάτων για ομάδες των 5-10 ατόμων και χαμηλότερες τιμές για μικρότερες ή μεγαλύτερες ομάδες. Ας σημειωθεί ότι η τελευταία στήλη του πίνακα αφορά στην ίδια κατηγορία με την αμέσως αριστερά της, απ' όπου όμως έχουν αφαιρεθεί δύο επιπλέον ακραίες τιμές – βλέπουμε ότι υπάρχει δραστική διαφοροποίηση στα στατιστικά στοιχεία κεντρικής τάσης (όχι όμως και στη διάμεσο, που άλλωστε παρουσιάζει πολύ μεγαλύτερη ευστάθεια στην επίδραση ακραίων τιμών). Όμως, δεν έχουμε τρόπο να γνωρίζουμε εάν οι τιμές αυτές πρέπει να αφαιρεθούν ή εάν αντιστοιχούν σε

περιπτώσεις που μπορεί και στο μέλλον να συναντήσουμε στην πράξη. Αυτό δίνει ένα μέτρο της δυσκολίας στην εξαγωγή αξιόπιστων αποτελεσμάτων με τις συνθήκες του συγκεκριμένου δείγματος.

Ακολουθεί η ανάλυση ως προς το λειτουργικό μέγεθος[32] (Πίνακας 11-11 και Πίνακας 11-12):

Λειτουργικό μέγεθος σε FP	***Ερωτηθέντες***	***Αποκριθέντες***	***Ποσοστό Μη-Απόκρισης***
3 <= FP <= 150	1229	196	**0.84**
151 <= FP <= 350	749	177	**0.76**
351 <= FP <= 850	510	118	**0.77**
851 <= FP <= 1800	215	54	**0.75**
MTS>= 1800	111	18	**0.84**

Πίνακας 11-11 Ανάλυση απόκρισης ως προς Λειτ. Μέγεθος

	FP<=150	*151<FP<=350*	*351<FP<=850*	*851<FP<=1800*	*FP>1800*
Mean	**22.05211**	**20.87574**	**15.01739**	**10.19245**	28.17222
Standard Error	2.589959	3.03772	2.500325	2.114208	15.46461
Median	**9**	**7.9**	**5.1**	**4.7**	3
Mode	0	0	0	0	0
Standard Deviation	35.70012	39.49036	26.813	15.39167	65.61078
Sample Variance	1274.498	1559.489	718.9369	236.9034	4304.774
Kurtosis	8.720937	18.84748	15.7996	11.24238	12.33581
Skewness	2.773843	3.949048	3.530857	2.914483	3.385556
Range	202.7	285.1	164.8	86.8	271.5
Minimum	0	0	0	0	0
Maximum	202.7	285.1	164.8	86.8	271.5
Sum	4189.9	3528	1727	540.2	507.1
Count	190	169	115	53	**18**

Πίνακας 11-12 Ανάλυση εξαρτ. μεταβλητής ως προς Λειτ. Μέγεθος

Ο Πίνακας 11-11 δείχνει μια ελαφρά διαφοροποίηση ανάμεσα στα «μικρά» (έως 150 FP) και τα «μεγάλα» έργα (πλέον των 1800 FP) από τη μια μεριά απέναντι στα «μεσαίου μεγέθους» (οι ενδιάμεσες υπο-κατηγορίες) σε ό,τι αφορά στην συμπεριφορά τους ως προς την καταγραφή ή μη πληροφοριών για τα σφάλματα. Διακρίνεται συγκεκριμένα μια καλύτερη συμπεριφορά ως προς την αποχή για τα «μεσαία» έργα. Η χειρότερη επίδοση για τις «μικρές» εφαρμογές εξηγείται μάλλον σε κάποιο βαθμό από την αδυναμία της ομάδας ανάπτυξης να συλλέξει επαρκή στοιχεία.

Ο Πίνακας 11-12 αναδεικνύει μια σημαντική πτωτική τάση στις μέσες τιμές και τις διαμέσους για τις τέσσερεις πρώτες κατηγορίες. Αν μάλιστα για τις 3 μεσαίες κατηγορίες ληφθεί παράλληλα υπόψη το

[32] Η κατηγοριοποίηση ως προ το μέγεθος του έργου ακολουθεί αυτήν της παραγράφου 10.3

παραπλήσιο ποσοστό μη-απόκρισης (75-77%), μπορούμε (με την παραδοχή, όπως παραπάνω, ότι δεν υφίσταται σοβαρό ποσό μεροληψίας λόγω της αποχής) να θεωρήσουμε ότι αυτή η τάση αντανακλά όχι μόνο στους αποκριθέντες αλλά συνολικά στον πληθυσμό.

Ας σημειωθεί ότι τα δεδομένα της τελευταίας κατηγορίας (που αφορούν σε έργα μεγαλύτερα των 1800 FP) δεν μπορεί να θεωρηθεί ότι αντιπροσωπεύουν αξιόπιστα την κατάσταση, αφενός λόγω του αυξημένου ποσοστού αποχής και αφετέρου λόγω του μικρού αριθμού των αποκριθέντων. Είναι αναμενόμενο μονάδες με μεγάλα μεγέθη να παρουσιάζουν και μεγαλύτερη διασπορά, είναι δε χαρακτηριστικό το πολύ μεγάλο τυπικό σφάλμα (ενδεικτικό της διακύμανσης), σχεδόν ίσο με το μισό της μέσης τιμής. Πάντως, η πολύ πιο ευσταθής διάμεσος φαίνεται να ακολουθεί τη γενική τάση μείωσης.

11.2.4 Παρατηρήσεις για την ανάλυση στο πεδίο της ποιότητας

Από την ανάλυση που προηγήθηκε βλέπουμε ότι το **διάνυσμα των βοηθητικών μεταβλητών** πρέπει καταρχήν να αποτελείται από τα ακόλουθα πεδία:

- Είδος Εφαρμογής
- Μέγιστος Αριθμός Μελών στην Ομάδα
- Λειτουργικό Μέγεθος

Η πρώτη από αυτές τις μεταβλητές είναι ονομαστική, οι άλλες δύο αριθμητικές (στις οποίες έχουμε κάνει ομαδοποίηση binning)

Περαιτέρω ανάλυση θα απαιτούσε ομαδοποιήσεις ανά συνδυασμούς των βασικών κατηγοριών που μελετήθηκαν στην Ενότητα 11.2.3 παραπάνω. Είναι δυνατό να υπάρξουν διαφοροποιημένα αποτελέσματα για καθεμιά από τις υπο-κατηγορίες που θα προκύψουν, καθώς η αρχική μας ανάλυση δεν μπορεί να διακρίνει πιθανές αλληλοσυσχετίσεις και πολυ-συγγραμικότητες. Πάντως, μια τέτοια λεπτομερέστερη κατηγοριοποίηση δεν θα έχει μεγάλη αξία στην περίπτωσή μας λόγω των εξαιρετικά μεγάλων ποσοστών μη-απόκρισης. Οποιαδήποτε τέτοια ανάλυση ενέχει σοβαρό κίνδυνο εξαγωγής εσφαλμένων συμπερασμάτων και επομένως είναι σκοπιμότερο, με τα στοιχεία που διαθέτουμε, να παραμείνουμε σε αυτό το πρώτο στάδιο ποιοτικής ανάλυσης και σε κάποια ενδεικτικά «ποιοτικά» συμπεράσματα γενικής φύσης, που παρατίθενται στην Ενότητα 11.3 που ακολουθεί.

Ο καταλογισμός τιμής στην περίπτωση αυτή είναι επίσης παρακινδυνευμένος, λόγω της πολύ μεγάλης αποχής (θα πρέπει να καταλογίσουμε 3-4 τιμές σε κενές θέσεις για κάθε μία υπάρχουσα στη βάση...). Πάντως, εάν κανείς έχει στη διάθεσή του ένα καλύτερο ως προς την απόκριση δείγμα – έστω και για κάποιες, αν όχι για όλες τις υπο-κατηγορίες – θα μπορούσε να προχωρήσει σε καταλογισμούς τιμών ώστε να εξάγει ενδεικτικά συμπεράσματα για κάποιες περιπτώσεις.

11.3 Ανακεφαλαίωση – Πορίσματα

Ακολουθεί μια κωδικοποίηση των αποτελεσμάτων της ανάλυσης που προηγήθηκε. Περισσότερες λεπτομέρειες μπορεί να αναζητήσει κανείς στις οικείες ενότητες και στο Παράρτημα με τα σχετικά γραφήματα.

Α. Παραγωγικότητα έργων λογισμικού

Τα πεδία που φαίνεται γενικά να έχουν την ισχυρότερη επίδραση είναι ο *Τύπος Γλώσσας Προγραμματισμού* και η *Αρχιτεκτονική*.Η χαμηλή ισχύ πρόγνωσης 11% είναι αναμενόμενη ως ένα βαθμό λόγω της μη-κανονικότητας που παρουσιάζει το ενιαίο δείγμα, πριν την κατάτμησή του σε κλάσεις κατά το λειτουργικό μέγεθος.

Ως προς τον Τύπο Γλώσσας, παρουσιάζεται καταρχήν μια ευκρινής διαμέριση σε δύο κλάσεις: {2GL, 3GL} και (4GL, ApG), που αποτελεί την πρώτη και πιο ισχυρή κατηγοριοποίηση που ανιχνεύεται στο δείγμα (βλ. Εικόνα 11-1). Παράλληλα, παρατηρείται μια σαφής βελτίωση στις μέσες τιμές PDR καθώς μετακινούμαστε διαδοχικά στη γραμμή 2GL -> 3GL -> 4GL -> ApG. Σε αντίθετη φορά φαίνονται να κινούνται οι εφαρμογές Multi-tier with web interface από τις γλώσσες 3GL στις 4GL. Το φαινόμενο εμφανίζεται εντονότερο στα έργα μικρού μεγέθους (βλ. παρακάτω).

Σε ό,τι αφορά στο πεδίο της Αρχιτεκτονικής, τα αποτελέσματα είναι αρκετά σύνθετα και όχι τόσο σαφή, πάντως διακρίνεται μια καλύτερη επίδοση για τις πολλαπλών επιπέδων αρχιτεκτονικές Multi-tier, προκειμένου για εφαρμογές σε γλώσσες 3GL & 4GL, ενώ για έργα 4GL & ApG καλύτερες αποτελέσματα φαίνεται να εμφανίζουν οι λιγότερο σύνθετες αρχιτεκτονικές Stand-Alone και Client-Server.

Στα «μικρά» έργα (μέχρι 150 FP) η **Αρχιτεκτονική** φαίνεται να είναι ο ισχυρότερος παράγοντας πρόγνωσης της PDR, ενώ ακολουθεί ο **Τύπος Γλώσσας Προγραμματισμού**. Ο συνδυασμός των δύο παρουσιάζει αυξημένη ισχύ πρόγνωσης 19%, η δε αλληλεπίδρασή τους κρίνεται στατιστικώς σημαντική. Αξιοσημείωτη είναι η μη αναμενόμενη αλλά στατιστικά σημαντική *αύξηση* της μέσης τιμής της PDR για τα έργα Multi-tier with Web Interface που έχουν υλοποιηθεί σε γλώσσα 4GL σε συνάρτηση με αυτά που χρησιμοποιούν γλώσσες 3GL.

Στην επόμενη κατηγορία μεγέθους (έως 350 FP) ισχυροποιείται ακόμη περισσότερο (28%) ο **συνδυασμός Αρχιτεκτονικής και Τύπου Γλώσσας** ως του σημαντικότερου παράγοντα πρόγνωσης της PDR. Δεν διακρίνονται άλλες στατιστικώς σημαντικές επιδράσεις από κάποιον άλλο συνδυασμό μεταβλητών.

Για τα μετρίου μεγέθους έργα (μεταξύ 350 και 850 FP) είναι κυρίως ο **Τύπος Γλώσσας** και δευτερευόντως η Αρχιτεκτονική που αποτελούν τους στατιστικώς σημαντικούς παράγοντες πρόγνωσης.

Για τα έργα μεγέθους 850-1800 FP, βασικός παράγοντας πρόγνωσης καθίσταται η **Πλατφόρμα Ανάπτυξης**, παρατηρείται δε μια σαφής βελτίωση στις μέσες τιμές της PDR όσο κινούμαστε στη γραμμή Mainframe -> Mid-Range -> Multi-Platform -> PC.

Τέλος, για τα έργα με μέγεθος πάνω από 1800 FP, δεν προκύπτουν αποτελέσματα λόγω των μεγάλων αποκλίσεων και των ακραίων τιμών. Είναι χαρακτηριστικό ότι η αποβολή ή όχι λίγων ακραίων τιμών (μάλλον περιστασιακών ή τυχαίων) δίνει ριζικά διαφορετικά ευρήματα για το συνδυασμό Αρχιτεκτονικής και Πλατφόρμας Ανάπτυξης, γεγονός που δεν επιτρέπει κατά τη γνώμη μας την εξαγωγή ασφαλούς συμπεράσματος.

Β. Ποιότητα έργων λογισμικού

Αναφορικά με την πυκνότητα σφαλμάτων, ως δείκτη της ποιότητας του λογισμικού, είχαμε να αντιμετωπίσουμε τα απαγορευτικά υψηλά ποσοστά μη-απόκρισης. Με αυτές τις συνθήκες είναι πολύ παρακινδυνευμένη οιαδήποτε πρόγνωση, καθώς η μεροληψία μπορεί να κινείται σε επίπεδα τέτοια που να διαστρέφει σημαντικά τις τιμές των στατιστικών εκτιμητριών. Το βάρος της ανάλυσής μας έπεσε στην ανάδειξη κάποιων πεδίων της βάσης καταγραφής που μπορούν να λειτουργήσουν ως βοηθητικές μεταβλητές και επεξηγούν από τη μια μεριά τη συμπεριφορά της αποχής, από την άλλη την κατανομή της υπό μελέτη μεταβλητής. Από την ανάλυση απόκρισης προέκυψε ένα αισθητά μεγαλύτερο ποσοστό αποχής από το μέσο όρο για τα μικρά έργα (έως 150 FP) και ένα αντίστοιχα χαμηλότερο ποσοστό για μεγαλύτερες εφαρμογές. Επίσης, είχαμε μια αξιοσημείωτη πτωτική τάση στην αποχή όσο μεγαλύτερος ήταν ο Μέγιστος Αριθμός Μελών στην Ομάδα ανάπτυξης της εφαρμογής. Σε ό,τι αφορά στην μεταβλητή που μελετάμε, δηλ. την πυκνότητα σφαλμάτων, προέκυψε μια έντονη διαφοροποίηση στις μέσες τιμές πυκνότητας σφαλμάτων (όχι όμως ανάλογη και στις διαμέσους) ανάλογα με το Είδος Εφαρμογής (Νέα Εφαρμογή ή Αναβάθμιση). Τέλος, προέκυψε σημαντική διαφοροποίηση στην πυκνότητα σφαλμάτων ανάλογα με το μέγεθος του έργου, και συγκεκριμένα μια εμφανής πτωτική τάση όσο το μέγεθος ήταν μεγαλύτερο.

Με τα δεδομένα αυτά, οι μεταβλητές που φαίνεται να λειτουργούν ως βοηθητικές σε πρώτη φάση είναι το **Είδος Εφαρμογής**, ο **Μέγιστος Αριθμός Μελών στην Ομάδα** και το **Λειτουργικό Μέγεθος** του έργου. Ενδεχομένως να υπάρχει ένα διάνυσμα με περισσότερες μεταβλητές ή συνδυασμούς τους, οι οποίες να επεξηγούν καλύτερα τη συμπεριφορά του πληθυσμού. Απαραίτητη όμως προϋπόθεση για μια επιτυχημένη αναζήτηση ενός τέτοιου διανύσματος είναι η ριζική μείωση του ποσοστού της αποχής.

11.4 Προτάσεις για περαιτέρω έρευνα

Όπως έχει επισημανθεί προηγούμενα, ένα μεγάλο προτέρημα της βάσης καταγραφής ISBSG είναι το μεγάλο μέγεθός της συγκριτικά με άλλες παρόμοιες έρευνες. Από την άλλη μεριά, ένα υπαρκτό πρόβλημα είναι η μη ελεγχόμενη κάλυψη του πληθυσμού αλλά και (κυρίως) η μέθοδος καταγραφής των δεδομένων. Καθώς τα δεδομένα συλλέγονται σε εθελοντική βάση χωρίς να επανέρχεται ο μηχανισμός συλλογής σε περιπτώσεις μη-απάντησης ή ελλιπούς συμπλήρωσης, εγείρονται σοβαρά ζητήματα ως προς την αντιπροσωπευτικότητα του δείγματος. Μια κατεύθυνση λοιπόν στην οποία αξίζει ίσως να επικεντρωθούμε από δω και πέρα είναι αυτή της καλύτερης δειγματοληψίας, εν ανάγκη με μικρότερα δείγματα. Τα «πιλοτικά» λίγο πολύ αποτελέσματα της παρούσας ανάλυσης μπορούν να χρησιμεύσουν καταρχήν ως υποθέσεις σε στατιστικούς ελέγχους ώστε να επιβεβαιωθούν ή να απορριφθούν στο επιθυμητό επίπεδο σημαντικότητας από τη μελλοντική έρευνα. Ιδιαίτερο ενδιαφέρον θα έχει μια τέτοια εργασία πάνω σε έργα των οποίων το λειτουργικό μέγεθος έχει

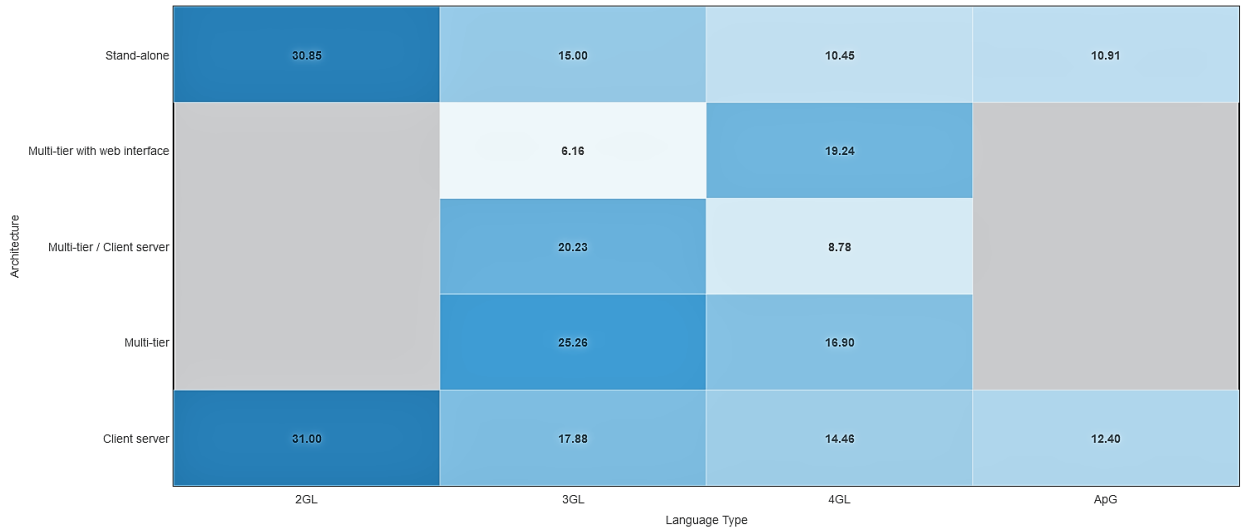
μετρηθεί με τις εναλλακτικές μεθόδους COSMIC, NESMA κτλ., έργα για τα οποία δεν υπήρξε πρόνοια στην παρούσα φάση. Σημαντικές θα είναι επίσης οι μελέτες περίπτωσης με λεπτομερή ερωτηματολόγια πάω σε επιμέρους ευρήματα που αναδεικνύει η παρούσα έρευνα, όπως για παράδειγμα η χειρότερη επίδοση των web-εφαρμογών μικρού μεγέθους που έχουν αναπτυχθεί με γλώσσες 4GL (βλ. Ενότητα 11.1)

Ένα πολύ σοβαρό πρόβλημα που αντιμετωπίσαμε στην παραπάνω ανάλυση είναι, όπως έχει σημειωθεί επανειλημμένα, το απαγορευτικά υψηλό ποσοστό μη-απόκρισης. Αυτό παρουσιάστηκε ως έντονο πρόβλημα ήδη κατά την επεξεργασία των στοιχείων γύρω από την παραγωγικότητα, στη φάση όμως της ανάλυσης της ποιότητας του λογισμικού έφτασε σε σημείο να μην επιτρέψει την εξαγωγή συγκεκριμένων αποτελεσμάτων, έστω και γενικής φύσης – πέρα από κάποιες πληροφορίες γύρω από τα ποιοτικά χαρακτηριστικά της συμπεριφοράς του πληθυσμού ως προς την απόκριση και ως προς την ίδια την τιμή της πυκνότητας σφαλμάτων. Διεύρυνση του δείγματος δεν θα βελτιώσει την κατάσταση, καθώς η μεροληψία λόγω μη-απόκρισης δεν τείνει στο μηδέν για μεγαλύτερα δείγματα – μπορεί μάλιστα και να αυξηθεί σε κάποιες περιπτώσεις. Είναι επομένως αυτονόητο ότι στο μέλλον πρέπει να επιδιώξουμε σημαντικά καλύτερες επιδόσεις στην απόκριση. Ο μηχανισμός συλλογής στοιχείων του ISBSG, «εκ γενετής» εθελοντικός, δεν προσφέρεται ίσως για την επίτευξη ενός τέτοιου στόχου. Θα απαιτηθεί ένα διαφορετικό δείγμα στο οποίο θα μπορεί ο ερευνητής να επιμείνει επανερχόμενος σε περιπτώσεις άρνησης ή ελλιπούς καταγραφής πεδίων. Υπάρχουν τεχνικές της σύγχρονης θεωρίας δειγματοληψίας που μπορεί να φανούν χρήσιμες στο σημείο αυτό (44) (47). Μια τέτοια τεχνική είναι η επανάκληση (call-back), όπου επιδιώκεται σε δεύτερη φάση να απαντήσει ένα σχετικά μικρό υπο-δείγμα της ομάδας των αρχικά μη-αποκρινόμενων, ώστε να μπορεί κανείς από τις απαντήσεις αυτών να εκτιμήσει και τις υπόλοιπες. Μια άλλη μέθοδος είναι αυτή της βασικής ερώτησης (Basic Question Approach). Εδώ το ερωτηματολόγιο συμπεριλαμβάνει κάποια (ελάχιστα) βασικά ερωτήματα στα οποία επιδιώκεται να ληφθούν απαντήσεις από το σύνολο του πληθυσμού, απαντήσεις που θα επιτρέψουν να προκύψει μια εικόνα ως προς τη συμπεριφορά και τις διαφορές μεταξύ αποκρινόμενων και μη. Ακόμα και αν, παρά τις όποιες προσπάθειες, παραμείνει ένα αρκετά υψηλό ποσοστό μη-απόκρισης, ο δραστικός περιορισμός στα επίπεδα της αποχής, σε συνδυασμό με τεχνικές όπως οι παραπάνω, θα μας επιτρέψει να εκμεταλλευτούμε τις βοηθητικές μεταβλητές που προέκυψαν στην παρούσα έρευνα. Έτσι θα πετύχουμε μια επιτυχή διαμέριση σε κελιά προσαρμογής και θα μπορέσουμε να εφαρμόσουμε στο καθένα από αυτά κατάλληλες αναπροσαρμογές βαρών ή τεχνικές καταλογισμού τιμής όπως π.χ. hot-deck με στοχαστικά υπόλοιπα ή αυτή του πλησιέστερου συγγενή (με βάση την απόσταση ανάμεσα στις τιμές των βοηθητικών μεταβλητών).

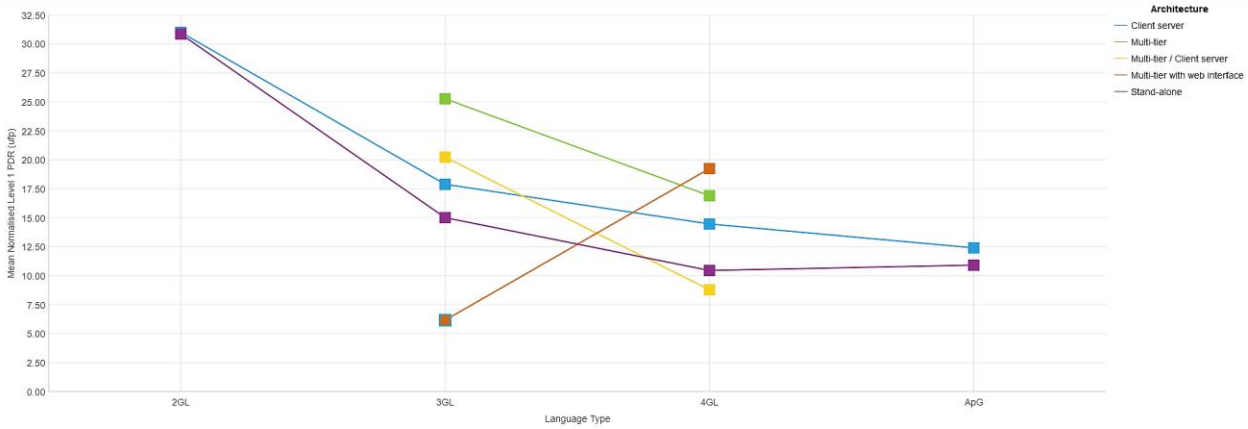
Η ανάλυση που πραγματοποιήθηκε στις προηγούμενες ενότητες φαίνεται να πάσχει από αδυναμίες που επηρεάζουν την αντικειμενικότητα των αποτελεσμάτων. Το γεγονός αυτό πρέπει να μας προβληματίσει, όχι όμως και να μας αποθαρρύνει. Είναι αναμενόμενο να παρουσιάζονται τέτοιες και τέτοιας έκτασης δυσκολίες σε μια έρευνα γύρω από την τεχνολογία λογισμικού, όπως συμβαίνει και σε αντίστοιχες μελέτες στις κοινωνικές επιστήμες, με τις οποίες υπάρχουν πολλά κοινά. Τα συμπεράσματα ερευνών όπως η παρούσα έχουν τους περιορισμούς τους, αλλά δε στερούνται αξίας. Σε κάθε περίπτωση, μας βοηθούν να προσεγγίσουμε το βασικό στόχο μας, δηλ. να εμβαθύνουμε στη φύση των φαινομένων και τελικά να κατασκευάσουμε νέες θεωρίες. Στην τεχνολογία λογισμικού, μια θεωρία που προσφέρει *εξήγηση* (το «γιατί» ενός φαινομένου) και *πρόγνωση* (το «τι θα συμβεί») έχει πολύ μεγάλη ακαδημαϊκή αλλά και πρακτική αξία, ακόμα και αν δεν διαθέτει την αδήριτη ισχύ ενός νόμου της

φυσικής (21). Αποτελεί ένα εννοιολογικό πλαίσιο και ένα μέσο αναλυτικής γενίκευσης εκεί όπου η στατιστική γενίκευση φτάνει στα όριά της, όπως π.χ. στις μελέτες περιπτώσεων ή στις δειγματοληψίες από μεγάλες πληθυσμιακές ομάδες, καταστάσεις που συναντάμε πολύ συχνά στην τεχνολογία λογισμικού. Άλλωστε, κάθε καινούργια θεωρία θα χρειαστεί να περάσει στη συνέχεια από αυστηρή αξιολόγηση με κριτήρια την ελεγξιμότητα, την επαρκή υποστήριξη σε εμπειρικά ευρήματα, την ερμηνευτική της ισχύ, τη γενικότητα και την χρηστικότητά της. (22)
Σε τελική ανάλυση, ακόμα και μια ασθενής θεωρία θα είναι πάντοτε προτιμότερη από το τίποτα. Γι' αυτό και είναι προς το συμφέρον των μηχανικών που ασχολούνται με την εμπειρική τεχνολογία λογισμικού να επιδιώξουν με συνέπεια την κατασκευή θεωριών και, συνακόλουθα, να επιμείνουν στην ερευνητική εργασία που θα ανοίξει αυτό το δρόμο.

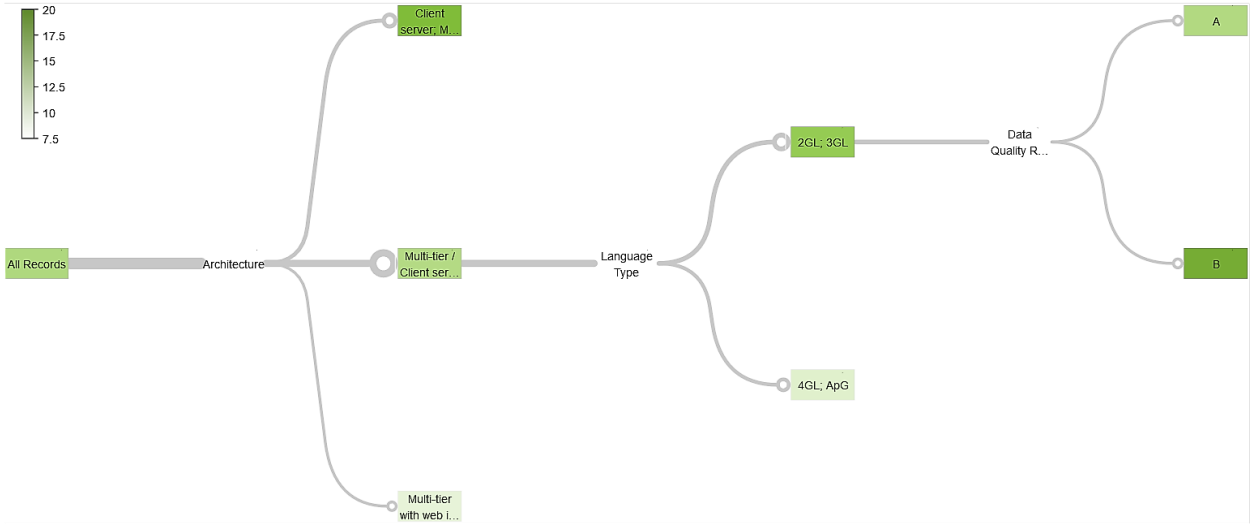
12 ΠΑΡΑΡΤΗΜΑ



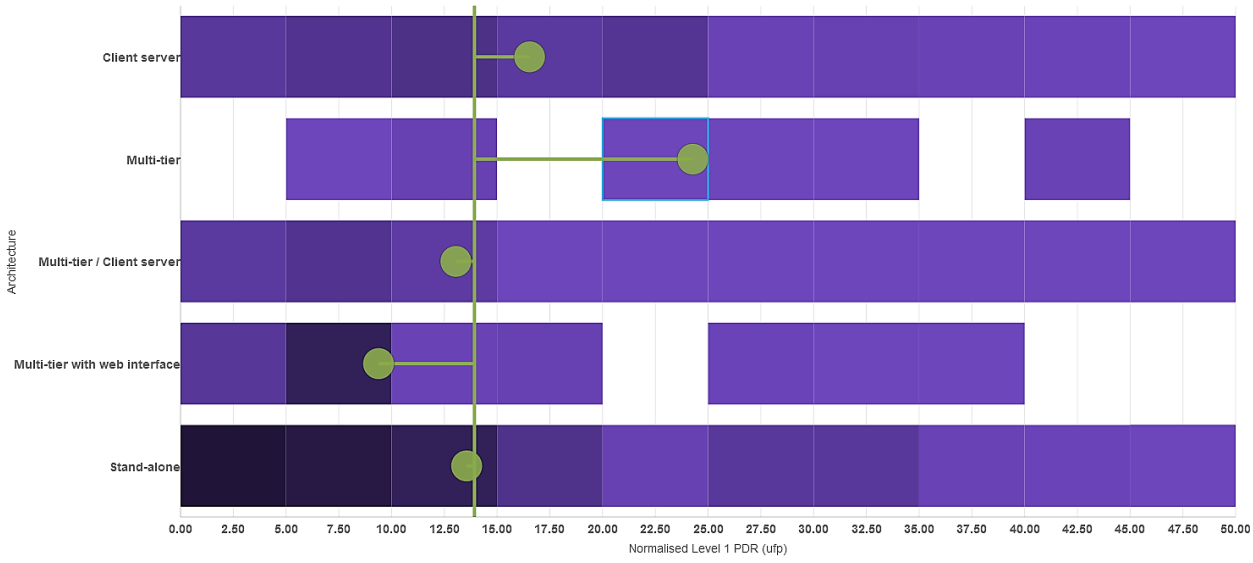
Εικόνα 12-1: Στρώμα A / 2 μεταβλητές (Αρχιτεκτονική και Τύπος Γλώσσας, I)



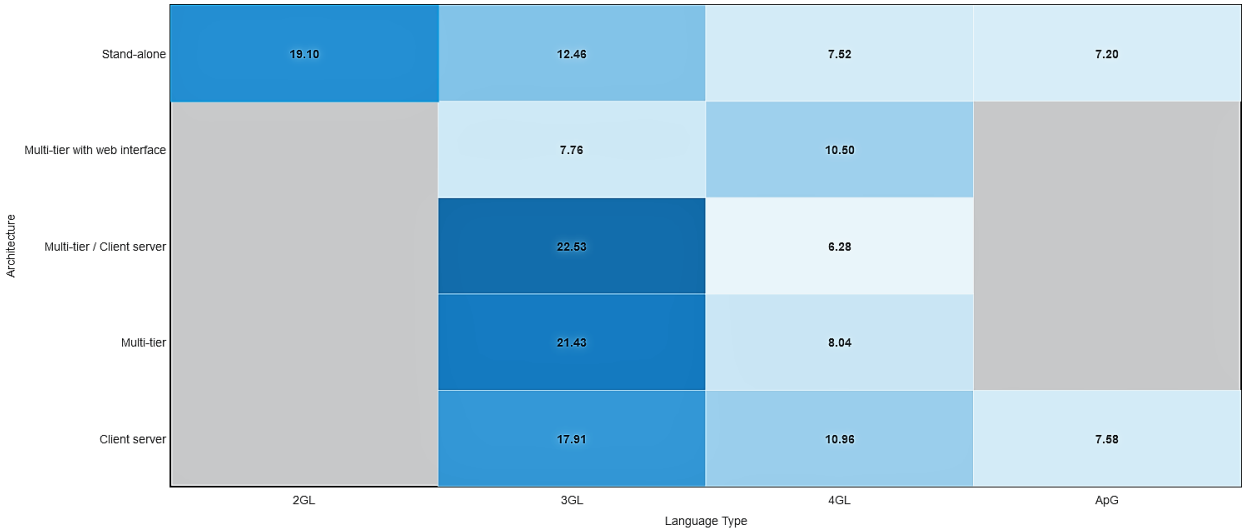
Εικόνα 12-2: : Στρώμα A / 2 μεταβλητές (Αρχιτεκτονική και Τύπος Γλώσσας, II)



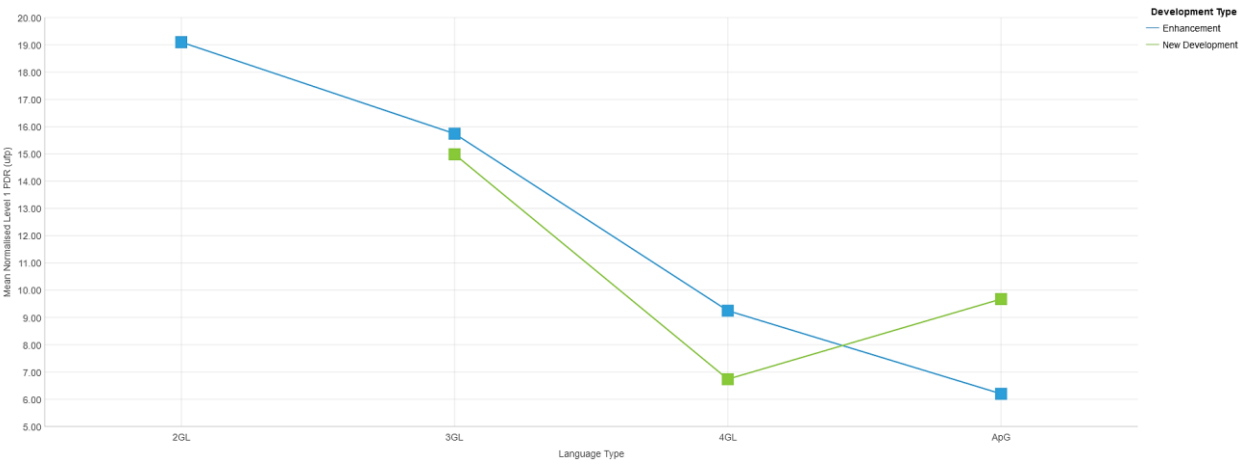
Εικόνα 12-3: Στρώμα Α / Δέντρο παλινδρόμησης



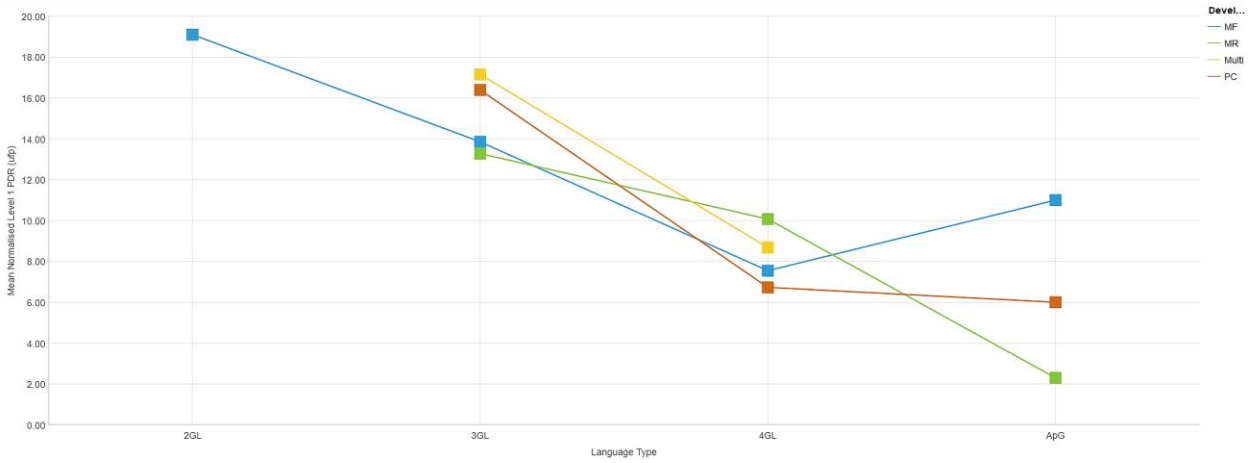
Εικόνα 12-4: Στρώμα Α / PDR και Αρχιτεκτονική



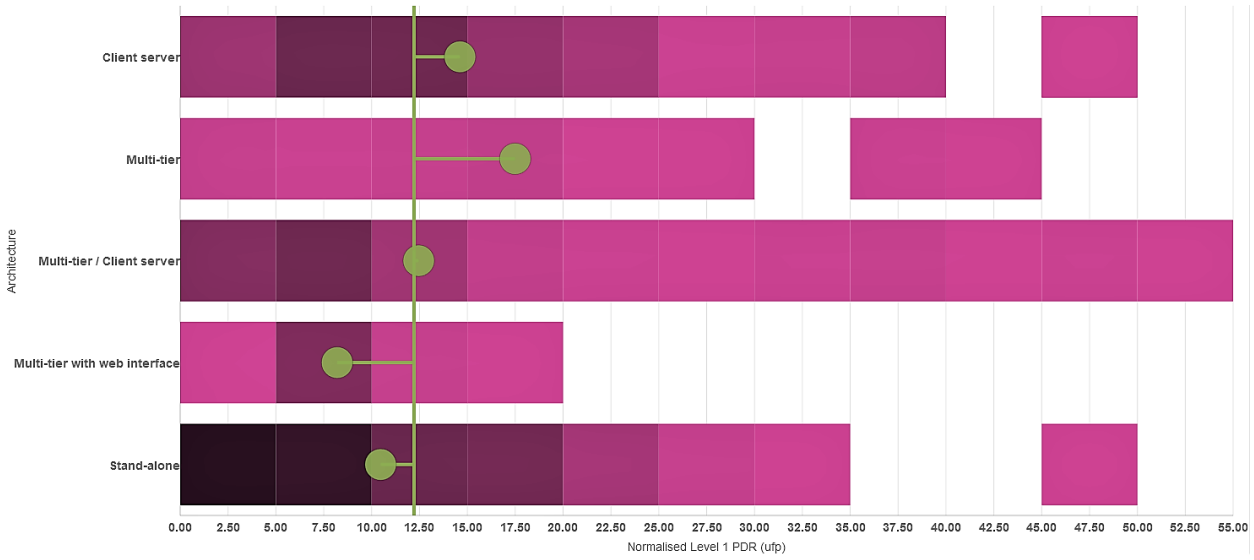
Εικόνα 12-5: Στρώμα Β / 2 μεταβλητές (Αρχιτεκτονική και Τύπος Γλώσσας)



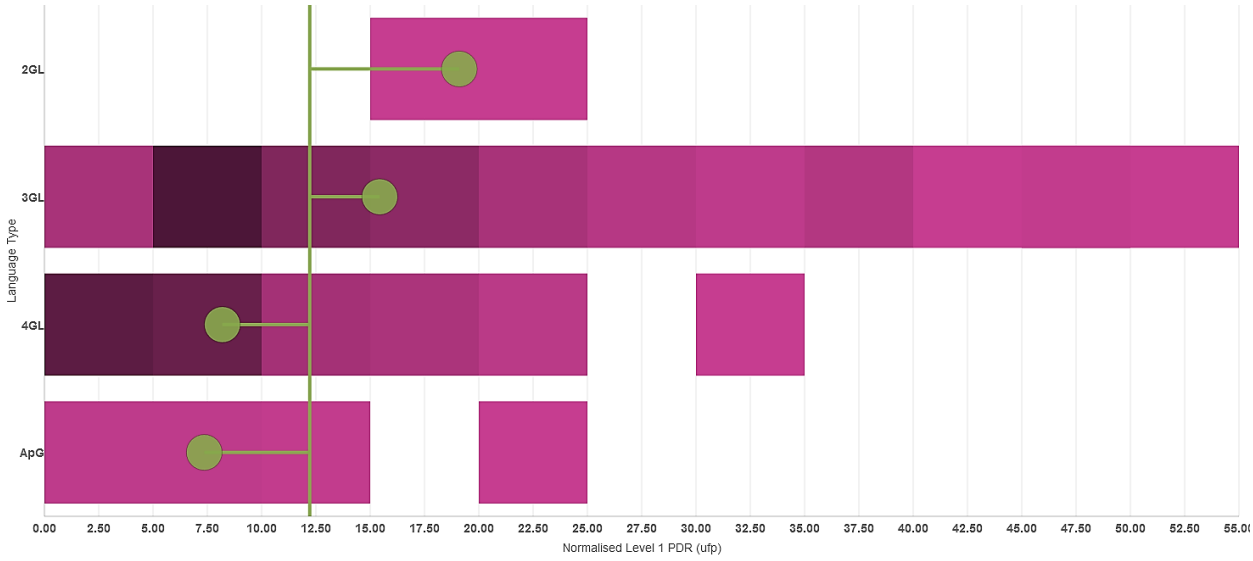
Εικόνα 12-6: Στρώμα Β / 2 μεταβλητές (Είδος Εφαρμογής και Τύπος Γλώσσας)



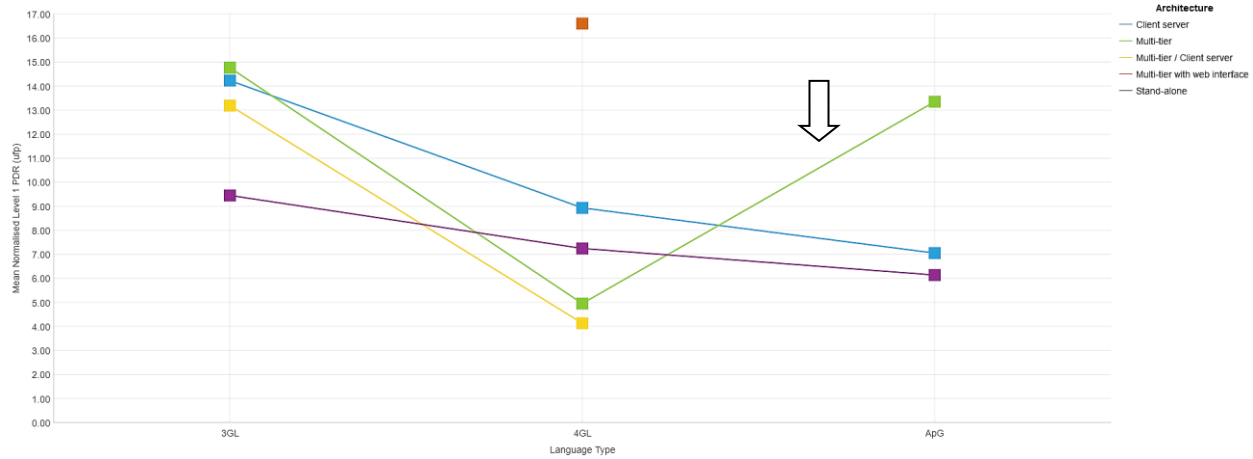
Εικόνα 12-7: Στρώμα Β / 2 μεταβλητές (Πλατφόρμα Ανάπτυξης και Τύπος Γλώσσας)



Εικόνα 12-8: Στρώμα Β / PDR και Αρχιτεκτονική



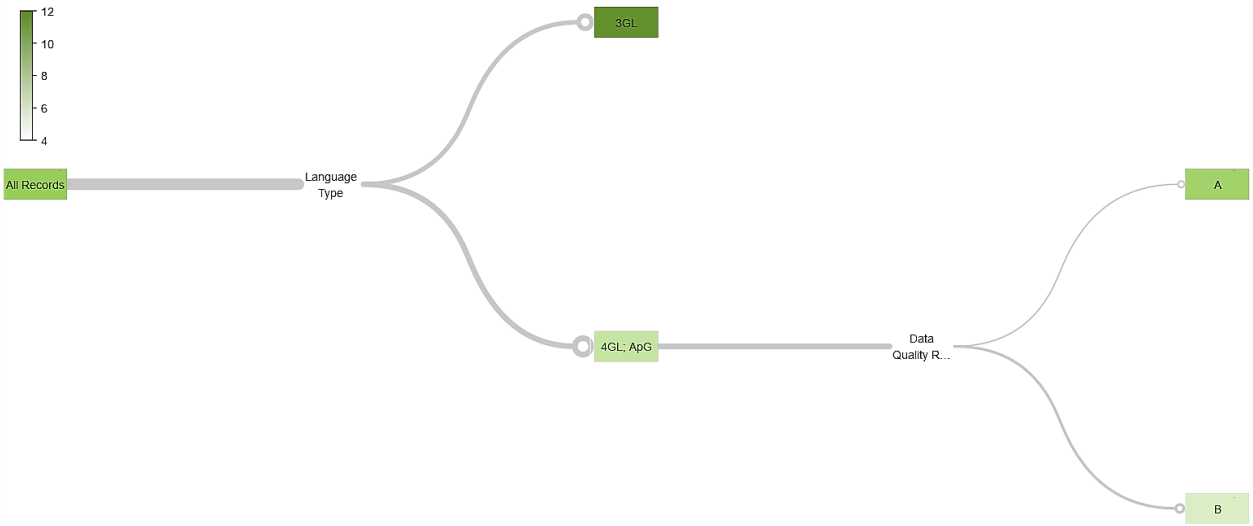
Εικόνα 12-9: Στρώμα B / PDR και Τύπος Γλώσσας



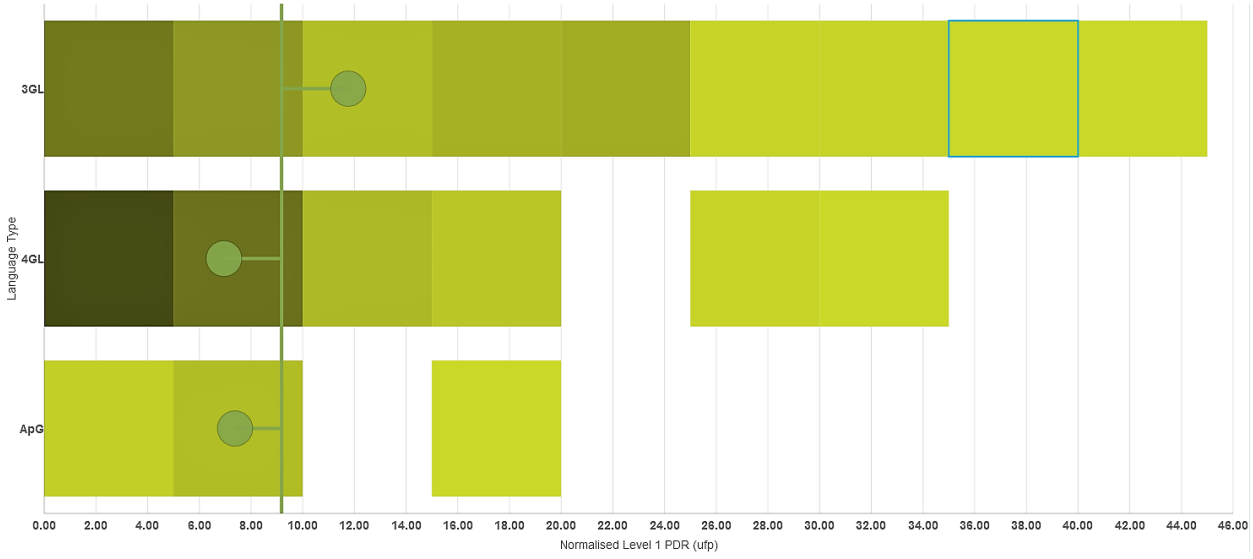
33

Εικόνα 12-10: Στρώμα C / 2 μεταβλητές (Αρχιτεκτονική και Τύπος Γλώσσας)

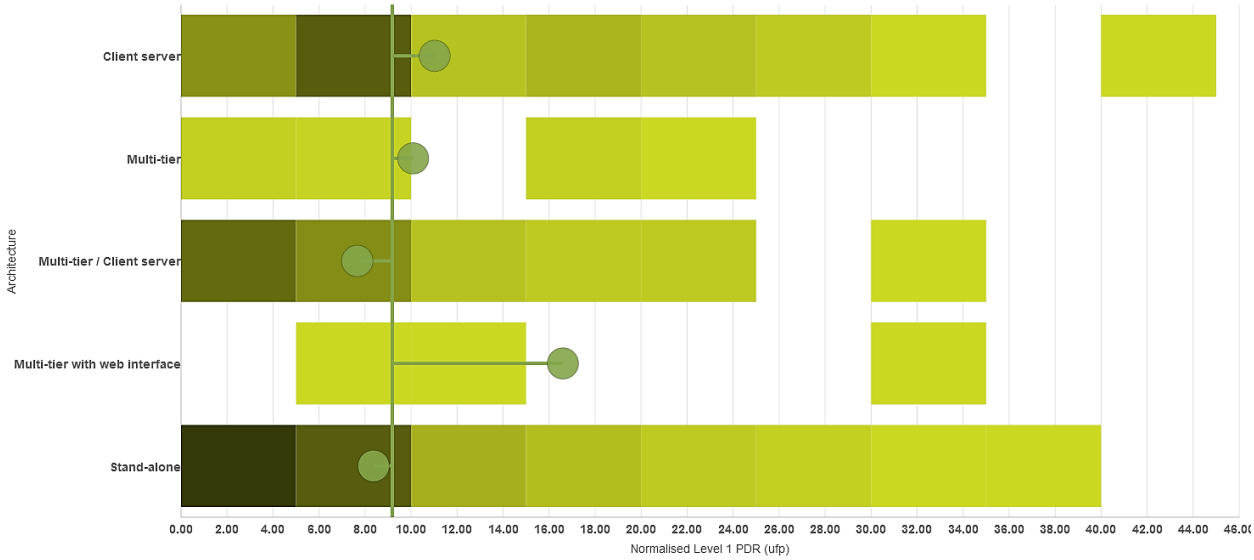
[33] Η απότομη άνοδος για τις εφαρμογές Multi-tier από 3GL σε 4GL (βλ. βέλος στην Εικόνα 12-10) δεν έχει νόημα να αναλυθεί ως προς τη σημαντικότητα καθώς αναφέρεται σε πολύ μικρούς πληθυσμούς (μόλις 3 έργα για την περίπτωση 3GL και 4 για την περίπτωση 4GL)



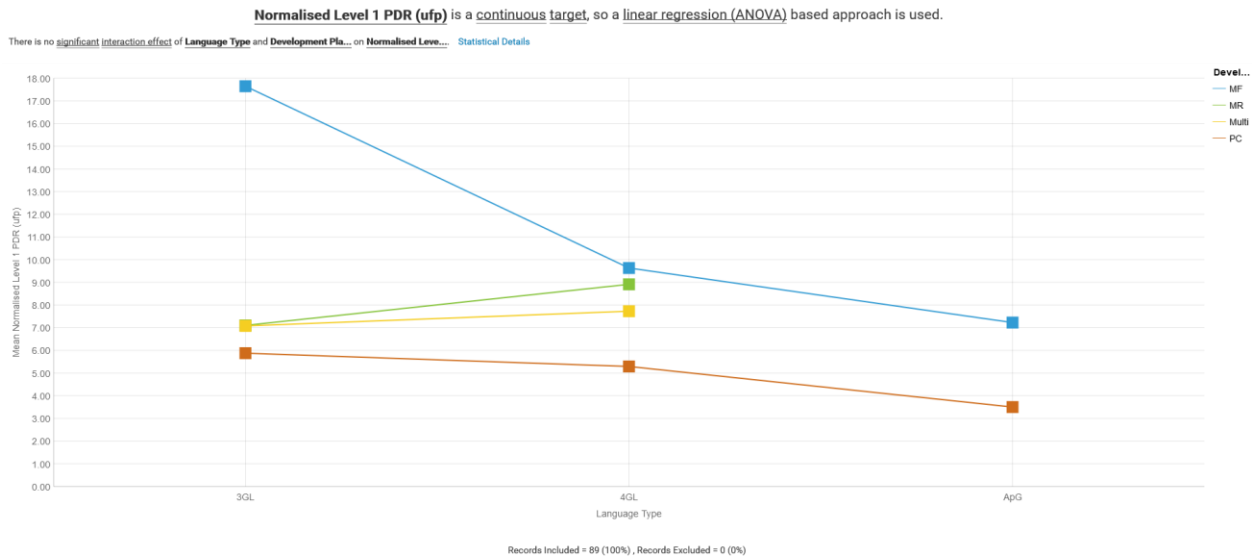
Εικόνα 12-11: Στρώμα C / Δέντρο παλινδρόμησης



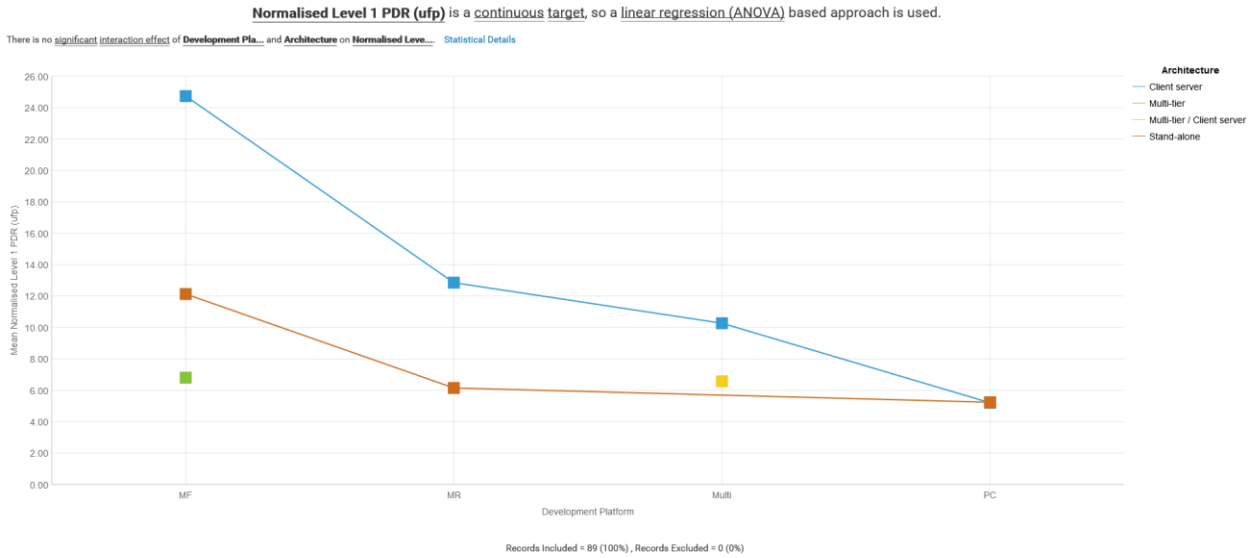
Εικόνα 12-12: Στρώμα C / PDR και Τύπος Γλώσσας



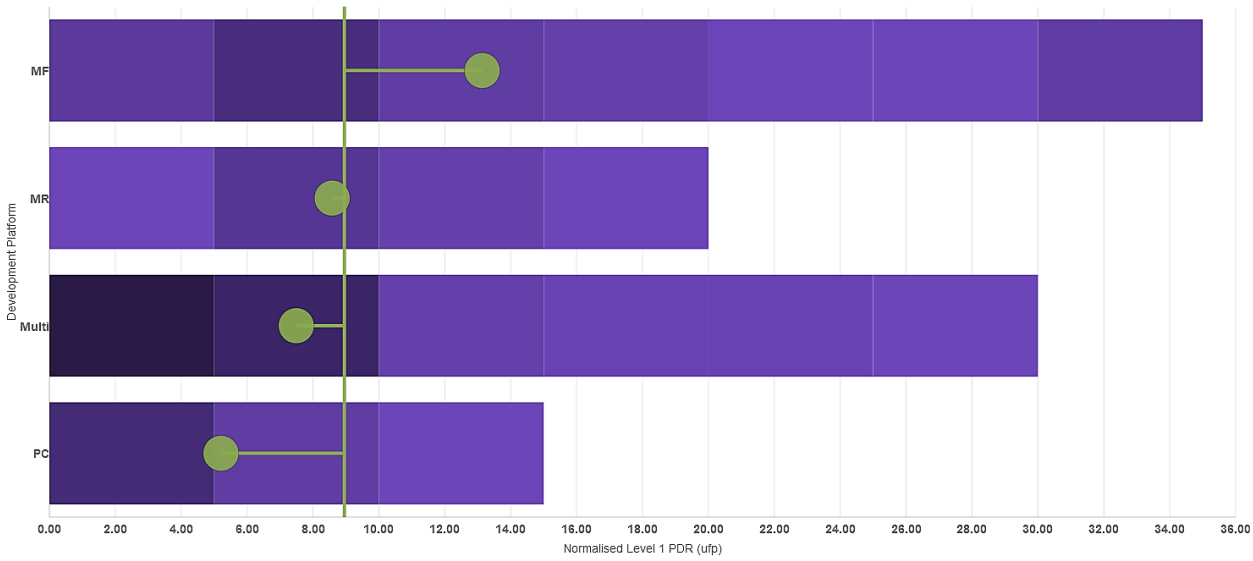
Εικόνα 12-13: Στρώμα C / PDR και Αρχιτεκτονική



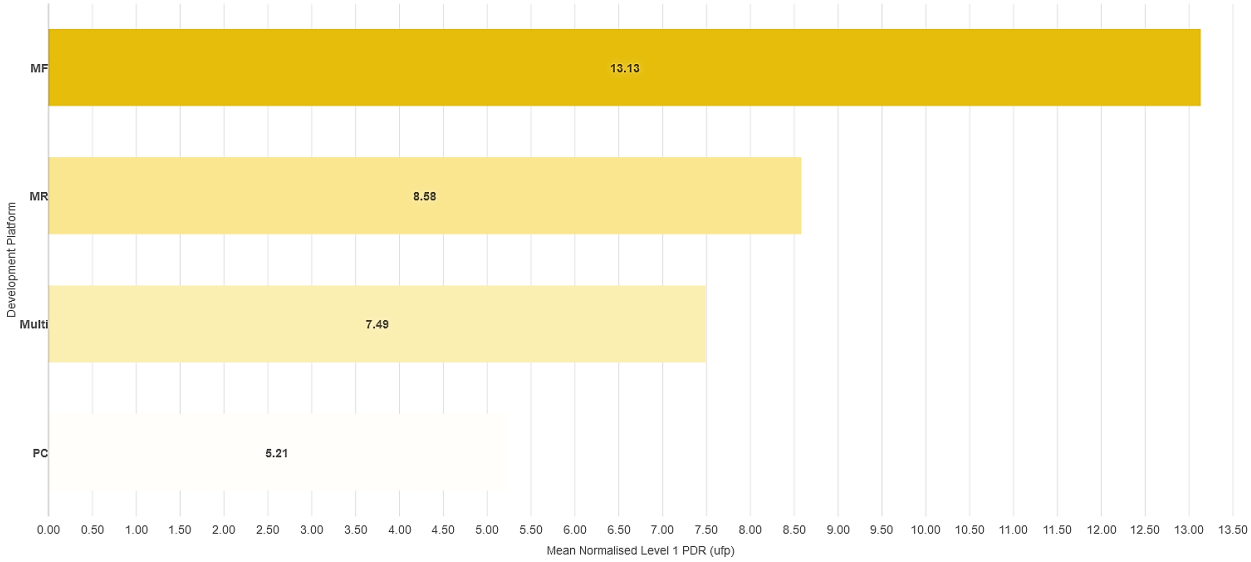
Εικόνα 12-14: Στρώμα D / 2 μεταβλητές (Πλατφόρμα Ανάπτυξης και Τύπος Γλώσσας)



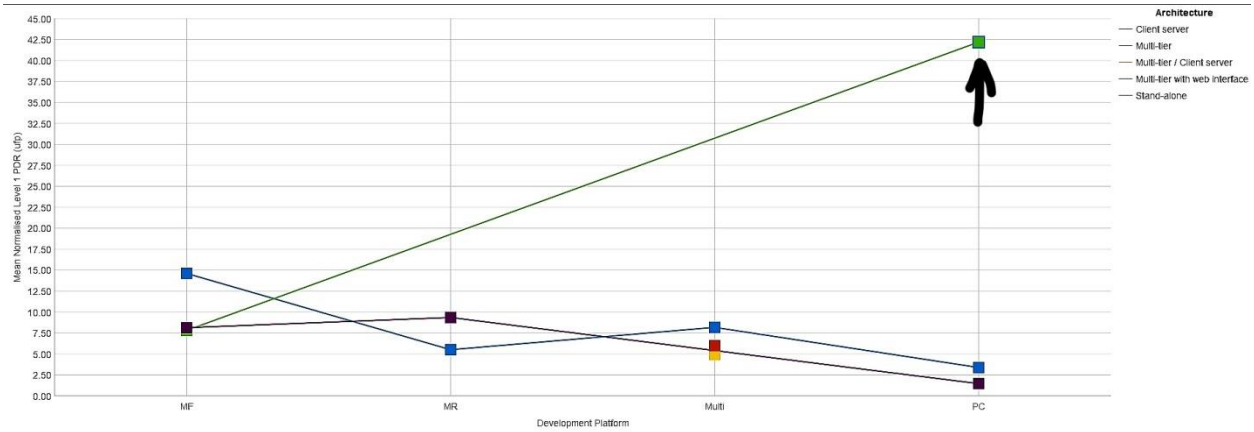
Εικόνα 12-15: Στρώμα D / 2 μεταβλητές (Πλατφόρμα Ανάπτυξης και Αρχιτεκτονική)



Εικόνα 12-16: Στρώμα D / PDR και Πλατφόρμα Ανάπτυξης (I)



Εικόνα 12-17: Στρώμα D / PDR και Πλατφόρμα Ανάπτυξης (II)



Εικόνα 12-18: Στρώμα E / 2 μεταβλητές (Πλατφόρμα Ανάπτυξης και Αρχιτεκτονική)

13 ΒΙΒΛΙΟΓΡΑΦΙΚΕΣ ΠΑΡΑΠΟΜΠΕΣ

1. **Boehm, Barry W.** Software Engineering Economics. *IEEE Transactions on Software Engineering.* 1984, Τόμ. 10, 1.

2. **Jones Capers, Bonsignour Olivier.** *The Economics of Software Quality.* s.l. : Addison-Wesley, 2012.

3. **Jones, Capers.** *Applied Software Measurement. Global analysis of productivity and quality.* 3rd edition. s.l. : Mc Graw Hill, 2008.

4. **Kenett Ron S., Baker Emmanuel R.** *Software Process Quality, Management and Control.* s.l. : Marcel Dekker Inc., 1999.

5. *Software Quality Model Requirements ofr Software Quality Engineering.* **Cote, Marc-Alexis, Suryn, Witold και Georgiadou, Elli.** 2006. Software Quality Management & INSPIRE Conference (BSI).

6. **Fenton, Norman και Neil, Martin.** Software Metrics: successes, failures and new directions. *The Journal of Systems and Software.* 1999, Τόμ. 47.

7. **Βεσκούκης, Βασίλειος.** *Τεχνολογία Λογισμικού ΙΙ.* s.l. : Ελληνικό Ανοικτό Πανεπιστήμιο, 2001.

8. **Sommerville, Ian.** *Software Engineering.* 9th edition. s.l. : Addison-Wesley, 2010.

9. **Tian, Jeff.** *Software Quality Engineering. Testing, quality assurance and quantifiable improvement.* s.l. : Wiley-Interscience, 2005.

10. **Laird Linda L., Brennan Carol M.** *Software Measurement and Estimation. A practical approach.* s.l. : Wiley, 2006.

11. **Basili, Victor R., Caldiera, Gianluigi και Rombach, H. Dieter.** Goal Question Metric Paradigm. *Encyclopedia of Software Engineering.* s.l. : John Wiley and Sons, Inc., 1994.

12. **Rosenberg, Jarrett.** Statistical Methods and Measurement. [επιμ.] Forrest Shull, Janice Singer και Dag I. K. Sjoberg. *Guide to Advanced Empirical Software Engineering.* s.l. : Springer, 2008.

13. *Software Metrics: Roadmap.* **Fenton, Norman E, και Neil, Martin.** s.l. : ISCE, 2000. Proceedings of the 22nd Conference on the Future of Software Engineering.

14. **Fenton Norman, Bieman James.** *Software Metrics.* 3rd edition. s.l. : CRC Press, 2014.

15. **Pandian, Ravindranath C.** *Software Metrics, A guide to planning, analysis and application.* s.l. : Auerbach Publications, 2003.

16. **Hill, Peter R.** *Practical Software Project Estimation.* 1st edition. s.l. : Mc Graw Hill, 2010.

17. **Abran, Alain.** *Software Metrics and Software Metrology.* s.l. : Wiley, 2010.

18. **Fenton, Norman E. και Neil, Martin.** A Critique of Software Defect Prediction Models. *IEEE Transactions on Software Engineering.* 1999, Τόμ. 25, 3.

19. **Myers, Glenford J.** *The Art of Software Testing.* 2nd Edition. s.l. : John Wiley & Sons, Inc., 2004.

20. **Wohlin, Claes, και συν.** *Experimentation in Software Engineering.* s.l. : Springer, 2012.

21. **Sjoberg, Dag I.K., Dyba, Tore και Jorgensen, Magne.** The Future of Empirical Methods in Software Engineering Research. *Future of Software Engineering.* s.l. : IEEE Copmputer Society, 2007.

22. **Sjoberg, Dag I. K., και συν.** Building Theories in Software Engineering. [επιμ.] Forrest Shull, Janice Singer και Dag I. K. Sjoberg. *Guide to Advanced Empirical Software Engineering.* s.l. : Springer, 2008.

23. **Easterbrook, Steve, και συν.** Selecting Empirical Methods for Software Engineering Research. [επιμ.] Forrest Shull, Janice Singer και Dag I. K. Sjoberg. *Guide to Advanced Empirical Software Engineering.* s.l. : Springer, 2008.

24. **Παπαγεωργίου, Ιουλία.** *Θεωρία δειγματοληψίας.* s.l. : Ελληνικά Ακαδημαϊκά Ηλεκτρονικά Συγγράματα και Βοηθήματα, 2015.

25. **Lohr, Sharon L.** Coverage and Sampling. [επιμ.] Edith D. de Leeuw, Joop J. Hox και Don A. Dillman. *International Handbook of Survey Methodology.* s.l. : European Association of Methodology, 2008.

26. **Biemer, Paul P. και Christ, Sharon L.** Weighting Survey Data. [επιμ.] Edith D. de Leeuw, Joop J. Hox και Don A. Dillman. *International Handbook of Survey Methodology.* s.l. : European Association of Methodology, 2008.

27. *Inference and Missing Data.* **Rubin, Donald B.** 3, 1976, Biometrika, Τόμ. 63.

28. **Little, Roderick I. A. και Rubin, Donald B.** *Statistical Analysis with Missing Data.* s.l. : John Wiley & Sons, 1987.

29. **Mockus, Audris.** Missing Data in Software Engineering. [επιμ.] Forrest Shull, Janice Singer και Dag I. K. Sjoberg. *Guide to Advanced Empirical Software Engineering.* s.l. : Springer, 2008.

30. **Lundstroem, Sixten και Saerndal, Carl-Erik.** *Estimation in the presence of Nonresponse and Frame Imperfections.* Oerebro : Statistics Sweden, 2001.

31. *Handling Missing Data in Survey Research.* **Kalton, G. και Brick, JM.** s.l. : SAGE Publications, 1996, Statistical Methods in Medical Research.

32. **Kalton, Graham.** Compensating for Missing Survey Data. *Research Report Series.* s.l. : University of Michigan, 1983.

33. **Raessler, Susanne, Rubin, Donald B. και Schenker, Nathaniel.** Incomplete Data: Diagnosis, Imputation and Estimation. [επιμ.] Edith D. de Leeuw, Joop J. Hox και Don A. Dillman. *International Handbook of Survey Methodology.* s.l. : European Association of Methodology, 2008.

34. **Ben-Gal, Irad.** Outlier Detection. [επιμ.] Oded Maimon και Lior Rokash. *The Data Mining and Knowledge Discovery Handbook.* s.l. : Springer, 2006.

35. **Aggarwal, Charu C.** *Outlier Analysis.* 2nd Edition. s.l. : Springer, 2016.

36. **Dasu, Tamraparni και Johnson, Theodore.** *Exploratory Data Mining and Data Cleaning.* s.l. : Wiley-Interscience, 2003.

37. **Hand, David, Mannila, Heikki και Smyth, Padhraic.** *Principles of Data Mining.* s.l. : A Bradford Book, MIT Press, 2001.

38. **Witten, Ian H., Frank, Eibe και Hall, Mark A.** *Data Mining. Practical machine learning tools and techniques.* 3rd Edition. s.l. : Elsevier / Morgan Kaufmann, 2011.

39. **Sutton, Clifton D.** Classification and Regression Trees, Bagging and Boosting. [επιμ.] C. R. Rao, E. J. Wegman και J. L. Solka. *Data Mining and Data Visualization.* s.l. : Elsevier, 2005.

40. **Rokach, Lior και Maimon, Oded.** Decision Trees. *The Data Mining and Knowledge Discovery Handbook.* s.l. : Springer, 2006.

41. **Pfleeger, Shari Lawrence και Atlee, Joanne M.** *Software Engineering, Theory and Practice.* 4th Edition. s.l. : Pearson, 2009.

42. Data Demographics. *ISBSG Repository - Release 11.* 2009.

43. Field Descriptions. *ISBSG Repository - Release 11.* 2009.

44. **Levy, Paul S. και Lemeshow, Stanley.** *Sampling of Populations. Methods and applications.* 3rd Edition. s.l. : John Wiley & Sons, Inc., 1999.

45. **Siegel, Sidney.** *Nonparametric Statistics for the Behavioral Sciences.* s.l. : McGraw-Hill Book Company, Inc., 1956.

46. *The impact of nonresponse on survey quality.* **Bethlehem, Jelke και Bakker, Bart.** Hong Kong : s.n., 2013. Proceedings of the 59th ISI World Statistics Congress.

47. **Cobben, Fannie.** *Nonresponse in sample surveys. Methods for analysis and adjustment.* Amsterdam University. 2009. doctoral thesis.

48. **Sarndal, Carl-Erik και Lundstrom, Sixten.** *Assessing auxiliary vectors for control of nonresponse bias in the calibration estimator.* s.l. : Statistics Sweden, 2007.

49. **Groves, Robert M.** Nonresponse rates and nonresponse bias in household surveys. *Public Opinion Quarterly.* 2006, Τόμ. 70, 5.

50. **Pfleeger, Shari Lawrence και Kitchenham, Barbara A.** Principles of Survey Research. *Software Engineering Notes.* 2001-2003, Τόμ. 26-28.

14 ΒΙΒΛΙΟΓΡΑΦΙΑ

Bell, Douglas; *Software Engineering for Students, A programming approach*, 4th ed., Addison-Wesley, 2005

Chemuturi, Murali; *Mastering Software Quality Assurance,* J. Ross Publishing, 2011

Emam, Khaled El; *The ROI from Software Quality,* Auerbach Publications, 2007

Galorath, Daniel D. and Evans, Michael W.; *Software Sizing, Estimation and Risk Management*, Auerbach Publications, 2006

http://www.esem-conferences.org

Johnson, Richard A. and Wichern, Dean W.; *Applied Multivariate Statistical Analysis*, 6th ed., Pearson, 2007

Juristo, Natalia and Moreno, Ana M.; *Basics of Software Engineering Experimentation*, Universidad Politecnica de Madrid

Juristo, Natalia and Moreno, Ana M. (ed.); *Lecture Notes on Empirical Software Engineering*, World Scientific, 2003

Lohr, Sharon L.; *Sampling, Design and analysis*, 2nd ed., Brooks/Cole, 2010

Naik, Kshirasagar and Tripathy, Priyadarshi; *Software Testing and Quality Assurance, Theory and Practice*, Wiley, 2008

Pressman, Roger S.; *Software Engineering, A practitioner's approach*, 7th ed., McGraw Hill, 2009

Rokash, Lior and Maimon, Oded; *Data Mining with Decision Trees, Theory and applications*, World Scientific, 2007

Särndal, Carl-Erik, Swensson, Bengt and Wretman, Jan; *Model Assisted Survey Sampling*, Springer, 1992

Βεσκούκης, Β.; *Στοιχεία Τεχνολογίας Λογισμικού*, Ελληνικά Ακαδημαϊκά Ηλεκτρονικά Συγγράμματα και Βοηθήματα, 2015

Ξεκαλάκη, Ε.; *Τεχνικές Δειγματοληψίας,* 2η Έκδοση, Αθήνα, 2004