# Μετρικές μέθοδοι λογισμικού

Γεωργουδάκης Ιωάννης

# ΠΕΡΙΕΧΟΜΕΝΑ

# 1-Ποιότητα λογισμικού

Στο πλαίσιο της μηχανικής λογισμικού η ποιότητα λογισμικού είναι ένα μέτρο που δείχνει την ποιότητα της σχεδίασης και κατά πόσο το παραχθέν λογισμικό ανταποκρίνεται σε αυτήν την σχεδίαση. Ο όρος ποιότητα στην μηχανική λογισμικού είναι μία ασαφής έννοια και πολλές φορές γίνονται λάθη όταν προσπαθούμε να την προσεγγίσουμε. Αυτό το λάθος γίνεται γιατί ακριβώς η ποιότητα δεν είναι μια απλή ιδέα αλλά μάλλον τις περισσότερες φορές παίρνει πολλές διαστάσεις .Μερικοί χρησιμοποιούν αυτήν την έννοια με την ευρύτερη σημασία της, ενώ άλλοι μιλάνε με συγκεκριμένους όρους.

**Μία διάσημη άποψη** για τον όρο ποιότητα λογισμικού , είναι ότι αυτός ο όρος είναι ένα άυλο γνώρισμα το οποίο μπορεί να συζητηθεί και να κριθεί δεν μπορεί όμως να μετρηθεί . Γενικά οι άνθρωποι μιλούν και συζητούν για γνωρίσματα της ζωής τους χωρίς αυτά να έχουν οριστεί. Για παράδειγμα μιλούν για ποιότητα ζωής χωρίς κάποιος να έχει ορίσει αυτό το μέγεθος . Μέχρι και στην επιστήμη οι επιστήμονες μιλούν για ύλη , χρησιμοποιούν την ύλη χωρίς κάποιος να έχει καταφέρει να δώσει κάποιον σαφή όρο για το τι είναι ύλη . Αυτό το σφάλμα πηγάζει από το γεγονός ότι οι άνθρωποι αντιλαμβάνονται τα πράγματα διαφορετικά . Η ποιότητα όμως λογισμικού είναι ένα μέγεθος το οποίο πρέπει να οριστεί , να μετρηθεί , να διαχειριστεί και τελικά να βελτιωθεί .

**Μία άλλη άποψη** είναι ότι ο όρος ποιότητα λογισμικού υποδηλώνει την πολυτέλεια . Ακριβά περίτεχνα και πολύπλοκα προϊόντα , θεωρείται ότι παρέχουν υψηλότερα επίπεδα ποιότητας από άλλα ποιο απλά . Επομένως ένα αυτοκίνητο Cadillac είναι ποιο ποιοτικό προϊόν από ‘τι είναι μια Chevrolet χωρίς να μιλάει κανείς για κόστος επισκευής και συντήρησης . Σύμφωνα με αυτήν την άποψη λοιπόν ο όρος ποιότητα αναφέρεται μόνο σε μία κλάση λογισμικών που είναι ακριβά και περίπλοκα . Ένα φθηνό και απλό λογισμικό δεν μπορεί να είναι

ταυτόχρονα και ποιοτικό . Από ‘τι καταλαβαίνετε αυτή η άποψη είναι λανθασμένη .

Οι παραπάνω ασαφής ερμηνείες όμως δεν θα βοηθήσουν στην βελτίωση ποιότητας ενός λογισμικού . Πρέπει να εξηγήσουμε αυτόν τον όρο με ένα πιο επιστημονικό τρόπο.

Συχνά η λέξη ποιότητα στο λογισμικό συνδέεται στενά με την έλλειψη ‘bugs’ που υπάρχουν σε ένα πρόγραμμα. Επίσης ένας δείκτης ποιότητας μπορεί να θεωρηθεί και πόσο στενή σχέση υπάρχει ανάμεσα στις απαιτήσεις και στο παραχθέν λογισμικό. Βέβαια υπάρχουν και επίσημες θέσεις και απόψεις σχετικά με την έννοια ποιότητα λογισμικού.

Η λέξη ποιότητα αναφέρεται σύμφωνα με το μοντέλο CUPRIMDA της IBM στους παρακάτω όρους:

**C**apability (Ικανότητα)

**U**sability (Ευχρηστία)

**P**erformance (Επίδοση)

**R**eliability (Αξιοπιστία)

**I**nstallability (Δυνατότητας εγκατάστασης)

**M**aintainability(Συντηρησιμότητα)

**D**ocumentation(Οδηγοί χρήσης)

**A**vailability(Διαθεσιμότητα)

Θα λέγαμε ότι η ‘μέτρηση’ της ποιότητας ενός προϊόντος είναι πολύ σημαντική , διότι αποτελεί ένα δείκτη που μας δείχνει κατά πόσο η

ανάπτυξη προγράμματος κινείται προς την σωστή κατεύθυνση η αποκλίνει από τον αρχικό στόχο και σχεδιασμό .

## 2-Θεωρία μετρήσεων

Η μέτρηση έχει παίξει καθοριστικό ρόλο στην εξέλιξη όλων των επιστημών . Η επιστήμες περνούν τα στάδια της μελέτης , συλλογής πληροφοριών και ανάλυσή τους ώστε αργότερα να καταλήξουν σε μία επιστημονική θεωρία . Έπειτα από αυτό και με νέες παρατηρήσεις μπορεί να έρθει η επιβεβαίωση της θεωρίας αυτής ή ακόμα και η απόρριψή της. Έτσι λοιπόν λειτουργεί και ο μηχανισμός της ανάπτυξης λογισμικού . Όσο πιο σκληρούς κανόνες θέτουμε τόσο πιο καλή ποιότητα κρύβεται πίσω από το έργο . Για να 'πάρουμε' τις μετρήσεις μας όμως και να αποφανθούμε εάν το αποτέλεσμα αντιπροσωπεύει επάξια την θεωρία χρησιμοποιούμε τέσσερις κλίμακες:

**→ Ονομαστικές κλίμακες**

Η πιο απλή διαδικασία μέτρησης που χρησιμοποιεί η επιστήμη είναι η κατάταξη. Υπάρχει η ανάγκη λοιπόν να κατατάξουμε τα στοιχεί μας σε κατηγορίες σύμφωνα με τα κοινά τους χαρακτηριστικά . Για παράδειγμα αν κατατάξουμε τα προγράμματα σε κατηγορίες σύμφωνα με την μεθοδολογία ανάπτυξής τους , θα έχουμε προγράμματα ανεπτυγμένα με την διαδικασία του καταρράκτη , με την σπειροειδή διαδικασία κτλ.

**→ Τακτικές κλίμακες**

Η τακτική κλίμακα είναι ένα είδος μέτρησης που αποδίδει τιμές στα αντικείμενα μας με βάση το πόσο αυτά ανταποκρίνονται καλύτερα στο θέμα για το οποίο βαθμολογούνται. Τακτικές κλίμακες για παράδειγμα χρησιμοποιούνται στα σχολεία , τακτική κλίμακα χρησιμοποιεί ο γιατρός για να δείξει το πόσο καλή βελτίωση έχει ένας ασθενής ( με αριθμούς από 1-10) και άλλα .

**→ Κλίμακες διαστημάτων**

Είναι μία κλίμακα που μετρά ένα κοινό χαρακτηριστικό για τα μεγέθη μας . Για παράδειγμα ας υποθέσουμε ότι κατασκευάζουμε ένα μετρητή άγχους για να μετρήσουμε το χαρακτηριστικό άγχος για τους ανθρώπους .Φυσικά αυτόν τον μετρητή θα τον χωρίζαμε σε υποδιαιρέσεις . Κάποιος άνθρωπος που θα είχε άγχος 30 με έναν άλλο που είχε άγχος 31 , θα είχαν την ίδια διαφορά που θα είχε ένας άλλος άνθρωπος με άγχος 50 με έναν που είχε 51 . Ωστόσο δεν είναι δυνατόν να κάνουμε δηλώσεις του τύπου : Ο άνθρωπος με άγχος 50 έχει το διπλάσιο άγχος από ότι έχει ένας άλλος με 25 . Ένα καλό παράδειγμα για να το καταλάβουμε αυτό είναι η θερμοκρασία . Μία θερμοκρασία 30 βαθμών δεν είναι διπλάσιας ζέστης από ότι είναι μία θερμοκρασία 15 .

**→ Ποσοστιαίες κλίμακες**

Είναι οι κλίμακες μέτρησης οι οποίες υποδηλώνουν την απόσταση ενός αντικειμένου από κάποιο χαρακτηριστικό. Για παράδειγμα κάτι που απέχει «4» από το «0» , απέχει διπλάσια από κάτι το οποίο απέχει «2» από το «0» .

## 3-Βασικά εργαλεία μετρήσεων

Ανεξαρτήτως από τις κλίμακες μετρήσεως , όταν συλλέγουμε διάφορα δεδομένα θέλουμε να εξάγουμε χρήσιμες πληροφορίες από αυτά. Υπάρχουν πολλές τεχνικές έτσι ώστε να συλλέξουμε αυτήν την ακατέργαστη πληροφορία να την επεξεργαστούμε και να εξάγουμε από αυτήν χρήσιμα συμπεράσματα. Μερικές από τις τεχνικές αυτές παρουσιάζονται παρακάτω .

> **ΛΟΓΟΣ-RATIO**

Όταν μιλάμε για λόγο αναφερόμαστε στον αριθμό που προκύπτει διαιρώντας δύο κοινά μεγέθη ενδιαφέροντος . Για παράδειγμα εάν θέλαμε να δούμε την σχέση μεταξύ ευχαριστημένων και δυσαρεστημένων πελατών θα κάναμε:

$$\frac{Αριθμός\ Ευχαρηστημένων\ Πελατών}{Αριθμός\ Δυσαρεστημένων\ Πελατών} \times 100\%$$

Εάν αυτό το νούμερο βγει μικρότερο του 100 αυτό σημαίνει ότι ο αριθμός των δυσαρεστημένων πελατών είναι μεγαλύτερος από αυτόν των ευχαριστημένων.

> **ΑΝΑΛΟΓΙΑ-PROPORTION**

Η αναλογία διαφέρει από τον λόγο που είδαμε πιο πριν στο εξής σημείο: Ο λόγος μπορεί να χρησιμοποιηθεί για ένα συγκεκριμένο πληθυσμό – κατηγορία , ενώ η αναλογία για πολλές κατηγορίες . Για παράδειγμα λοιπόν εάν θέλαμε σε αυτήν την περίπτωση να βρούμε πόσους ευχαριστημένους πελάτες έχουμε θα κάναμε :

$$\frac{Αριθμός\ ευχαριστημένων\ πελατών}{Συνολικός\ αριθμός\ πελατών}$$

> **ΠΟΣΟΣΤΟ-PERCENTAGE**

Χρησιμοποιείται συχνά για να προβάλλουμε μία αναφορά αποτελεσμάτων κανονικοποιημένη στην κλίμακα του 100 .Η μηχανική λογισμικού χρησιμοποιεί κατά κόρον αυτό το μηχανισμό. Αν για παράδειγμα είχαμε 2000 κώδικα σε ένα πρόγραμμα ( 2 KLOC ) και μέσα σε αυτές παρουσιαζόταν 200 bugs τότε θα λέγαμε ότι :

$$\frac{200}{2000} \times 100\% = 10\%$$

Με άλλα λόγια το 10% του προγράμματός μας θα ήταν κατοικημένο από bugs .

> **ΒΑΘΜΟΣ-RATE**

Τα προηγούμενα μέτρα που είδαμε είναι στατικά. Παρουσιάζουν τα αποτελέσματα μίας δεδομένης χρονικής στιγμής. Το σημείο κλειδί για να καταλάβουμε το εργαλείο μετρήσεως 'βαθμός' , είναι ότι ο βαθμός αλλάζει δυναμικά , σύμφωνα με το βάρος που δίνουμε στα σημεία ενδιαφέροντος μιας δεδομένης χρονικής στιγμής. Για παράδειγμα εάν θέλαμε να διαπιστώσουμε τα ελαττώματα ενός έργου μιας χρονικής στιγμής θα λέγαμε ότι:

$$Ελαττώματα = \frac{Αριθμός\ ελαττωμάτων}{Περιθώρια\ ελαττωμάτων} \times K$$

Όπου το Κ παίρνει συνήθως την τιμή 1000 . Η μεταβλητή που αλλάζει δυναμικά εδώ είναι 'τα περιθώρια ελαττωμάτων' μιας χρονικής στιγμής. Για παράδειγμα αυτός ο αριθμός θα ήταν μεγάλος στην αρχή ενός έργου και μικρότερος όσο περνούσε ο χρόνος (θα μίκραιναν τα περιθώρια) .

> **6σ**

Είναι μία στρατηγική διαχείρισης των επιχειρήσεων που παρουσιάστηκε για πρώτη φορά από την εταιρία Motorola . Το 6σ επιδιώκει να βελτιστοποιήσει την ποιότητα των αποτελεσμάτων κατά την παραγωγή ενός έργου , εντοπίζοντας και εξαλείφοντας τα προβλήματα που εντοπίζονται σε αυτό . Για να επιτύχει το παραπάνω χρησιμοποιεί τον μηχανισμό της στατιστικής από τα μαθηματικά .

Το 6σ μας προσφέρει δύο βασικές μεθόδους που ακούν στα ονόματα:

**DMAIC** και **DMADV** .

## DMAIC

Προέρχεται από τα αρχικά των λέξεων :

- **D**efine the problem, the voice of the customer, and the project goals, specifically.
- **M**easure key aspects of the current process and collect relevant data.
- **A**nalyze the data to investigate and verify cause-and-effect relationships. Determine what the relationships are, and attempt to ensure that all factors have been considered. Seek out root cause of the defect under investigation.
- **I**mprove or optimize the current process based upon data analysis using techniques such as design of experiments, poka yoke or mistake proofing, and standard work to create a new, future state process. Set up pilot runs to establish process capability.
- **C**ontrol the future state process to ensure that any deviations from target are corrected before they result in defects. Implement control systems such as statistical process control, production boards, and visual workplaces, and continuously monitor the process.

## DMADV

Προέρχεται από τα αρχικά των λέξεων :

- Define design goals that are consistent with customer demands and the enterprise strategy.
- Measure and identify CTQs (characteristics that are Critical To Quality), product capabilities, production process capability, and risks.
- Analyze to develop and design alternatives, create a high-level design and evaluate design capability to select the best design.
- Design details, optimize the design, and plan for design verification. This phase may require simulations.
- Verify the design, set up pilot runs, implement the production process and hand it over to the process owner(s).

# 4-Μετρικά εργαλεία ποιότητας λογισμικού

Συνήθως όπως είδαμε και πριν η ποιότητα ενός λογισμικού μπορεί να μετρηθεί με το πόσα bugs υπάρχουν σε ένα πρόγραμμα ανάλογα με τις γραμμές που περιέχει . Ο μέσος χρόνος αποτυχίας ενός προγράμματος αναφέρεται στο πόσο χρόνο αυτό μπορεί να εκτελείται χωρίς να παρουσιάσει πρόβλημα . Συνήθως γράφεται MTTF(Mean Time To Failure) . Χρησιμοποιείται κατά κόρον αυτός ο όρος σε ευαίσθητα συστήματα ασφαλείας . Για παράδειγμα η εταιρία που έχει κατασκευάσει το σύστημα εναέριου ελέγχου στην Αμερική αναφέρει ότι το σύστημα μπορεί να μην είναι διαθέσιμο έως 3 δευτερόλεπτα . Η μέτρηση των γραμμών κώδικα όμως και η χρησιμοποίησή του ως μετρικό εργαλείο ποιότητας μπορεί να είναι περίπλοκη . Κάποιος μπορεί να μετρήσει το LOC ως οδηγίες προς τον Η/Υ , όπως γινόταν στην assembly ,όπου κάθε εντολή ήταν και μία οδηγία , με τις καινούριες γλώσσες προγραμματισμού όμως αυτό δεν ισχύει . Υπάρχουν διάφοροι τρόποι μέτρησης για αυτόν το σκοπό :

→ Μέτρηση μόνο των εκτελέσιμων εντολών

→ Μέτρηση εντολών συν των σχολίων .

→ Μέτρηση εντολών και μεταβλητών.

κοκ .

Στην εταιρία IBM η μέτρηση αυτή γίνεται ως εξής : Επειδή η μέτρηση των γραμμών κώδικα βασίζεται μόνο στην μέτρηση των εντολών οδηγιών ( Logical LOC ) , δηλαδή των δηλώσεων μεταβλητών εκτελέσιμων οδηγιών χωρίς τα σχόλια , τα μεγέθη αυτά ονομάζονται shipped source instructions (SSI) και changed source instructions (CSI) .

Το πρώτο αναφέρετε στις παλιές εντολές που έχουν συγγραφεί και το δεύτερο στις αλλαγμένες γραμμές κώδικα . Ο τύπος είναι ο εξής :

$$SSI(Τωρινή\ έκδοση) = SSI(Προηγούμενη\ έκδοση) + CSI(Νέες\ γραμμές\ κώδικα) - Διαγραμμένος\ κώδικας - Αλλαγμένος\ κώδικας$$

Όπως διαπιστώσαμε η μέτρηση γραμμών κώδικα δεν μας δίνει και πολύ αξιόπιστα αποτελέσματα . Για αυτό καταφεύγουμε στα μαθηματικά και στην στατιστική . Ο Ishikawa παρουσίασε τα οκτώ εργαλεία που πλέον χρησιμοποιούνται από πολλές κατασκευαστικές εταιρίες ( είτε software είτε άλλες ) . Αυτά τα εργαλεία είναι :

**→ Λίστες ελέγχου**

Μια λίστα ελέγχου παίζει σημαντικό ρόλο στην ανάπτυξη λογισμικού. Η διαδικασία ανάπτυξης ενός προγράμματος περνά μέσα από πολλές φάσεις , όπως η συλλογή και καταγραφή των απαιτήσεων , η αρχιτεκτονική και άλλα . Κάθε διαδικασία λοιπόν από αυτές περνάει από επιμέρους στάδια . Μία λίστα ελέγχου λοιπόν βοηθά τους προγραμματιστές να διαπιστώσουν ότι αυτά τα στάδια έχουν διεκπεραιωθεί πλήρως και δεν υπάρχει κάποια έλλειψη . Ένα παράδειγμα είναι το παρακάτω:

**Σημειώστε με Χ**

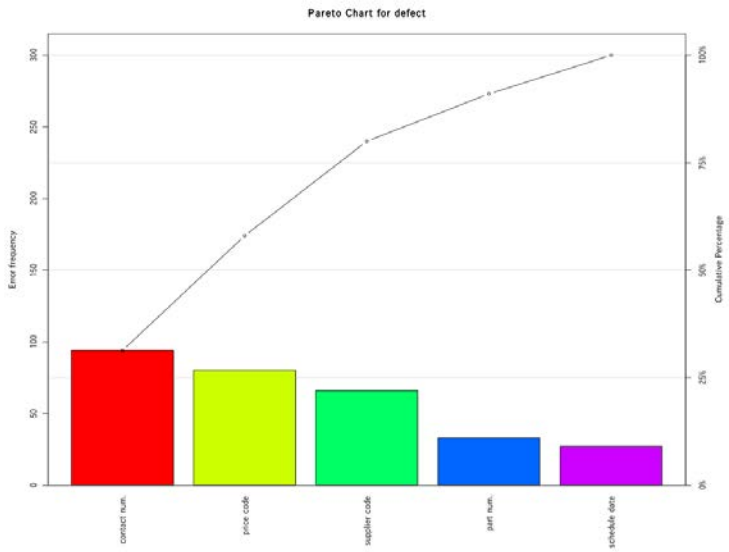
| | ΝΑΙ | ΟΧΙ |
|---|---|---|
| **1-Έχετε διαπιστώσει ποιο θα είναι το βασικό σενάριο χρήσης?** | \| \| | \| \| |
| **2-Πιστεύετε ότι στην συνέχεια θα προκύψουν Και άλλα σενάρια χρήσης ?** | \| \| | \| \| |

.....

Επίσης λίστες ελέγχου μπορούν να χρησιμοποιηθούν και για τους πελάτες και έτσι να έχουμε μια γενική άποψη για την γνώμη των πελατών για το προϊόν μας .

**→Διάγραμμα Pareto**

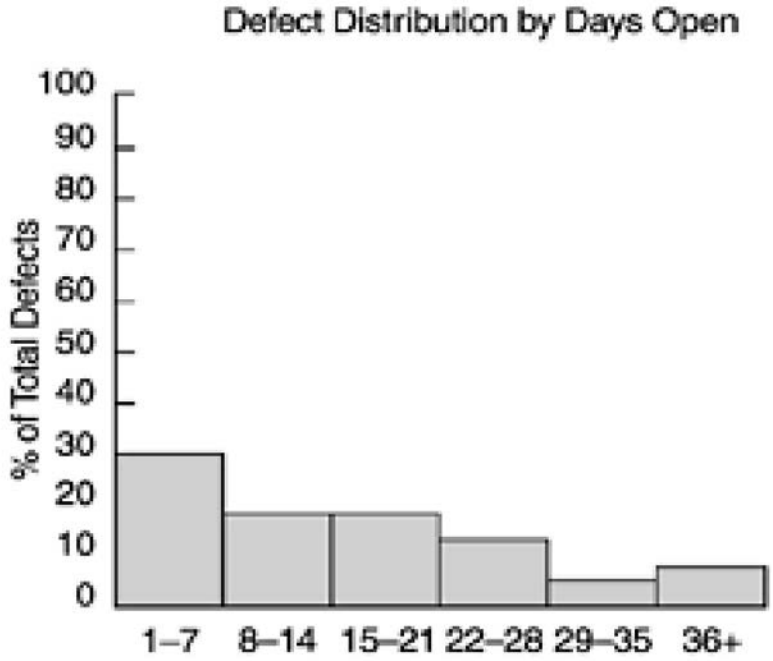
Η ανάλυση Pareto μας βοηθά να εντοπίσουμε περιοχές οι οποίες δημιουργούν τον μεγαλύτερο όγκο των προβλημάτων μας. Το επόμενο σχήμα μας βοηθά να κατανοήσουμε καλύτερα ένα διάγραμμα Pareto :



Όπου κάθε μπάρα θα μπορούσε να είναι κάποιο πρόβλημα , για παράδειγμα το κόκκινο να είναι «αργή εκκίνηση» το δεύτερο «πρόβλημα με βάση» κοκ .

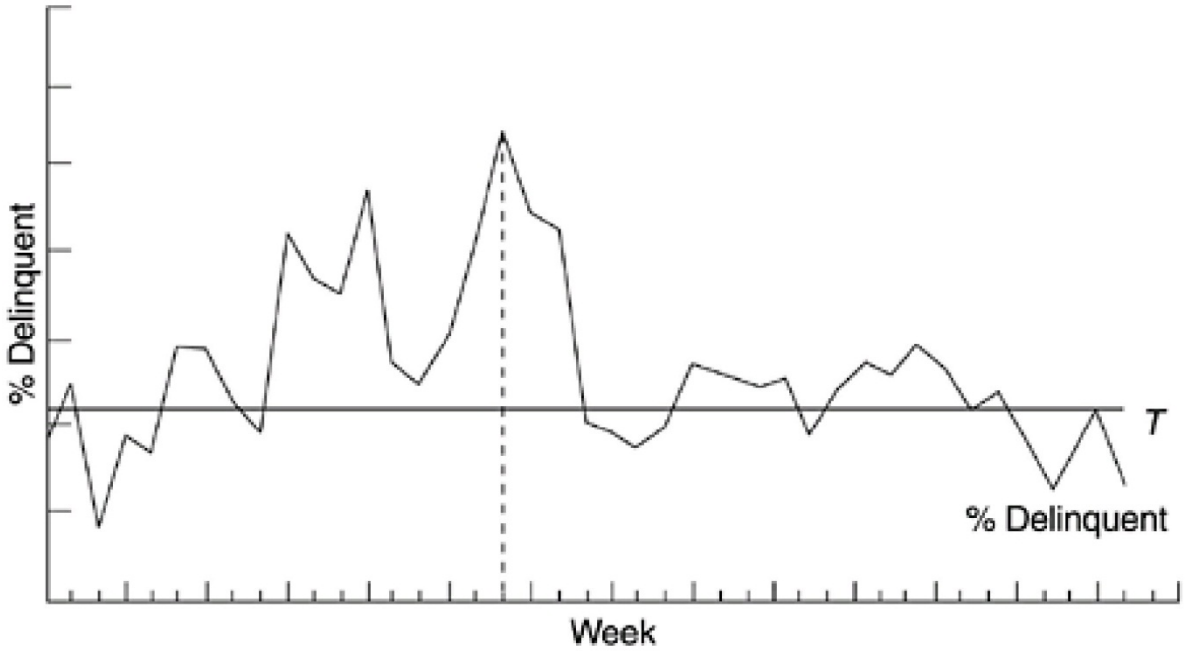
**→Ιστόγραμμα**

Το ιστόγραμμα χρησιμοποιείται κατά κόρο σε πολλές επιστήμες εκτός της μηχανικής λογισμικού .Για παράδειγμα στην ψηφιακή επεξεργασία εικόνας χρησιμοποιούμε ιστόγραμμα για να παρουσιάσουμε πόσες φορές εμφανίζεται κάποια απόχρωση . Στο παράδειγμα εδώ χρησιμοποιούμε το εργαλείο του ιστογράμματος για να παρουσιάσουμε πόσα προβλήματα εμφανίστηκαν από την κυκλοφορία του λογισμικού μας .



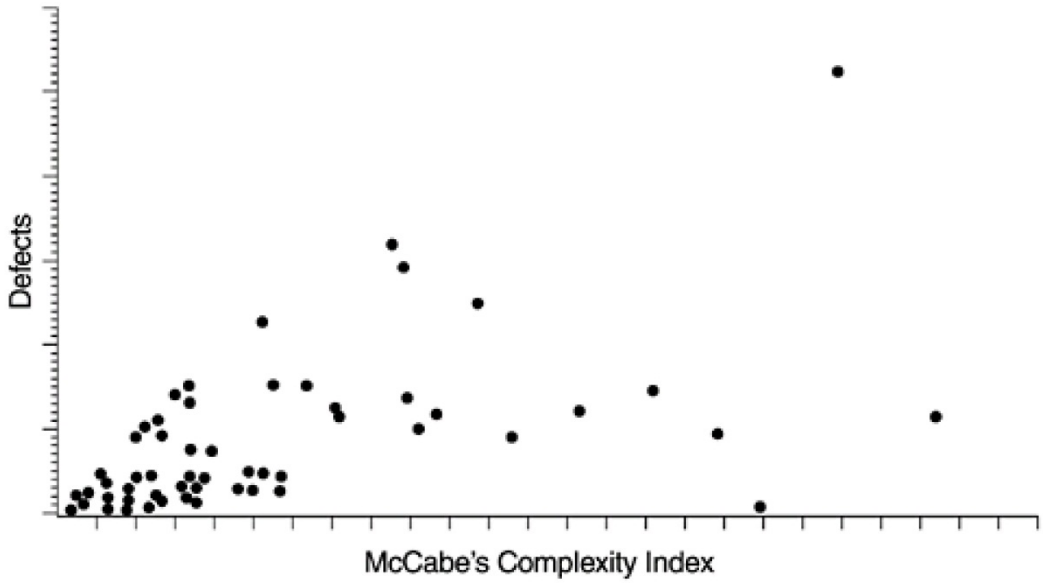
**→Χρονοδιαγράμματα**

Τα χρονοδιαγράμματα επίσης χρησιμοποιούνται πολύ στην μηχανική λογισμικού . Για παράδειγμα η εμφάνιση ελαττωμάτων ενός προγράμματος σε μία βδομάδα μπορεί να παρασταθεί μέσω χρονοδιαγράμματος :




**→Διαγράμματα διασποράς**

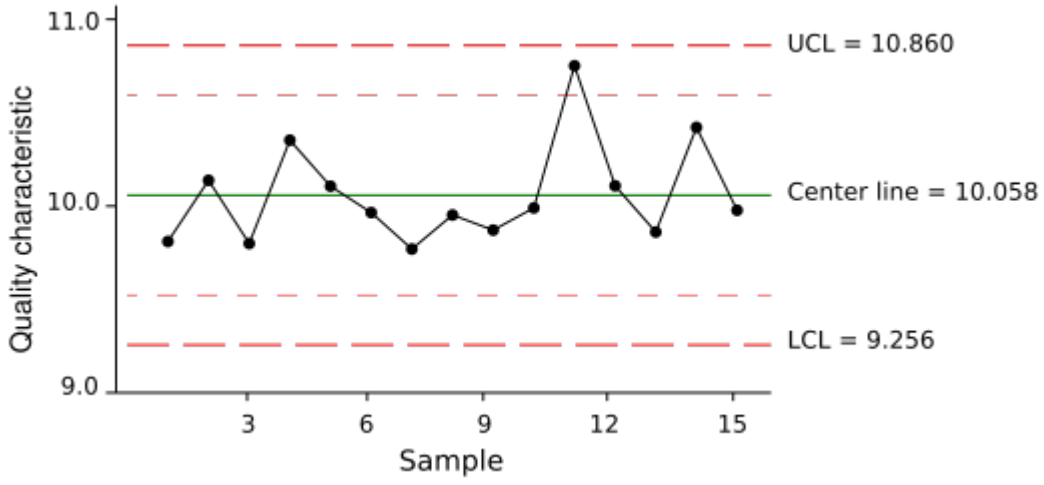
Αυτό το διάγραμμα έχει την πιο δύσκολη εφαρμογή από όλα τα άλλα . Ο σκοπός χρήσης του είναι για την επιτέλεση μιας έρευνας . Χρησιμοποιείται με τεχνικές παλινδρόμησης και στατιστική μοντελοποίηση .




Το παραπάνω διάγραμμα παρουσιάζει τα προβλήματα που προκύπτουν σε σχέση με το πόση πολυπλοκότητα εμπεριέχεται πίσω από το κάθε πρόβλημα .

**→ Διαγράμματα ελέγχου**

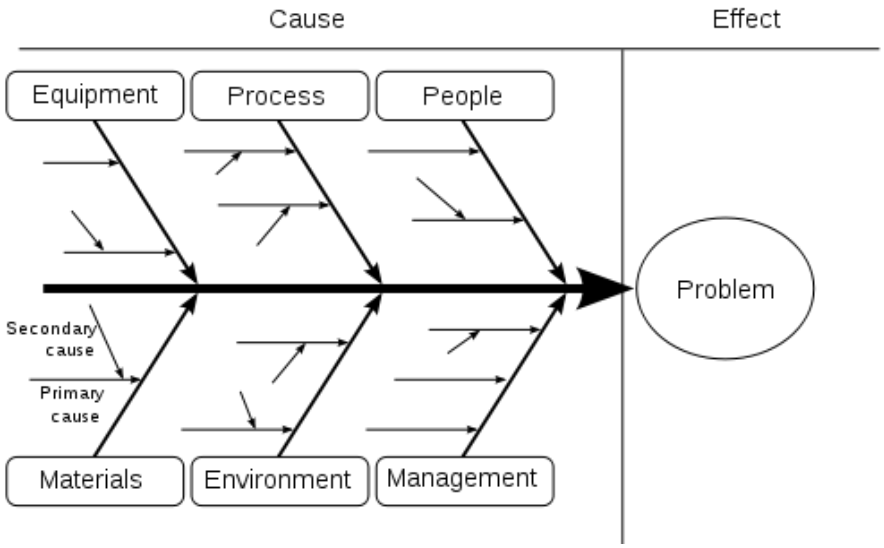
Είναι διαγράμματα τα οποία ελέγχουν την διαδικασία της παραγωγής χρησιμοποιώντας στατιστικά στοιχεία . Διαπιστώνουμε αν μία διαδικασία κινείται στον σωστό δρόμο και αν αυτή χρήζει προσοχής και ελέγχου .



Όπως φαίνεται και στο διάγραμμα πρέπει το δείγμα μας να κινείται ανάμεσα στις δύο κόκκινες γραμμές (UCL , LCL) που έχουμε θέσει ως όρια .

**→Διαγράμματα αιτίου-αποτελέσματος**

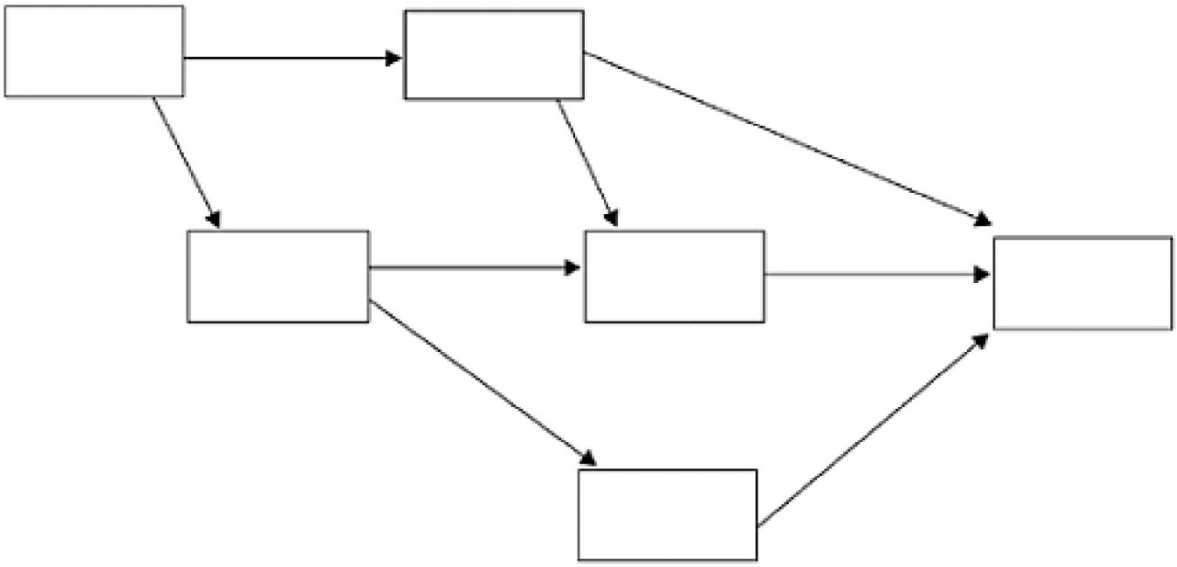
Αυτά τα διαγράμματα αν και δεν χρησιμοποιούνται πολύ από τις εταιρίες μηχανικής λογισμικού ωστόσο μας βοηθούν να αντλήσουμε σημαντικές πληροφορίες .




Το διαγράμματα αυτά σε περίπτωση που βρεθούν προβλήματα στην λειτουργία του λογισμικού (αποτέλεσμα) μας βοηθούν να ξεκαθαρίσουμε τι είναι αυτό που συμβάλλει πραγματικά στην δημιουργία του προβλήματος αυτού (αίτιο) .

**→ Σχεσιακά διαγράμματα**

Είναι ένα σημαντικό εργαλείο με το οποίο είναι εύκολο να παραστήσουμε πολύπλοκες σχέσεις αιτίου-αποτελέσματος . Οργανώνει την πληροφορία έτσι ώστε να κατανοήσουμε τα πλήρη αίτια και τα πλήρη αποτελέσματα . Διαφέρει από τα διαγράμματα αιτίου αποτελέσματος διότι αυτά είναι ικανά να παραστήσουν πολλαπλά αίτια και επιπτώσεις .



**Σημαντικά επίσης μέτρα αποτελούν και το μοντέλο Rayleigh και τα εκθετικά μέτρα .**

# 5-Συχνές μετρικές μέθοδοι λογισμικού

Πέρα από τα διαγράμματα και τις μεθόδους που μιλήσαμε πριν εδώ συγκεντρώνονται οι συχνές χρησιμοποιούμενες μέθοδοι για την μέτρηση της ποιότητας ενός λογισμικού.

- Σφάλματα ανά γραμμή κώδικα

Ο όρος σφάλμα αναφέρεται σε ένα λάθος που μπορεί να προκύψει κατά την διάρκεια εκτέλεσης του προγράμματος. Το σφάλμα μπορεί να προέλθει είτε από προγραμματιστικό λάθος είτε από λάθος χρήστη.

- Κάλυψη κώδικα

Αυτός ο όρος χρησιμοποιείται για δοκιμή λογισμικού (software testing).Περιγράφει τον βαθμό που ο πηγαίος κώδικας έχει δοκιμαστεί .

- Μέτρο συνοχής

Αναφέρετε στο πόσο στενά συνδεδεμένος είναι ο πηγαίος κώδικας με μια λειτουργία του λογισμικού . Έτσι έχουμε υψηλή και χαμηλή συνοχή ανάλογα με την περίπτωση . Προτιμάται η υψηλή συνοχή γιατί αυτή συνδέεται με την επαναχρησιμοποίηση της συγκεκριμένης λειτουργίας και την αξιοπιστία .

- Σύζευξη

Είναι ο βαθμός ο οποίος μετράει το πόσο εξάρτηση υπάρχει μεταξύ των ‘βιβλιοθηκών’ του προγράμματος , των λειτουργιών . Η χαμηλή σύζευξη προτιμάται σε αυτήν την περίπτωση και χαμηλή σύζευξη σημαίνει υψηλή συνοχή . Υψηλή σύζευξη σημαίνει χαμηλή συνοχή.

- Πολυπλοκότητα κυκλωμάτων

  Χρησιμοποιείται για να δείξει τον βαθμό πολυπλοκότητας του προγράμματος . Μετρά άμεσα τις γραμμές κώδικα που δουλεύουν ανεξάρτητες από άλλες εξαρτήσεις .

- Βαθμοί λειτουργικότητας

Υπολογίζονται με βάση τον αριθμό εισόδων του λογισμικού , τον αριθμό εξόδων ( δηλαδή τις εισόδους που βάζουμε στο σύστημα και τα αποτελέσματα που παίρνουμε) τον αριθμό ερωτημάτων ( είτε ερωτήματα σε μία βάση δεδομένων είτε άντληση πληροφοριών από αρχεία κοκ ) και τέλος τον αριθμό εσωτερικών λογικών αρχείων και τον αριθμό εξωτερικών λογικών αρχείων ( μπορεί κάποια αρχεία να υπάρχουν στον Η/Υ που τρέχει το λογισμικό αλλά μπορεί να υπάρχουν και έξω από αυτό παράδειγμα σε ένα σύστημα τράπεζας ) . Τα παραπάνω που αναφέρθηκαν παίρνουν έναν βαθμό ανάλογα με τη δυσκολία ( απλό , μέτριο , δύσκολο ) και προκύπτουν οι βαθμοί λειτουργικότητας .

- Χρόνος εκτέλεσης προγράμματος

Αυτό το μέγεθος διαφέρει ανάλογα με τον χρήστη του συστήματος. Για τον χρήστη αυτός ο χρόνος αναφέρετε στο πόσο χρόνο κάνει το πρόγραμμα να τρέξει και για τον προγραμματιστή στο πόση ώρα γίνεται build το πρόγραμμα .

## ΒΙΒΛΙΟΓΡΑΦΙΑ

1. The definition of software quality : A practical approach, (1999) Dr. Roland Petrasch.

2. The Goal Question Metric Approach, Victor R. Basili, Gianluigi Caldiera,
H.Dieter Rombach.

3. The Measurement of User Information Satisfaction, Blake Ives (Dartmouth College) , Margrethe H. Olson & Jack Baroudi (New York University),October 1983 Communications of the ACM.

4. http://en.wikipedia.org/wiki/Software_metric

5. Metrics And Models In Software Quality Engineering - Stephen H. Kan