ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΔΥΤΙΚΗΣ ΕΛΛΑΔΑΣ
ΣΧΟΛΗ ΔΙΟΙΚΗΣΗΣ ΚΑΙ ΟΙΚΟΝΟΜΙΑΣ
ΤΜΗΜΑ ΔΙΟΙΚΗΣΗΣ ΕΠΙΧΕΙΡΗΣΕΩΝ

Πτυχιακή Εργασία

Τεχνολογία Λογισμικού και Επιχειρησιακό Περιβάλλον

Πτυχιακή Εργασία των:

Κεφάλαιο 1: Αντώνης Ανδρέου

Κεφάλαιο 2: Δήμητρα Πλέγα

Κεφάλαιο 3: Νεκτάριος Νικολάου

Κεφάλαιο 4: Δημήτρης Στασινόπουλος

Επιβλέπων: κ.Καναβός Αντρέας

Πάτρα 2012

Περιεχόμενα

Κεφάλαιο 1: Γνωριμία με την Τεχνολογία του Λογισμικού

ΣΚΟΠΟΣ

Σκοπός του πρώτου κεφαλαίου είναι να αντιληφθεί ο αναγνώστης του τι είναι το λογισμικό. Ποια είναι η επίδραση του στον πραγματικό κόσμο και την βοήθεια που δίνει στην επιτέλεση των διαφόρων δραστηριοτήτων. Να μπορεί να εντοπίζει τα προβλήματα που στην ανάπτυξη του λογισμικού καθώς και να μπορέσει να εξηγήσει πώς προέκυψε η τεχνολογία λογισμικού.

Εισαγωγικές παρατηρήσεις

Το λογισμικό όπως θα γνωρίζουμε είναι ένα ανθρώπινο κατασκεύασμα χάρης στο οποίο ο άνθρωπος πέρασε στην εποχή της πληροφορικής και της επικοινωνίας στην οποία ζούμε σήμερα. Το λογισμικό αποτελεί ένα από τα πιο δύσκολα και πολυπλοκότερα δημιουργήματα του ανθρώπου. Η μελέτη του λογισμικού Η δημιουργία του δεν είναι μια απλή υπόθεση καθώς ο πραγματικός κόσμος μεταβάλετε συνεχώς δημιουργώντας νέες απαιτήσεις. Ένα λογισμικό πρέπει να ισορροπεί μεταξύ ποιότητας και οικονομικής εφικτότητας, καθώς πολλά από τα επιθυμητά του χαρακτηριστικά οφείλονται στην υπόσταση του ως προϊόντος . Η μελέτη του λογισμικού για κάποιον που θα ασχοληθεί με την τεχνολογία λογισμικού θα συνάντηση πλουραλισμό απόψεων για αυτό το προϊόν.

ΕΝΟΤΗΤΕΣ ΚΕΦΑΛΑΙΟΥ

1.2.ΗΛΕΚΤΡΟΝΙΚΟΣ ΥΠΟΛΟΓΙΣΤΗΣ-ΛΟΓΙΣΜΙΚΟ

Από τις σπουδαιότερες αν όχι η σπουδαιότερη εφεύρεση του 20ου αιώνα, του αιώνα που μας πέρασε ήταν η εφεύρεση του ηλεκτρονικού υπολογιστή. Η επιμονή του ανθρώπου να ανοίξει νέους ορίζοντες για ένα καλύτερο κόσμο, για ένα καλύτερο τρόπο ζωής οδήγησε στην εφεύρεση του ηλεκτρονικού υπολογιστή. Η πιο πάνω άποψη αντικατοπτρίζετε και δικαιολογείται στον καθρέφτη στον καθρέφτη της κοινωνίας καθώς ο ηλεκτρονικός υπολογιστής έχει ένα με την επιβίωση του ανθρώπου.

Από την στιγμή που εφευρέθηκε ο ηλεκτρονικός υπολογιστής από τον άνθρωπο η πρόοδος και ο ρυθμός ανάπτυξης του ήταν ραγδαίος και συνεχής με τον άνθρωπο να αναζητά συνεχώς την βελτίωση και την ανάπτυξη των χαρακτηριστικών και δυνατοτήτων του, και ήδη βρισκόμαστε στην 4η εποχή ηλεκτρονικών υπολογιστών.

Η είσοδος των ηλεκτρονικών υπολογιστών στις πλείστες βαθμίδες της δημόσιας εκπαίδευσης μας δείχνει ότι ο Η/Υ έχει για τα καλά στην ζωή μας.

Η βοήθεια που μας παρέχει ο Η/Υ είναι τεράστια. Εργασίες όπου στο παρελθόν πάρα πολύ χρονοβόρες και χρειαζόταν απόλυτη συγκέντρωση και μεγάλα αποθέματα ενέργειας για να επιτελεστούν, ακόμα και εργασίες που ήταν αδύνατον να πραγματοποιηθούν, στην μετά-Η/Υ εποχή όχι μόνο έχουν αυτοματοποιηθεί αλλά έχουν εξαλειφθεί και όλες οι πιθανότητες λάθους. Η εκτέλεση των διαφόρων εργασιών είτε αυτές λέγονται επαγγελματικές είτε λέγονται ιδιωτικές έχουν ως κύριο χαρακτηριστικό για την διεκπεραίωση τους την χρήση του Η/Υ.

Στην μεταβατική εποχή που ζούμε ο Η/Υ έχει σκεπάσει και τις πλευρές τις ανθρώπινης ζωής και επιβίωσης πλέον τις περισσότερες φορές χωρίς να γίνετε και αντιληπτός. Τα μέσα μαζικής μεταφοράς, τα αεροπλάνα, τα πλοία ,οι υπηρεσίες όπως έλεγχος οδικής ασφάλειας και εναέριας κυκλοφορίας περνά μέσα από την χρήση του Η/Υ.

Για τον Η/Υ έχουν δοθεί διάφοροι ορισμοί. Ένας αποδεκτός ορισμός είναι:

Ο Ηλεκτρονικός Υπολογιστής (Η/Υ) είναι ένα σύνολο ηλεκτρονικών-ηλεκτρικών συσκευών που μας δίνει τη δυνατότητα να εκτελούμε διάφορες εργασίες (δακτυλογράφηση, ταξινόμηση, ζωγραφική, σχέδιο, υπολογισμοί, επικοινωνία κ.ά.) που χρειαζόμαστε στις καθημερινές μας δραστηριότητες. Η σημαντική αλλαγή που επέφερε η χρήση του Η/Υστην οργάνωση της καθημερινής μας ζωής είναι ότι όλες οι εργασίες και δραστηριότητές μας εκτελούνται με μεγάλη ταχύτητα και ακρίβεια.

Σας παρουσιάζω συνοπτικά τις 4 εποχές Η/Υ

- Η Πρώτη εποχή ήταν η εποχή του τύπου Main Frame των μεγάλων υπολογιστικών συστημάτων
- Η Δεύτερη εποχή στιγματίστηκε από μικρότερα υπολογιστικά συστήματα που χρησιμοποιούνταν για επαγγελματική χρήση μόνο.

- Η Τρίτη εποχή είναι η εποχή των προσωπικών υπολογιστών οι οποίοι μπήκαν με γοργούς ρυθμούς τόσο στην εργασία όσο και στο σπίτι μας όσο και στην καθημερινότητα μας.

- Η Τέταρτη εποχή είναι η εποχή της δικτύωσης και της διασύνδεσης όλων των υπολογιστικών συστημάτων σε παγκόσμιο επίπεδο.

Όπως έχουμε δει έχουμε περάσει στην 4η εποχή Η/Υ στην εποχή που όλοι οι Η/Υ θα είναι συνδεδεμένοι με το δίκτυο και θα εκτελούν σύνθετες εργασίες και σύνθετες εφαρμογές. Εφαρμογές και εργασίες που έχουν αντίκτυπο από τον ίδιο μας τον πολιτισμό μέχρι και την λειτουργία του δημοκρατικού πολιτεύματος. Έχουμε περάσει πλέον στην ψηφιακή εποχή και όπως αντιλαμβανόμαστε σαν κοινωνία πρέπει να τροποποιήσουμε τόσο τις κοινωνικές και πολιτικές σχέσεις όσο και τις οικονομικές σχέσεις μας για να προσαρμοστούμε στην νέα εποχή του μετασχηματισμού της κοινωνίας. Την νέα αυτή κατάσταση που έχει διαμορφωθεί την ονομάζουμε ως κοινωνία της πληροφορίας ή ως κοινωνία δικτύων, φαινόμενο το οποίο έχει πολύ δυναμικό χαρακτήρα και εξελίσσετε συνεχώς από την διείσδυση της τεχνολογίας, με το δίκτυο να έχει καίρια θέση στη όλη κατάσταση.

Όπως έχουμε δει ο ηλεκτρονικός υπολογιστής είναι το μεταφορικό μέσο που μας έχει φέρει στη εποχή που ζούμε σημερα.2 ερωτήματα που θα μπορούσε ένας αρχάριος να κάνει διαβάζοντας όσα έχω αναφέρει είναι:

1)Μπορούσε όμως ο Η/Υ από μόνος του σαν μια ηλεκτρονική-ηλεκτρικη συσκευή να εξαπλωθεί σε όλες σχεδόν τις πτυχές της καθημερινότητας μας;

2)Μπορούσε όλες αυτές οι εξελίξεις να τις έχει επιτύχει από μόνη της μια συσκευή;

Ο Η/Υ σαν μια ηλεκτρονική-ηλεκτρικη συσκευή μπορεί να μόνο να επιτελεί κάποιες λειτουργίες με υψηλότατη ταχύτητα και με τρόπο ο οποίος είναι ιδιαίτερα δύσκολος να το προσεγγίσει ο άνθρωπος. Όλες αυτές οι εξελίξεις στο παγκόσμιο στερέωμα που επηρεάζουν ποικίλες πτυχές ζωής μας έγιναν, γίνονται και θα γίνονται δυνατές λόγω της ενσωμάτωσης στον Η/Υ του λογισμικού, που εκτελεί ένα σύνολο δυσεπίλυτων εφαρμογών.

Θα μπορούσαμε να πούμε ότι το λογισμικό είναι ο τρόπος συμπεριφοράς του Η/Υ. Το λογισμικό προσδίδει <<χαρακτήρα>> στον Η/Υ και καταντά τον Η/Υ χρήσιμο για τον άνθρωπο, διότι ο άνθρωπος τον αξιοποιεί τον Η/Υ μόνο σαν λογισμικό.

Για να γίνει πιο κατανοητό αυτό που ανάφερα ας σκεφτούμε ένα στεγνωτήριο ρούχων. Χωρίς το λογισμικό το στεγνωτήριο θα ήταν απλά ένα τετράγωνο κουτί. Ο άνθρωπος αξιοποιεί το συγκεκριμένο μηχάνημα μέσω του λογισμικού του για να στεγνώσει προφανώς τα ρούχα του.

Λαμβάνοντας τα προαναφερθέντα καταλαβαίνει αμέσως κάποιος ότι είναι δύσκολο να δοθεί ένας συγκεκριμένος και κοινά αποδεκτός ορισμός για την έννοια του λογισμικού. Με το πέρασμα των χρόνων ένας θεωρητικά ορισμός για την έννοια του λογισμικού μπορεί με πολύ ευκολία να αμφισβητηθεί.

Ένας αποδεκτός ορισμός για την έννοια του λογισμικού που μπορούμε να αποδεχτούμε για αυτήν την διπλωματική εργασία είναι:

<<Το λογισμικό είναι ένα σύνολο από εντολές, τα όποια είναι προγράμματα του ηλεκτρονικού υπολογιστή τα οποία υπαγορεύουν στον υπολογιστή το τρόπο με τον οποίο θα επεξεργαστεί τα δεδομένα που του δίνουμε για να μας επιστρέψει τις επιθυμητές πληροφορίες η αποτελέσματα που θέλουμε>>.

1.3.ΤΕΧΝΙΚΕΣ ΚΑΤΑΣΚΕΥΕΣ ΚΑΙ ΛΟΓΙΣΜΙΚΟ

Το λογισμικό όπως γνωρίζουμε είναι ένα τεχνικό κατασκεύασμα με ένα σύνολο από εντολές που υπαγορεύει στον ηλεκτρονικό υπολογιστή με ποιο τρόπο να ενεργεί ούτως ώστε να παράγει δεδομένα έπειτα από την εκτέλεση κάποιων ενεργειών. Το λογισμικό δεν είναι ένα σύστημα που μπορεί από μόνο του να λειτουργήσει χωρίς να εξαρτάτε από κάτι άλλο δηλαδή δεν είναι αυτοτελή.

Το λογισμικό σε σχέση με τις διάφορες τεχνικές κατασκευές παρουσιάζει πολλές ομοιότητες αλλά και πολλές διαφορές τις οποίες θα δούμε παρακάτω.

ΟΜΟΙΟΤΗΤΕΣ

1) Στις τεχνικές κατασκευές όπως και στο λογισμικό συντάσσετε ένα λεπτομερές σχέδιο προκειμένου να καταγραφούν οι φάσεις της κατασκευής και με ποια σειρά.
2) Και στις δύο περιπτώσεις πριν αρχίσει η διαδικασία κατασκευής τους γίνετε ένας προϋπολογισμός σχετικά με το κόστος του έργου.
3) Παρόμοιος υπολογίζετε και ο χρόνος ο οποίος θα χρειαστεί για να διεκπεραιωθεί το έργο με συγκεκριμένα περιοριστικά χρονοδιαγράμματα.
4) Η κατασκευή γίνετε σε διακριτές φάσεις οι οποίες έχουν σχέση μεταξύ τους.
5) Η σχεδιασμένη τοποθέτηση και άλληλα σύνδεση των επιμέρους στοιχείων επηρεάζουν την συνολική συμπεριφορά και εικόνα προς τους χρήστες.
6) Οι απαιτήσεις που έχουν από το έργο οι χρήστες τίθενται αποκλειστικά από αυτούς.

ΔΙΑΦΟΡΕΣ

1. Η φύση του λογισμικού δεν γίνετε αντιληπτή με τις αισθήσεις μας, μόνο τα αποτελέσματα της χρήσης τα οποία παράγει γίνονται αντιληπτά σε αντίθεση με τις τεχνικές κατασκευές που γίνονται ορατές και αντιληπτές. Η δομή του λογισμικού σε μακροσκοπικό και μικροσκοπικό επίπεδο είναι ένα κατασκεύασμα το οποίο συλλαμβάνετε από το μυαλό του ανθρώπου και γίνετε αντιληπτό με ποικίλους τρόπους.
2. Η σταθερότητα και η ισορροπία μιας τεχνικής κατασκευής και η διάρκεια της στο χρόνο μπορεί να απειληθεί μόνο από φυσικά φαινόμενα(ανεμοστρόβιλοι, τυφώνες, ισχυρές βροχοπτώσεις κτλ) ενώ στο λογισμικό απειλείτε μόνο από τη φυσιολογική χρήση.
3. Στις τεχνικές κατασκευές η πορεία κατασκευής ενός έργου ακολουθεί μια σειρά η οποία είναι καθορισμένη και γνωστή από την αρχή και σπανίως ή και ποτέ δεν αλλάζει. Στην κατασκευή του λογισμικού υπάρχει μια πολυπλοκότητα καθώς τα πάντα αλλάζουν με ραγδαίους ρυθμούς και η αρχική του προκαθορισμένη διαδικασία κατασκευής αλλάζει όπως

και οι προδιαγραφές. Σαν μια παρομοίωση θα μπορούσαμε να πούμε ότι μοιάζει <<σκόπευση κινούμενου στόχου από κινούμενο έδαφος και με όπλο που αλλάζει συνεχώς την συμπεριφορά του>>.

a. Οι απαιτήσεις των χρηστών μεταβάλλονται συχνά σε αντίθεση με τις τεχνικές κατασκευές που σπανίως μεταβάλλονται, που συνεπάγετε ότι αλλάζει και ο στόχος για να ικανοποιήσει τις απαιτήσεις αυτές. Οι απαιτήσεις αυτές μπορεί να αλλάξουν και κατά την διάρκεια της διαδικασίας η οποία προορίζετε να τις ικανοποιήσει όποτε αλλάζει και η διαδικασία κατασκευής σε αντίθεση με τις τεχνικές κατασκευές που αφού δεν παρατηρείτε καμία μεταβολή συνεχίζετε η κατασκευή με το αρχικό πλάνο.
b. Οι μεθοδολογίες τα εργαλεία και το περιβάλλον ανάπτυξης μεταβάλλονται συνεχώς και έτσι χρειάζεται να μεταβληθούν και αυτά που χρησιμοποιούνται για την διαδικασία ανάπτυξης σε αντίθεση με τις τεχνικές κατασκευές που δεν παρατηρείτε τέτοια μεταβλητότητα.
c. Στις τεχνητές κατασκευές το έδαφος το οποίο γίνετε η κατασκευή δεν μεταβάλλετε σε αντίθεση με του λογισμικού όπου υπάρχει ραγδαία μεταβλητότητα γιατί το περιβάλλον ανάπτυξης εξελίσσετε.

Όπως βλέπουμε οι διαφορές του λογισμικού με άλλες τεχνικές κατασκευές είναι πολύ μεγάλες. Η πρόοδος και ανάπτυξη της τεχνολογίας πολλές φορές μας κάνει να αναθεωρούμε απόψεις μας. Ο ανταγωνισμός για το μεγαλύτερο μερίδιο αγοράς στα διάφορα επιχειρηματικά πεδία φέρνει την εξέλιξη των διαφόρων ειδών της τεχνολογίας με αποτέλεσμα η κατασκευή ενός λογισμικού που να έχει διάρκεια στο χρόνο να είναι προς το παρόν αδύνατη. Τα προϊόντα τεχνολογίας αναπτύσσονται με ρυθμό οποίος είναι ταχύτατος και ορμητικός και επιβάλλει στις αγορές να τα χρησιμοποιήσουν για να είναι ανταγωνιστικές.

Παρότι για το λογισμικό γίνονται προσπάθειες ούτος ώστε να αυτοματοποιηθεί και να δοθεί ένας τύπος που μπορεί να συνάδει με την ανάπτυξη και να ικανοποιεί τις διάφορες πτυχές της καθημερινής και επιχειρηματικής ζωής αυτό δεν έχει επιτευχθεί στις μέρες μας, και το λογισμικό παραμένει στις μέρες το μοναδικό ανθρώπινο κατασκεύασμα του οποίου η εξέλιξη δεν έχει ορατό τέλος.

1.4.ΚΡΙΣΗ ΛΟΓΙΣΜΙΚΟΥ

Από τον περασμένο αιώνα που εφευρέθηκε ο Η/Υ όπως γνωρίζουμε έχει σημειώσει μεγάλη ανάπτυξη. Όπως έχουμε τονίσει σε προηγούμενη αναφορά μας οι Η/Υ για να καταστούν χρήσιμοι και αποδοτικοί για τον άνθρωπο πρέπει να έχουν στην συσκευή τους ένα σύνολο πολύπλοκων εφαρμογών λογισμικού.

Ένα ερώτημα που πηγάζει από αυτή την αναφορά είναι:

Τι γίνετε στην περίπτωση όπου ο ρυθμός ανάπτυξης των Η/Υ έχει ταχύτητα πολύ μεγαλύτερη από τον ρυθμό ανάπτυξης του λογισμικού;

Από εδώ πηγάζει και ο όρος <<ΚΡΙΣΗ ΛΟΓΙΣΜΙΚΟΥ>>. Ο όρος αυτός εισήχθηκε από τον Φ.Λ. BAYER στην πρώτη διάσκεψη τεχνολογίας λογισμικού του ΝΑΤΟ στην Γερμανία. Ο όρος κρίση λογισμικού μας προσδιορίζει τα προβλήματα και τα στίγματα που αφήνει η ραγδαία αύξηση της δύναμης των Η/Υ.

Ο όρος αυτός χρησιμοποιήθηκε από τις πρώτες μέρες της τεχνολογίας του λογισμικού χωρίς να είναι τόσο μεγάλο ζήτημα και γίνει ένα καθιερωμένο θέμα και να εξελιχθεί στο τέλος σε μια χρόνια πάθηση που μας ταλαιπωρεί εδώ και σχεδόν μισό αιώνα. Η παράληψη της σπουδαιότητας του λογισμικού στα πρώτα χρόνια της πληροφορικής που έφερε σαν αποτέλεσμα την λανθασμένη νοοτροπία που ακολουθούμε από τότε, οδήγησαν στην κρίση του λογισμικού. Η συνεχής και μεγάλες απαιτήσεις για καινούργια προγράμματα και η λανθασμένη σχεδίαση τους, οδηγούν τις εταιρείες παραγωγής λογισμικού σε αποτυχημένα έργα λογισμικού. Ένα χαρακτηριστικό το οποίο αποτυπώνετε στο λογισμικό είναι ότι είναι ένα σύνολο πολύπλοκων εφαρμογών το οποίο πωλείτε ως έχει χωρίς εγγύηση για τυχόν ζημιές που μπορεί να προκαλέσει.

Κάνοντας την μελέτη μου πάνω στην κρίση του λογισμικού κατέληξα στο συμπέρασμα ότι ο κυριότερος λόγος που οδηγεί στην αποτυχία εξασφάλισης ποιότητας λογισμικού είναι ότι υπάρχει ένα τεράστιο κενό στην έλλειψη αυστηρά καθορισμένων μετρήσιμων στόχων και διαδικασιών μέτρησης. Για παράδειγμα ας σκεφτούμε ένα βιομηχανικό προϊόν το οποίο χρησιμοποιούμε συχνά την καθημερινή μας ζωή όπως ένα κουτάκι αναψυκτικού. Για το συγκεκριμένο προϊόν μπορούμε να ορίσουμε τιμές, να μετρήσουμε το εμβαδόν, την χωρητικότητα του, το ύψος του κτλ. Αυτήν την εργασία δεν μπορούμε να την κάνουμε και στα προϊόντα λογισμικού, στοιχείο το οποίο συνεπάγετε με παραγωγή προϊόντων λογισμικού κακής ποιότητας, χαμηλής αξιοπιστίας και χαμηλής λειτουργικότητας.

Παρακάτω αποτυπώνονται τα βασικότερα σημεία της κρίσης του λογισμικού:

- Δύσκολη διαδικασία κατασκευής: Τις περισσότερες φορές διακρίνετε ασάφεια ως προς τα ποια είναι τα βήματα τα οποία θα πρέπει να ακολουθηθούν και με ποια σειρά ακόμα και για τα ενδιάμεσα βήματα.
- Μεγάλο κόστος ανάπτυξης και συντήρησης: ο λόγος κόστους του λογισμικού έναντι του υλικού αυξάνετε με μεγάλες ταχύτητες. Αν πάρουμε την δεκαετία του 60 όπου ο συγκεκριμένος λόγος έχει αυξηθεί κατά 20% περίπου στην δεκαετία την σημερινή ο λόγος αυτός έχει σημειώσει την ραγδαία αύξηση που φθάνει το μέχρι και το 90%.Μεγαλης

προσοχής πρέπει να τύχη ότι από το συνολικό κόστος του λογισμικού ποσοστό που αρχίζει από το 50% και μπορεί να φθάσει το αστρονομικό 90% διατίθεται αποκλειστικά για την συντήρηση του.

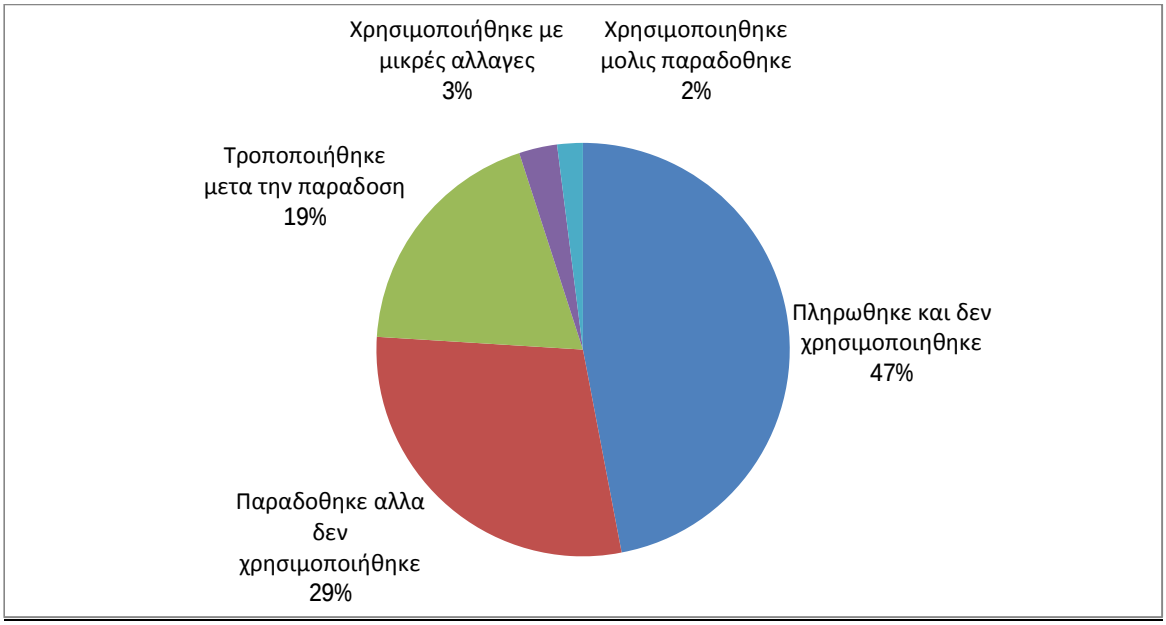
- Ø Ελλείπεις και κακή ποιότητα τελικού προϊόντος: Τα λάθη που γίνονται στην κατασκευή του λογισμικού και φέρνουν την αναποτελεσματικότητα και την μη ικανοποίηση των απαιτήσεων του χρήστη, προβλήματα το οποίο αυξάνετε με αμείωτους ρυθμούς οι ανάγκες των χρηστών μεταβάλλονται που συνεπάγετε και ίδια μεταβολή των αναγκών τους.
- Ø Ελάχιστη ή και καμία προσαρμογή σε μεταβαλλόμενες απαιτήσεις.
- Ø Δυσκολία στην επικοινωνία μεταξύ του χρήστη και του κατασκευαστή.
- Ø Ανακάλυψη σφαλμάτων μετά την παράδοση του προϊόντος
- Ø Υπερβάσεις προϋπολογισμών: Γίνονται τις περισσότερες φορές κακές εκτίμησης για το τελικό κόστος και πωλούνται με μεγαλύτερο κόστος από ότι είχε αρχικά προβλεφτεί.
- Ø Υπέρβαση χρονοδιαγραμμάτων: Κακή εκτίμηση του χρόνου και αδικαιολόγητες καθυστερήσεις που μερικές φορές είναι υπερβολικές.
- Ø Δύσκολη κατανόηση από τρίτους: Πρακτικά πλην του ίδιου του κατασκευαστή η κατανόηση του λογισμικού από τρίτους είναι δύσκολη έως αδύνατη.

Στοιχειά έρευνας έδειξαν εντυπωσιακά αποτελέσματα σε σχέση με τη χρήση του λογισμικού. Το πιο εντυπωσιακό είναι ότι λογισμικό το οποίο παραδόθηκε σε χρήστες και το χρησιμοποίησαν ήταν μόνο με 2% ενώ το ποσοστό αυτό δεν σημείωσε αισθητή άνοδο ακόμα και μετά την έλευση των μικρών φθάνοντας μόνο στο 5%.Ειναι προφανές ότι το συντριπτικό μέρος των λογισμικών ήταν αποτυχημένα.

Από στοιχεία του 1979, από έργα 6.8 εκατομμυρίων δολαρίων συλλέχθηκαν:

- v 47% πληρώθηκε αλλά δεν παραδόθηκε προς χρήση
- v 29% παραδόθηκε αλλά δεν χρησιμοποιήθηκε
- v 19% τροποποιήθηκε μετά την παράδοση
- v 3% χρησιμοποιήθηκε με μικρές αλλαγές
- v 2% χρησιμοποιήθηκε όπως παραδόθηκε

ΣΧΗΜΑ



Πιο κάτω αποτυπώνονται μερικά περιστατικά που δείχνουν το πρόβλημα που έχει το λογισμικό σήμερα. Περιπτώσεις επιχειρήσεων που όχι μόνο εγκατέλειψαν την προσπάθεια αλλά χρεοκόπησαν μέχρι και θάνατο ασθενών λόγο της αναποτελεσματικότητας του λογισμικού.

Παραδείγματα:

- Αρχές του 1980 το IRS ανάθεσε στην Sperry Corp την ανάπτυξη ενός συστήματος για την αυτόματη επεξεργασία φόρων. Από διορθώσεις στο λογισμικό μετά από περίπου μια πενταετία είχε ως αποτέλεσμα ήταν να διπλασιαστεί ο αρχικός προϋπολογισμός του και το σύστημα να μην βελτιωθεί αισθητά.
- Στις αρχές του 1990 το σύστημα ελέγχου ακτινοβόλησης ασθενών Therac-25 προκάλεσε τον θάνατο πολλών ασθενών από υπερβολικές δόσεις ακτινοβολίας.
- Τον Ιούνιο του 1996 καταστράφηκε ο πύραυλος Arian-5 μετά από εκτροπή στην πορεία του λόγο σφάλματος του λογισμικού στις προδιαγραφές ζημία που κόστισε περίπου 500 εκατομμύρια δολάρια.
- Το 2000 η Kmart Corp για τον εκσυγχρονισμό του IT δαπάνησε περίπου 1,4 δισεκατομμύρια δολάρια με αποτέλεσμα να χρεοκοπήσει.
- Το 2004 ηUK Revenue από σφάλματα στο λογισμικό της υποχρεώθηκε να πληρώσει 3,45 δισεκατομμύρια δολάρια για υπέρ-πληρωμές φόρων.

Παρακάτω εντοπίζονται οι παράγοντες της αποτυχίας του λογισμικού:

Απαιτήσεις: Το προϊόν το οποίο παραδίδετε στους χρήστες κατά μεγάλο ποσοστό δεν τόσο είναι και πολύ χρήσιμο.

Διαχείριση έργου: Όταν η διαχείριση του έργου είναι αναποτελεσματική έχει ως αποτέλεσμα να καθυστερεί και να αυξάνετε ο προϋπολογισμός.

Ποιότητα: Όταν το έργο παραδίδετε με σφάλματα και προβλήματα στις υπηρεσίες του οι χρήστες δεν μπορούν να το χρησιμοποιήσουν και κρίνετε αναποτελεσματικό.

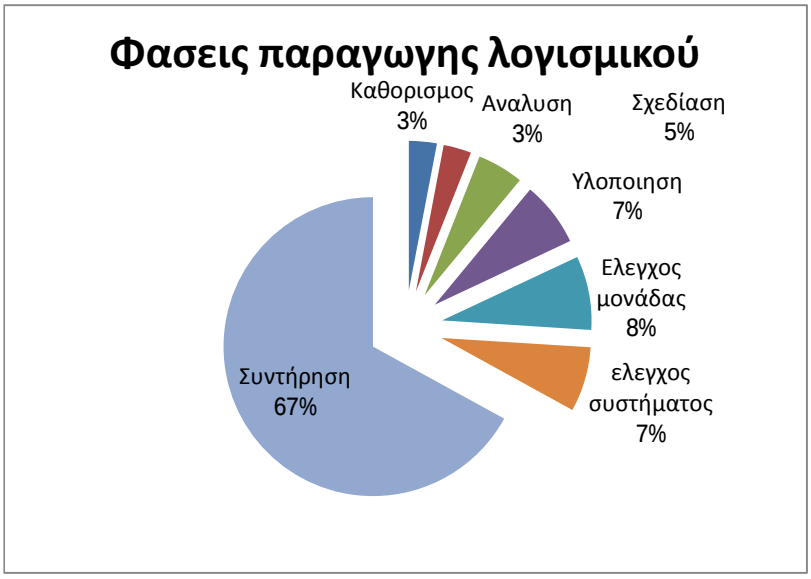
ΑΚΟΜΑ

Εξωτερικές συνθήκες: Οι συνθήκες όπως η οικονομική κρίση που βιώνουμε και οι μεταβαλλόμενες απαιτήσεις της αγοράς επιδρούν αρνητικά στο έργο

Οι φάσεις του κύκλου ζωής τους του λογισμικού διέπονται από 7 στάδια:

1. Τον καθορισμό
2. Την ανάλυση
3. Την σχεδίαση
4. Την υλοποίηση
5. Τον έλεγχο της μονάδας
6. Τον έλεγχο συστήματος
7. Την συντήρηση

Στις φάσεις του κύκλου ζωής του προϊόντος όπως παρατηρούμε στο παρακάτω σχήμα ότι είναι αναγκαία να δίνετε μια μεγάλη προσπάθεια για την διορθωτική συντήρηση που πηγάζει από την έλλειψη μιας συγκεκριμένης μεθοδολογίας που οδηγούν στα λάθη και στις παραλήψεις.



Τα προβλήματα αυτά δεν είναι ανακαλύφθηκαν στην σημερινή εποχή που ζούμε. Προϋπήρχαν από παλαιοτέρα, από την πρώτη εποχή ηλεκτρονικών υπολογιστών και έγιναν εντονότερα στην δεκαετία του 70 αφού τα πληροφοριακά συστήματα έγιναν μεγαλύτερα και πολύπλοκα το υλικό αυξανόταν με το ίδιο ρυθμό επιτρέποντας την εκτέλεση τέτοιων προγραμμάτων ομοίως μεταβαλλόταν και οι απαιτήσεις των χρηστών με ραγδαία ταχύτητα.

Όπως προαναφέραμε ο κύριος λόγος στην αποτυχία εξασφάλισης ποιότητας λογισμικού είναι η μη εξασφάλιση μετρίσιμων στόχων και διαδικασιών μέτρησης καθώς για τα χαρακτηριστικά που επιθυμούμε να μετρήσουμε δεν υπάρχει δυνατότητα μέτρησης.

Σαν προτεινόμενες λύσεις προτάθηκαν στην διάσκεψη του ΝΑΤΟ το 1968 οι εξής:

- Ομαδοποίηση ανθρώπινων πόρων
- Ενδελεχής ανάλυση του προβλήματος για να αντιμετωπιστεί η πολυπλοκότητα
- Έμφαση στην συντήρηση

Η ορθή στρατηγική που ο DE MARKO προτείνει στις εταιρίες παραγωγής λογισμικών προϊόντων πρέπει να περιλαμβάνει τα εξής στοιχεία:

- Να υλοποιούν πολύ λιγότερο κώδικα
- Να επιλέγουν πολύ προσεκτικότερα τι υλοποιούν
- Να θέτουν ελαστικότερα όρια ολοκλήρωσης και παράδοσης των έργων τους

Τα τελευταία χρόνια έχουν γίνει μεγάλες προσπάθειες ούτως ώστε να αποκρυπτογραφηθεί η αδυναμία αυτή. Έχουν δημιουργηθεί διεθνή πρότυπα τα οποία περιλαμβάνουν κανόνες οι οποίοι θα πρέπει να τηρηθούν από τις εταιρίες λογισμικού χωρίς όμως να υπαγορεύουν συγκεκριμένες λύσεις ως προς τον καθορισμό μετρίσιμων στόχων και διαδικασιών μέτρησης.

ΜΕΡΙΚΑ ΔΙΕΘΝΗ ΠΡΟΤΥΠΑ ΚΑΙ ΒΡΑΒΕΙΑ:

- ISO9000 (ISO00)
- IEEE (IEEE89)
- BALDRIGE (BRO91) (STE93)
- CAPABILITY MATURITY MODEL (CMM) (PAU93)
- CAPABILITY MATURITY MODEL INTERGRATION (CMMI) (AHE04)

Για τον σκοπό αυτό αναπτύχτηκε ο ειδικός κλάδος της επιστήμης της πληροφορικής με την ονομασία <<ΤΕΧΝΟΛΟΓΙΑ ΛΟΓΙΣΜΙΚΟΥ>>, κλάδος ο οποίος αναπτύχτηκε έπειτα από εισήγηση του συνεδρίου με κύριο θέμα την κρίση λογισμικού το 1968 στις ΗΠΑ, σαν αποτέλεσμα όλων αυτών των προσπαθειών που γίνονται για την ποιότητα λογισμικού να αντεπεξέλθει στις απαιτήσεις των χρηστών και της νέας εποχής, το οποίο θα δούμε αμέσως μετά.

1.5.ΤΕΧΝΟΛΟΓΙΑ ΛΟΓΙΣΜΙΚΟΥ

ΟΡΙΣΜΟΣ

Τομέας που πραγματεύεται *τεχνικές*, μεθοδολογίες, πρακτικές και εργαλεία για την συστηματική, μεθοδική και ποσοτικοποιημένη προδιαγραφή, σχεδίαση, υλοποίηση, έλεγχο, και συντήρηση συστημάτων λογισμικού υψηλής ποιότητας και εντός δεδομένου προϋπολογισμού και χρόνου εκτέλεσης.

Τεχνική – Μεθοδολογία – Εργαλεία

Τεχνική (methods): Φορμαλιστικές διαδικασίες για την επίλυση προβλημάτων με τη χρήση καλώς ορισμένων συστημάτων παρουσίασης και κωδικοποίησης

Μεθοδολογία: Συλλογή από τεχνικές που εφαρμόζονται επιλεκτικά κατά την διάρκεια όλων των φάσεων ενός έργου και συνδυάζονται σύμφωνα με κάποια γενική πρακτική και πλάνο και καθορίζει πως πρέπει να εκτελούνται οι διαδικασίες ανάπτυξης δηλαδή

- Ποιες επιμέρους ενέργειες περιλαμβάνουν
- Ποια βήματα έχει κάθε διαδικασία
- Ποια προϊόντα παράγονται
- Πότε ολοκληρώνεται κάθε διαδικασία

Εργαλεία: Αυτοματοποιημένα συστήματα που διεκπεραιώνουν μια τεχνική και υποστηρίζουν την εφαρμογή των μεθοδολογιών.

Η τεχνολογία λογισμικού δεν ασχολείται μόνο με την κατασκευή ενός προϊόντος λογισμικού αλλά με ολόκληρο τον κύκλο ζωής του, δηλαδή από την στιγμή που συλλαμβάνετε η ιδέα για την κατασκευή μιας εφαρμογής, στην διαδικασία κατασκευής της εφαρμογής και χρησιμοποίησης της έως ότου αποσυρθεί λόγο παλαιότητας ή ακαταλληλότητας και αντικατασταθεί από άλλη εφαρμογή με πιο πολλές δυνατότητες.

Αντικείμενο της τεχνολογίας λογισμικού είναι να δημιουργήσει μια εφαρμογή λογισμικού που να κάνει την εργασία για την οποία δημιουργήθηκε σωστά και αποδοτικά.

Οι χρήστες επιδιώκουν από ένα λογισμικό να έχει:

1. Μεγαλύτερη δυνατή ποιότητα
2. Μεγαλύτερη δυνατή αυτοματοποίηση
3. Μεγαλύτερη δυνατή παραγωγικότητα
4. Ελάχιστο κόστος παραγωγής
5. Ελάχιστο κόστος συντήρησης

Τα πέντε αυτά χαρακτηριστικά για την εποχή που ζούμε είναι μεταξύ τους αδύνατον να συνυπάρξουν σε ένα λογισμικό, καθώς δεν είναι δυνατόν να ζητούμε μεγιστοποίηση της αυτοματοποίησης και της παραγωγικότητας και την μέγιστη ελαχιστοποίηση του κόστους και της συντήρησης. Η λύση συνήθως βρίσκετε κάπου στην μέση, καθώς θα πρέπει να υπάρξει μια ισορροπία μεταξύ τους.

Η τεχνολογία Λογισμικού περιλαμβάνει τον προσδιορισμό και τις φάσεις των ενεργειών σύμφωνα με ορισμένη σειρά οι οποίες πρέπει να εκτελούνται και περιγράφει με σαφήνεια και κατανόηση το τρόπο με τον οποίο παράγονται τα προϊόντα έπειτα από τις πραγματοποιηθείσες ενέργειες και καθορίζει τις μεθόδους ελέγχου, επαλήθευσης και διασφάλισης της ποιότητας, έτσι ώστε να παραχθεί λογισμικό καλής ποιότητας. Το σύνολο όλων αυτών των ενεργειών τον ονομάζουμε εκτελέσιμο κώδικα ο οποίος όπως αντιλαμβανόμαστε είναι ένα πακέτο εντολών οι οποίες μας δείχνουν τι τρόπο εκτέλεσης τους και που στην συνέχεια θα αυτοματοποιηθούν από μια εφαρμογή λογισμικού.

Όπως γνωρίζουμε για να παραχθεί ο εκτελέσιμος κώδικας πρέπει να προηγηθούν μια σειρά από άλλες φάσεις τόσο για το λογισμικό όσο και για τις διάφορες τεχνητές κατασκευές. Ένα στοιχείο το οποίο μας δείχνει την μοναδικότητα του λογισμικού και το κάνει να ξεχωρίζει από άλλες τεχνητές κατασκευές είναι ότι πολύ συχνά η πρώτη έκδοση του εκτελέσιμου κώδικα και το υλικό τεκμηρίωσης υπόκεινται σε διορθώσεις και μεταβολές.

<u>Συνήθεις Αιτίες για τις οποίες το λογισμικό τροποποιείτε είναι</u>:

- Διόρθωση των λαθών
- Βελτιστοποίηση της απόδοσης
- Για να αυτοματοποιηθούν οι νέες εργασίες
- Για να ενσωματωθούν οι μεταβολές που συμβαίνουν στο επιχειρηματικό περιβάλλον και στη καθημερινή μας ζωή

Από το συνεχές μεταβαλλόμενο περιβάλλον και τις νέες απαιτήσεις οι οποίες αλλοιώνουν τα μέχρι πρότινος χαρακτηριστικά που πρέπει να τροποποιηθούν για να ικανοποιούν αυτές τις απαιτήσεις.

Η τεχνολογία λογισμικού έχει αναπτυχθεί ούτος ώστε να δώσει λύση στο πρόβλημα που λέγετε λογισμικό. Μερικές βασικές προκλήσεις για την τεχνολογία λογισμικού είναι:

- Ετερογένεια: να αναπτύξει τεχνικές για να κατασκευάσει ένα λογισμικό το οποίο να είναι αξιόπιστο και ευέλικτο ούτος ώστε να υπάρχει ομοιογένεια.
- Χρόνος παράδοσης: Είναι ένα από τα σοβαρά σημεία της κρίσης του λογισμικού και η τεχνολογία λογισμικού στοχεύει να αναπτύξει τεχνικές οι οποίες να δώσουν λύση στο χρόνο παράδοσης για τα πολύπλοκα έργα χωρίς ποιότητα που τις περισσότερες φορές παραδίδονται με μεγάλες αποστάσεις από τον προγραμματισμένο χρόνο.
- Εμπιστοσύνη: Για να μπορέσουν οι χρήστες να εμπιστευτούν το λογισμικό μέσα από τις τεχνικές που θα προσπαθήσει να αναπτύξει η τεχνολογία λογισμικού.

Η τεχνολογία λογισμικού είναι ένας πολύ σημαντικός κλάδος για τον άνθρωπο πού καθώς ολοένα και περισσότερα συστήματα πλέον εξαρτώνται από το λογισμικό, λογισμικό για το οποίο μέσα από

τον κλάδο αυτό της επιστήμης προσπαθεί να αυτοματοποιηθεί στις μεταβολές της καθημερινής μας ζωής.

Φθάνει να δούμε ότι οι οικονομίες όλων των ανεπτυγμένων χωρών του κόσμου εξαρτώνται από το λογισμικό και ότι οι δαπάνες για λογισμικό καλύπτουν μεγάλο μέρος του Ακαθάριστου Εθνικού προϊόντος στις ανεπτυγμένες χώρες, ο κλάδος αυτός της επιστήμης χρίζει ιδιαίτερης προσοχής και αποκτά ιδιαίτερη αξία.

1.6.ΤΟ ΛΟΓΙΣΜΙΚΟ ΩΣ ΜΕΡΟΣ ΣΥΣΤΗΜΑΤΩΝ

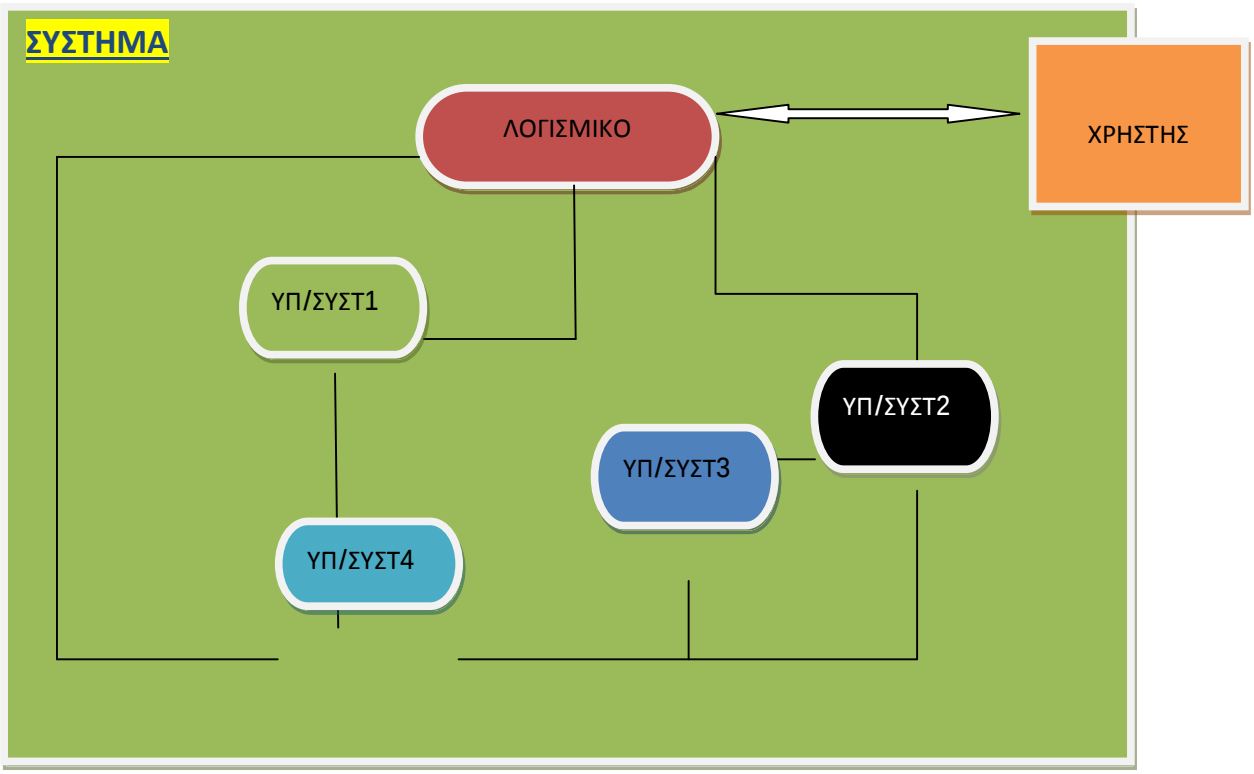
Τις πλείστες φορές το λογισμικό αντιμετωπίζετε λανθασμένα σαν μια οντότητα η οποία έχει δική της ανεξάρτητη και αυτοτελή ύπαρξη. Στον κλάδο της επιστήμης τεχνολογία λογισμικού γίνετε τις περισσότερες φορές γίνετε λόγος για το σύστημα και ξεχωριστά για το λογισμικό. Στο σημείο αυτό δημιουργείτε μια κατάσταση η οποία διακρίνετε από την αντιφατικότητα και ασάφεια σε περιπτώσεις ως προς τις έννοιες και την προσπάθεια προσέγγισης των οντοτήτων του κόσμου στον οποίο ζούμε.

Δυο περιπτώσεις που διακρίνονται μέσα σε όλη αυτή τη σύγχυση είναι:

1. Το λογισμικό αποτελεί εσωτερικό συστατικό ενός τεχνητού μη υπολογιστικού συστήματος.
2. Το λογισμικό λειτουργεί αυτοτελώς σε ένα υπολογιστικό σύστημα

Στην πρώτη περίπτωση διακρίνονται οι τα μη υπολογιστικά συστήματα τα οποία χρειάζονται ένα λογισμικό το οποίο να κατευθύνει τις λειτουργίες τους όπως είναι ψηφιακοί αυτοματισμοί, οι μηχανές αυτόματης πώλησης και οικιακές συσκευές όπως πλυντήριο, στεγνωτήριο ρούχων κτλ. Στην πρώτη περίπτωση ανήκουν και τα σύνθετα συστήματα όπου το λογισμικό είναι μέρος πολλών συστημάτων ταυτόχρονα οι οποίες είναι συνδεδεμένες μεταξύ τους και ο χρήστης αλληλεπιδρά μαζί τους όπως είναι τα ιατρικά μηχανήματα ανάλυσης και απεικόνισης, τα συστήματα ελέγχου της εναέριας κυκλοφορίας συστήματα χρονομέτρησης αγώνων κτλ.

Ένα σημαντικό στοιχείο που χρίζει ιδιαίτερης προσοχής σε τέτοιες περιπτώσεις οι κατασκευαστές ή οι οίκοι λογισμικών που θα αναπτύξουν το λογισμικό που θα ενσωματωθεί στις εν λόγω συσκευές θα πρέπει να λάβουν υπόψη τους τα ειδικά χαρακτηριστικά των συσκευών αυτών καθώς επηρεάζουν την δομή και την συμπεριφορά του λογισμικού ακόμα και στην περίπτωση που δεν είναι χαρακτηριστικά του.

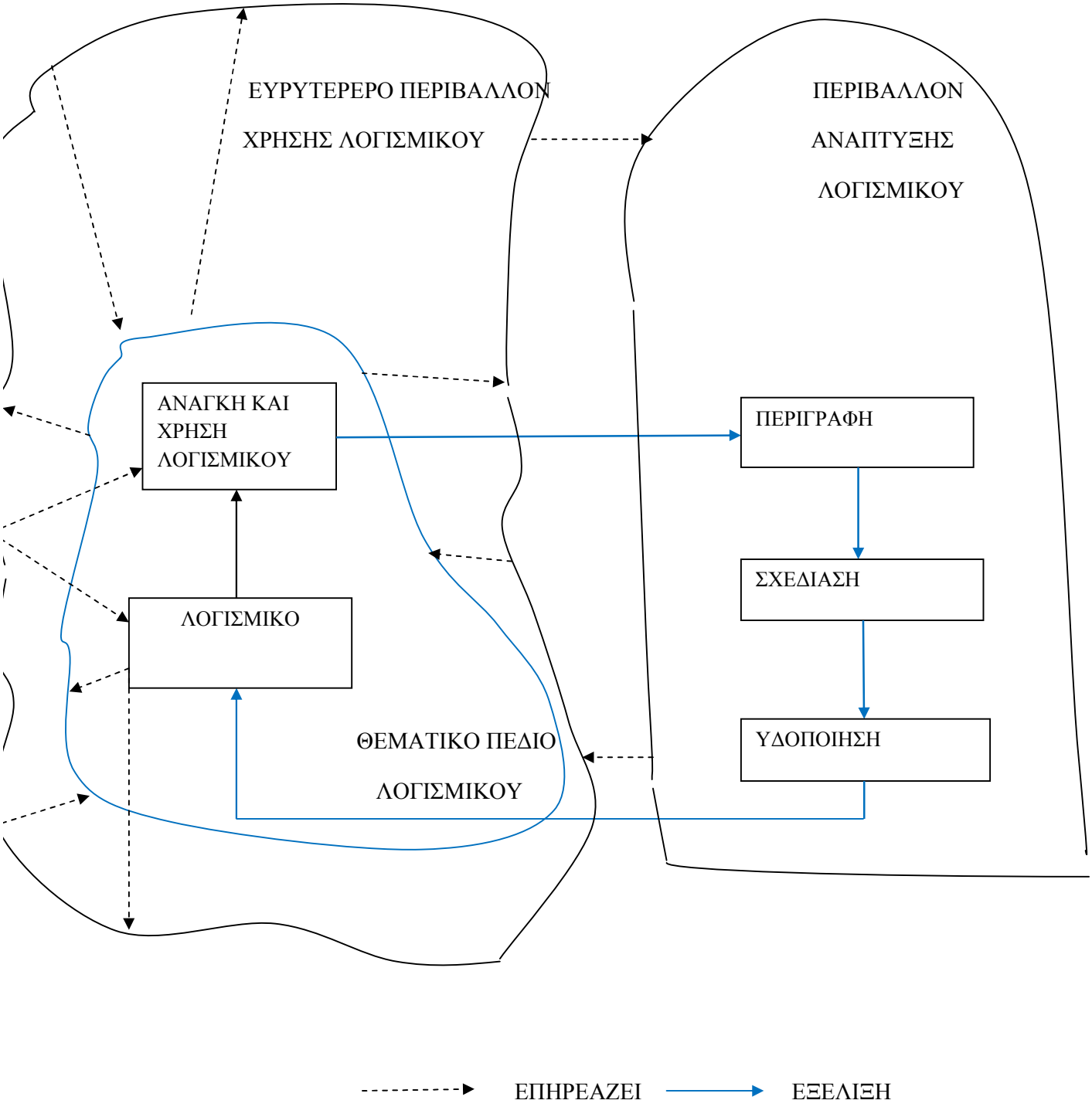


Στο παραπάνω σχήμα βλέπουμε ότι το λογισμικό μπορεί να είναι μέρος πολλών συστημάτων . Ο χρήστης αλληλεπιδρά με τα συστήματα και μπορεί να χρησιμοποιήσει το λογισμικό χωρίς να πέσει στην αντίληψη του.

Στην δεύτερη όπου το λογισμικό λειτουργεί αυτοτελώς σε ένα υπολογιστικό σύστημα αποτελεί λειτουργικός, μέρος ενός οργανισμού που έχει και άλλα λειτουργικά πεδία και αλληλεπιδρά με αυτά με αποτέλεσμα να αλληλό-καθορίζονται.

Όπως είναι γνωστό στα σταδία ανάπτυξης μιας εφαρμογής λογισμικού από την ανάγκη σύλληψης λογισμικού στην έναρξη μέχρι την παραγωγή του εκτελέσιμου κώδικα συμβαίνουν πολλές αλλαγές στο πεδίο χρήσης και τις περισσότερες φορές αποτελεί αιτιολογικό για την αποτυχία της ανάπτυξης εφαρμογής του.

Όπως διακρίνουμε ανάμεσα στο λογισμικό και στον χώρο όπου αναπτύσσετε υπάρχει μια κατάσταση στην οποία παρατηρούμε ότι υπάρχει μια εσωτερική αντίθεση. Η ανάγκη για λογισμικό ξεκινά λόγο της αυτοματοποίησης των εργασιών στην καθημερινή επιχειρηματική και όχι μόνο ζωή. Η συνένωση της αυτοματοποίησης στην περιοχή που ανακαλύφθηκε η ανάγκη έχει ως αποτέλεσμα ο κύκλος του λογισμικού να αρχίσει πάλι από την αρχή καθώς οι αλλαγές θα πρέπει να ενσωματωθούν στο λογισμικό και έτσι αυτό ξαναρχίζει.


ΕΥΡΥΤΕΡΕΡΟ ΠΕΡΙΒΑΛΛΟΝ
ΧΡΗΣΗΣ ΛΟΓΙΣΜΙΚΟΥ
ΠΕΡΙΒΑΛΛΟΝ
ΑΝΑΠΤΥΞΗΣ
ΛΟΓΙΣΜΙΚΟΥ
ΑΝΑΓΚΗ ΚΑΙ
ΧΡΗΣΗ
ΛΟΓΙΣΜΙΚΟΥ
ΠΕΡΙΓΡΑΦΗ
ΣΧΕΔΙΑΣΗ
ΛΟΓΙΣΜΙΚΟ
ΥΔΟΠΟΙΗΣΗ
ΘΕΜΑΤΙΚΟ ΠΕΔΙΟ
ΛΟΓΙΣΜΙΚΟΥ
ΕΠΗΡΕΑΖΕΙ
ΕΞΕΛΙΞΗ

1.7.ΛΟΓΙΣΜΙΚΟ ΩΣ ΠΡΟΙΟΝ

ΣΤΟΧΟΣ

Ο στόχος για τον οποίο γίνονται προσπάθειες για την ανάπτυξη του λογισμικού είναι να κατασκευαστή ένα προϊόν το οποίο να μπορεί να απευθύνετε τόσο στο σύνολο της μάζας των καταναλωτών όσο και για τις εξειδικευμένες ανάγκες ενός συγκεκριμένου χρήστη και να κατέχει υψηλό μερίδιο αγοράς.

Ένα λογισμικό μπορεί να αναπτυχθεί για τους εξής σκοπούς:

- *Σαν προϊόν μαζικής κατανάλωσης*

Οίκοι λογισμικών αναπτύσσουν λογισμικά προϊόντα τα οποία τοποθετούν στην συνέχεια στην αγορά και πωλούνται σε κάθε ενδιαφερόμενο χρήστη που τα έχει ανάγκη.

- *Customized λογισμικό*

Είναι συστήματα τα οποία δημιουργούνται αποκλειστικά για τις ανάγκες ενός πελάτη και έχουν αναπτυχθεί στα πλαίσια συγκεκριμένων συμβολαίων για το λογαριασμό του.

- *Λογισμικό προσωπικής χρήσης*

Ένας χρήστης μπορεί να κατασκευάσει ένα λογισμικό αποκλειστικά για τις δικές του ανάγκες.

Ταξινόμηση του λογισμικού

Τα λογισμικό χωρίζετε σε 2 βασικές κατηγορίες που ονομάζονται ΛΟΓΙΣΜΙΚΟ ΣΥΣΤΥΜΑΤΟΣ ΚΑΙ ΛΟΓΙΣΜΙΚΟ ΕΦΑΡΜΟΓΩΝ τα οποία θα αναλύσουμε πιο κάτω.

Ø Λογισμικό συστήματος ή λειτουργικό σύστημα (operating system):

Το λειτουργικό σύστημα είναι η βάση του λογισμικού, και αποτελείτε από ένα σύνολο προγραμμάτων το που είναι υπεύθυνα να ελέγχουν και να κατευθύνουν τις όλες τις λειτουργίες του ηλεκτρονικού υπολογιστή και λειτουργούν ως συνδετικός κρίκος μεταξύ του υλικού και των εφαρμογών του υπολογιστή, συμβάλουν στην δημιουργία και εξυπηρέτηση και εκτέλεση άλλων προγραμμάτων και συντελεί στην καλύτερη δυνατή επικοινωνία του χρήστη με τον ηλεκτρονικό υπολογιστή. Χωρίς αυτό δεν είναι δυνατή η λειτουργία του ηλεκτρονικού υπολογιστή.

Το λειτουργικό σύστημα χωρίζεται σε 2 βασικές κατηγορίες:

Single user:

Είναι συστήματα που έχουν την δυνατότητα να εξυπηρετούν ένα και μόνο χρήστη και μπορούν να χρησιμοποιηθούν από μικρό-υπολογιστές.

Multi user:

Είναι συστήματα που έχουν την δυνατότητα να εξυπηρετούν πολλούς χρήστες ταυτοχρόνως διαμοιράζοντας το χρόνο για να μπορεί να εξυπηρετεί όλους τους χρήστες που επικοινωνεί.

ΡΟΛΟΣ ΤΟΥ ΛΕΙΤΟΥΡΓΙΚΟΥ ΣΥΣΤΥΜΑΤΟΣ

- **Το Λειτουργικό Σύστημα συντελεί στον προγραμματισμό ενός Η/Υ σε επίπεδα όπως:**
  - Υλικό *(hardware)*
  - Μικροπρόγραμμα (*microcode*)
  - Γλώσσα μηχανής *(machine language)*
  - Λειτουργικό σύστημα (*operating system*)
  - Προγράμματα εφαρμογών (*application programs*)
- **Το λειτουργικό σύστημα ως εκτεταμένη μηχανή (extended machine):**
  - Απομονώνει τον προγραμματιστή από την πολυπλοκότητα του υπολογιστή
  - Διευκολύνει τη μεταφερσιμότητα (*portability*) των εφαρμογών.
  - Παρουσιάζει στις εφαρμογές ιδεατό (*virtual*) υλικό
- **Το λειτουργικό σύστημα ως διαχειριστής πόρων:**
  - Επιτρέπει τον πολυπρογραμματισμό
  - Επιτρέπει τη χρήση από πολλούς χρήστες
  - Επιβάλλει πολιτικές διαχείρισης και ασφάλειας

ΤΑ ΠΙΟ ΓΝΩΣΤΑ ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΥΜΑΤΑ

- Windows(Microsoft)
- Unix (AT&T)
- Linoux (LINOUS TORVALDS)
- MacOS (APPLE)
- OS/2(IBM)
- SOLARIS
- MS DOS
- VMS

Το λειτουργικό σύστημα για να εκτελέσει την εργασία του χρησιμοποιεί προγράμματα ελέγχου και λειτουργιών που:

- Επιτρέπουν την επικοινωνία χρήστη και ηλεκτρονικού υπολογιστή
- Καταγράφουν εργασίες

- Εκτελούν εργασίες
- Επιθεωρούν την συνολική λειτουργία του ηλεκτρονικού υπολογιστή για να εξασφαλίσουν ότι οι διάφορες λειτουργίες εκτελούνται χωρίς προβλήματα μέχρι την ολοκλήρωση τους.

Λογισμικό εφαρμογών (Application Software)

Το λογισμικό εφαρμογών είναι τα προγράμματα εκείνα που δημιουργεί ή αγοράζει ο χρήστης για την ικανοποίηση των αναγκών του και την εκτέλεση των πολλών και διαφόρων εργασιών. Κάθε πρόγραμμα αποκλειστικά προσανατολισμένο να εκτελεί μόνο κάποιες συγκεκριμένες εργασίες.

Πιο κάτω θα αναφέρουμε και θα περιγράψουμε τις κατηγορίες του λογισμικού εφαρμογών

- Προγράμματα εφαρμογών: Είναι προγράμματα που παραγγέλλονται σε ειδικά γραφεία ανάπτυξης λογισμικού ικανά έτσι ώστε να ικανοποιούν τις ανάγκες μιας επιχείρησης για την επεξεργασία των πληροφοριών π.χ. όπως προγράμματα που λειτουργεί μια ασφαλιστική εταιρεία.
- Πακέτα εφαρμογών: Είναι ένα πακέτο από διάφορα προγράμματα που έχουν σαν κύριο χαρακτηριστικό την παραμετροποίηση και έχουν στόχο να ικανοποιήσουν συγκεκριμένες ανάγκες μια επιχείρησης όπως η μισθοδοσία.
- Λογισμικό γενικής χρήσης: Είναι προγράμματα που αυτοματοποιούν τις περισσότερες λειτουργίες του γραφείου μια επιχείρησης π.χ. βάσεις δεδομένων, διαχείριση αρχείων, επεξεργασίας κειμένου κτλ.
- Πακέτα Επιτραπέζιων Εκδόσεων (Desk Top Publishing): Είναι προγράμματα που καλύπτουν όλο το φάσμα των εργασιών των εκδόσεων, από δημιουργία και επεξεργασία εικόνας και γραφικών έως και ηλεκτρονική επεξεργασία σελίδας όπου ενσωματώνονται όλα τα στοιχεία μιας σελίδας προς έκδοση (κείμενα, εικόνες και γραφικά).
- Λογισμικό επικοινωνιών: Είναι προγράμματα που επιτρέπουν την επικοινωνία μεταξύ πολλών υπολογιστών που βρίσκονται στο ίδιο ή διαφορετικό χώρο.

Δυνατότητες και εργασίες που εκτελούνται στο λογισμικό εφαρμογών

- Επεξεργασία κειμένου: Όπου παρέχεται στο χρήστη η δυνατότητα να δημιουργήσει, να μορφοποιήσει, να διορθώσει και να αποθηκεύσει κείμενα. Ακόμα και ήχους, γραφικά και εικόνες.
- Παρουσιάσεις: Στηρίζονται στην δημιουργία ειδικών οθόνων που ονομάζονται διαφάνειες και Δίνετε η δυνατότητα στο χρήστη να δημιουργήσει παρουσιάσεις με κείμενο, ήχο και εικόνες μέσα από μια γκάμα γραφικών, ειδικών εφέ, ήχων, κινουμένων εικόνων, εικονιδίων και έλεγχο της ροής της παρουσίασης. (powerpiont)
- Επιτραπέζια έκδοση εντύπων: Είναι προγράμματα που χρησιμεύουν για την στοιχειοθεσία ενός κειμένου όπως αρίθμηση σελίδων, χειρισμός γραφημάτων, φωτογραφιών κτλ και χρησιμεύουν στις εκδόσεις βιβλίων, εφημερίδων, περιοδικών, διαφημιστικών κτλ.
- Διαχείριση βάσης δεδομένων :Με αυτή την εφαρμογή ο χρήστης μπορεί να εισάγει να διαγράψει να τροποποιήσει, να διαχειριστεί τα προγράμματα που υπάρχουν σε μια βάση

δεδομένων και να δημιουργήσει καινούργιες βάσεις δεδομένων. Για τους προσωπικούς Η/Υ είναι η acssess και στα μεγάλα συστήματα η oracle.

- Λογιστικά φύλλα :Δίνετε η δυνατότητα στο χρήστη να διαχειριστεί πληροφορίες σε μορφή δισδιάστατου πίνακα και να επιλύσει τα γενικότερα αριθμητικά προβλήματα. Είναι χωρισμένο σε τετραγωνάκια που ονομάζονται κελιά, μπορούν να περιέχουν αριθμούς ακόμα και κείμενα και ο χρήστης μπορεί να ενσωματώσει στατιστικά στοιχεία, ημερομηνίες κτλ. Το πιο γνωστό πρόγραμμα είναι η excel.

Ένα από τα πολλά προβλήματα που αντιμετωπίζει το λογισμικό στη σημερινή εποχή είναι η εξασφάλιση της ποιότητας. Οι νόμοι της αγοράς, ο ανταγωνισμός για την εμπορική επικράτηση και η πίεση του χρόνου είναι μειονέκτημα για την εξασφάλιση λογισμικού ποιότητας, που σε αντίθεση με άλλες θεωρητικές επιστήμες όπου υπάρχει απεριόριστος χρόνος για να επιτευχθεί ποιότητα ή κάποιος άλλος στόχος, στο λογισμικό ισχύει το ακριβώς αντίθετο, τρομερή πίεση χρόνου.

Ο ανταγωνισμός είναι ένα επιπλέον σημείο στην πληγές του λογισμικού σήμερα. Ο ανταγωνισμός δεν βοηθά στην συνεργασία για να βγει το αγαθό στην αγορά με περισσότερες δυνατότητες και να μπορεί να ανταποκριθεί στις απαιτήσεις της. Η αγορές μεταβάλλονται συνεχώς και έχουν νέες απαιτήσεις. Οι διάφοροι οίκοι λογισμικού σπεύδουν να δημιουργήσουν ένα σύστημα το οποίο να καλύπτει τις ανάγκες αυτές, και να ικανοποιεί τις απαιτήσεις των χρηστών. Στην πιο συνηθισμένη περίπτωση οι διάφοροι οίκοι κατέχουν ένα μέρος της λύσης που χρειάζονται οι καταναλωτές χωρίς καμιά να έχει το ολοκληρωμένο πακέτο και κινούμενοι σε διαφορετικούς δρόμους. Η όλη αυτή κατάσταση μερικές φορές μπορεί να λειτουργήσει θετικά και μερικές φορές αρνητικά για τον καταναλωτή.

Όπως καταλαβαίνουμε θα πρέπει να εξομαλυνθεί η κατάσταση αυτή για να μπορέσει να υπάρξει σωστή ανάπτυξη λογισμικού. Απόψεις που έχουν διατυπωθεί ως προς την θεμελίωση κανόνων και διαδικασιών ανάπτυξης μόνο θετικές μπορούμε να τις κρίνουμε για την περαιτέρω ανάπτυξη του λογισμικού. Ταυτόχρονα όμως πρέπει να υπάρχει μια ισορροπία μεταξύ τεχνικής ορθότητας και της τιμής.

Θα πρέπει όμως το προϊόν να τοποθετείτε στην αγορά την κατάλληλη στιγμή, δηλαδή όταν υπάρχει ζήτηση για αυτό ούτως ώστε να ο κατασκευαστής του να έχει οικονομικά κέρδη.

1.8.ΣΥΣΤΑΤΙΚΑ ΣΤΟΙΧΕΙΑ ΛΟΓΙΣΜΙΚΟΥ

Συστατικά στοιχεία λογισμικού είναι τα ενδιάμεσα προϊόντα που παράγονται μεταξύ των φάσεων από τον καθορισμό των εργασιών τις οποίες το λογισμικό θα αυτοματοποιήσει στη συνέχεια μέχρι την παραγωγή του εκτελέσιμου κώδικα, και αποτελούν μέρος αυτού.

Τα ενδιάμεσα αυτά προϊόντα χωρίζονται σε 2 τμήματα και είναι είτε:

Α)Ενδιάμεσα συστατικά λογισμικού

Β)Τεκμηρίωση λογισμικού

Στην Α περίπτωση είναι τα συστατικά που παράγονται κατά την διάρκεια του λογισμικού μέχρι να φτάσουμε στην φάση του εκτελέσιμου κώδικα τα οποία είναι:

- Ο εκτελέσιμος κώδικας: είναι το σύνολο των εντολών του προγράμματος που εκτελείτε άμεσα από τον υπολογιστή εφόσον πληρούνται συγκεκριμένες προϋποθέσεις.
- Πηγαίος κώδικας: είναι πρόγραμμα του υπολογιστή το οποίο έχει γραφεί σε κάποια γλώσσα προγραμματισμού και είναι σχεδόν πάντα μυστικός. Είναι γραμμένος πάντα στα αγγλικά και ο υπολογιστής για να μπορέσει να το εκτελέσει πρέπει να μεταγλωττιστεί από τον μεταγλωττιστή σε γλώσσα μηχανής για να γίνει αντικειμενικός κώδικας. Αν ακόμα ο υπολογιστής δεν μπορεί να διαβάσει τον αντικειμενικό κώδικα τότε θα πρέπει ο μετεσκευαστής να κάνει την τελευταία μετάφραση σε γλώσσα μηχανής.
- Βιβλιοθήκες: όταν υπάρχει εξάρτηση μεταξύ εκτελέσιμων συστατικών εννοούμε τις βιβλιοθήκες που χρειάζεται το πρόγραμμα για να εκτελεστεί.
- Κώδικας σε μορφή object: είναι ο αντικειμενικός κώδικας που αναφέραμε πιο πάνω.
- Εκθέσεις :
- Αναφορές :
- Κείμενα:
- Σχέδια :
- Διαγράμματα:

Στην Β περίπτωση στην τεκμηρίωση λογισμικού η οποία περιγράφει την δομή και την συμπεριφορά του λογισμικού και καταγράφει.

- Ποιές εργασίες εκτελεί
- Πως εκτελούνται οι εργασίες
- Ποιες δομές δεδομένων θα χρησιμοποιήσει

Τα συστατικά λογισμικού μπορούν να ταξινομηθούν ως :

- Ως προς την φύση τους
  Διακρίνονται σε:
  - Ηλεκτρονική μορφή
  - Έντυπη μορφή
- Ως προς τον τρόπο παραγωγής τους
  Διακρίνονται σε αυτά που:
  - Παράγονται αυτόματα με τον εκτελέσιμο κώδικα
  - Παράγονται με το χέρι

- Ως προς την φάση κύκλου ζωής στην οποία παράγονται
- Ως προς την εσωτερική δομή:
  - Ανάλογα με την τεχνική φύση του περιβάλλοντος ανάπτυξης και λειτουργίας
- Ως προς τα πρότυπα στα οποία συμμορφώνονται
  - Αναφέρονται στη δομημένη περιγραφή κάποιων χαρακτηριστικών

Στο 5^{ο} μέρος όσον αφορά ως προς τα πρότυπα τα οποία συμμορφώνονται υπάρχει μια σύγκρουση η οποία και αποτελεί μεγάλο πρόβλημα και εμποδίζει την ανάπτυξη του λογισμικού. Λόγοι και αιτίες που οδηγούν σε αυτή την σύγκρουση στην σημερινή εποχή αποτυπώνονται πιο κάτω

Υπάρχουν πολλαπλά και αυτοτελή σύμβολα, τίτλοι, ορισμοί, δομές και έννοιες που κάνουν λόγο παρόμοιες ή παραπλήσιες οντότητες πράγμα το οποίο συνεπάγετε στην μη κατανόηση των οντοτήτων που σχετίζονται με το λογισμικό και επηρεάζουν αρνητικά την εξεύρεση και αποδοχή μιας κοινά αποδεκτής εκδοχής για τις οντότητες.

Η κατάσταση που δημιουργείτε από την ασάφεια και την αντιφατικότητα αυξάνει το πρόβλημα σχετικά με το λογισμικό. Η πιο πάνω κατάσταση καθρεφτίζετε παντού στις γλώσσες προγραμματισμού στις μεθοδολογίες στα εργαλεία πράγμα το οποίο τους οδηγεί τους κατασκευαστές στην επιλογή και χρησιμοποίηση δικών τους λύσεων σε διαφορετικά θέματα, σε σύμβολα και δομές.

Υπάρχουν πολυάριθμα επίπεδα στα οποία μπορεί να εμφανιστούν οι αιτίες που προκαλούν την σύγχυση, τα οποία είναι:

- Αρχική ανωριμότητα σε κάθε νέο ερευνητικό πεδίο
- Ανταγωνισμός για την κατοχή του μεγαλύτερου μεριδίου αγοράς σε τομείς όπως:
    - Εργαλεία ανάπτυξης
    - Παροχή τεχνογνωσίας
- Ορολογία
- Συμβολισμοί

Σε μια προσπάθεια ούτως ώστε να αναπτυχθούν πρότυπα και να δοθεί μια περιγραφή των συστατικών λογισμικού ήταν και η προσπάθεια που έκανε ο οργανισμός IEEE, αναπτύσσοντας πρότυπα που να προσαρμόζονται ανάλογα με την μεθοδολογία και τον ακολουθούμενο κύκλο ζωής.

1.9.ΣΥΝΟΨΗ ΠΡΩΤΟΥ ΚΕΦΑΛΑΙΟΥ

Ένας θεωρητικός ορισμός για το λογισμικό διαχρονικά αμφιβάλετε από κάποιον άλλον ορισμό. Τα λογισμικό έχει ως στόχο να αυτοματοποιήσει τις ανθρώπινες εργασίες τόσο στην καθημερινή μας ζωή όσο και στο επιχειρηματικό περιβάλλον. Το λογισμικό δεν είναι μόνο ο εκτελέσιμος κώδικας. Είναι και όλα τα ενδιάμεσα προϊόντα που παράγονται όπως βιβλιοθήκες, πηγαίος κώδικας, σχέδια κτλ, τα οποία αποτελούν προϊόντα του κύκλου ζωής του λογισμικού. Το λογισμικό αποτελεί το πιο πολύπλοκο σύστημα που κατασκεύασε ο άνθρωπος και γι'αυτό και μέχρι σήμερα αντιμετωπίζει πολλά προβλήματα σε πολλά επίπεδα όπως ποιότητα, κόστος κτλ. Τα χρόνια αυτά προβλήματα τα αναγνωρίζουμε ως κρίση λογισμικού. Για τα προβλήματα αυτά αναπτύχτηκε και η περιοχή της επιστήμης που την ονομάζουμε Τεχνολογία λογισμικού που ασχολείται με την εύρεση και θεμελίωση μεθόδων για να κατασκευαστεί ένα λογισμικό το οποίο να έχει όσο μεγαλύτερη αυτοματοποίηση σε διάρκεια παραγωγικότητα και ελάχιστο κόστος συντήρησης. Η τεχνολογία λογισμικού στοχεύει με την μελέτη και την εξέταση του λογισμικού να καθοδηγήσει τους οίκους και εταιρείες παραγωγής λογισμικού στο να παράγουν προϊόντα λογισμικού που να αποδίδουν τα μέγιστα και να ενσωματώνουν τις απαιτήσεις του μεταβαλλόμενου περιβάλλοντος.

ΚΕΦΑΛΑΙΟ 2: Μοντέλα κύκλου ζωής του Λογισμικού

ΜΟΝΤΕΛΑ ΚΥΚΛΟΥ ΖΩΗΣ ΛΟΓΙΣΜΙΚΟΥ

Στο κεφάλαιο αυτό θα αναλύσουμε τον ορισμό της έννοιας Μοντέλα Κύκλου Ζωής Λογισμικού, τα μοντέλα κύκλου ζωής που υπάρχουν καθώς και την χρήση των μοντέλων αυτών.

Θα μελετηθούν κάποιες έννοιες με τις οποίες ο αναγνώστης θα είναι σε θέση να διακρίνει και να κατανοήσει τα θεμελιώδη συστατικά της πειθαρχημένης ανάπτυξης τα οποία επιβάλει η Τεχνολογία Λογισμικού.

Έννοιες προς ανάλυση:

►Μοντέλο Κύκλου Ζωής

►Δραστηριότητα ανάπτυξης λογισμικού

►Μεθοδολογία Ανάπτυξης

►Εργαλεία Λογισμικού

►Προδιαγραφή – Ανάπτυξη – Επαλήθευση – Εξέλιξη Λογισμικού

►Μοντέλο Καταρράκτη

►Μοντέλο Πρωτοτυποποίησης

►Μοντέλο Λειτουργικής Επαύξησης

►Σπειροειδές Μοντέλο

►Μοντέλο του Πίδακα

Σκοπός είναι μετά από την μελέτη του κεφαλαίου αυτού ο αναγνώστης να μπορεί να περιγράφει τις έννοιες «μοντέλο κύκλου ζωής», μεθοδολογία, εργαλεία και την διαδικασία ανάπτυξης και να μπορεί να τις χρησιμοποιεί. Εκτός από την περιγραφή των εννοιών αυτών να μπορεί να περιγράφει συγχρόνως και τα χαρακτηριστικά ανάπτυξης λογισμικού. Τέλος στα προσδοκώμενα αποτελέσματα θα πρέπει να διακρίνει τα θετικά και τα αρνητικά στοιχεία της υιοθέτησης ενός μοντέλου κύκλου ζωής σε συγκεκριμένες συνθήκες, να διακρίνει ποιο μοντέλο κύκλου ζωής εφαρμόζετε σε μια δεδομένη περίπτωση κατασκευαστή λογισμικού και τελευταίο να διακρίνει περιγραφές των συστατικών στοιχείων του λογισμικού σύμφωνα με μια ταξινόμηση σε διαφορετικά επίπεδα λεπτομέρειας.

Εισαγωγικές Παρατηρήσεις

Στο πρώτο κεφάλαιο είδαμε ότι «η Τεχνολογία Λογισμικού είναι η περιοχή εκείνη της επιστήμης της μηχανικής που ασχολείται με την εύρεση και την θεμελίωση μεθόδων για να περιγράφεται, να κατασκευάζεται και να συντηρείται λογισμικό καλής ποιότητας, με την μεγαλύτερη δυνατή αυτοματοποίηση και παραγωγικότητα και το ελάχιστο δυνατό κόστος. Για να επιτευχθούν οι σκοποί της Τεχνολογίας Λογισμικού, οφείλει και πρέπει να περιγράψει τις ενέργειες που πρέπει να γίνονται κατά την ανάπτυξη του λογισμικού, τόσο σε μακροσκοπικό επίπεδο όσο και σε μικροσκοπικό επίπεδο. Σε μακροσκοπικό επίπεδο πρέπει να οριστούν οι φάσεις από τις οποίες διέρχεται η κατασκευή του λογισμικού, ενώ σε μικροσκοπικό επίπεδο θα πρέπει να οριστούν οι ενέργειες που γίνονται σε κάθε φάση και τα προϊόντα που παράγονται.

Δεν υπάρχει ένας και μόνο, μοναδικός τρόπος για να προσδιορίσουμε τις γενικές φάσεις από τις οποίες διέρχεται η κατασκευή του λογισμικού. Όπως θα δούμε και θα αναλύσουμε πιο κάτω στο κεφάλαιο αυτό, πολλά εξαρτώνται από τις ιδιαίτερες συνθήκες που επικρατούν, όπως το είδος της εφαρμογής, η πιθανότητα να αλλάξουν οι απαιτήσεις από τη εφαρμογή αυτή αλλά και το είδος της εφαρμογής. Για τον λόγο αυτό έχουν προταθεί πολλές εναλλακτικές διαδρομές που μπορεί να ακολουθήσει κανείς στην κατασκευή και την συντήρηση του λογισμικού, οι οποίες ονομάζονται «Μοντέλα Κύκλου Ζωής» , στις οποίες θα αναφερθούμε στο κεφάλαιο αυτό.

Το κεφάλαιο που ακολουθεί , περιέχει και αναλύονται οι πιο κάτω ενότητες :

2.1 Η έννοια του μοντέλου κύκλου ζωής
2.2 Το μοντέλο του καταρράκτη
2.3 Το μοντέλο της πρωτοτυποποίησης
2.4 Το μοντέλο λειτουργικής επαύξησης

2.5 Το σπειροειδές μοντέλο

2.6 Το μοντέλο του πίδακα

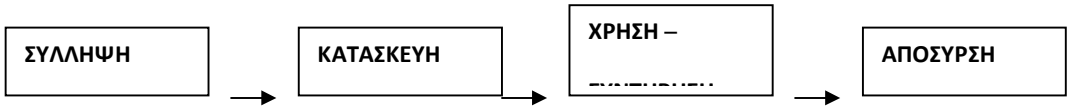
2.7 Σύγχρονα μοντέλα κύκλου ζωής λογισμικού

2.8 Περιγραφή διαδικασιών ανάπτυξης και προϊόντων λογισμικού

2.1 Η Έννοια του μοντέλου κύκλου ζωής

Από την αρχή μέχρι το τέλος, κάθε εφαρμογή λογισμικού, διέρχεται από διάφορα στάδια, εκ των οποίων πρέπει να γίνονται οι κατάλληλες εργασίες για να επιφέρουν τα επιθυμητά αποτελέσματα. Στο σχήμα 2.1.1 που ακολουθεί θα δούμε συνοπτικά τις γενικές φάσεις του κύκλου ζωής λογισμικού, σε μακροσκοπικό επίπεδο.

Σχήμα 2.1.1 Γενικές Φάσεις του κύκλου ζωής λογισμικού



Σε γενικές γραμμές το μοντέλο κύκλου ζωής είναι:

►Εξαγωγή και Ανάλυση Απαιτήσεων

►Σχεδίαση Προγραμμάτων

►Κωδικοποίηση

►Δοκιμή Μονάδων Ενοποίησης

►Δοκιμή Συστήματος

►Δοκιμή Αποδοχής

►Παράδοση, Λειτουργία και Συντήρηση Συστήματος

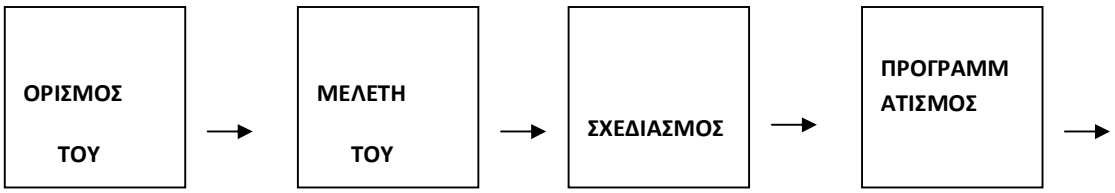
Είναι χρήσιμο να αναφερθούμε σε κάποιους χρήσιμους ορισμούς πριν αναφέρουμε τον ορισμό του μοντέλου κύκλου ζωής αλλά και τα πιο σημαντικά μοντέλα κύκλου ζωής που χρησιμοποιούμε σήμερα. Ενδεχομένως, φυσικά, ο αναγνώστης να συναντήσει διάφορες επεξηγήσεις για τον κάθε ορισμό, επειδή στην Τεχνολογία Λογισμικού υπάρχουν κάποιες απόψεις που είναι ελαφρώς διαφοροποιημένες για κάποιους ορισμούς.

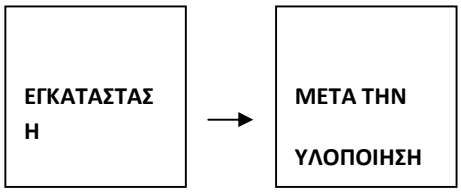
■ ΔΡΑΣΤΗΡΙΟΤΗΤΑ ΑΝΑΠΤΥΞΗΣ ΛΟΓΙΣΜΙΚΟΥ (software process)

Είναι η δραστηριότητα, η οποία καθορίζει στις γενικές φάσεις του κύκλου, πότε πρέπει να γίνουν οι κατάλληλες ενέργειες για να επιφέρει το επιθυμητό αποτέλεσμα. Μία δραστηριότητα μπορεί να αναλύεται σε περισσότερες από μια επιμέρους φάσεις.

■ ΜΕΘΟΔΟΛΟΓΙΑ ΑΝΑΠΤΥΞΗΣ (software development methodology)

Καθορίζει πώς πρέπει να γίνονταί οι δραστηριότητες ανάπτυξης Λογισμικού, δηλαδή ποιες ενέργειες και ποια βήματα. Η μεθοδολογία χωρίζεται στα εξής 6 στάδια :





Αναλυτικά τα 6 στάδια τις μεθοδολογίας είναι τα εξής :

1. Ορισμός έργου (project definition) : Το στάδιο κύκλου ζωής συστημάτων στο οποίο προσδιορίζεται αν ο οργανισμός έχει κάποιο πρόβλημα και αν αυτό το πρόβλημα μπορεί να λυθεί με ένα έργο συστήματος.
2. Μελέτη Συστήματος (system study) : Το στάδιο του κύκλου ζωής των συστημάτων στο οποίο αναλύονται τα προβλήματα των υπαρχόντων συστημάτων, καθορίζονται οι στόχοι της λύσης τους και τέλος αξιολογούνται διάφορες εναλλακτικές λύσεις.
3. Σχεδιασμός (design): Το στάδιο του κύκλου ζωής που παράγει τις προδιαγραφές του λογικού και του φυσικού σχεδιασμού για την επίλυση του προβλήματος.

4. Προγραμματισμός (programming) : Το στάδιο του κύκλου ζωής συστημάτων στο οποίο οι προδιαγραφές του λογικού και του φυσικού σχεδιασμού στο στάδιο σχεδιασμού μεταφράζονται σε κώδικα προγραμμάτων λογισμικού.

5. Εγκατάσταση (installation) : Το στάδιο του κύκλου ζωής συστημάτων που αποτελείται από τις δοκιμές, την εκπαίδευση και την μετατροπή – τα τελικά βήματα για να μπει ένα σύστημα σε λειτουργία.

6. Μετά την υλοποίηση (post implementation) : Το τελικό στάδιο του κύκλου ζωής συστημάτων στο οποίο χρησιμοποιείται και αξιολογείται ενώ βρίσκεται σε λειτουργία και τροποποιείται για να γίνουν βελτιώσεις ή να ικανοποιηθούν νέες απαιτήσεις.

■ ΕΡΓΑΛΕΙΟ ΛΟΓΙΣΜΙΚΟΥ

Είναι ένα βοηθητικό πρόγραμμα ή μια δυνατότητα που συμβάλει στην ολοκλήρωση μίας εργασίας που εκτελείται κατά την εφαρμογή των μεθοδολογιών της ανάπτυξης λογισμικού

■ ΜΟΝΤΕΛΟ ΚΥΚΛΟΥ ΖΩΗΣ ΛΟΓΙΣΜΙΚΟΥ

Είναι μια περιγραφή των διαδικασιών και των επιμέρους φάσεων από τις οποίες διέρχεται μια εφαρμογή από την σύλληψη μέχρι της απόσυρση.

Οι διαδικασίες ανάπτυξης προσδιορίζονται από τα μοντέλα ζωής λογισμικού. Σε διαδικασία ανάπτυξης μπορούμε να διακρίνουμε επιμέρους φάσεις και σε κάθε επιμέρους φάση να διακρίνουμε περισσότερες από μια εργασίες.

ΤΑΞΙΝΟΜΗΣΗ ΔΙΑΔΙΚΑΣΙΩΝ ΑΝΑΠΤΥΞΗΣ

***Προδιαγραφή (specification):** ο καθορισμός δηλαδή των εργασιών που θα επιτελεί το λογισμικό, καθώς και των περιορισμών και των παραδοχών που ισχύουν.

* **Ανάπτυξη (development) :** η κατασκευή του λογισμικού. Διακρίνουμε τρεις επιμέρους φάσεις : α) την ανάλυση, β)την σχεδίαση, γ)την συγγραφή του πηγαίου κώδικα- source code (κωδικοποίηση)

* **Επαλήθευση (verification) :** η επιβεβαίωση της ικανοποίησης των προδιαγραφών και της μη ύπαρξης σφαλμάτων.

* **Εξέλιξη (evolution) :** η επαύξηση των λειτουργικών χαρακτηριστικών του λογισμικού προκειμένου να ικανοποιούνται οι μεταβαλλόμενες ανάγκες.

ΣΤΟΧΟΣ ενός μοντέλου κύκλου ζωής λογισμικού είναι η καθοδήγηση του κατασκευαστή λογισμικού για την καλύτερη εκτέλεση των διαδικασιών ανάπτυξης, δηλαδή να είναι περισσότερο παραγωγική, να υπάρχουν λιγότερα σφάλματα και μικρότερο ρίσκο. Φυσικά τα πιο πάνω μπορεί να επηρεαστούν από :

Α) το μέγεθος της εφαρμογής

Β) το θέμα της εφαρμογής

Γ) την εμπειρία του κατασκευαστή

Δ) το περιβάλλον ανάπτυξης

Ε) το **κόστος εφαρμογής**

Θα αναφερθούμε συνοπτικά στο κόστος εφαρμογής το οποίο δεν αφορά απαραίτητα μόνο οικονομικούς πόρους που αποδίδονται στο έργο, αλλά πολλές φορές αναφέρεται και στο χρόνο καθυστέρησης. Η έλλειψη ή μη πιστή εφαρμογή του μοντέλου κύκλου ζωής έχει ως αποτέλεσμα αναθεώρηση αποφάσεων, διόρθωση σφαλμάτων, οπισθοδρόμηση διαδικασιών, τα οποία χρονοτριβούν το έργο, και το κόστος λογισμικού να μεγαλώνει.

Υπάρχουν αρκετά μοντέλα κύκλου ζωής λογισμικού τα οποία έχουν κάποια κριτήρια διαφοροποίησης όπως :

►σύλληψη ιδέας του τρόπου κατασκευής

►επιμέρους φάσεις

►επαναληπτικότητα εργασιών

►εμβέλεια εργασιών – έκταση στην οποία εφαρμόζονται οι διαδικασίες

►ενδιάμεσα συστατικά

►ενδιάμεσες αποτιμήσεις από πελάτη ή κατασκευαστή

►οικονομικά και επιχειρηματικά κριτήρια χρήσης τους

Η διαδικασία επίλυσης των προβλημάτων είναι κάθε μια από τις ενέργειες που περιγράφεται σε ένα μοντέλο κύκλου ζωής. Τα βήματα κάθε τέτοιας διαδικασίας είναι :

1) Αποτίμηση τρέχουσας κατάστασης – μονάδας

2) Ορισμός του προβλήματος

3) Επιλογή μιας λύσης

4) Υλοποίηση της λύσης

5) Ενσωμάτωση της λύσης στο σύστημα

Να αναφέρουμε ότι ο μηχανισμός Λογισμικού εκτελεί συνεχώς τέτοιες διαδικασίες επίλυσης προβλημάτων τόσο σε μικροσκοπικό αλλά τόσο και σε μακροσκοπικό επίπεδο.

Στην επόμενη ενότητα θα αναφέρουμε και θα αναλύσουμε εν συντομία μερικά από τα σημαντικότερα μοντέλα κύκλου ζωής λογισμικού.

Χαρακτηριστικά μοντέλα κύκλου ζωής λογισμικού

►Μοντέλο του καταρράκτη

►Μοντέλο πρωτοτυποποίησης

►Μοντέλο Λειτουργικής Επαύξησης

►Σπειροειδές μοντέλο

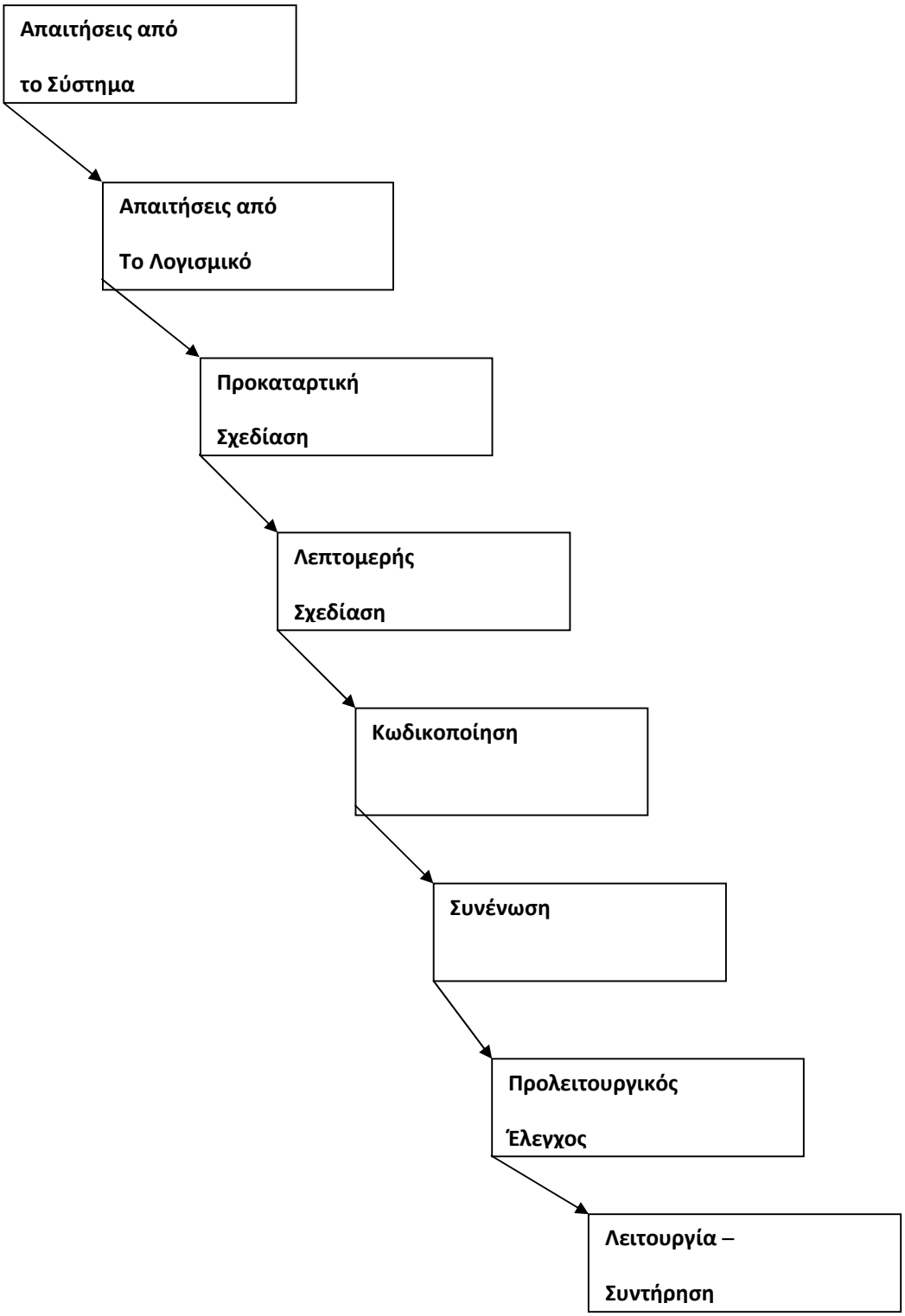
►Μοντέλο του Πίδακα

2.2 Το Μοντέλο του Καταρράκτη

Το μοντέλο του καταρράκτη (waterfall) είναι ένα από τα παλιότερα μοντέλα και συγχρόνως πιο διαδεδομένα. Αντιμετωπίζει την ανάπτυξη του λογισμικού ως την μεταφορά μεγάλου ογκόλιθου (=λογισμικό) από ένα σημείο στο άλλο με ενδιάμεσες στάσεις, χωρίς να μεταβάλλεται ο ογκόλιθος.

Το λογισμικό σύστημα, διευρύνεται περνώντας διακριτές φάσεις με μια ακολουθιακή σειρά. Κάθε φάση ολοκληρώνεται με μια εργασία επαλήθευσης των προϊόντων που παρήγαγε και στην συνέχεια, αποφασίζεται αν θα μεταβιβαστεί στην επόμενη φάση ή όχι. Όμως στην περίπτωση για να μεταβιβαστεί στην επόμενη φάση, βασικό κριτήριο είναι η ολοκλήρωση της προηγούμενης φάσης.

Σχήμα 2.2.1.Μοντέλο του Καταρράκτη (waterfall model)



Όπως βλέπουμε στο σχήμα 2.2.1 στην αρχή καθορίζονται οι απαιτήσεις από το σύστημα και από το λογισμικό. Στην συνέχεια ακολουθεί μια προκαταρτική σχεδίαση του συστήματος και των προγραμμάτων στην οποία καθορίζονται οι μονάδες που θα αποτελούν το λογισμικό. Αμέσως μετά η λεπτομερή σχεδίαση λογισμικού, στην οποία καθορίζεται η εσωτερική δομή κάθε μονάδας λογισμικού (αλγόριθμοι, δομές δεδομένων κτλ) και περιλαμβάνει την συγγραφή πηγαίου κώδικα. Ακολούθως βλέπουμε την συνένωση των μονάδων στο σύστημα καθώς και τον προλειτουργικό έλεγχο του συστήματος και αφού ολοκληρωθεί η παράδοση προϊόντος στον πελάτη επέρχεται η φάση της λειτουργίας και συντήρησης.

Παρ΄όλο που το μοντέλο του καταρράκτή είναι ιδιαίτερα χρήσιμο σε περιπτώσεις όπου οι απαιτήσεις από το λογισμικό είναι από την αρχή γνωστές και δεν μεταβάλλονται κατά την ανάπτυξη του λογισμικού παρατηρούνται κάποια μειονεκτήματα του μοντέλου αυτού τα οποία είναι τα εξής :

►όσο μεγαλώνει η έκταση εφαρμογής λογισμικού, τόσο δυσκολότερη είναι η μετάβαση από τη επόμενη και η αποφυγή σφαλμάτων τα οποία εντοπίζονται αργότερα

►όσο αργότερα εντοπίζεται στην ανάπτυξη ένα σφάλμα, τόσο μεγαλύτερες είναι οι επιπτώσεις που μπορεί να έχει η διόρθωση του, σε κόστος οπισθοδρόμησης, παρενέργειες,

Καθυστερήσεις, αλλά και δημιουργία νέων σφαλμάτων.

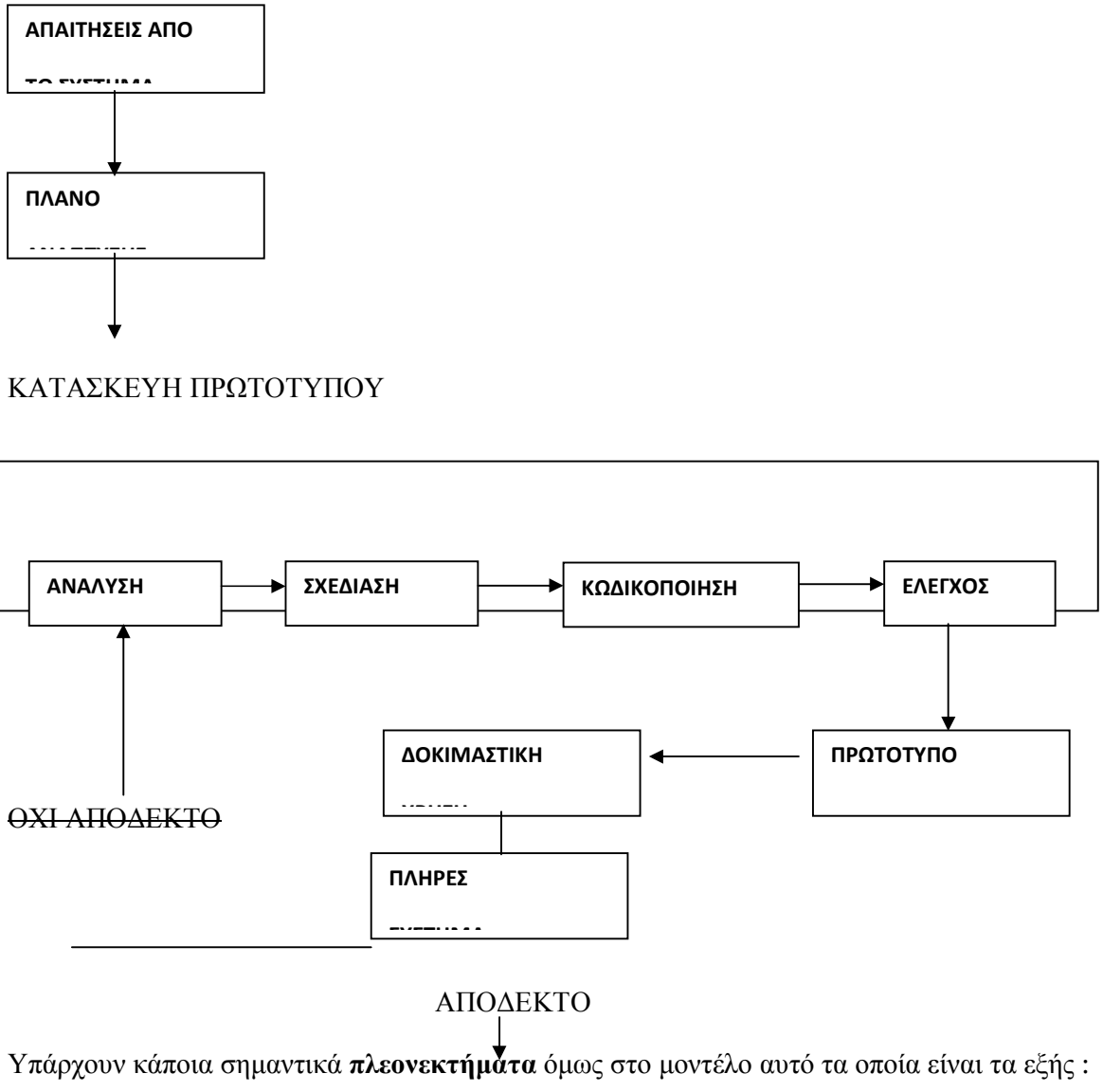
►Δεν είναι δυνατό να υπάρχει μια πρώτη εικόνα του συστήματος που κατασκευάζεται, παρά μόνο σε προχωρημένη φάση της ανάπτυξης.

2.3 Το μοντέλο της προτυποποίησης

Το μοντέλο προτυποποίησης (prototyping model) είναι η ανάπτυξη του λογισμικού σε στάδια που ονομάζονται «πρωτότυπα» , και για κάθε στάδιο φτιάχνεται ένα πρωτότυπο. Το μοντέλο αυτό ακόμη ονομάζεται «επαναληπτικό μοντέλο» επειδή οι διαδικασίες ανάπτυξης επαναλαμβάνονται για κάθε πρωτότυπο.

Παρατηρούμε ακόμη ότι κάθε πρωτότυπο περιέχει κάποιες βασικές λειτουργίες για την εκτέλεση του λογισμικού, και έτσι δοκιμάζεται από τον πελάτη-χρήστη και βελτιώνεται με νέα έκδοση πρωτοτύπου μέχρι να γίνει αποδεκτό.

Σχήμα 2.3.1 Το μοντέλο της προτυποποίησης (prototyping model)



Υπάρχουν κάποια σημαντικά **πλεονεκτήματα** όμως στο μοντέλο αυτό τα οποία είναι τα εξής :

►Η γρήγορη ανίχνευση αναγκών-προβλημάτων πριν την ανάπτυξη μεγάλου μέρους του Λογισμικού, σε σχέση με το μοντέλο του καταρράκτη, που αυτό επιφέρει λιγότερες καθυστερήσεις και λιγότερα κόστη

►Είναι κατάλληλο για την ανάπτυξη εφαρμογών με ασάφεια στις απαιτήσεις

►Επικοινωνία χρηστών /ομάδας ανάπτυξης

►Αυξανόμενη σταδιακά ικανοποίηση του πελάτη

►Αρχικά πρωτότυπα χρησιμοποιούνται για εξοικείωση από τους χρήστες

►Σημαντική η διοίκηση του έργου – υλοποιησιμότητα και εύκολη τροποποίηση πρωτοτύπου

Παρ ’όλο όμως , που το μοντέλο αυτό έχει αρκετά πλεονεκτήματα έχει και ένα μειονέκτημα. Το μέγεθος της εφαρμογής δεν μπορεί να είναι μεγάλο επειδή ο χρόνος ανάπτυξης κάθε πρωτότυπου μεγαλώνει και η απαιτούμενη ευελιξία μειώνεται.

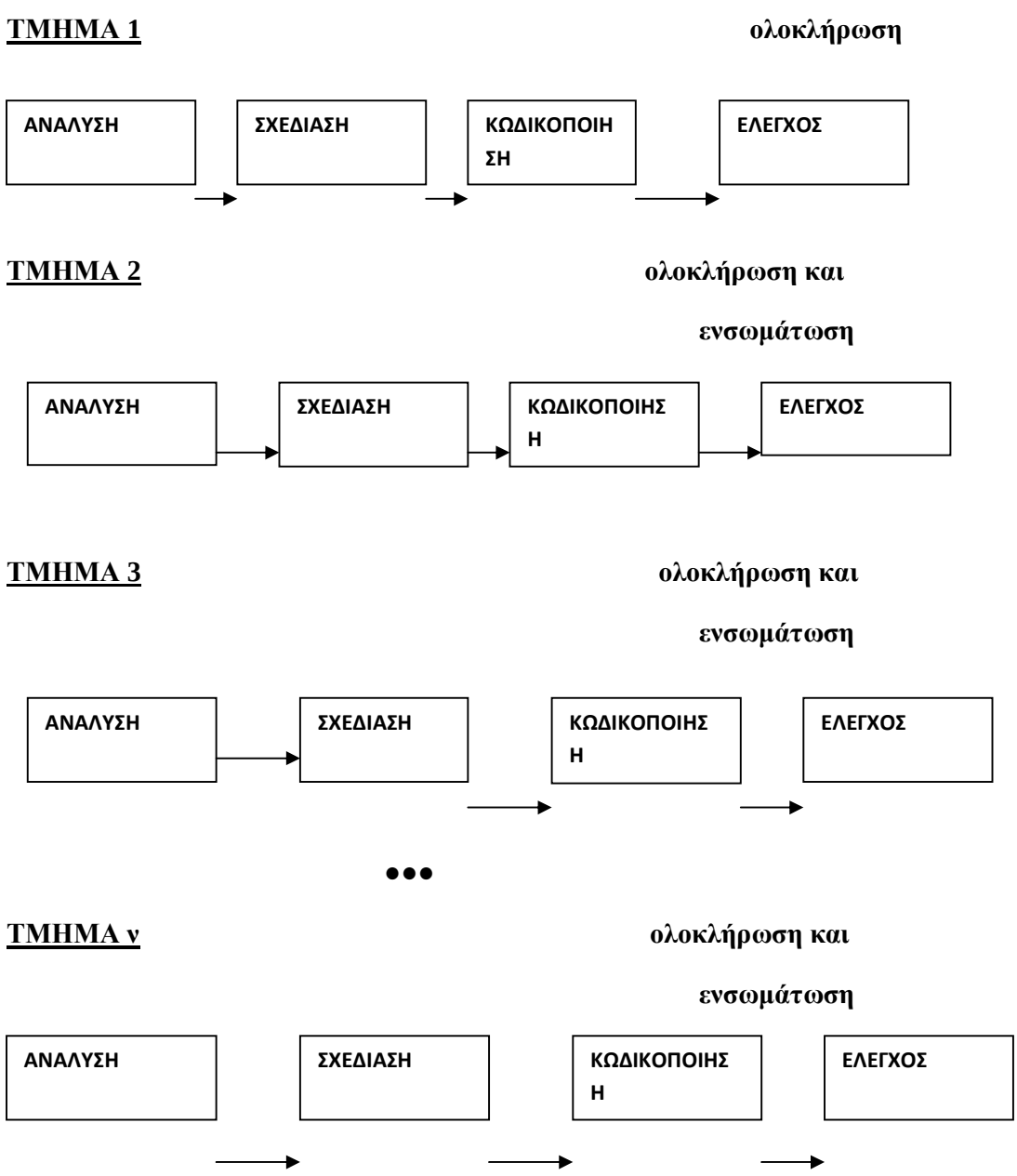
Το μοντέλο αυτό χρησιμοποιείται σε εφαρμογές που οι απαιτήσεις δεν είναι γνωστές από την αρχή της ανάπτυξης.

2.4 Το μοντέλο λειτουργικής επαύξησης

Στο μοντέλο αυτό βλέπουμε την ανάπτυξη κάθε τμήματος με βάση το μοντέλο του καταρράκτη (ακολουθιακή ανάπτυξη), σε συνδυασμό με την τμηματική ανάπτυξη του μοντέλου πρωτοτυποποίησης. Είναι μια κατάτμηση δηλαδή Λογισμικού σε ανεξάρτητα τμήματα και το κάθε τμήμα ακολουθεί ,ακολουθιακή ανάπτυξη.

Αρχικά βλέπουμε την ανάλυση και την σχεδίαση, σε ποια τμήματα δηλαδή θα κατατμηθεί η εφαρμογή. Μετά επέρχεται ανεξάρτητα και παράλληλα, η ανάπτυξη των τμημάτων, και τέλος παρατηρούμε την ενσωμάτωση κάθε τμήματος μετά την ολοκλήρωση της ανάπτυξης του στο σύνολο της εφαρμογής η οποία ονομάζεται «λειτουργική επαύξηση». Στο σχήμα 2.4.1 βλέπουμε αναλυτικά το μοντέλο της λειτουργικής επαύξησης.

Σχήμα 2.4.1 Το μοντέλο της λειτουργικής επαύξησης



Χρησιμοποιείται σε μεγάλες εφαρμογές λογισμικού, για τις οποίες ισχύουν οι απαιτήσεις του καταρράκτη και φυσικά υπάρχει μικρή μεταβλητότητα των απαιτήσεων κατά την ανάπτυξη.

Στα πλεονεκτήματα του μοντέλου αυτού παρατηρούμε ότι υπάρχει δυνατότητα παράλληλης ανάπτυξης αλλά και διαδοχικός εμπλουτισμός λειτουργικών χαρακτηριστικών. Επιπλέον όμως παρατηρούμε και μειονεκτήματα. Η αρχική κατάτμηση και γενική σχεδίαση του συστήματος είναι σημαντική και γι ’αυτό αν υπάρξει κάποιο σφάλμα κατά την διάρκεια της σχεδίασης μπορεί να επιφέρει επιπτώσεις στο λογισμικό. Ακόμη μπορεί να προκύψει πρόβλημα λόγω ακατάλληλης κατάτμηση εφαρμογής, αν έχουμε μεταβολή λειτουργικών απαιτήσεων κατά την χρήση του ημιτελούς συστήματος.

2.5 Το Σπειροειδές Μοντέλο

Το σπειροειδές μοντέλο (σχήμα 2.5.1) είναι η ανάπτυξη συστήματος που χωρίζεται σε κύκλους εργασιών με σταδιακή επέκταση των λειτουργικών χαρακτηριστικών της εφαρμογής. Μπορούμε να δούμε και να αναλύσουμε τους κινδύνους πριν από κάθε έναρξη του κύκλου. Είναι ένας συνεχής καθορισμός λεπτομερειών υλοποίησης (σε σύγκριση με τα προηγούμενα μοντέλα).

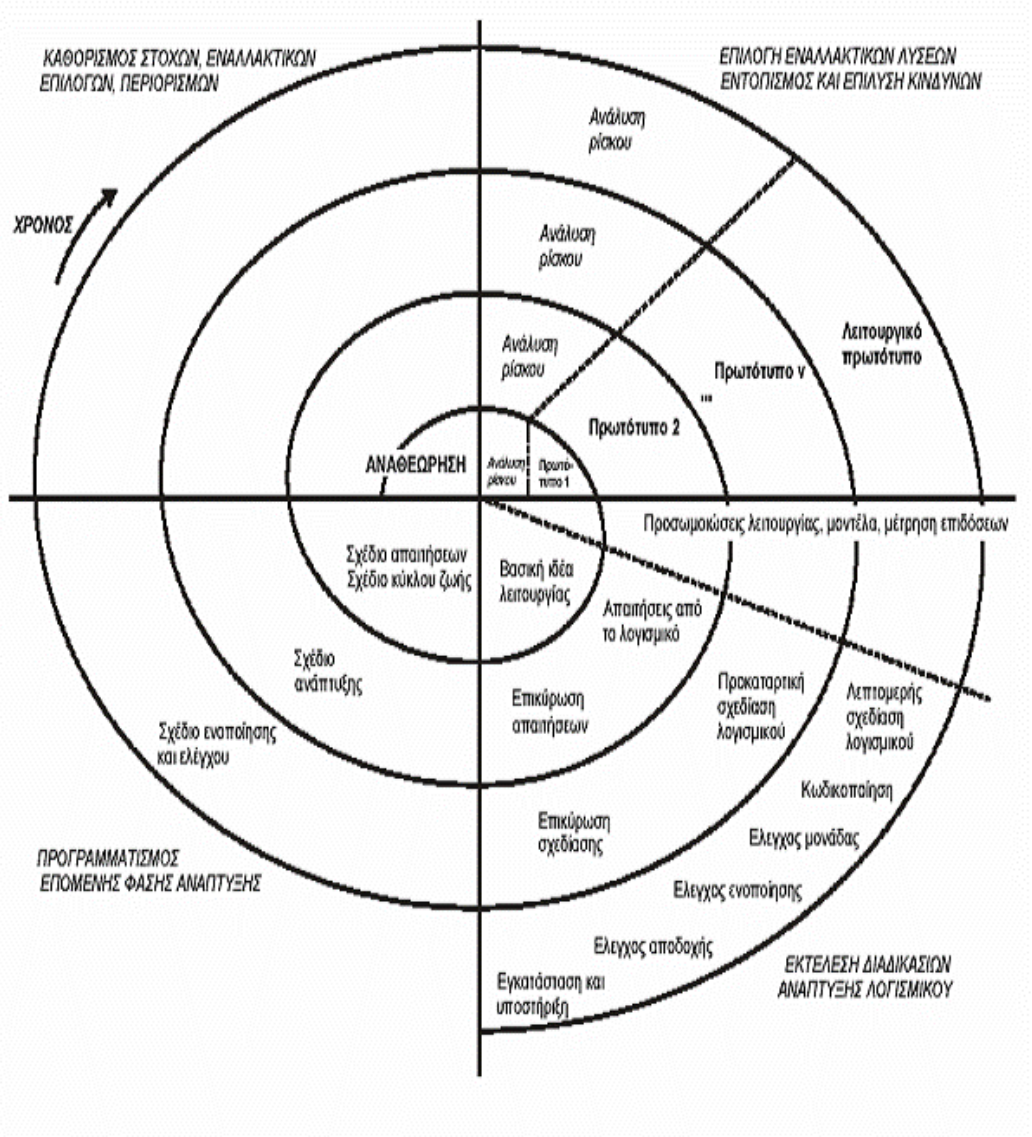
Το μοντέλο αυτό είναι η γενίκευση των μοντέλων ΠΡΩΤΟΤΥΠΟΙΗΣΗΣ + ΛΕΙΤΟΥΡΓΙΚΗΣ ΕΠΑΥΞΗΣΗΣ . Θα δούμε ότι :

► Οι φάσεις και οι διαδικασίες ανάπτυξης λογισμικού δεν είναι προκαθορισμένες από το μοντέλο αλλά εξειδικεύονται στο χώρο εφαρμογής του.

►Η ανάπτυξη ολόκληρου του συστήματος χωρίζεται σε πολλούς κύκλους και σε κάθε ένα προστίθενται λειτουργικά χαρακτηριστικά

► Πριν από κάθε κύκλο, γίνεται μια μελέτη σκοπιμότητα και ανάλυσης κινδύνων κατά την οποία προκύπτουν συγκεκριμένες εργασίες που θα εκτελεστούν μέσα στον κύκλο, καθώς και η εφικτότητα εκτέλεσης του κύκλου.

Σχήμα 2.5.1 Το σπειροειδές μοντέλο



'Όπως φαίνεται στο πιο πάνω σχήμα 2.5.1 το σπειροειδές μοντέλο χωρίζεται σε τέσσερις κατηγορίες εργασιών :

1)Προσδιορισμός Στόχων

2)Επίλυση Κινδύνων

3)Εκτέλεση διαδικασιών ανάπτυξης λογισμικού

4) Προγραμματισμός επόμενης φάσης ανάπτυξης

ΑΝΑΛΥΤΙΚΑ

1) ΠΡΟΣΔΙΟΡΙΣΜΟΣ ΣΤΟΧΩΝ

►Καθορισμός αντικείμενων εργασιών κάθε επανάληψης

►Καταγραφή [περιορισμών για το προϊόν και την διαδικασία

►Κατασκευή αναλυτικού πλάνου διοίκησης

►Καταγραφή κινδύνων εργασίας

►Καταγραφή εναλλακτικών λύσεων αν υπάρχουν

2) ΕΠΙΛΥΣΗ ΚΙΝΔΥΝΩΝ

►Ανάλυση κινδύνων

►Αποτίμηση κάθε εναλλακτικής λύσης

►Λήψη αποφάσεων για συνέχιση ή όχι, ποιο μοντέλο θα ακολουθήσει, κατασκευή ή όχι πρωτοτύπου

3) ΕΚΤΕΛΕΣΗ ΔΙΑΔΙΚΑΣΙΩΝ ΑΝΑΠΤΥΞΗΣ ΛΟΓΙΣΜΙΚΟΥ

►Εκτέλεση βημάτων, όπως αποφασίστηκαν για το τμήμα του συστήματος που αφορά η τρέχουσα επανάληψη

4) ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΕΠΟΜΕΝΗΣ ΦΑΣΗΣ ΑΝΑΠΤΥΞΗΣ

►Επέρχεται μετά την επαλήθευση των αποτελεσμάτων και έτσι έχουμε την συνέχιση της ανάπτυξης

Η εφαρμογή του σπειροειδούς μοντέλου είναι δύσκολη. Σημαντικό είναι ο καθορισμός λεπτομερειών υλοποίησης να γίνεται συνεχώς κατά την ανάπτυξη με ευθύνη και τεκμηρίωση από τον κατασκευαστή. Δεν καθορίζει όμως ποιες εργασίες ανάπτυξης λογισμικού πρέπει να γίνουν και σε ποια έκταση του συστήματος θα εφαρμοστούν.

Ακόμη βλέπουμε ότι διαφορετικές διαδικασίες ανάπτυξης επιλέγονται για διαφορετικά τμήματα λογισμικού. Συγχρόνως παρατηρείται εισαγωγή νέων εργασιών που αφορούν τεκμηρίωση σκοπιμότητας και τμηματικό προγραμματισμό ανάπτυξης. Αυτό φυσικά συνεπάγεται με κάποιο

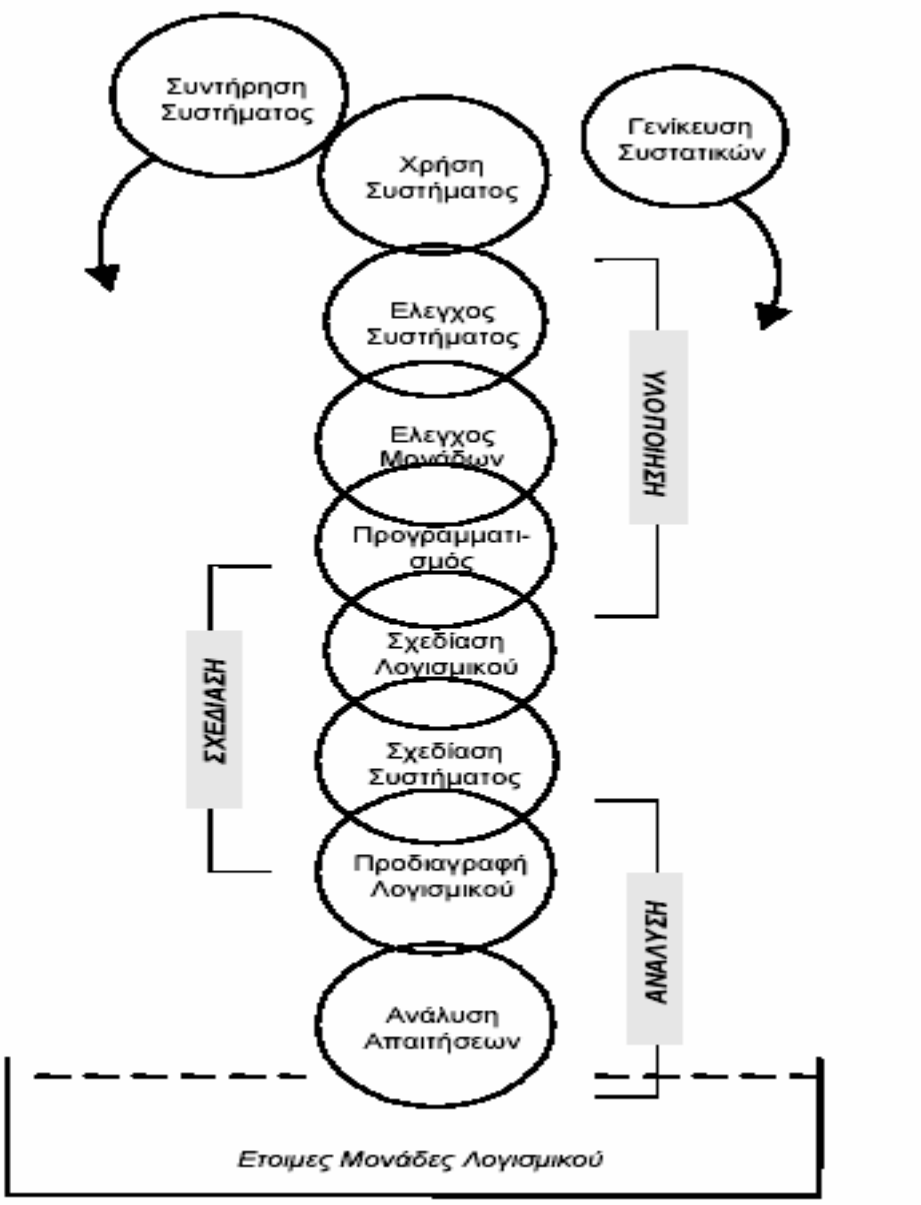
κόστος το οποίο μπορεί να απαλειφθεί όταν εντοπιστούν έγκαιρα τα προβλήματα και να αποφευχθούν πιθανές αποτυχίες.

Όσο αφορά τις μικρές εφαρμογές λογισμικού , το μοντέλο έχει δυσανάλογο κόστος, άρα δεν είναι και η καλύτερη επιλογή από οικονομική πλευρά.

2.6 Το μοντέλο του Πίδακα

Ένα άλλο μοντέλο που θα αναλύσουμε είναι το μοντέλο του πίδακα. Είναι μια παραλλαγή των προηγούμενων μοντέλων. Βασίζεται στην αντικειμενοστραφή τεχνολογία (object-oriented) και οι έννοιες «ανάλυση -σχεδίαση- κωδικοποίηση» έρχονται στο αντικειμενοστραφές παράδειγμα πολύ κοντά. Ακόμη το αποτέλεσμα κάθε διαδικασία κατασκευής λογισμικού είναι όχι μόνο ένα σύστημα αλλά και επαναχρησιμοποιημένες μονάδες.

Σχήμα 2.6.1 Το μοντέλο κύκλου ζωής του πίδακα, το οποίο βασίζεται στην αντικειμενοστραφή τεχνολογία ανάπτυξης λογισμικού.



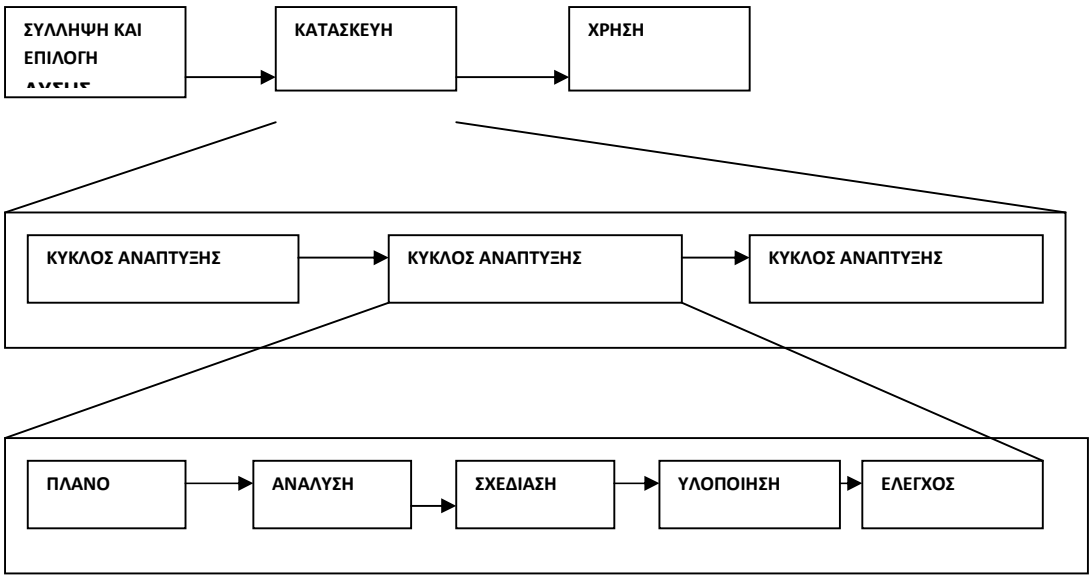
Κατά την ανάπτυξη του μοντέλου παρατηρούνται επικαλύψεις φάσεων «ανάλυση – σχεδίαση – κωδικοποίηση». Τέλος ορισμένα συστατικά λογισμικού ενσωματώνονται σε δεξαμενή συστατικών και διατίθενται για χρήση στην ανάπτυξη νέων συστημάτων.

Τέλος να επισημάνουμε ότι το μοντέλο του πίδακα τονίζει περισσότερο τα επιθυμητά χαρακτηριστικά της μεθοδολογίας κατασκευής του λογισμικού σύμφωνα με την αντικειμενοστρεφική λογική.

2.7 Σύγχρονα μοντέλα κύκλου ζωής λογισμικού

Τα σύγχρονά μοντέλα κύκλου ζωής λογισμικού, περιέχουν γενικές κατευθύνσεις εφαρμογής των ιδεών. Εξειδικεύονται στο εκάστοτε περιβάλλον ανάπτυξης και παρατηρείται μια ελευθερία στο κατασκευαστή λογισμικού. Χαρακτηριστικό του μοντέλου αυτού είναι η εξάλειψη ογκωδέστατων παραδοτέων με σχέδια και προδιαγραφές που στην ουσία όμως ήταν υποχρεωτικό να υπάρχουν. Περιέχει φάσεις σύλληψης, κατασκευής και λειτουργίας. Έτσι αναλύονται επιμέρους εργασίες αλλά και φάσεις κατασκευής σε «κύκλους ανάπτυξης, ο καθένας δηλαδή να προσθέτει νέα χαρακτηριστικά και λειτουργίες. Ακόμη τα μοντέλα δεν είναι συνδεδεμένα με συγκεκριμένη μεθοδολογία ανάπτυξης λογισμικού.

Σχέδιο 2.7.1 Ένα γενικό μοντέλο κύκλου ζωής το οποίο ενσωματώνει χαρακτηριστικά πολλών από τα μοντέλα που αναφέρθηκαν



Όλα τα μοντέλα κύκλου ζωής που αναφέρθηκαν μπορούν να θεωρηθούν ειδικές εκδοχές του γενικού μοντέλου κύκλου ζωής. Σε κάθε κύκλο ανάπτυξης παρατηρείται επανάληψη αλλά και παράλληλη εκτέλεση τμήματος έργου.

Η κατασκευή αναλύεται σε κύκλους ανάπτυξης, για τους οποίους στην αρχή μπορεί να προηγηθεί ένα πλάνο εργασιών και να εκτιμηθεί το ρίσκο και η σκοπιμότητα.

2.8 Περιγραφή διαδικασιών ανάπτυξης και προϊόντων Λογισμικού

Στην περιγραφή των διαδικασιών ανάπτυξης λογισμικού παρατηρούνται τρία διαδοχικά επίπεδα λεπτομέρειας, ανάλογα με το σημείο εστίασης της προσοχής του παρατηρητή.

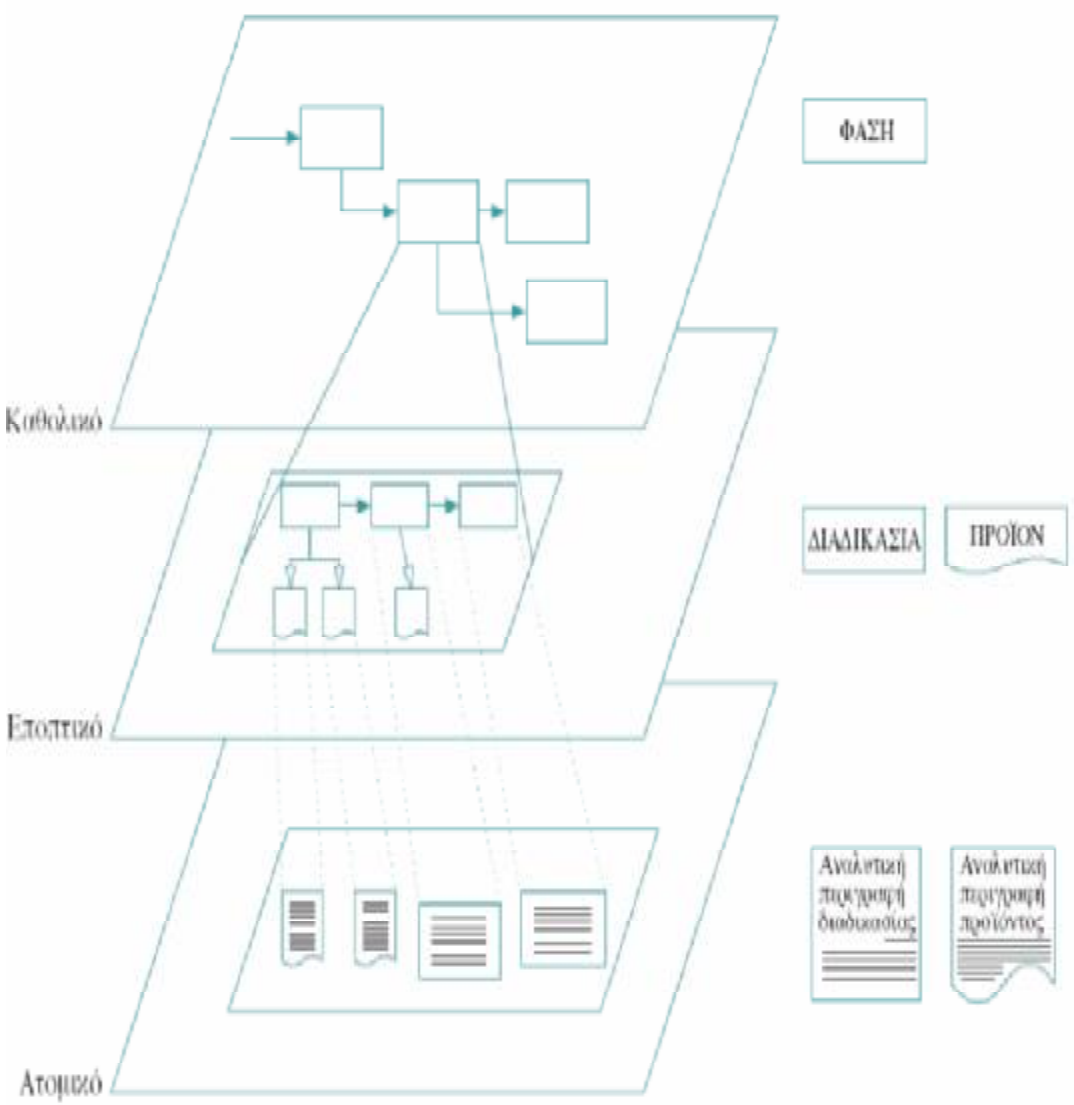
Τα τρία στάδια που διακρίνουμε είναι :

Α) το καθολικό : το οποίο έχει την γενική εποπτεία της διαδικασίας

Β) το εποπτικό : που για κάθε γενική αναφορά βήμα περιέχονται σε γενική αναφορά οι επιμέρους εργασίες των προϊόντων

Γ) το ατομικό : όπου όλες οι διαδικασίες και τα προϊόντα αποκτούν πλήρεις αναλυτικές περιγραφές.

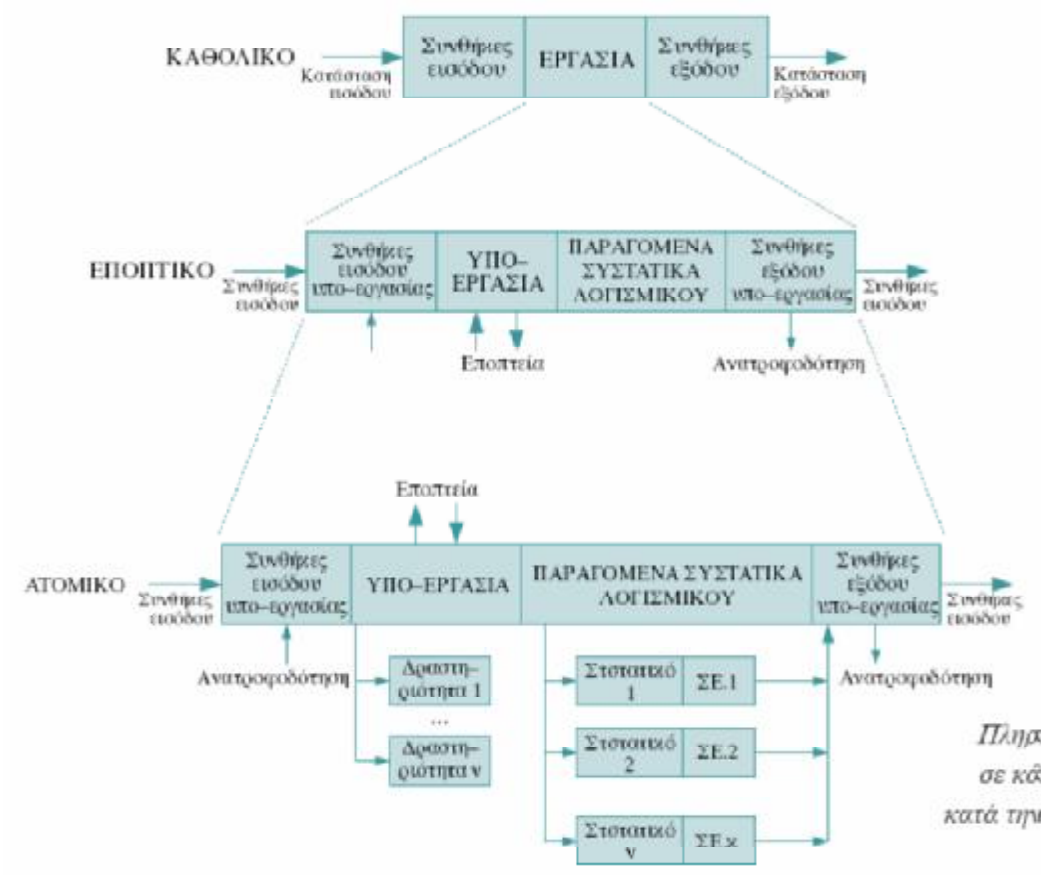
Σχήμα 2.8.1 Επίπεδα λεπτομέρειας στην περιγραφή διαδικασιών ανάπτυξης λογισμικού.



Τα επίπεδα λεπτομέρειας διακρίνονται ανάλογα με το σκοπό της περιγραφής, τους αποδέκτες αλλά και με τις εκάστοτε συνθήκες. Η περιγραφή σε κάθε επίπεδο είναι διαφορετική, για παράδειγμα στο ατομικό επίπεδο η περιγραφή είναι πιο λεπτομερής σε σύγκριση με το καθολικό επίπεδο.

Στο πιο κάτω σχήμα απεικονίζονται οι πληροφορίες που περιέχονται σε κάθε επίπεδο λεπτομέρειας κατά την περιγραφή διαδικασιών ανάπτυξης λογισμικού. Παρατηρούμε ότι ανάλογα με την φύση των εργασιών που γίνονται στην φάση του κύκλου ζωής, υπάρχει η δυνατότητα λήψης και παραγωγής πληροφοριών ανατροφοδότησης (feedback) προς άλλες φάσεις του κύκλου ζωής.

Σχήμα 2.8..2 Πληροφορίες που παρέχονται σε κάθε επίπεδο λεπτομέρειας κατά την περιγραφή διαδικασιών ανάπτυξης λογισμικού



ΣΥΝΟΨΗ

Συνοπτική παράθεση χαρακτηριστικών των μοντέλων κύκλου ζωής λογισμικού

Μοντέλο	**Μέγεθος Εφαρμογών**	**Μεταβολές στις Απαιτήσεις**	**Προσαρμοστικότητα Στον κατασκευαστή**	**Διάδοση**
Καταρράκτη	Μικρό έως μεσαίο	Ανεπιθύμητες	Καμία	Μεγάλη με Τάση μείωσης
Πρωτοτυποποιήσης	Μικρό έως μεσαίο	Δεκτές	Μικρή	Μικρή με τάση αύξησης
Λειτουργικής επαύξησης	Μικρό έως μεγάλο	Ανεπιθύμητες	Καμία	Μικρή με τάση μείωσης
Σπειροειδές	Μικρό έως μεγάλο	Δεκτές	Αρκετή	Μικρή με τάση μείωσης
Πίδακα	Οποιοδήποτε	Δεκτές	Αρκετή	Μικρή
Γενικό	Οποιοδήποτε	Δεκτές	Μεγάλη	Μικρή με ισχυρές τάσεις αύξησης

Το μοντέλο ζωής λογισμικού περιγράφει τις φάσεις από τις οποίες διέρχεται η εφαρμογή λογισμικού μέχρι την απόσυρση της και τις ενέργειες που λάμβαναν χώρα σε κάθε μια από αυτές. Τα μοντέλα διακρίνονται σε ακολουθιακά και σε επαναληπτικά. Στα ακολουθιακά μοντέλα η ανάπτυξη γίνεται σε διαδοχικές διακριτές φάσεις και για ολόκληρο το σύστημα λογισμικού, ενώ στα επαναληπτικά είναι το σπειροειδές. Πρακτικά, χρησιμότερα είναι τα μοντέλα του κύκλου ζωής που αφήνουν ελευθερία εξειδίκευσης στις εκάστοτε συνθήκες και δεν προσδιορίζουν με αυστηρότητα τις ενέργειες που πρέπει να γίνουν, τα προϊόντα κτλ. Τέλος δεν υπάρχει ένα καλύτερο μοντέλο ζωής , αλλά ένα καταλληλότερο στις εκάστοτε συνθήκες τόσο του κατασκευαστή όσο και του θεαματικού πεδίου εφαρμογής του λογισμικού.

Κεφάλαιο 3:Προδιαγραφή Απαιτήσεων

Περιεχόμενα

3.2. Διαγράμματα οντοτήτων-συσχετίσεων

3.6.Συμβολισμοί Διαγραμμάτων Ροής Δεδομένων

3.7. Διαγράμματα Οντοτήτων- Συσχετίσεων

3.8.Διαγράμματα μετάβασης καταστάσεων

3.9.Λεξικό Δεδομένων

3.10.Προβλήματα στον προσδιορισμό των απαιτήσεων

3.10.1.Επικοινωνίας

3.10.2.Προτύπων

3.10.3.Γλώσσας

3.10.4.Οικονομικά

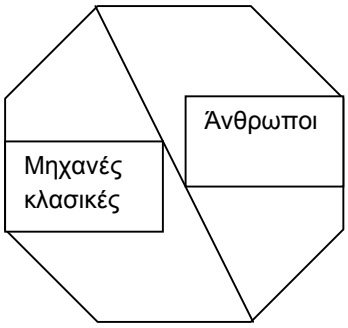
3.11. Έρευνες

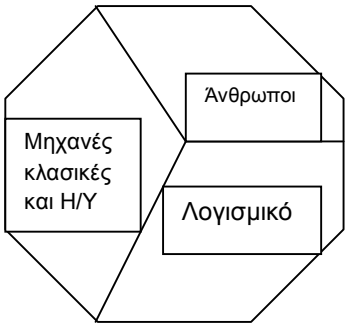
Σκοπός του κεφαλαίου είναι να αντιληφθούμε την έννοια των απαιτήσεων του λογισμικού, να εξάγουμε απαιτήσεις και προδιαγραφές των απαιτήσεων και να δημιουργήσουμε μοντέλα παράστασης λογισμικού με βάση τη δομημένη ανάλυση απαιτήσεων η οποία βασίζεται σε:

- διαγράμματα ροής δεδομένων
- διαγράμματα οντοτήτων-συσχετίσεων
- διαγράμματα μετάβασης καταστάσεων

Έννοια των απαιτήσεων:

Σε ένα σύστημα ενός πραγματικού κόσμου μπορούμε να διακρίνουμε 2 συνιστώσες τους ανθρώπους και τις μηχανές. Με το όρο μηχανές αναφερόμαστε στις κλασσικές μηχανές με εξαίρεση τον ηλεκτρονικό υπολογιστή ο οποίος δεν θεωρείται ότι έχει υπόσταση παρά μόνο με τη βοήθεια του λογισμικού. Σε τέτοια περίπτωση, όταν στο σύστημά μας υπάρχει ηλεκτρονικός υπολογιστής, θεωρούμε σκόπιμο να πούμε ότι υπάρχει και μια τρίτη συνιστώσα, αυτή του λογισμικού. Χαρακτηριστικά είναι τα παρακάτω σχήματα





Διακρίνεται σε απαιτήσεις συστήματος και λογισμικού

1. Απαίτηση από το σύστημα

Ορισμός: είναι μια περιγραφή μιας εργασίας που θα πρέπει να εκτελείται από ένα παράγοντα του συστήματος π.χ. άνθρωπο, μηχανή ή λογισμικό ή ενός χαρακτηριστικού το οποίο θα πρέπει να έχει το σύστημα. Είναι μια σύνθετη εργασία και αρκετές απαιτήσεις από το σύστημα μπορεί να σχετίζονται άμεσα ή έμμεσα με τις απαιτήσεις από το λογισμικό. Πολλές φορές είναι δύσκολη η διάκριση τους.

2. Απαίτηση από το λογισμικό

Ορισμός: Μια λειτουργία που θα πρέπει να επιτελεί ή μια συνθήκη που θα πρέπει να ικανοποιεί όταν θα έχει ολοκληρωθεί η κατασκευή του λογισμικού.

Όταν πρόκειται να κατασκευαστεί ένα λογισμικό θα πρέπει πρώτα από όλα να αντιληφθούμε επακριβώς το τι εργασία αυτό θα επιτελεί καθώς και άλλα δευτερεύοντα χαρακτηριστικά του όπως εμφάνιση, τρόπος χρήσης, επίδοση κτλ. Λογικό είναι ότι λανθασμένη αντίληψη των απαιτήσεων του πελάτη για το λογισμικό που πρόκειται να κατασκευαστεί, οδηγεί στην κατασκευή λογισμικού το οποίο δεν θα επιτελεί το σκοπό του.

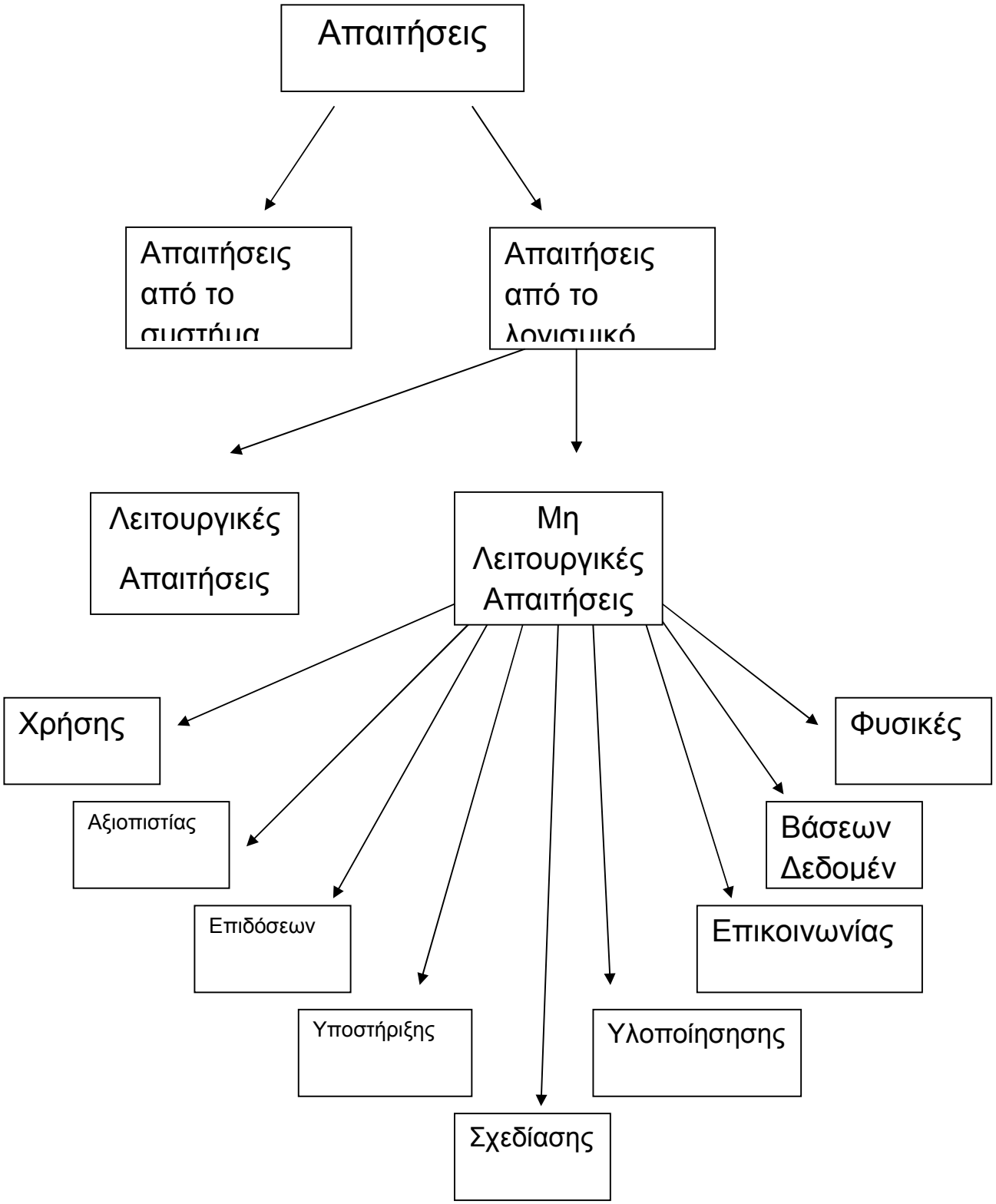
Ο πελάτης εκφράζει τις απαιτήσεις του για το λογισμικό με μεγάλο βαθμό γενικότητας και ο κατασκευαστής θα πρέπει να τις διατυπώσει με περισσότερες λεπτομέρειες και ακρίβεια ώστε να μπορέσει να φτιάξει το κατάλληλο λογισμικό αναλόγως των απαιτήσεων του πελάτη.

Οι απαιτήσεις λογισμικού ταξινομούνται σε λειτουργικές απαιτήσεις και μη λειτουργικές απαιτήσεις. Οι λειτουργικές απαιτήσεις περιγράφουν τις λειτουργίες που θα εκτελεί το λογισμικό. Οι μη λειτουργικές απαιτήσεις περιγράφουν χαρακτηριστικά που πρέπει να έχει το λογισμικό τα οποία δεν αφορούν την εκτέλεση κάποιας λειτουργίας από αυτό. Οι λειτουργικές απαιτήσεις καθορίζουν την συμπεριφορά του συστήματος δηλαδή τα επιθυμητά αποτελέσματα που θα πρέπει να παράγει το λογισμικό. Οι μη λειτουργικές απαιτήσεις καθορίζουν την εμφάνιση του λογισμικού, επιδόσεων, υλοποίησης κτλ τα οποία γενικά καθορίζουν το λογισμικό χωρίς όμως να θεωρούνται ως λειτουργίες που αυτό θα επιτελεί.

Οι μη λειτουργικές απαιτήσεις διακρίνονται σε διάφορες κατηγορίες όπως:

- Απαιτήσεις Χρήσης
- Απαιτήσεις Αξιοπιστίας
- Απαιτήσεις Επιδόσεων
- Απαιτήσεις Υποστήριξης
- Απαιτήσεις Σχεδίασης
- Απαιτήσεις Υλοποίησης
- Απαιτήσεις Επικοινωνίας με άλλα συστήματα
- Απαιτήσεις Βάσεων δεδομένων
- Φυσικές Απαιτήσεις

Ο διαχωρισμός των απαιτήσεων και η ταξινόμηση των απαιτήσεων λογισμικού που αναφέραμε παραπάνω φαίνεται παραστατικά στο παρακάτω σχήμα:



3.1.Μηχανική των απαιτήσεων

Ορισμός: Η γενική διαδικασία ανάλυσης και προσδιορισμού απαιτήσεων του λογισμικού, δηλαδή ο τρόπος επίλυσης του προβλήματος προσδιορισμού των απαιτήσεων που προτείνει κάθε διαφορετική προσέγγιση σ' αυτό.

Ένα πρόβλημα προσεγγίζεται με διαφορετικούς τρόπους κάθε ένας εκ των οποίων προτείνει δικά του βήματα και μεθοδολογίες για επίλυση του. Ο τρόπος επίλυσης του προβλήματος προσδιορισμού των απαιτήσεων που προτείνει κάθε ένας από αυτούς τους τρόπους προσέγγισης αναφέρεται ως μηχανική των απαιτήσεων.

Μερικά από τα σημαντικότερα προβλήματα που συναντά κανείς όταν πρόκειται να αντιμετωπίσει ένα πρόβλημα προσδιορισμού απαιτήσεων από το λογισμικό είναι το πλήθος και η πολυπλοκότητα που χαρακτηρίζει πολλές από τις απαιτήσεις από το λογισμικό, ο ικανοποιητικός προσδιορισμός και αντίληψη των συσχετίσεων μεταξύ αυτών των δύο και τέλος τα διαφορετικά επίπεδα στην περιγραφή τους.

Η προσέγγιση αυτή γίνεται με τη μέθοδο της δομημένης ανάλυσης. Είναι μια μέθοδος ανάλυσης που χρησιμοποιείται περισσότερο από είκοσι χρόνια. Είναι ανάλυση προσαρμοσμένη στη ροή των δεδομένων. Το μοντέλο εστιάζει στην περιγραφή της ροής των δεδομένων προς, και μέσα από το σύστημα που πρόκειται να αναπτύξουμε. Πλεονέκτημα της είναι το ότι καταφέρνει να μειώσει την πολυπλοκότητα του προβλήματος που αντιμετωπίζουμε. Τέλος βασίζεται στα διαγράμματα ροής δεδομένων και στα λεξικά δεδομένων τα οποία θα αναπτύξουμε παρακάτω.

3.1.1.Βήματα στον προσδιορισμό των απαιτήσεων

1. **Μελέτη του εγγράφου απαιτήσεων από το σύστημα ή και των αναγκών του πελάτη.**

Συνήθως αυτό δίνεται με τη μορφή έκθεσης αναγκών από τον πελάτη. Στοχεύει στην αρχική κατανόηση του προβλήματος για την επίλυση του οποίου καλείται να χρησιμοποιηθεί το υπό κατασκευή λογισμικό.

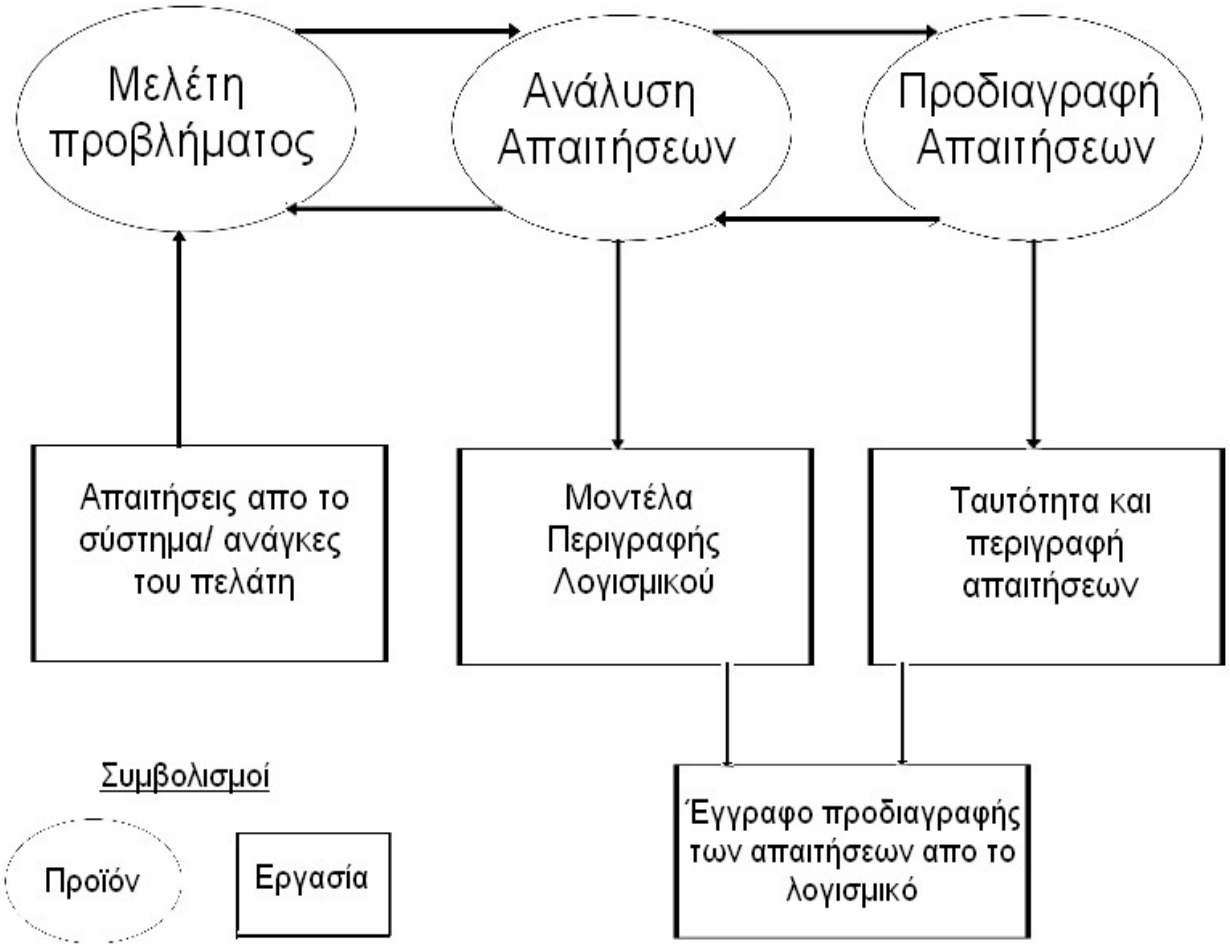
2. **Ανάλυση των απαιτήσεων**

Στοχεύει στη δημιουργία μοντέλων που περιγράφουν διαφορετικές πλευρές του λογισμικού. Τα μοντέλα αυτά παρουσιάζονται με τη βοήθεια διαγραμμάτων ροής δεδομένων, οντοτήτων-συσχετίσεων και μετάβασης καταστάσεων. Επίσης χρησιμοποιούμε και ένα πίνακα λεξικού δεδομένων.

3. **Διάκριση και προδιαγραφή κάθε συγκεκριμένης απαίτησης από το λογισμικό**

Συμπληρώνεται έγγραφο «ΠΡΟΔΙΑΓΡΑΦΕΣ ΤΩΝ ΑΠΑΙΤΗΣΕΩΝ ΑΠΟ ΤΟ ΛΟΓΙΣΜΙΚΟ» το οποίο είναι το επιθυμητό αποτέλεσμα της διαδικασίας που ακολουθήσαμε. Το έγγραφο αυτό περιγράφει με λεπτομέρεια τις απαιτήσεις από το λογισμικό τις ταξινομεί και τις ιεραρχεί. Το έγγραφο αυτό συμφωνεί πλήρως με τα διαγράμματα που έχουν δημιουργηθεί στο προηγούμενο βήμα.

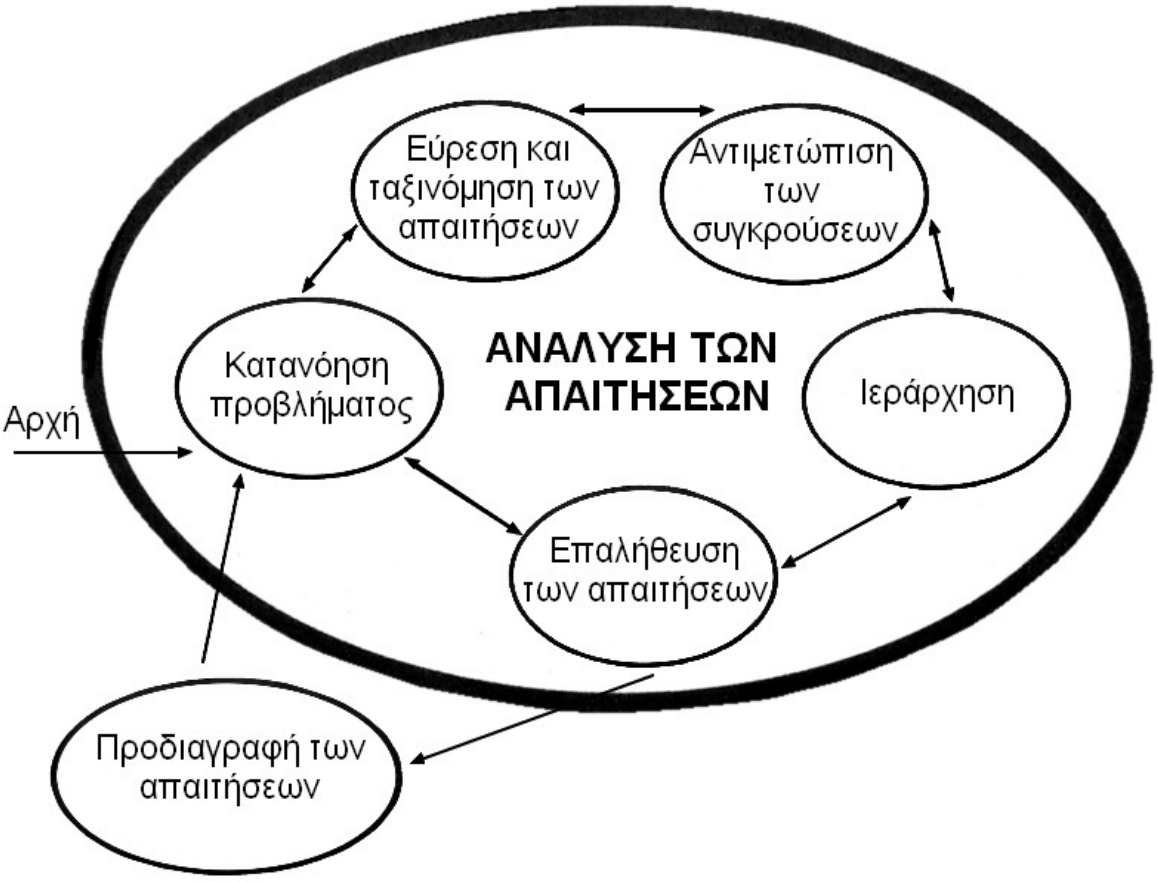
Η γενική μορφή της διαδικασίας προσδιορισμού των απαιτήσεων από το λογισμικό παρουσιάζεται ξεκάθαρα στο παρακάτω σχήμα:



3.2.Ανάλυση απαιτήσεων

Κατά την ανάλυση απαιτήσεων εντοπίζονται για πρώτη φορά οι απαιτήσεις από το λογισμικό και ακολουθούν έναν κύκλο ταξινόμησης, ιεράρχησης και επαλήθευσης.

Χαρακτηριστικό είναι το σχήμα παρακάτω:



3.2.1.Κατανόηση προβλήματος: κατά την διαδικασία αυτή ο αναλυτής εισάγεται ο ίδιος στην ουσία του προβλήματος όσο πιο πολύ γίνεται. Συνήθως η έκθεση του πελάτη για το τι απαιτήσεις έχει δεν είναι επαρκής για να αντιληφθεί πλήρως ο κατασκευαστής την ουσία του αντικειμένου σχετικά με το οποίο θα αναπτυχθεί λογισμικό. Η ανάπτυξη του λογισμικού που αφορά την επίλυση του προβλήματος του πελάτη δεν είναι εφικτή αν ο κατασκευαστής δεν εξοικειωθεί πλήρως με την ουσία του προβλήματος, στο αρχικό στάδιο ανάπτυξης της διαδικασίας που θα ακολουθήσει.

3.2.2.Ευρεση και ταξινόμηση των απαιτήσεων: συλλέγονται οι απαιτήσεις που έχουν σχέση με το λογισμικό και γίνεται μια καταγραφή τους σε λίστα. Η λίστα αυτή που δημιουργείται περιέχει τις απαιτήσεις και τις πληροφορίες που χρειάζονται για το λογισμικό και είναι «ατάκτως

κατανεμημένες» . Έπειτα αυτές οι απαιτήσεις ταξινομούνται σε κατηγορίες ανάλογα με το υποσύνολο του προβλήματος που αφορούν.

3.2.3.Αντιμετώπιση των συγκρούσεων: σε αυτό το στάδιο γίνεται επίλυση των συγκρούσεων, η οποία μπορεί να φέρει επαφές ξανά με τον πελάτη για να βρεθεί λύση. Π.χ. μπορεί κάποιες απαιτήσεις να μην μπορούν να ικανοποιηθούν ταυτόχρονα έτσι να χρειάζεται επίλυση.

3.2.4.Ιεράρχηση: όταν τελικά επιλυθούν οι συγκρούσεις, οι απαιτήσεις αυτές ιεραρχούνται σε μια σειρά προτεραιότητας ως προς τη σειρά ικανοποίησης τους. Αυτή η σειρά καθορίζει την χρονική αλληλουχία με την οποία θα ενσωματωθούν στο λογισμικό οι λειτουργίες που θα ικανοποιούν τις απαιτήσεις του πελάτη αλλά και ποιες από αυτές δεν θα ικανοποιούν καθόλου τις απαιτήσεις.

3.2.5.Επαλήθευση των απαιτήσεων: Η όλη διαδικασία ολοκληρώνεται σε αυτό το στάδιο. Για να γίνει όμως ολοκλήρωση θα πρέπει να υπάρξει μια νέα επαφή με τον πελάτη με τη μορφή σύσκεψης και με ανταλλαγή εγγράφων. Όταν δεν ικανοποιηθεί ο πελάτης, τότε είναι αναγκαίο να τροποποιήσουμε ξανά τα επιμέρους βήματα που ακολουθήσαμε. Υπάρχουν και οι περιπτώσεις στις οποίες χρειάζεται να επανέλθουμε στο στάδιο της κατανόησης του προβλήματος και να ξεκινήσουμε από την αρχή την όλη διαδικασία.

3.2.6.Επιθυμητά χαρακτηριστικά των απαιτήσεων:

- **Ορθότητα:** οι απαιτήσεις να έχουν καταγραφεί χωρίς λάθη.
- **Συνέπεια:** δεν υπάρχουν αλληλοσυγκρουόμενες απαιτήσεις.
- **Πληρότητα:** περιγράφουν όλες τις πιθανές καταστάσεις, αλλαγές καταστάσεων, είσοδοι, προϊόντα και περιορισμοί.
- **Ρεαλισμός:** να μπορούν να υλοποιηθούν
- Αναγκαιότητα: περιγράφουν κάτι που πραγματικά χρειάζεται ο πελάτης.
- **Επαληθευσιμότητα:** είναι δυνατός ο έλεγχος ότι οι απαιτήσεις πληρούνται εξ΄ ολοκλήρου.
- **Ιχνηλασιμότητα:** είναι εύκολο να βρεθεί το σύνολο των απαιτήσεων που σχετίζονται με μια συγκεκριμένη άποψη του συστήματος.

3.3.Προδιαγραφή των απαιτήσεων

Με την έννοια προδιαγραφή αναφερόμαστε στη δομημένη και λεπτομερή περιγραφή των απαιτήσεων από το λογισμικό η οποία γίνεται με τη μορφή γραπτού λόγου και, όπου απαιτείται, διαγραμμάτων.

3.3.1.Επιθυμητά χαρακτηριστικά του εγγράφου προδιαγραφών:

- Θα πρέπει να περιγράφει την συμπεριφορά του λογισμικού προς το εξωτερικό του περιβάλλον και όχι εσωτερικά του στοιχεία.
- Θα πρέπει να περιγράφει όλους τους περιορισμούς που αφορούν την ανάπτυξη του λογισμικού.
- Θα πρέπει να είναι εύκολο να αλλαχτεί.
- Θα πρέπει να είναι χρήσιμο στη συντήρηση του λογισμικού.
- Θα πρέπει να περιγράφει τη συμπεριφορά του λογισμικού σε ανεπιθύμητες καταστάσεις

3.3.2.Γενική δομή εγγράφου προδιαγραφών των απαιτήσεων από το λογισμικό:

Έγγραφο προδιαγραφών απατήσεων

1. Εισαγωγή
   - α. Σκοπός
   - β.Έκταση
   - γ.Ορισμοί, ακρονυμίες και συντομογραφίες
   - δ.Αναφορές
   - ε.Γενική εικόνα
2. Γενική περιγραφή
   - α.Προοπτική του προϊόντος
   - β.Λειτουργίες
   - γ.Χαρακτηριστικά των χρηστών
   - δ.Γενικοί περιορισμοί
   - ε.Παραδοχές και εξαρτήσεις
3. Ειδικές απαιτήσεις
   - α.Λειτουργικές απαιτήσεις
     - ι.Λειτουργική απαίτηση 1
       1. Εισαγωγή
       2. Είσοδοι
       3. Επεξεργασία
       4. Έξοδοι
     - ιι.Λειτουργική απαίτηση 2
     - ...
     - ν.Λειτουργική απαίτηση ν
   - β. Απαιτήσεις εξωτερικών διαπροσωπειών
     - ι. Διαπροσωπείες χρήστη
     - ιι.Διαπροσωπείες υλικού
     - ιιι.Διαπροσωπείες λογισμικού
     - ιν.Διαπροσωπείες επικοινωνιών
   - γ.Απαιτήσεις επίδοσης
   - δ.Περιορισμοί σχεδίασης
   - ι.Συμμόρφωση με τα πρότυπα
   - ιι.Περιορισμοί από το υλικό
   - ε.Ιδιώματα
     - ι.Διαθεσιμότητα
     - ιι.Ασφάλεια
     - ιιι.Συντηρισιμότητα
     - ιν.Μεταφερσιμότητα
   - ζ. Άλλες απαιτήσεις
     - ι.Βάση δεδομένων
     - ιι.Τρόποι λειτουργίας
     - ιιι.Προσαρμογή στο χώρο εγκατάστασης
4. Παραρτήματα
5. Ευρετήριο

à Δεδομένου ότι έχουν ολοκληρωθεί όλα τα παραπάνω βήματα επιτυχώς, επόμενο βήμα είναι η ανάπτυξη/ δημιουργία του λογισμικού δηλαδή του προγράμματος που πρόκειται να δώσει λύση στο πρόβλημα/ απαίτηση του πελάτη.

3.3.3.Τι είναι προγραμματιστικό περιβάλλον;

Ένα πρόγραμμα είναι ένα σύνολο εντολών σε κάποια γλώσσα προγραμματισμού που έχει ως σκοπό της επίλυση ενός προβλήματος με τη βοήθεια ενός υπολογιστή. Υπάρχουν αρκετές γλώσσες προγραμματισμού κάθε μια από αυτές έχει διαφορετικό σύνολο εντολών. Ο υπολογιστής όμως μπορεί να αναγνωρίσει και να εκτελέσει μόνο εντολές σε γλώσσα μηχανής, προκειμένου να γίνει μετάφραση του προγράμματος από τη γλώσσα που χρησιμοποιεί ο προγραμματιστής στη γλώσσα μηχανής. Το εργαλείο λογισμικού με την βοήθεια του οποίου πραγματοποιείται η διαδικασία αυτή είναι ένα προγραμματιστικό περιβάλλον.

3.3.4.Κύκλος ανάπτυξης προγράμματος.

Η διαδικασία που ακολουθούμε για την ανάπτυξη ενός προγράμματος ονομάζεται κύκλος ανάπτυξης προγράμματος. Πρόκειται για μια σειρά από ενέργειες που πρέπει να γίνουν έτσι ώστε το τελικό αποτέλεσμα που θα προκύψει, να ικανοποιεί το σκοπό για τον οποίο κατασκευάστηκε. Ο κύκλος ανάπτυξης ενός προγράμματος αναλύεται σε 5 βασικά βήματα:

1. Περιγραφή και κατανόηση του προβλήματος

2. Σχεδίαση της λύσης του προβλήματος

3. Ανάπτυξη προγράμματος

4. Εκφαλμάτωση του προγράμματος

5. Συντήρηση του προγράμματος

3.3.4.1.Περιγραφή και κατανόηση του προβλήματος: Όπως είπαμε και παραπάνω σκοπός του προγράμματος είναι η επίλυση του προβλήματος που αντιμετωπίζουμε, το οποίο μπορεί να είναι ειδικό ή γενικό. Πριν κατασκευαστεί ένα πρόγραμμα πρέπει να αφιερωθεί χρόνος στη μελέτη και κατανόηση του προβλήματος. Ακόμα και ένα φαινομενικά απλό πρόβλημα μπορεί να κρύβει ανακρίβειες οι οποίες μπορεί να είναι δυνατό να προκαλέσουν προβλήματα στην εξέλιξη της ανάπτυξης του προγράμματος. Για παράδειγμα ας υποθέσουμε ότι ο πελάτης μας ανήκει σε μια εταιρία και ζητάει ένα πρόγραμμα παρακολούθησης των προϊόντων της αποθήκης του. Κάποια ερωτήματα που μπορούν να τεθούν είναι:

- Ποια είδη φυλάσσονται στις αποθήκες και σε τι ποσότητες;
- Ποιος είναι ο όγκος των καθημερινών εξαγωγών από την αποθήκη;
- Πόσες αποθήκες υπάρχουν;
- Ο πελάτης θέλει να γνωρίζει τη συνολική ποσότητα των ειδών που βρίσκονται στην αποθήκη και την αξία τους
- Σε τι υπολογιστικό σύστημα θα εγκατασταθεί το πρόγραμμα;
- Ποίος θα το χειρίζεται;

Αφού απαντηθούν οι παραπάνω ερωτήσεις θα μπορέσουμε να καθορίσουμε τις προδιαγραφές που πρέπει να ικανοποιεί το πρόγραμμα που θα φτιάξουμε για επίλυση του προβλήματος του πελάτη και

να απαντήσουμε σε τυχόν απορίες που μπορεί να προκύψουν, να έχουμε δηλαδή αποσαφηνίσει τις απαιτήσεις του πελάτη.

3.3.4.2.Σχεδίαση της λύσης του προβλήματος: Πριν αναπτύξουμε ένα πρόγραμμα θα πρέπει να αποτυπώσουμε σε ένα χαρτί τη λύση του προβλήματος. Η περιγραφή της λύσης του προβλήματος γίνεται με ένα αλγόριθμο, ο οποίος είναι η περιγραφή της ακολουθίας βημάτων που οδηγούν στην επίλυση ενός προβλήματος. Για την περιγραφή της λύσης ενός προβλήματος πιο διαδεδομένες τεχνικές είναι το λογικό διάγραμμα και ο ψευδοκώδικας. Το λογικό διάγραμμα είναι ένας σχηματικός τρόπος αναπαράστασης της ροής των οδηγιών που συνθέτουν ένα αλγόριθμο. Οι οδηγίες συμβολίζονται με απλά σχήματα ενώ η σειρά εκτέλεσης των οδηγιών υποδεικνύονται με βέλη που συνδέουν τα σχήματα αυτά. Στον ψευδοκώδικα ο αλγόριθμος εκφράζεται με απλές προστακτικές φράσεις σε φυσική γλώσσα.

3.3.4.3.Ανάπτυξη προγράμματος: Στο στάδιο αυτό πραγματοποιείται η συγγραφή του προγράμματος σε μια γλώσσα προγραμματισμού. Το πρόγραμμα εισάγεται σε ένα πρόγραμμα διόρθωσης κειμένου. Στη συνέχεια ειδικό πρόγραμμα αναλαμβάνει να μεταφράσει το κείμενο του προγράμματος σε γλώσσα μηχανής. Το πρόγραμμα αυτό ενημερώνει τον προγραμματιστή για λάθη τα οποία εντόπισε. Αφού ο χρήστης διορθώσει τα λάθη μετατρέπει το πρόγραμμα σε κώδικα μηχανής το οποίο μπορεί να εκτελεστεί στον υπολογιστή.

3.3.4.4. Εκφαλμάτωση του προγράμματος: Στη συνέχεια ο προγραμματιστής δοκιμάζει το πρόγραμμα για να βεβαιωθεί ότι δουλεύει σωστά, ότι δηλαδή ανταποκρίνεται στις προδιαγραφές που έχουν καθοριστεί. Τα λάθη που εντοπίζονται στη φάση αυτή ονομάζονται λογικά, ενώ τα λάθη που εμφανίζονται στο στάδιο της ανάπτυξης ονομάζονται συντακτικά. Αν ο προγραμματιστής διαπιστώσει την ύπαρξη λογικών λαθών επιστρέφει στην προηγούμενη φάση τροποποιεί το πρόγραμμα και στη συνέχεια το μετατρέπει πάλι σε γλώσσα μηχανής για να ελεγχθεί ξανά. Η φάση αυτή είναι πολύ σημαντική γιατί όσο λεπτομερής δουλειά και να έχει γίνει στις προηγούμενες φάσεις είναι πρακτικά αδύνατο να μην εντοπιστεί ούτε ένα λογικό λάθος στη φάση αυτή. Γι αυτό και οι προγραμματιστές δοκιμάζουν διεξοδικά τα προγράμματα που κατασκευάζουν.

3.3.4.5. Συντήρηση του προγράμματος: Θεωρώντας ότι το λογισμικό λειτουργεί σωστά και έχει εγκατασταθεί και ξεκινήσει τη λειτουργία του είναι δυνατό να χρειαστούν τροποποιήσεις. Οι τροποποιήσεις αυτές μπορεί να οφείλονται σε:

- Αλλαγές που ζητούν οι πελάτες για βελτίωση του προγράμματος.
- Προσαρμογή του λογισμικού σε νέες συνθήκες/ απαιτήσεις.
- Διορθώσεις και μεταβολές που πρέπει να γίνουν στο πρόγραμμα λόγω ύπαρξης λογικών λαθών.

Η συντήρηση είναι η πιο δαπανηρή φάση του κύκλου ζωής του λογισμικού και καλύπτει εώς και 60% του συνολικού κόστους.

3.4.Μοντέλο παράστασης λογισμικού

Είναι ένα διάγραμμα ή ένα σύνολο από ομοειδή διαγράμματα τα οποία περιγράφει το λογισμικό από μια συγκεκριμένη οπτική γωνία πράγμα το οποίο δηλώνει ότι κανένα μοντέλο παράστασης λογισμικού δεν είναι πλήρες, δηλαδή δεν περιέχει όλες τις δυνατές πληροφορίες για το λογισμικό.

Είναι μια αφαιρετική περιγραφή κάποιων επιλεγμένων χαρακτηριστικών του και μόνο. Αυτό δηλώνει ότι αφαιρούνται τα στοιχεία που δεν ενδιαφέρουν και περιγράφονται μόνο εκείνα στα οποία εστιάζεται η προσοχή.

Υπάρχουν τα διαγράμματα ροής δεδομένων, οντοτήτων-συσχετίσεων, μετάβασης καταστάσεων καθώς και λεξικά δεδομένων τα οποία περιγράφουν συμπληρωματικά μια εφαρμογή λογισμικού.

- Η συμπληρωματικότητα αυτή που περιγράφει το σχήμα έχει νόημα μόνο όταν κανένα μοντέλο από τα παραπάνω δεν περιγράφει πλήρως την εφαρμογή/ λογισμικό που πρόκειται να φτιάξουμε, από μόνο του, ενώ όλα μαζί το κάνουν.
- Ακόμη το σχήμα αυτό μας δείχνει συνέπεια, δηλαδή οι οντότητες που αναφέρονται σε κάθε τέτοιο μοντέλο δεν μπορεί να είναι εντελώς ξένες με αυτές που αναφέρονται στα υπόλοιπα.
- Για να εξασφαλιστεί και να επιταχυνθεί αυτό χρειάζεται το λεξικό δεδομένων, το οποίο περιέχει αναφορές σε όλες τις οντότητες που περιλαμβάνονται στα μοντέλα παράστασης λογισμικού. Το λεξικό δεδομένων θα αναλυθεί περισσότερο παρακάτω.

3.5.Διαγράμματα Ροής Δεδομένων

Ένα διάγραμμα ροής δεδομένων είναι μια γραφική αναπαράσταση της ροής των δεδομένων διαμέσου ενός πληροφοριακού συστήματος . Τα διαγράμματα αυτά μπορούν να χρησιμοποιηθούν επίσης για οπτικοποίηση της επεξεργασίας των δεδομένων.

Σε ένα διάγραμμα ροής δεδομένων, τα δεδομένα ρέουν από μια εξωτερική πηγή δεδομένων προς έναν εσωτερικό αποθηκευτικό χώρο δεδομένων ή έναν εξωτερικό προορισμό δεδομένων, μέσω μιας εσωτερικής διεργασίας.

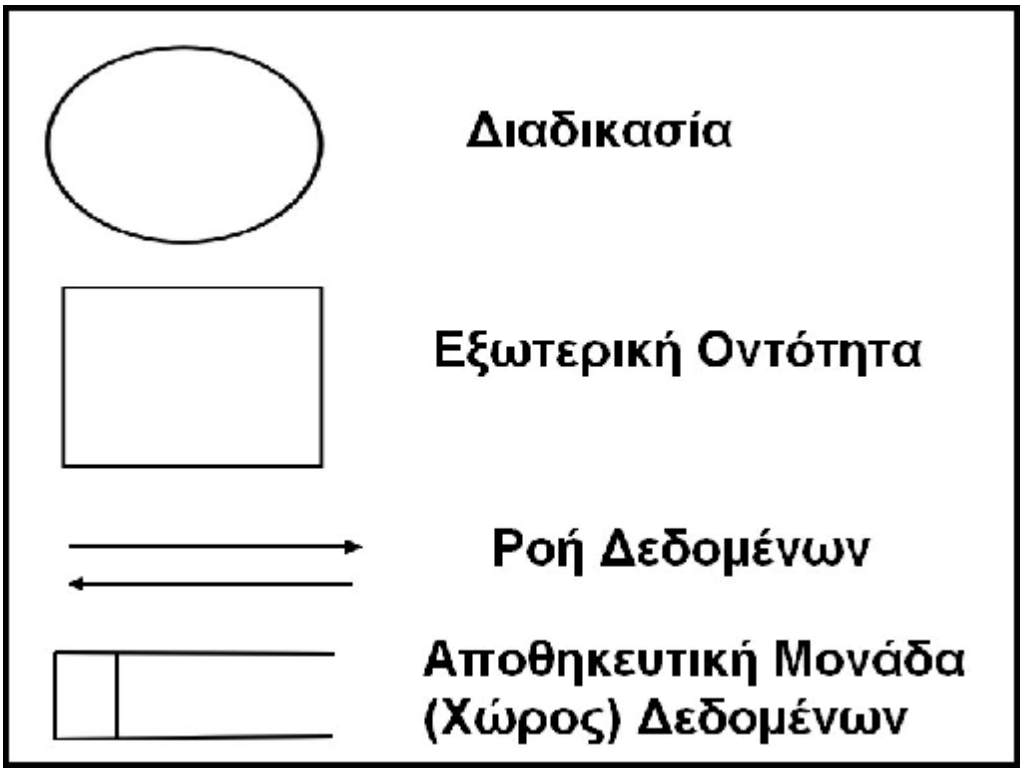
Ένα διάγραμμα ροής δεδομένων δεν παρέχει πληροφορίες για το χρονισμό των διεργασιών ή το αν αυτές λειτουργούν ακολουθιακά ή παράλληλα. Είναι επομένως διαφορετικό από ένα διάγραμμα ροής, το οποίο δείχνει τη ροή του ελέγχου μέσα σε έναν αλγόριθμο και επιτρέπει στον αναγνώστη να βρίσκει ποιες λειτουργίες θα εκτελεστούν, με ποια σειρά και κάτω από ποιες συνθήκες, αλλά όχι τι είδους δεδομένα θα εισαχθούν και θα εξαχθούν από το σύστημα, ούτε από πού έρχονται τα δεδομένα και προς τα πού κατευθύνονται, ούτε πού αυτά αποθηκεύονται (όλες αυτές οι πληροφορίες εμφανίζονται σε ένα διάγραμμα ροής δεδομένων).

Ένα διάγραμμα ροής δεδομένων:

- Είναι εύκολα κατανοητό.
- Είναι ακριβές στο επίπεδο του *ποιες λειτουργίες γίνονται και όχι στο πως*.
- Αποτελείται από πολλά διαφορετικά τμήματα τα οποία αφορούν επιμέρους τμήματα του λογισμικού.
- Μπορεί να σχεδιάζεται σε διαφορετικά επίπεδα λεπτομέρειας.
- Δεν περιέχει πληροφορία για την χρονική αλληλουχία με την οποία συμβαίνουν οι μετασχηματισμοί δεδομένων.
- Είναι εύκολο να υποστεί μεταβολές, όταν κάτι τέτοιο χρειαστεί.

Οι απαιτήσεις του πελάτη από το λογισμικό σε αρχικό στάδιο δίνονται με τη μορφή έκθεσης σε φυσική γλώσσα. Αυτό είναι ιδιαίτερα χρήσιμο για τον πελάτη, όχι όμως για τον κατασκευαστή, ο οποίος χρειάζεται κάτι πιο κοντινό στην υλοποίηση του λογισμικού τρόπου. Αυτό γίνεται με την βοήθεια του διαγράμματος ροής δεδομένων, το οποίο περιέχει τις απαιτήσεις του πελάτη με τη μορφή ενός δικτύου στο οποίο ρέουν δεδομένα τα οποία μετασχηματίζονται σε νέα δεδομένα από μονάδες λογισμικού. Αποτελούν βάση για αρκετά από τα βήματα για την ανάπτυξη/ δημιουργία του λογισμικού.

3.6.Συμβολισμοί Διαγραμμάτων Ροής δεδομένων



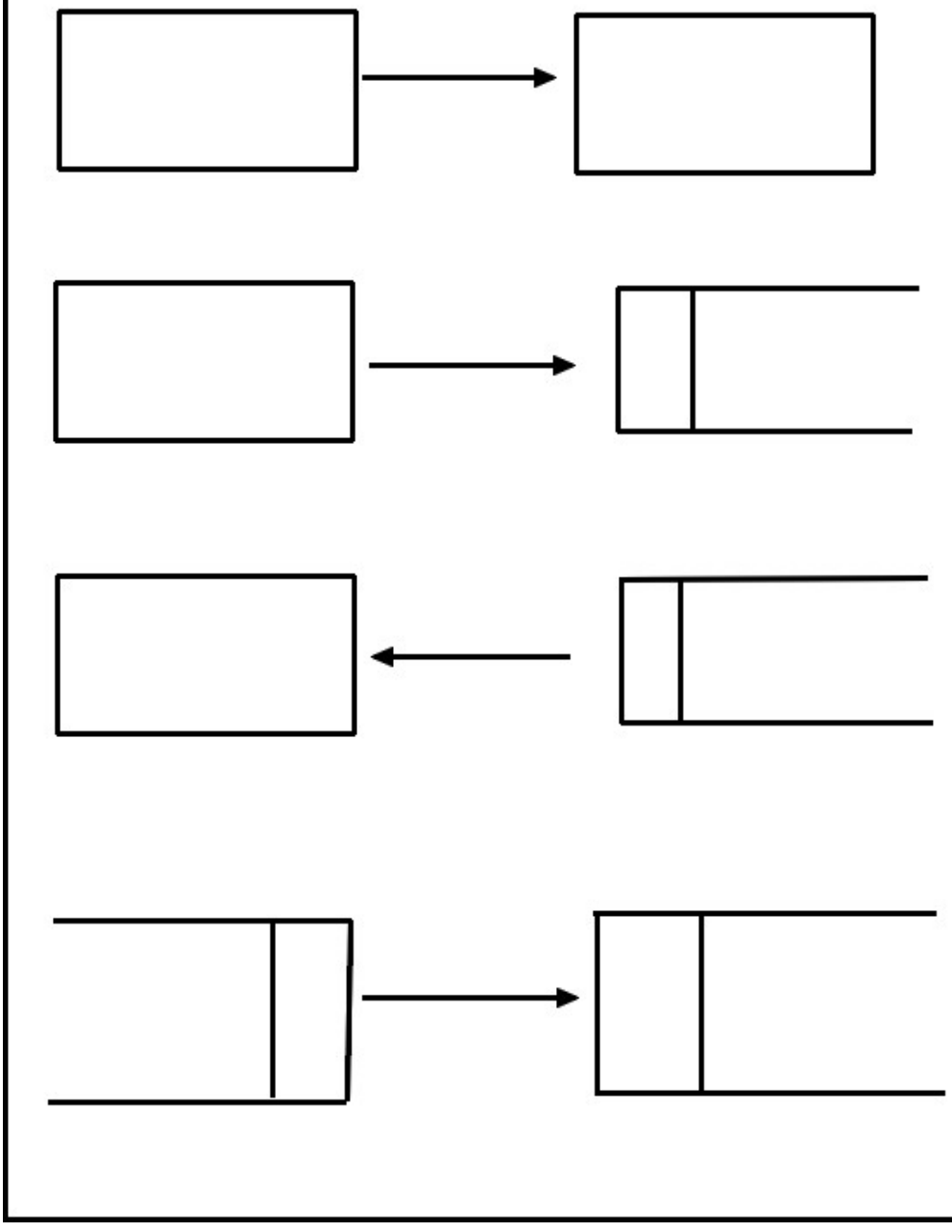
- Οι **διαδικασίες** είναι η διάφορες επεξεργασίες οι οποίες δέχονται κάποια δεδομένα ως είσοδο και παράγουν άλλα δεδομένα ως έξοδο. Κάθε διαδικασία πρέπει να έχει ένα τουλάχιστον βέλος εισερχόμενης ροής δεδομένων, επειδή μια διαδικασία δεν μπορεί να παράγει πληροφορίες χωρίς είσοδο. Επιπλέον πρέπει να έχει και ένα βέλος εξερχόμενης ροής δεδομένων, επειδή μια διαδικασία η οποία δεν παράγει καθόλου πληροφορίες είναι άχρηστη.

- **Εξωτερική οντότητα** καθορίζει τα σύνορα του συστήματος. Μπορεί να είναι πράκτορες, μονάδες του οργανισμού, κάποιο άλλο σύστημα ή άλλοι οργανισμοί οι οποίοι αλληλεπιδρούν με το σύστημά μας ανταλλάσσοντας δεδομένα, είτε για είσοδο είτε για έξοδο είτε και για τα δύο.

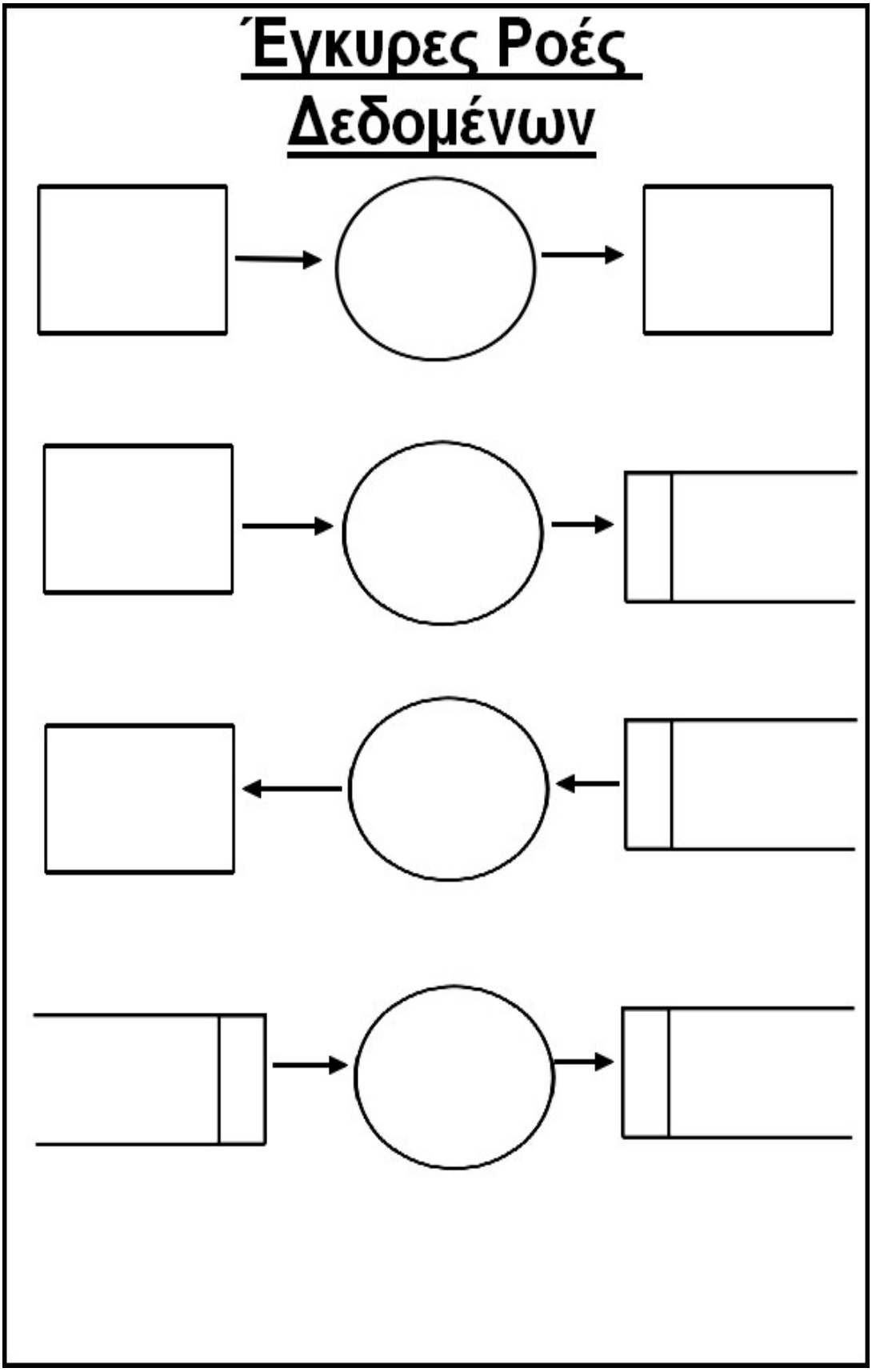
- Οι **ροές δεδομένων** αναπαριστάνουν εισόδους και εξόδους δεδομένων από και προς μια διαδικασία. Οι ροές δεδομένων προέρχονται από εξωτερικές οντότητες οι οποίες μπορούν να είναι μέσα στην επιχείρηση ή εκτός από αυτήν. Στην ροή πρέπει να αναγράφονται τα δεδομένα που εμπλέκονται.

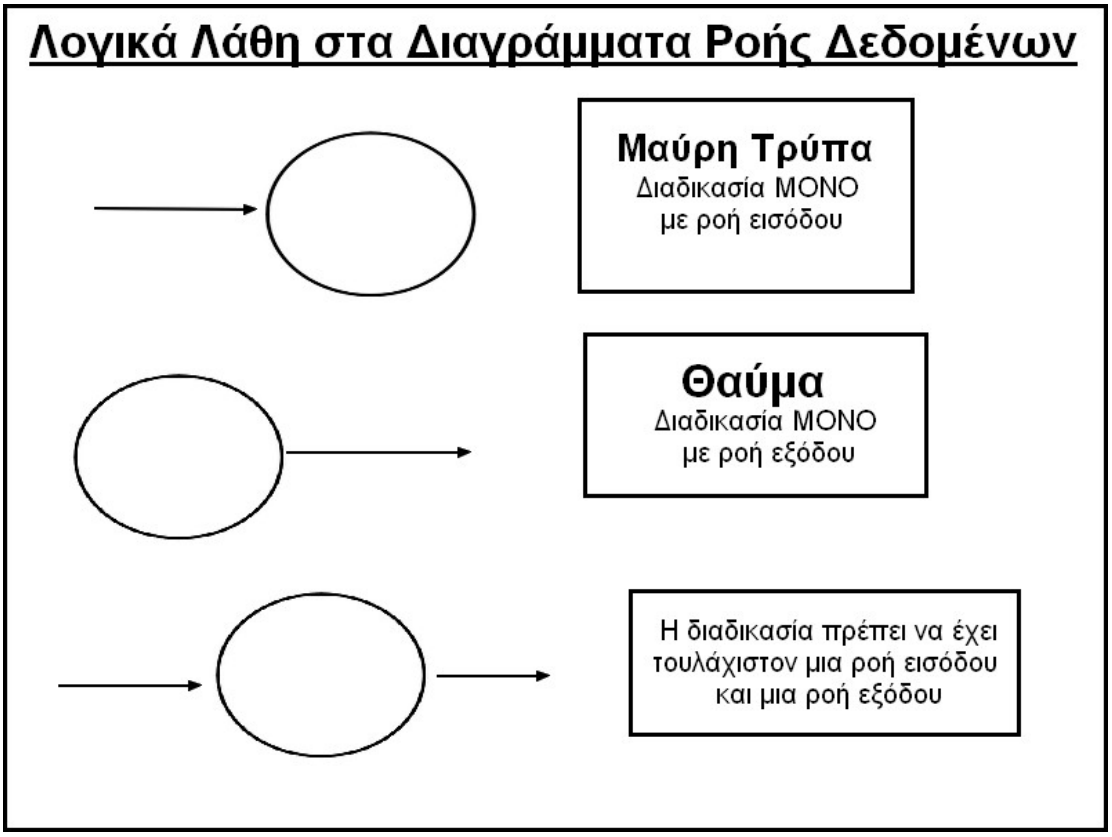
- **Αποθηκευτικές μονάδες** είναι οι χώροι που φυλάγονται επί μόνιμης βάσεως δεδομένα. Περιγράφουν πράγματα για τα οποία η επιχείρηση θέλει να διαφυλάξει δεδομένα και πληροφορίες, όπως πρόσωπα, τοποθεσίες, αντικείμενα, γεγονότα και έννοιες. Συνήθως υλοποιούνται ως αρχεία ή βάσεις δεδομένων.

Μη έγκυρες ροές δεδομένων



Έγκυρες Ροές Δεδομένων





à Επίσης υπάρχουν και οι **γκρίζες τρύπες** κατά τις οποίες τα δεδομένα που εισέρχονται σε μια διαδικασία θα πρέπει με την επεξεργασία που γίνεται σε αυτά, να μπορούν να παράγουν τις ζητούμενες πληροφορίες. Είναι από τα πιο δύσκολα λάθη ως προς την ανίχνευσή τους.

Αυτό που μας ενδιαφέρει στα διαγράμματα ροής δεδομένων, είναι η παρουσίαση των γενικών ποιοτικών στοιχείων των ροών δεδομένων και των μετασχηματισμών που εφαρμόζει σε αυτά ένα συγκεκριμένο λογισμικό. Γι αυτό και θα πρέπει να:

- Να επικεντρώνουμε την προσοχή μας στη ουσία της εφαρμογής/ λογισμικού.
- Κατά την μετάβαση από ένα επίπεδο λεπτομέριας στο επόμενο, εκτός από τις διαδικασίες είναι πιθανόν να αποσυντίθενται οι ροές δεδομένων, πράγμα που μπορεί να προκαλέσει σύγχυση. Είναι όμως συνήθως χρήσιμη, ιδιαίτερα εκεί που η δημιουργία διαγραμμάτων έχει περιορισμούς χώρου. Σε τέτοια περίπτωση η αποσύνθεση θα πρέπει να παρουσιάζεται στο λεξικό δεδομένων.
- Δεν πρέπει να μπερδεύουμε τη ροή δεδομένων με τη διαγραμματική αναπαράσταση αλγορίθμων ή δέντρων απόφασης και γενικά με οτιδήποτε κατασκευαστικό που εξαρτάται από τη γλώσσα προγραμματισμού, το περιβάλλον λειτουργίας κ.ά.
- Η παράσταση της χρονικής αλληλουχίας κατά την οποία λαμβάνουν χώρα οι μετασχηματισμοί δεν παριστάνεται με διάγραμμα ροής δεδομένων αλλά με διάγραμμα μετάβασης καταστάσεων ή με κάποιο είδος ψευδοκώδικα.

- Η αλληλουχία εμφάνισης των οθονών, του μενού, γενικά της συμπεριφοράς του λογισμικού κατά την αλληλεπίδραση του με το χρήστη παριστάνεται με διάγραμμα μετάβασης καταστάσεων και όχι ροής δεδομένων.
- Θα πρέπει να διατηρούμε ισορροπία μεταξύ λεπτομέρειας και αφαίρεσης γιατί ένα διάγραμμα ροής δεδομένων είναι μεγάλο, διαβάζεται δύσκολα και τροποποιείται δυσκολότερα. Ενώ ένα απλό διάγραμμα τροποποιείται εύκολα αλλά δεν είναι ιδιαίτερα χρήσιμο, γι αυτό και θα πρέπει να υπάρξει μια ισορροπία μεταξύ αυτών των δύο.

3.7.Διαγράμματα οντοτήτων- συσχετίσεων

Το διάγραμμα οντοτήτων συσχετίσεων χρησιμοποιείται συχνά από σχεδιαστές βάσεων για να επικοινωνούν με τους απλούς χρήστες. Αποτελεί μια λογική αναπαράσταση των δεδομένων που ενδιαφέρουν μια επιχείρηση και πάνω σε αυτό σχεδιάζεται η βάση δεδομένων. Το μοντέλο οντοτήτων-συσχετίσεων είναι ένα αφαιρετικό ιδεατό μοντέλο δεδομένων, τα οποία έχουν καθορισμένη δομή. Στη μηχανική λογισμικού χρησιμοποιείται για να παρέχει ένα εννοιολογικό σχήμα κατά τη σχεδίαση βάσεων δεδομένων, ως μοντέλο δεδομένων ενός συστήματος και των απαιτήσεών του. Ένα διάγραμμα που δημιουργείται με αυτή τη διαδικασία σχεδίασης καλείται διάγραμμα οντοτήτων-συσχετίσεων. Προτάθηκε αρχικά το 1976 από τον Peter Chen, ωστόσο στη συνέχεια επινοήθηκαν πολλές παραλλαγές της διαδικασίας.

Ένα διάγραμμα οντοτήτων- συσχετίσεων χρησιμοποιείται στο πρώτο στάδιο σχεδίασης ενός συστήματος πληροφοριών, κατά την ανάλυση των απαιτήσεών του. Σκοπός του είναι να περιγράφει τις αναγκαίες πληροφορίες οι οποίες πρόκειται να αποθηκευτούν στη βάση δεδομένων ή τον τύπο τους. Η μοντελοποίηση δεδομένων γίνεται για την περιγραφή των χρησιμοποιούμενων όρων και των σχέσεών τους σε έναν ορισμένο τομέα ενδιαφέροντος. Στην περίπτωση σχεδιασμού ενός συστήματος πληροφοριών, που στηρίζεται σε μια βάση δεδομένων, το εννοιολογικό μοντέλο δεδομένων χαρτογραφείται σε προχωρημένο στάδιο σε ένα λογικό μοντέλο δεδομένων. Το στάδιο αυτό ονομάζεται συνήθως στάδιο λογικού σχεδιασμού. Ύστερα, κατά τη διάρκεια του φυσικού σχεδιασμού το λογικό μοντέλο χαρτογραφείται σε κάποιο φυσικό μοντέλο. Ας σημειωθεί ότι ορισμένες φορές και οι δύο φάσεις αναφέρονται ως "φυσικός σχεδιασμός".

Οντότητα:

Οντότητα είναι ένα αντικείμενο ενδιαφέροντος στον πραγματικό κόσμο το οποίο ξεχωρίζει από τα υπόλοιπα. Μια οντότητα λειτουργεί αφαιρετικά σε έναν πολύπλοκο τομέα. Οντότητες μπορεί να είναι άνθρωποι, μέρη, αντικείμενα, γεγονότα, έννοιες κλπ. Στιγμιότυπο μιας οντότητας είναι μια συγκεκριμένη περίπτωση ενός τύπου οντότητας. Είναι ουσιαστικά ένα σύνολο από αντικείμενα, πρόσωπα ή γεγονότα του πραγματικού κόσμου τα οποία βρίσκονται εντός του πεδίου ενδιαφέροντος της εφαρμογής λογισμικού η οποία κατασκευάζεται.

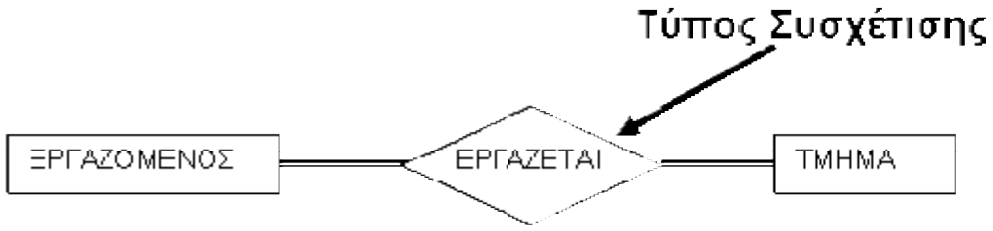
Κάθε οντότητα χαρακτηρίζεται από ένα σύνολο στοιχείων, τα οποία ονομάζονται πεδία και περιέχουν τιμές ορισμένων χαρακτηριστικών ιδιωμάτων της οντότητας. Κάθε πεδίο περιέχει συγκεκριμένη πληροφορία που αφορά μια οντότητα. Το σύνολο των πεδίων που αφορούν μια συγκεκριμένη οντότητα ονομάζεται εγγραφή. Το σύνολο των εγγράφων αποθηκεύεται με τη βοήθεια ενός πίνακα.

Ως σχέση νοείται μια αντιστοίχηση μεταξύ διαφορετικών οντοτήτων, η οποία περιγράφεται με ένα ρήμα.

Συσχέτιση:

είναι η σύνδεση δύο ή περισσότερων τύπων οντοτήτων που παρουσιάζει ενδιαφέρον για σχεδιασμό. Με συσχετίσεις μπορούν να συνδέονται και χαρακτηριστικά οντοτήτων.

Ένας τύπος συσχέτισης (σύνολο συσχετίσεων) παριστάνεται με ρόμβο. Στο εσωτερικό αναγράφεται το όνομα με μικρά γράμματα. Υποδεικνύουμε τα όρια της συσχέτισης με ένα δείκτη.



Ως όρια μπούμε να συναντήσουμε:

0 έως άπειρο: κατώτατο όριο 0, ανώτατο όριο άπειρο
τουλάχιστον 1: κατώτατο όριο 1, ανώτατο όριο άπειρο
ακριβώς 1: κατώτατο όριο 1, ανώτατο όριο 1
το πολύ 1: κατώτατο όριο 0, ανώτατο όριο 1

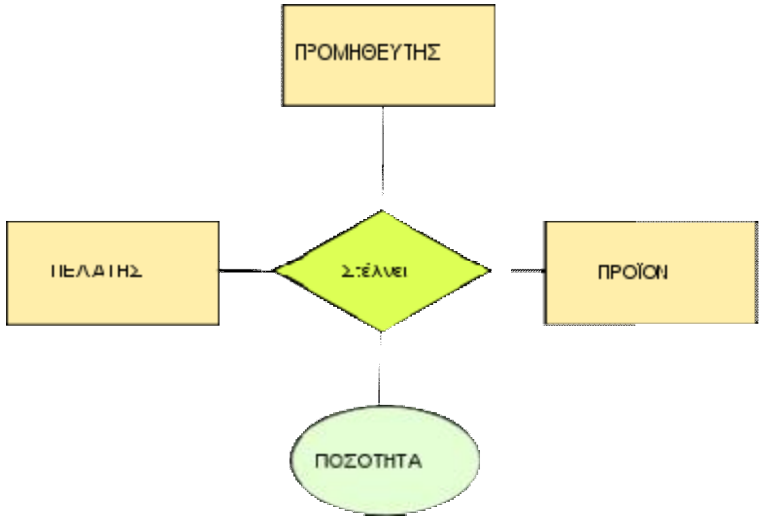
Η συσχέτιση σε αυτό το παράδειγμα ονομάζεται "Παρακολουθεί". Τα βέλη δείχνουν μια συσχέτιση πολλά-προς-πολλά. Κάθε φοιτητής μπορεί να παρακολουθεί ένα ή περισσότερα μαθήματα. Κάθε μάθημα μπορεί να παρακολουθείται από έναν ή περισσότερους μαθητές. Από τη σχέση αυτή μπορούμε να βρούμε ποιοι φοιτητές παρακολουθούν ένα μάθημα ή ποια μαθήματα παρακολουθεί ένας φοιτητής.

Βαθμός ή πολυπλοκότητα ενός τύπου συσχετίσεων:

Ο βαθμός μιας συσχέτισης είναι ο αριθμός των τύπων οντοτήτων που παίρνουν μέρος στη συσχέτιση. Οι πιο συνηθισμένες συσχετίσεις είναι:

- μοναδικές, ο βαθμός τους τότε είναι 1
- δυαδικές, ο βαθμός τους τότε είναι 2
- τριαδικές, ο βαθμός τους τότε είναι 3

Παράδειγμα αποτελεί το παρακάτω σχήμα στο οποίο ο βαθμός συσχέτισης είναι 3 .Οι τύποι οντοτήτων ΠΡΟΜΗΘΕΥΤΗΣ, ΠΕΛΑΤΗΣ και ΠΡΟΪΟΝ παίρνουν μέρος στη συσχέτιση:



Πληθικότητα:

Η πληθικότητα , περιγράφει τον αριθμό στιγμιότυπων ενός τύπου οντοτήτων που μπορούν να αντιστοιχίζονται με μία οντότητα ενός άλλου τύπου σε μια συσχέτιση.

Ο λόγος πληθικότητας ή πληθικός λόγος (cardinality ratio), είναι ο λόγος των πληθικοτήτων μιας συσχέτισης.

Μπορούμε να έχουμε συσχετίσεις με λόγο πληθικότητας:

- 1:1 (ένα- προς- ένα): Αντιστοιχίζεται μια οντότητα ενός τύπου με το πολύ ή ακριβώς μια οντότητα ενός άλλου τύπου.
- 1:N (ένα- προς- πολλά): Αντιστοιχίζεται μια οντότητα ενός τύπου με κανένα, ένα ή πολλά στιγμιότυπα ενός άλλου τύπου.
- M-N (πολλά- προς- πολλά): Αντιστοιχίζεται κάθε στιγμιότυπο του ενός τύπου με ένα, κανένα ή πολλά στιγμιότυπα του άλλου τύπου.

Ασθενής Τύπος Οντότητας:

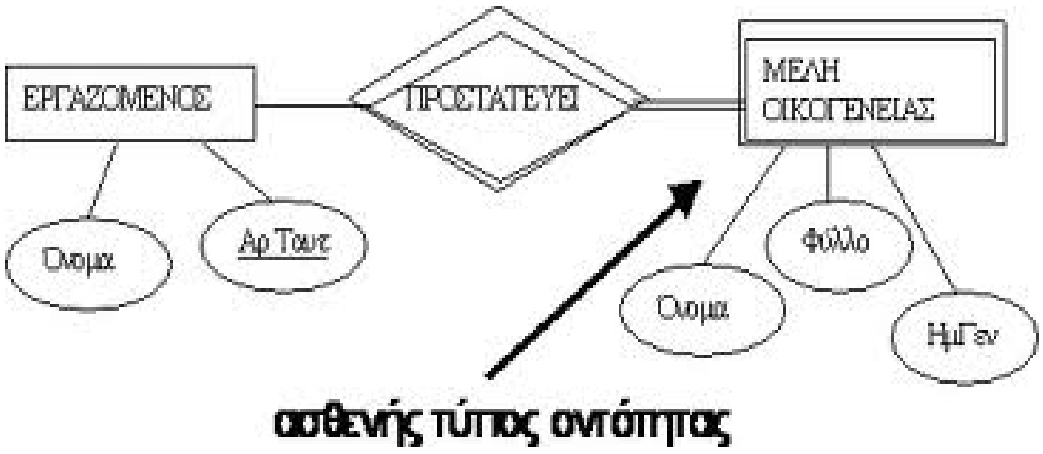
Αδύναμη ή ασθενής οντότητα λέγεται μια οντότητα που εξαρτάται από την ύπαρξη κάποιας άλλης. Οι αδύναμες οντότητες συμμετέχουν σε συσχετίσεις μέσω ταυτοποιητικών συσχετίσεων με ισχυρή οντότητα.

Ταυτοποιητική συσχέτιση ονομάζεται η συσχέτιση στην οποία το πρωτεύον κλειδί της ισχυρής οντότητας χρησιμοποιείται ως μέρος του πρωτεύοντος κλειδιού της αδύναμης οντότητας.

Διακριτικό ή μερικό κλειδί ονομάζεται το χαρακτηριστικό της αδύναμης οντότητας το οποίο μαζί με το πρωτεύον κλειδί της ισχυρής οντότητας είναι το πρωτεύον κλειδί της αδύναμης.

Κατά την αναπαράσταση αδύναμων οντοτήτων:

- η οντότητα παριστάνεται με διπλό ορθογώνιο
- ταυτοποιητική συσχέτιση με διπλό ρόμβο
- το μερικό κλειδί με διακεκομμένη γραμμή

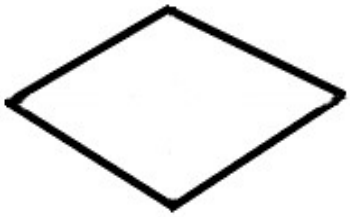
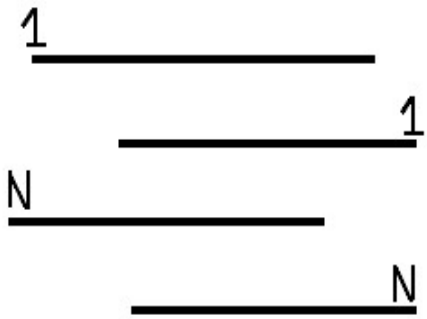
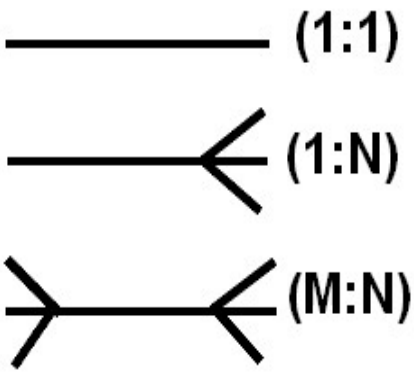


ασθενής τύπος οντότητας

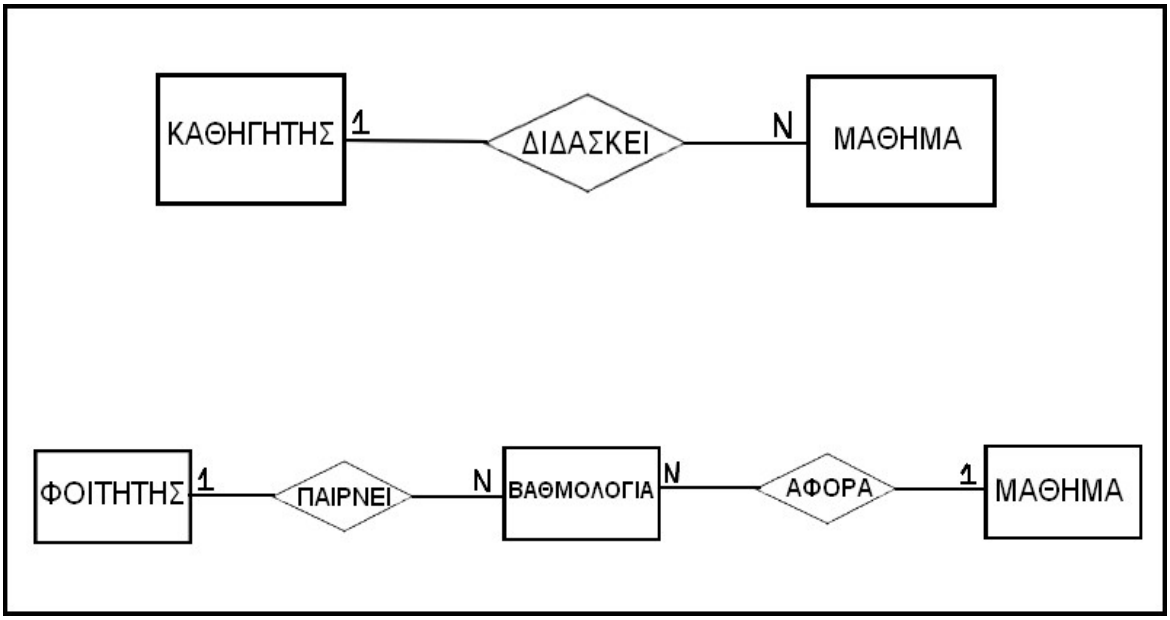
Με τη βοήθεια του διαγράμματος οντοτήτων συσχετίσεων καταγράφονται, στη φάση της προδιαγραφής των απαιτήσεων από το λογισμικό, οι απαιτήσεις σε μόνιμη αποθήκευση δεδομένων, χωρίς να ενδιαφέρει ιδιαίτερα η κατασκευαστική λεπτομέρεια. Κάθε οντότητα περιγράφεται με τη χρήση ενός παραλληλόγραμμου, μέσα στο οποίο σημειώνεται το όνομά της. Για την παράσταση των συσχετίσεων, στο σχήμα φαίνονται δύο τρόποι:

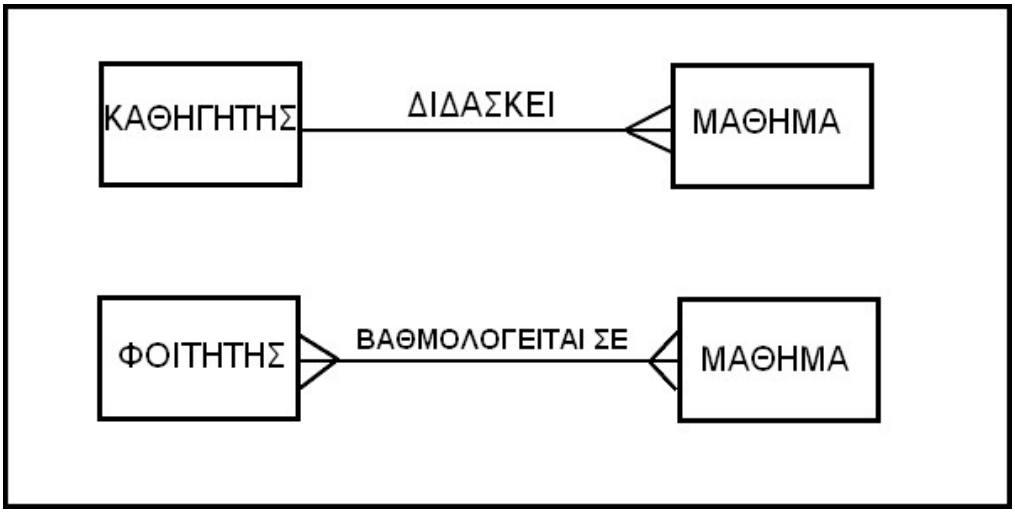
1. Με τη βοήθεια ενός ρόμβου μέσα στον οποίο σημειώνεται το ρήμα που τις χαρακτηρίζει και στις ακμές η πολυπλοκότητα της σχέσης.

2. Με τη βοήθεια γραμμών που συνδέουν τις συσχετιζόμενες οντότητες, ανάλογα με την κατάληξη των οποίων δηλώνεται το είδος της σχέσης. Το όνομα της σχέσης σημειώνεται επάνω από τη γραμμή, σε κάποιο πρόσφορο σημείο.

Συμβολισμοί Διαγραμμάτων Οντοτήτων- Συσχετίσεων	
	Οντότητα Δεδομένων
	Συσχέτιση μεταξύ οντοτήτων
	Ορισμός Πολλαπλότητας Συσχέτισης
	Συσχέτισης και Ορισμός Πολλαπλότητας

Παραδείγματα:





Επεξήγηση: Μια συσχέτιση δεν είναι αντιμεταθετική, δηλαδή δεν μπορεί να διαβαστεί με την αντίθετη φορά με το ίδιο ρήμα. Σε αυτό το παράδειγμα διαβάζεται ως εξής:

1. Κάθε καθηγητής διδάσκει πολλά μαθήματα.
2. Κάθε φοιτητής βαθμολογείται σε κάθε μάθημα πολλές φορές.

Αντίστροφα αυτό θα διαβαζόταν:

1. Κάθε μάθημα διδάσκεται από ένα καθηγητή.
2. Σε κάθε μάθημα κάθε φοιτητής λαμβάνει τουλάχιστον μία βαθμολογία.

Χρήσιμα σημεία που πρέπει να λαμβάνονται υπόψη στη σχεδίαση διαγραμμάτων οντοτήτων-συσχετίσεων στη φάση της προδιαγραφής των απαιτήσεων από το λογισμικό:

- Θα πρέπει καταρχάς να προσδιορίσουμε τις οντότητες και τις σχέσεις μεταξύ τους. Οι λεπτομέρειες που αφορούν κάθε οντότητα και σχέση, όπως το πλήθος και το είδος των πεδίων, προσδιορίζονται σε επόμενη φάση της ανάπτυξης.

- Η απλότητα και η εύκολη αναγνωσιμότητα του διαγράμματος θα πρέπει να αποτελεί επιδίωξη. Όσο πιο σύνθετο είναι ένα διάγραμμα οντοτήτων- συσχετίσεων, τόσο πιο δύσκολη είναι η υλοποίηση του και οι μεταβολές σε αυτό στη συνέχεια, ενώ συνήθως υπάρχει πάντα ένας εναλλακτικός τρόπος σχεδίασης.
- Δεν χρειάζεται σε τέτοια φάση να γίνει η οποιαδήποτε τροποποίηση με στόχο τη βελτιστοποίηση. Αυτό μπορεί να γίνει και στη συνέχεια της σχεδίασης του λογισμικού.
- Η σχεδίαση του διαγράμματος οντοτήτων- συσχετίσεων, στην παρούσα φάση δεν θα πρέπει να γίνεται έχοντας στο νου ένα συγκεκριμένο περιβάλλον υλοποίησης. Τα πλεονεκτήματα και μειονεκτήματα του συγκεκριμένου περιβάλλοντος θα απασχολήσουν τον κατασκευαστή αργότερα, κατά τη φάση της σχεδίασης του λογισμικού.

3.8.Διαγράμματα μετάβασης καταστάσεων

Τα Διαγράμματα μετάβασης καταστάσεων περιγράφουν την χρονική σειρά εκτέλεσης εργασιών ανάλογα με τα εξωτερικά γεγονότα τα οποία προκαλεί ο χρήστης ή άλλες εξωτερικές πηγές. Η περιγραφή της συμπεριφοράς αυτής είναι απαραίτητη προκειμένου να αποκτήσει κανείς μια καλή εικόνα του λογισμικού που κατασκευάζεται. Παρουσιάζει δηλαδή, τη δυναμική συμπεριφορά του λογισμικού.

Ανάλογα με το μήνυμα που λαμβάνει το λογισμικό από το περιβάλλον εκτελεί μια συγκεκριμένη εργασία. Επομένως δεν είναι δυνατή η εκτέλεση οποιασδήποτε εργασίας σε οποιαδήποτε στιγμή. Ανάλογα με την κατάσταση που βρίσκεται το σύστημα και με το μήνυμα που λαμβάνει, εκτελείται μια εργασία και το σύστημα μεταβαίνει, σε μια άλλη κατάσταση. Αυτό ακριβώς περιγράφουν τα διαγράμματα μετάβασης καταστάσεων.

Ορισμοί:

Γεγονός: είναι μια στιγμιαία μεταβολή στο περιβάλλον λειτουργίας του λογισμικού, η οποία προκαλείται από εξωτερικούς παράγοντες, οι οποίοι μπορεί να είναι ο χρήστης, το λειτουργικό σύστημα και άλλες εφαρμογές λογισμικού.

Απόκριση: είναι μια λειτουργία που εκτελεί το λογισμικό όταν προκαλείται ένα γεγονός. Στη δομημένη ανάλυση και σχεδίαση δεν είναι όλες οι λειτουργίες εκτελέσιμες ως αποκρίσεις σε γεγονότα.

Κατάσταση:

όταν το λογισμικό αναμένει γεγονότα, τότε λέμε ότι βρίσκεται σε μια κατάσταση. Όταν το λογισμικό δεχτεί ένα γεγονός τότε μπορεί να εκτελέσει μια λειτουργία και να μεταβεί σε μια άλλη κατάσταση.

Σε κάθε κατάσταση τα γεγονότα τα οποία μπορούν να προκαλέσουν μετάβαση είναι συγκεκριμένα και σαφή. Θα πρέπει να γνωρίζουμε ποιο γεγονός θα προκαλέσει αυτή τη μετάβαση και σε ποια νέα κατάσταση καθώς και το ποια νέα λειτουργία θα εκτελείται από το λογισμικό στη νέα αυτή κατάσταση που μετέβηκε.

Συμβολισμοί Διαγραμμάτων Μετάβασης Καταστάσεων	
	Κατάσταση
	Κατάσταση Έναρξης
	Κατάσταση Τέλους
	Μετάβαση σε άλλη κατάσταση/λειτουργία που εκτελείται
	Μετάβαση στην ίδια κατάσταση/λειτουργία που εκτελείται

Στο διάγραμμα μετάβασης καταστάσεων οι καταστάσεις παριστάνονται με ένα παραλληλόγραμμο με στρογγυλεμένες γωνίες. Οι καταστάσεις έναρξης και τέλους παριστάνονται με μια μεγάλη μαύρη κουκκίδα εντός και εκτός κύκλου, αντίστοιχα. Οι μεταβάσεις μεταξύ καταστάσεων περιγράφονται με ένα βέλος από την αρχική στην τελική κατάσταση. Πάνω στο βέλος, σημειώνεται το γεγονός που προκαλεί την μετάβαση.

- Επίσης ένα διάγραμμα μετάβασης καταστάσεων είναι χρήσιμο για ορισμένες από τις εργασίες, και κυρίως για εκείνες των οποίων η ροή της εκτέλεσης δεν είναι εύκολα ή με μοναδικό τρόπο αντιληπτή.

Υπάρχουν δύο κριτήρια για να κατασκευάσει κανείς ένα διάγραμμα μετάβασης καταστάσεων:

Όταν η περιγραφή της συμπεριφοράς μιας μονάδας λογισμικού με ένα τέτοιο διάγραμμα βοηθά στην καλύτερη κατανόηση και σχεδίαση του αντίστοιχου πηγαίου κώδικα.

1. Όταν πρόκειται να περιγράψουμε τη συμπεριφορά μιας μονάδας η οποία στηρίζεται με τη διαχείριση δεδομένων που αφορούν γεγονότα του πραγματικού κόσμου.

ΠΑΡΑΔΕΙΓΜΑ:

Ζητείται να κατασκευαστεί ένα διάγραμμα μετάβασης κατάστασης που αναφέρεται στην οντότητα ΑΥΤΟΚΙΝΗΤΟ που να περιγράφει τη διαδικασία από τη φάση της παραγγελίας ενός αυτοκινήτου έως τη φάση της παράδοσης του.

Προδιαγραφή 1:

- Αρχικά το αυτοκίνητο γίνεται παραγγελία από ένα πελάτη και ο πελάτης περιμένει να εκτελεστεί η παραγγελία του.
- Στη συνέχεια ξεκινάει η εκτέλεση της παραγγελίας και το αυτοκίνητο περνά στη φάση της παραγωγής. Στο σημείο αυτό μπορεί ο πελάτης να ζητήσει ακύρωση της παραγγελίας και τότε σταματά η διαδικασία της παραγγελίας.

Προδιαγραφή 2:

- Με την ολοκλήρωση της παραγωγής του αυτοκινήτου, το αυτοκίνητο μεταφέρεται στο σημείο πώλησης.
- Με την ολοκλήρωση της μεταφοράς του αυτοκινήτου, το αυτοκίνητο θεωρείται ετοιμοπαράδοτο.
- Ακολουθεί ενημέρωση του πελάτη και έτσι ολοκληρώνεται η διαδικασία της παραγγελίας.

Αναζήτηση καταστάσεων γεγονότων 1:

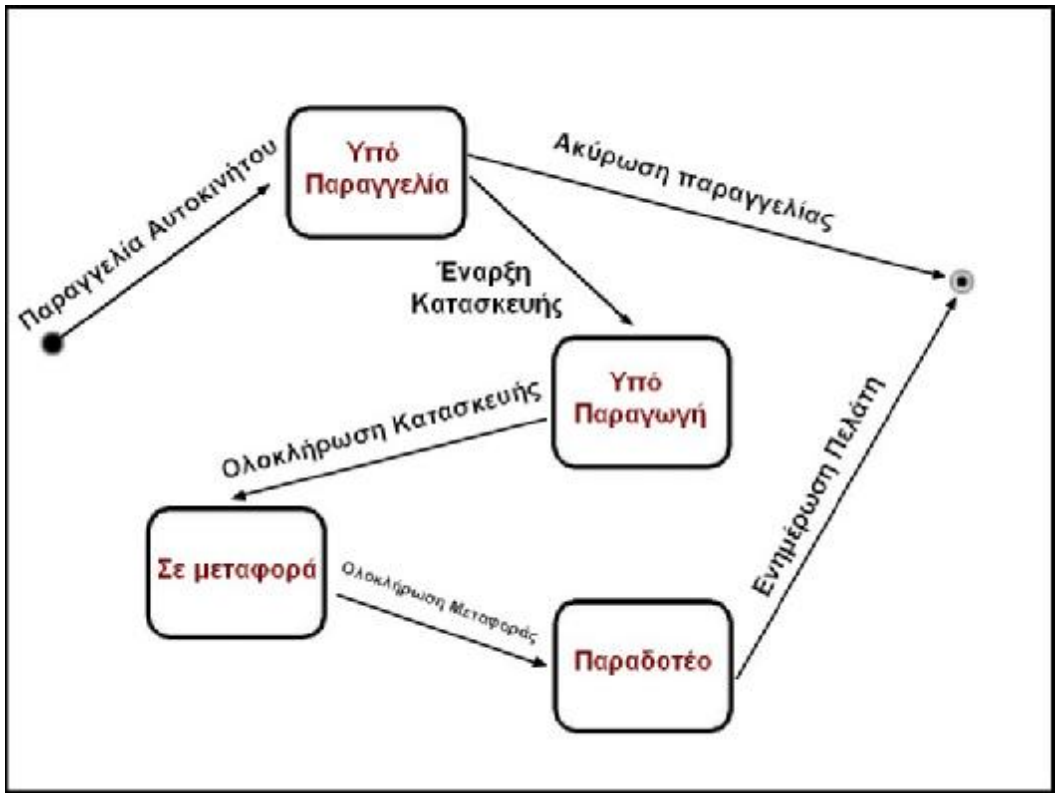
- Αρχικά το αυτοκίνητο γίνεται παραγγελία από το πελάτη και έτσι το αυτοκίνητο βρίσκεται στην κατάσταση **υπό παραγγελία**.
- Στη συνέχεια ξεκινάει η εκτέλεση της παραγγελίας και το αυτοκίνητο περνά στην κατάσταση **υπό παραγωγή**. Στην περίπτωση, όπως αναφέρθηκε και παραπάνω, που πελάτης ζητήσει ακύρωση της παραγγελίας τότε σταματά η διαδικασία της παραγγελίας.

Αναζήτηση καταστάσεων γεγονότων 2:

- Με την ολοκλήρωση τη ς παραγωγής το αυτοκίνητο περνά στη φάση της μεταφοράς, δηλαδή στην κατάσταση **σε μεταφορά**.
- Με την ολοκλήρωση της μεταφοράς το αυτοκίνητο βρίσκεται στην κατάσταση **παραδοτέο**. Ακολουθεί ενημέρωση του πελάτη και σταματά η διαδικασία της παραγγελίας.

Καταστάσεις	**Γεγονότα**
Υπό παραγγελία	Παραγγελία αυτοκινήτου
Υπό παραγωγή	Έναρξη κατασκευής
Σε μεταφορά	Ακύρωση παραγγελίας
Παραδοτέο	Ολοκλήρωση κατασκευής
	Ολοκλήρωση μεταφοράς
	Ενημέρωση πελάτη

Σχήμα:



3.9.Λεξικό Δεδομένων

Το λεξικό δεδομένων είναι μια οργανωμένη ταξινόμηση όλων των σχετιζόμενων, με δεδομένα, στοιχείων των μοντέλων παράστασης λογισμικού με όσο το δυνατό μεγαλύτερη σαφήνεια και πληρότητα, έτσι ώστε να γίνονται κατανοητά τόσο από τον αναλυτή του συστήματος όσο και από το χρήστη.

Το λεξικό δεδομένων είναι ένας πίνακας που χρησιμοποιείται σε σχετικά πρόωρο στάδιο της ανάπτυξης του λογισμικού, όπου η δημιουργία συγχύσεων είναι εύκολη, αλλά και το γεγονός ότι η πολυπλοκότητα στη σχεδίαση ενός διαγράμματος μπορεί να μην επιτρέπει να περιγράφονται με κατατοπιστικούς τίτλους τα όσα φαίνονται σε αυτό. Καταλαβαίνει εύκολα την αναγκαιότητα ενός πιο οργανωμένου τρόπου περιγραφής των εννοιών που περιέχονται στα μοντέλα παράστασης του λογισμικού. Απαιτείται μια συνεχής ενημέρωση του λεξικού δεδομένων πράγμα αρκετά δύσκολο και χρονοβόρο.

Μια συνηθισμένη δομή ενός λεξικού δεδομένων είναι ένας πίνακας που περιλαμβάνει τα παρακάτω πεδία:

- Ονομασία
- Βοηθητικές Ονομασίες
- Πού χρησιμοποιείται
- Πώς χρησιμοποιείται
- Τι περιέχει
- Όρια τιμών
- Αρχική τιμή
- Λοιπά στοιχεία

Ο παρακάτω πίνακας περιέχει ένα παράδειγμα από λεξικό δεδομένων για μια υποθετική εφαρμογή τράπεζας:

Όνομα	**Τύπος**	**Περιγραφή**
Πελάτης	Δομή	Η εγγραφή με τα στοιχεία ενός πελάτη. Περιέχει όνομα, κωδικό λογαριασμού και υπόλοιπο λογαριασμού.
Κωδικός Λογαριασμού	INT(20)	Ο μοναδικός κωδικός του λογαριασμού.
Όνομα	CHAR(30)	Το ονοματεπώνυμο ενός πελάτη.
Υπόλοιπο Λογαριασμού	INT(12)	Το καθαρό υπόλοιπο του λογαριασμού σε ευρώ.

Το λεξικό δεδομένων είναι το ίδιο με μια βάση δεδομένων στην οποία αποθηκεύονται περισσότερες πληροφορίες ή αποσυντίθενται σύνθετα δεδομένα σε απλούστερα τα οποία εμφανίζονται στα διαγράμματα ροής δεδομένων. Είναι μια ανεξάρτητη εργασία, αλλά γίνεται παράλληλα με την κατασκευή των μοντέλων παράστασης λογισμικού. ,ε την πρώτη εμφάνιση ενός στοιχείου δεδομένων, αυτό εισάγεται στο λεξικό και χαρακτηρίζεται όσο ολοκληρωμένα γίνεται. Ο χαρακτηρισμός του τελειοποιείται καθώς προχωρά η ανάπτυξη.

Τα λεξικά δεδομένων έχουν εξελιχθεί σε αποθήκες πληροφοριών λογισμικού και εκτός από δεδομένα, περιλαμβάνουν και ενεργά συστατικά στοιχεία λογισμικού, τα οποία μπορούν να επαναχρησιμοποιηθούν.

3.10.Προβλήματα στον προσδιορισμό απαιτήσεων

3.10.1.Επικοινωνίας

3.10.2.Προτύπων

3.10.3.Γλώσσας

3.10.4.Οικονομικά

3.10.1.Επικοινωνίας:

Απαραίτητη προϋπόθεση για να μην υπάρξουν προβλήματα επικοινωνίας μεταξύ πελάτη και κατασκευαστή, είναι να μιλάνε την ίδια γλώσσα, ούτως ώστε να μπορέσουν να επικοινωνήσουν.

Συνήθως ο πελάτης δυσκολεύεται να παρουσιάσει στον κατασκευαστή τι πραγματικά θέλει, τις ακριβείς απαιτήσεις του από το λογισμικό που πρόκειται να κατασκευαστεί. Σε τέτοια περίπτωση ο κατασκευαστής θα πρέπει όχι μόνο να καταγράψει τις απαιτήσεις του πελάτη αλλά και να τις διαμορφώσει κατάλληλα. Αυτό επιτυγχάνεται συνήθως προκαλώντας στον πελάτη συνεχείς αμφισβητήσεις και ερωτήματα για τις πληροφορίες που δίνει στον κατασκευαστή, ώστε να αναθεωρήσει τις απόψεις και απαιτήσεις του από πολλές πλευρές και να είναι σαφής και τεκμηριωμένες.

Για να επιτευχθεί αυτό θα πρέπει ο κατασκευαστής να μπορεί να αντιλαμβάνεται την ουσία του προβλήματος και να έχει τη δυνατότητα να αντλήσει τις σωστές πληροφορίες από τον πελάτη μέσω του διαλόγου.

3.10.2.Προτύπων:

Αυτού του είδους προβλήματα αναφέρονται στα πρότυπα μοντέλα παράστασης λογισμικού που χρησιμοποιούνται για την καταγραφή των απαιτήσεων από το λογισμικό. Δεν είναι όμως τα μοναδικά μέσα που χρησιμοποιούνται για το σκοπό αυτό και έχουν αμφισβητηθεί αρκετές φορές. Παρόλα αυτά δεν έχει βρεθεί καλύτερη λύση για την καταγραφή των απαιτήσεων και οποιαδήποτε ερευνητική εργασία έχει γίνει γι αυτό το θέμα καταλήγει σε αυτή την κατεύθυνση.

Ένας κατασκευαστής ακολουθεί όλα τα παραπάνω βήματα που αναπτύξαμε προηγουμένως κάτι το οποίο είναι επαρκές για να αντιληφθούν όλοι οι εμπλεκόμενοι, το τι θα κάνει το λογισμικό ενώ παράλληλα μπορεί ο κατασκευαστής να προχωρήσει στην επόμενη φάση σχεδίασης του λογισμικού.

Όταν όμως χρησιμοποιήσουμε περίσσεια μη προτυποποιημένων διαγραμμάτων, κειμένων κτλ αυτό μπορεί να δημιουργήσει σύγχυση στον κατασκευαστή που τα εισήγαγε, αφού αυτά συνήθως δεν καλύπτουν τις απαραίτητες ανάγκες του κατασκευαστή για τη δημιουργία του λογισμικού και δεν μπορούν να μελετηθούν πλήρως. Ακόμη αυτό μπορεί να περιορίσει την

επικοινωνία με άλλους κατασκευαστές, που θα μπορούσαν να βοηθήσουν, καθώς και με εμπλεκόμενους στην ανάπτυξη του προγράμματος του λογισμικού.

Αυτή η σύγχυση εκτείνεται όταν εμπλεκόμενοι εργαζόμενοι για την ανάπτυξη του λογισμικού, εισάγουν νέους συμβολισμούς και έννοιες οι οποίες έχουν παρεμφερή σημασία τα πρότυπα, αλλά οι υπόλοιποι κατασκευαστές δεν γνωρίζουν την έννοια τους.

3.10.3.Γλώσσας:

Η φυσική γλώσσα αποτελεί το σημαντικότερο εργαλείο για την περιγραφή και καταγραφή των απαιτήσεων από το λογισμικό. Αυτή είναι που θα καθορίσει την ακρίβεια και την περιγραφή των απαιτήσεων.

Συχνά όμως περιέχει ασάφειες και μπορεί να είναι ελλιπής παρά την προσπάθεια να γίνει το αντίθετο. Πολλές φορές οι περιγραφές των απαιτήσεων είναι απαραίτητο να διευκρινιστούν πάλι με την βοήθεια της φυσικής γλώσσας και της χρήσης κειμένου, το οποίο θα εξηγεί τις ασάφειες. Πρέπει δηλαδή να υπάρχει σωστός χειρισμός της γλώσσας. Ένα τέτοιο κείμενο θα πρέπει να είναι απλό, να περιέχει σύντομες και μικρές προτάσεις, να μην αφήνει περιθώρια παρερμηνείας και να προβάλει με ευστοχία αυτό που θέλει να πει.

Πρόβλημα επίσης αποτελούν οι όροι που προέρχονται από την αγγλική γλώσσα και πρέπει να αποδοθούν στην ελληνική. Οι αποδόσεις αυτές είναι αρκετές αλλά πολλές φορές όχι και τόσο σωστές/ συχνά λανθασμένες και γίνονται διαφορετικά αντιληπτές από τον καθένα ή συγχέονται με άλλους όρους.

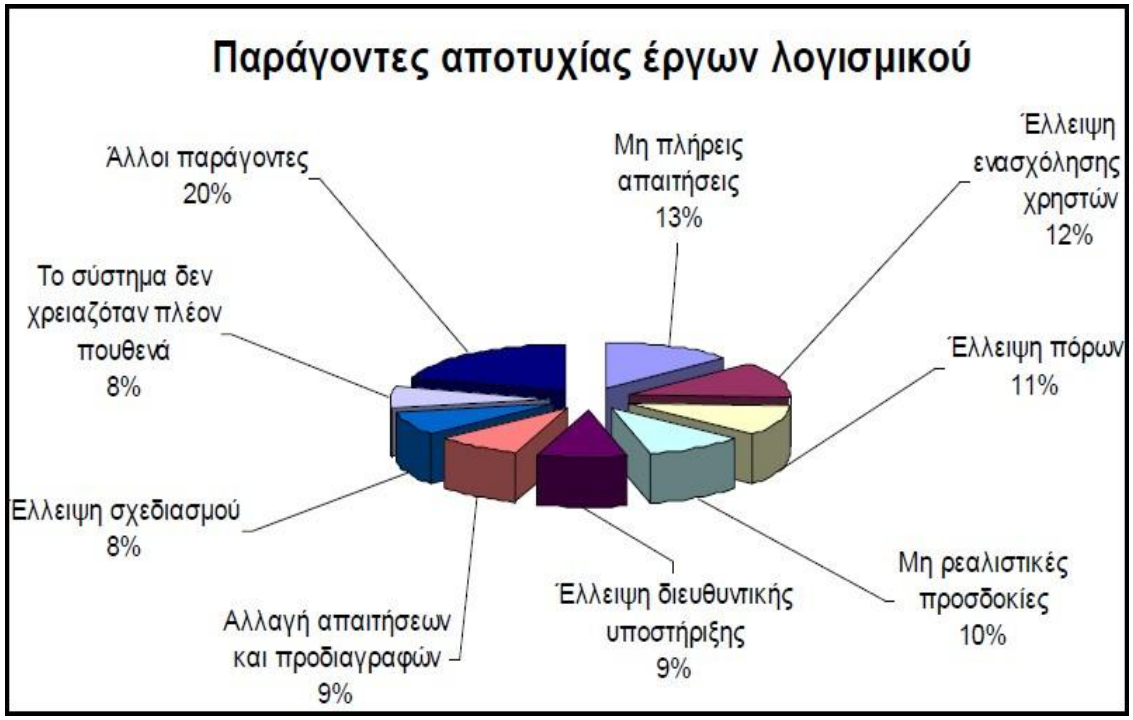
3.10.4.Οικονομικά:

Το λογισμικό που κατασκευάζεται έχει σκοπό το κέρδος. Η άϋλη υπόσταση του λογισμικού συχνά δεν επιτρέπει στον πελάτη να αντιληφθεί πάντα το τι πληρώνει και συνήθως προσπαθεί να εμπλουτίσει το λογισμικό με χαρακτηριστικά και λειτουργίες και συγχρόνως να ελαχιστοποιήσει το κόστος καλύπτοντας τις απαιτήσεις του.

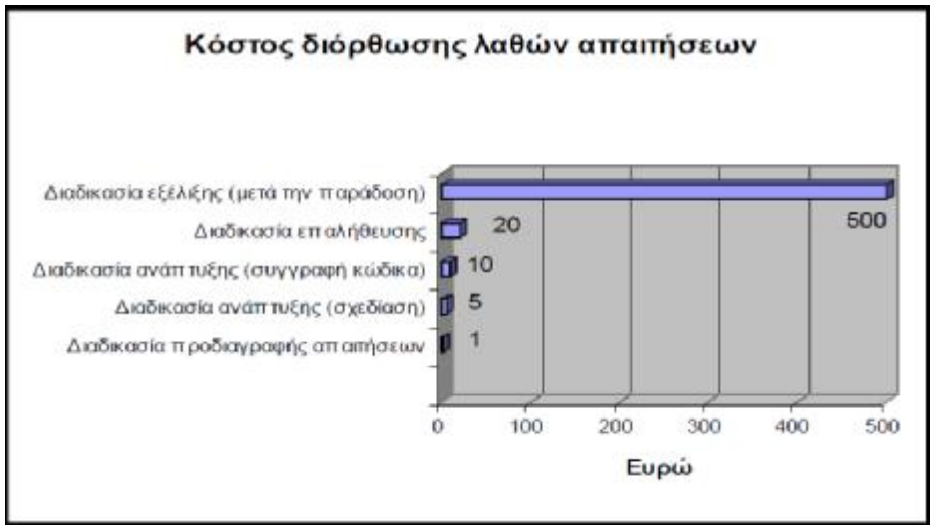
Από την άλλη πλευρά ο κατασκευαστής προσπαθεί να κάνει τα λιγότερο δυνατά με όσο το δυνατό μεγαλύτερο κέρδος, πράγμα φέρνει πελάτη και κατασκευαστή σε δύο διαφορετικές πλευρές.

3.11.Έρευνες:

Σύμφωνα με έρευνα της Standish Group σε 350 εταιρίες και 8000 έργα που έγινε το 1995 βρέθηκαν τα εξής αποτελέσματα για την αποτυχία έργων λογισμικού:



Επίσης το παρακάτω σχήμα δείχνει την σημασία εύρεση λαθών και το κόστος που υπάρχει αναλόγως σε τι στάδιο ανακαλύφθηκε το λάθος:



Κεφάλαιο 4: Σχεδίαση

4.1 Σκοπός σχεδίασης

Ο σκοπός της σχεδίασης λογισμικού είναι η δημιουργική διαδικασία μετατροπής του προβλήματος σε λύση. Αυτό που ζητείτε μέσω της κατασκευής του λογισμικού είναι να εκτελεί τις επιθυμητές λειτουργίες και να έχει τα επιθυμητά χαρακτηριστικά. Η λύση του προβλήματος της σχεδίασης δεν είναι καθόλου εύκολη καθώς γεννιούνται τα παρακάτω ερωτήματα:

- Με ποιά στρατηγική πρέπει να αντιμετωπίσουμε την μετάβαση από τις προδιαγραφές στην σχεδίαση έτσι ώστε η εργασία μας να είναι αποτελεσματική;
- Ποιός είναι ο καλύτερος τρόπος υλοποίησης μιας προδιαγραφής και πως τεκμηριώνεται;
- Σε ποιό βαθμό δεσμεύεται η σχεδίαση από το περιβάλλον;
- Ποια είναι η περισσότερο επαρκής, δηλαδή κατάλληλη, χωρίς να

είναι ούτε ελλιπής ούτε φλύαρη, περιγραφή του σχεδίου του λογισμικού;

- Πώς εξασφαλίζεται η ποιότητα του παραγόμενου λογισμικού μέσα από τις εργασίες που λαμβάνουν χώρα κατά τη σχεδίαση;

Η σχεδίαση θα πρέπει να δώσει την καλύτερη δυνατή απάντηση στα παραπάνω ερωτήματα. Σε αυτό το σημείο θα πρέπει να σημειωθεί ότι δεν υπάρχει βέλτιστη λύση αλλά η καλύτερη δυνατή στις εκάτοστες συνθήκες.
Τέλος, θα πρέπει να επισημανθεί ότι εκτός από την βέλτιστη λύση που πρέπει να έχουμε, θα πρέπει να υπάρχουν και κριτήρια τα οποία να συμφωνούν μεταξύ τους.

Τέσσερα από τα κριτήρια αυτά είναι:

1. Το σχέδιο θα πρέπει να ικανοποιεί όλες τις προδιαγραφές των απαιτήσεων από το λογισμικό.
2. Το σχέδιο πρέπει να περιγράφει πλήρως όλες τις πλευρές του λογισμικού δεδομένα, λειτουργίες και συμπεριφορά, όπως αυτές θεωρούνται από την πλευρά του κατασκευαστή.
3. Το σχέδιο θα πρέπει να είναι εύκολα κατανοητό από τους προγραμματιστές.
4. Το σχέδιο δεν πρέπει να περιέχει σφάλματα.

4.2 ΤΕΧΝΟΤΡΟΠΙΕΣ ΣΧΕΔΙΑΣΗΣ

Το πρόβλημα της σχεδίασης λογισμικού μπορεί να αντιμετωπιστεί με διάφορες στρατηγικές προσεγγίσεις. Οι διάφορες μεθοδολογίες που έχουν παρουσιαστεί μπορούν να ενταχθούν σε δύο μεγάλες κατηγορίες: τις προσανατολισμένες στις διαδικασίες (function oriented) και τις προσανατολισμένες στα αντικείμενα (object oriented). Μια σύντομη αναφορά στα χαρακτηριστικά των δύο κατηγοριών δίνεται στην ενότητα αυτή. Στην ερώτηση για το τι θα κάνει το λογισμικό έχουν δοθεί πολλές απόψεις.

4.2.1. ΔΟΜΗΜΕΝΗ ΣΧΕΔΙΑΣΗ

Οι μεθοδολογίες που ακολουθούν αυτή την προσέγγιση προτείνουν τρόπους αποσύνθεσης του συστήματος από πάνω προς τα κάτω (top-down) σε μια ιεραρχία διαδικασιών, συναρτήσεων και άλλων ενεργών μονάδων λογισμικού. Όσο κατεβαίνει κανείς στην ιεραρχία αυτή, τόσο μεγαλύτερη λεπτομέρεια συναντά μέχρις ότου φτάσει στις απλές δομικές μονάδες, δηλαδή τις εντολές της γλώσσα προγραμματισμού.

4.2.2 ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΕΦΗΣ ΣΧΕΔΙΑΣΗ

Η αντικειμενοστραφής προσέγγιση (object-oriented) ακολουθεί ένα διαφορετικό δρόμο: αντί το σύστημα να θεωρείται ως μια ιεραρχία διαδικασιών, ανεξάρτητων από τα δεδομένα, θεωρείται ως μια συλλογή οντοτήτων, καθεμία εκ των οποίων περικλείει και διαδικασίες και δεδομένα. Η προσέγγιση βασίζεται στην ιδέα ότι στον πραγματικό κόσμο δεδομένα και διαδικασίες μπορούν να ιδωθούν ενιαία με βάση το πεδίο ευθύνης κάποιων οντοτήτων που ονομάζονται «αντικείμενα». Κάθε αντικείμενο παρέχει στο περιβάλλον του ένα σύνολο υπηρεσιών της ευθύνης του. Η συνεργασία του συνόλου των αντικειμένων του πεδίου μιας εφαρμογής λογισμικού παράγει το επιθυμητό αποτέλεσμα. Μερικές από τις πιο γνωστές προσεγγίσεις που ανήκουν στην κατηγορία αυτή προτάθηκαν από τους Meyer (1988), Booch (1994), Jacobson (1993), Rumbaugh (1992).

4.3 ΑΝΤΙΚΕΙΜΕΝΟ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑ ΤΗΣ ΣΧΕΔΙΑΣΗΣ

Στην ενότητα αυτή θα καθοριστούν τα αντικείμενα εργασίας κατά την σχεδίαση καθώς και θα παρουσιαστεί ένας τρόπος παράστασης των αποτελεσμάτων της σχεδίασης με δομημένο κείμενο, αναλόγως με αυτόν του εγγράφου προδιαγραφών των απαιτήσεων από το λογισμικό.

4.3.1 ΕΙΣΑΓΩΓΗ

Στον ορισμό που δόθηκε στην προηγούμενη ενότητα το σχέδιο λογισμικού φέρεται ως «η περιγραφή των μονάδων που αποτελούν το λογισμικό, των συσχετίσεων μεταξύ τους, της διάταξής τους, καθώς και της εσωτερικής τους λεπτομέρειας». Προκειμένου να γίνει δυνατή η περιγραφή αυτή, πρέπει να αντιμετωπίσουμε το πρόβλημα σε τέσσερα επίπεδα ως ακολούθως:

- **Αρχιτεκτονική σχεδίαση**: Αφορά τον καθορισμό του ,ποιές είναι οι μονάδες που συγκροτούν το σύστημα λογισμικού και πως αυτές ανατίθενται για εκτέλεση στις υπολογιστικές μονάδες που είναι διαθέσιμες.

- **Σχεδίαση διαπροσωπειών:** Αφορά τον καθορισμό επικοινωνίας με άλλες συσκευές, με τον άνθρωπο καθώς και με άλλα συστήματα λογισμικού. Με τον όρο επικοινωνία εννοούμε ποια μονάδα επικοινωνεί με ποια καθώς , με ποιές παραμέτρους καθώς και με όποιο άλλο στοιχείο απαιτείται.

- **Λεπτομερής σχεδίαση μονάδων:** Αφορά τον καθορισμό της εσωτερικής δομής κάθε μονάδας ώστε να ικανοποιεί τις λειτουργικές απαιτήσεις καθώς και να συνεργάζεται με τις άλλες μονάδες όπως έχει καθοριστεί κατά την αρχιτεκτονική και την σχεδίαση δεδομένων.

- **Σχεδίαση δεδομένων:** Πρόκειται για τη λεπτομερή σχεδίαση των δεδομένων, η τήρηση των οποίων αποτελεί απαίτηση από το λογισμικό, η οποία έχει τεθεί στη φάση προδιαγραφής των απαιτήσεων

4.3.2 ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΣΧΕΔΙΑΣΗ

Κατά την αρχιτεκτονική σχεδίαση προσδιορίζεται το ποιες είναι οι μονάδες που αποτελούν το λογισμικό και πώς σχετίζονται μεταξύ τους. Ο προσδιορισμός αυτός αναφέρεται και ως σχεδίαση συστήματος. Στη δομημένη σχεδίαση η εργασία αυτή γίνεται λαμβάνοντας ως είσοδο τα διαγράμματα ροής δεδομένων. Σε αυτά εφαρμόζεται ένα σύνολο κανόνων και καθορίζονται οι μονάδες λογισμικού που είναι αναγκαίες για την υλοποίηση των μετασχηματισμών και των ροών δεδομένων. Η αρχιτεκτονική λογισμικού καθορίζει ποιες θα είναι αυτές και στη συνέχεια το λεπτομερές σχέδιο μονάδων θα καθορίσει πώς ακριβώς θα υλοποιηθούν. Σε όσο χαμηλότερο επίπεδο βρισκόμαστε, τόσο πιθανότερο είναι να αναγνωρίζονται «κοινοί παράγοντες», δηλαδή μονάδες λογισμικού που ανήκουν σε περισσότερα του ενός υποσυστήματα του αμέσως παραπάνω επιπέδου. Το φαινόμενο εμφανίζεται κυρίως μετά από την εκτέλεση της συναρτησιακής αποσύνθεσης (functional decomposition). Οι μονάδες που αποτελούν το λογισμικό μπορεί να ανατίθενται προς εκτέλεση σε πολλούς διατιθέμενους υπολογιστικούς πόρους (υπολογιστικά συστήματα, μονάδες επεξεργασίας). Στην απλούστερη περίπτωση, το λογισμικό είναι κατασκευασμένο ώστε να τρέχει εξ ολοκλήρου σε ένα μόνο υπολογιστικό σύστημα. Στη γενικότερη περί-πτωση, τα υποσυστήματα του λογισμικού μπορούν να ανατίθενται σε διαφορετικούς υπολογιστικούς πόρους.

4.3.3 ΣΧΕΔΙΑΣΗ ΔΙΑΠΡΟΣΩΠΕΙΩΝ

Καμία μονάδα λογισμικού δεν είναι ανεξάρτητη. Είτε χρησιμοποιεί άλλες μονάδες προκειμένου να επιτελέσει τη λειτουργία για την οποία προορίζεται είτε χρησιμοποιείται από άλλες μονάδες για τον ίδιο σκοπό. Η διατύπωση ότι μια μονάδα λογισμικού χρησιμοποιεί άλλες μονάδες δηλώνει την κλήση προγραμμάτων, υπό ρουτινών, συναρτήσεων και άλλων δομικών στοιχείων λογισμικού. Η κλήση μιας μονάδας Β από μια μονάδα Α σημαίνει:

- ενεργοποίηση της μονάδας Β, δηλαδή η ροή του προγράμματος μεταφέρεται στη μονάδα Β και εκτελούνται οι εντολές που περιέχονται σε αυτή, καθώς και
- επικοινωνία της μονάδας Α με τη Β μέσω παραμέτρων που περνά η Α στη Β και ενδεχομένως και αντίστροφα.

Κατά τη σχεδίαση διαπροσωπειών (interface design) καθορίζονται οι λεπτομέρειες του περάσματος των παραμέτρων αυτών σε όλα τα επίπεδα επικοινωνίας. Οι μονάδες λογισμικού δεν επικοινωνούν μόνο μεταξύ τους, αλλά και με εξωτερικά συστήματα ή συσκευές, καθώς και με τον άνθρωπο. Συγκεντρώνοντας το σύνολο των απαιτήσεων από τη σχεδίαση διαπροσωπειών, καταλήγουμε ότι κατά την εργασία αυτή θα πρέπει να καθοριστούν τα ακόλουθα:

- Ο αριθμός και ο τύπος των παραμέτρων κατά την κλήση μονάδων λογισμικού.

- Το είδος και οι λεπτομέρειες της επικοινωνίας με άλλα συστήματα λογισμικού.
- Το είδος και οι λεπτομέρειες της επικοινωνίας με εξωτερικές συσκευές.
- Η επικοινωνία του λογισμικού με τον άνθρωπο.

Οι παράμετροι κατά την επικοινωνία με άλλες μονάδες λογισμικού (στην ίδια εφαρμογή) καθορίζονται κατά την αρχιτεκτονική και τη λεπτομερή σχεδίαση και περιγράφονται στα διαγράμματα δομής προγράμματος και το λεπτομερές σχέδιο μονάδων, στα οποία θα αναφερθούμε σε επόμενες παραγράφους. Οι εξωτερικές διαπροσωπείες περιγράφονται αυτοτελώς με τον προσφορότερο κατά περίπτωση τρόπο, ανάλογα με τα εξωτερικά συστήματα λογισμικού ή τις εξωτερικές συσκευές. Η σχεδίαση της διαπροσωπείας με τον άνθρωπο είναι ένα ιδιαίτερα πολύπλοκο πρόβλημα, στο οποίο εκτός από μηχανικούς λογισμικού μπορεί να έχουν λόγο και άλλες ειδικότητες, όπως κοινωνιολόγοι και ψυχολόγοι, καθώς και κοινοί χρήστες.

4.3.4 ΛΕΠΤΟΜΕΡΗΣ ΣΧΕΔΙΑΣΗ ΜΟΝΑΔΩΝ

Το περισσότερο λεπτομερές επίπεδο της σχεδίασης είναι η σχεδίαση μονάδων. Πρόκειται για το σημείο όπου πρέπει να καθοριστούν όλες οι κατασκευαστικές λεπτομέρειες που απαιτούνται για τη δημιουργία του πηγαίου κώδικα για όλες τις μονάδες λογισμικού. Η γνώση της ύπαρξης μιας μονάδας και των χαρακτηριστικών επικοινωνίας αυτής με άλλες μονάδες δεν επαρκεί, στη γενική περίπτωση, για την κατασκευή του πηγαίου κώδικα της ίδιας της μονάδας. Σε τετριμμένες περιπτώσεις, η λεπτομερής περιγραφή της εσωτερικής δομής δεν είναι αναγκαία για την κατασκευή της μονάδας. Στις περισσότερες όμως των περιπτώσεων αυτό δεν ισχύει και η λεπτομερής σχεδίαση μονάδων είναι αναγκαίο υλικό που χρησιμοποιείται ως είσοδος στη φάση της λεπτομερούς σχεδίασης μονάδων είναι το «αρχιτεκτονικό σχέδιο», το «σχέδιο διαπροσωπιών», το «έγγραφο προδιαγραφών απαιτήσεων από το λογισμικό», καθώς και το «διάγραμμα μετάβασης καταστάσεων». Τα δύο πρώτα έχουν κατασκευαστεί κατά τη σχεδίαση, ενώ τα δύο τελευταία κατασκευάστηκαν κατά την προδιαγραφή των απαιτήσεων και περιέχουν περιγραφή της λειτουργικής συμπεριφοράς του λογισμικού με τη μορφή απαιτήσεων. Κατά τη λεπτομερή σχεδίαση, για κάθε μονάδα θα δοθεί ένα περίγραμμα - ομοίωμα της δομής της, χρήσιμο για την κατασκευή της. Αποτέλεσμα της λεπτομερούς σχεδίασης είναι το λεπτομερές σχέδιο μονάδων, το οποίο στη συνέχεια θα δοθεί στους προγραμματιστές που θα συγγράψουν τον πηγαίο κώδικα.

4.3.5 ΣΧΕΔΙΑΣΗ ΔΕΔΟΜΕΝΩΝ

Η σχεδίαση δεδομένων θεωρείται ως μια από τις σημαντικότερες διαδικασίες στην ανάπτυξη λογισμικού. Η σύλληψη της δομής και των συσχετίσεων των δεδομένων μπορεί να καθορίζει πολλά από τα χαρακτηριστικά των λειτουργικών μονάδων λογισμικού. Στη δομημένη ανάλυση και σχεδίαση, ο προσδιορισμός των απαιτήσεων και η λεπτομερής σχεδίαση των δεδομένων, από τη μία, και η σχεδίαση των μονάδων λογισμικού, από την άλλη, είναι δύο διαφορετικές διακριτές εργασίες.

Προκειμένου να μπορεί να κατασκευαστεί με πληρότητα το λεπτομερές σχέδιο μονάδων λογισμικού, είναι απαραίτητο να διατίθεται το λεπτομερές σχέδιο δεδομένων. Αυτό ισχύει για τις περιπτώσεις λειτουργικών μονάδων λογισμικού που επιδρούν, με οποιονδήποτε τρόπο, πάνω σε δεδομένα. Έχοντας, λοιπόν, υπόψη το λεξικό δεδομένων και το διάγραμμα οντοτήτων -

συσχετίσεων, ο σχεδιαστής λογισμικού:

- Πραγματοποιεί βελτιστοποιήσεις στο μοντέλο οντοτήτων - συσχετίσεων, με σκοπό τη βελτιστοποίηση των επιδόσεων, την πληροφοριακή πληρότητα και την εξάλειψη πλεονάζουσας (redundant) πληροφορίας.

Καθορίζει τη δομή κάθε στο φυσικό επίπεδο.

- Καθορίζει τις δυνατές απόψεις των δεδομένων στο λογικό επίπεδο.

Το αποτέλεσμα της σχεδίασης δεδομένων είναι η λεπτομερής περιγραφή των οντοτήτων και των συσχετίσεων αυτών, σε επίπεδο που να είναι εφικτή η υλοποίηση της Βάσης Δεδομένων.

4.3.6 ΤΟ ΕΓΓΡΑΦΟ ΠΕΡΙΓΡΑΦΗΣ ΤΟΥ ΣΧΕΔΙΟΥ ΤΟΥ ΛΟΓΙΣΜΙΚΟΥ

Σε αναλογία με το έγγραφο προδιαγραφών των απαιτήσεων από το λογισμικό, είναι χρήσιμο το σχέδιο του λογισμικού να αποτυπώνεται με χρήση ενός δομημένου εγγράφου. Το έγγραφο αυτό αποτελεί ένα ενιαίο δομικό κέλυφος στο οποίο ενσωματώνονται όλα τα επιμέρους προϊόντα της σχεδίασης με τη μορφή διαγραμμάτων, αλλά και με τη μορφή κειμένων ή άλλων τρόπων περιγραφής σχεδίου, τους οποίους θα αναφέρουμε στη συνέχεια. Μια γενική δομή του εγγράφου περιγραφής είναι η παρακάτω:

1.Εισαγωγή

1.1 Γενική περιγραφή

1.2 Εμβέλεια

1.3 Ορισμοί και ακρωνύμια

1.4 Αναφορές

2. Περιγραφή αποσύνθεσης του λογισμικού

2.1 Αποσύνθεση σε μονάδες

2.1.1 Περιγραφή μονάδας 1

2.1.2 Περιγραφή μονάδας 2

2.2 Αποσύνθεση σε ταυτόχρονες διεργασίες

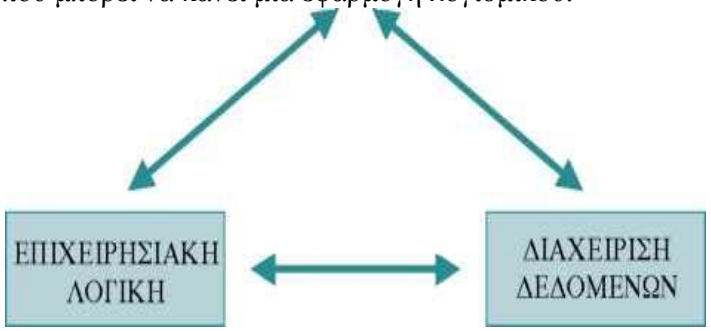
2.2.1 Περιγραφή διεργασίας 1

4.4 ΔΙΑΤΑΞΕΙΣ ΛΟΓΙΣΜΙΚΟΥ

Διάταξη λογισμικού είναι η κατάτμηση μιας εφαρμογής σε ανεξάρτητα λειτουργικά τμήματα και η ανάθεση αυτών σε διατιθέμενους υπολογιστικούς πόρους.

Με βάση τον παραπάνω ορισμό, με τον όρο «διάταξη» αναφερόμαστε στη γενική αρχιτεκτονική του λογισμικού. Σε αρκετές περιπτώσεις χρησιμοποιείται σκέτος ο όρος «αρχιτεκτονική», ιδιαίτερα εκεί όπου το λογισμικό θεωρείται από εξωτερική σκοπιά και όχι από αυτή του κατασκευαστή. Ακολούθως θα αναφερόμαστε στη γενική αρχιτεκτονική με τον όρο «διάταξη» και στην εσωτερική αρχιτεκτονική, στην οποία ήδη αναφερθήκαμε, με τον όρο «αρχιτεκτονική».

Ο ορισμός των διατάξεων που θα ακολουθήσει θα βασιστεί στην κατηγοριοποίηση των εργασιών που μπορεί να κάνει μια εφαρμογή λογισμικού.



Η κατηγοριοποίηση αυτή έχει γίνει γενικά αποδεκτή από κατασκευαστές και ερευνητές. Διακρίνονται τρία είδη εργασιών: οι εργασίες *παρουσίασης*, οι εργασίες *διαχείρισης δεδομένων* και οι εργασίες επιχειρησιακής λογικής.

Ως εργασίες **παρουσίασης** ορίζονται όλες οι εργασίες που σχετίζονται με την επικοινωνία του συστήματος με το χρήστη και με εξωτερικές συσκευές και συστήματα, δηλαδή εργασίες που υλοποιούν τις διαπροσωπείες του λογισμικού. Οι εργασίες **διαχείρισης δεδομένων** είναι εκείνες που ασχολούνται με την αποθήκευση και την ανάκτηση των δεδομένων. Τέλος, οι εργασίες **επιχειρησιακής λογικής** (business logic) είναι όλες οι υπόλοιπες εργασίες, δηλαδή εκείνες που υλοποιούν τις ιδιαίτερες λειτουργικές απαιτήσεις κάθε εφαρμογής λογισμικού.

Η διάκριση αυτή απαιτεί μια αυστηρότητα στον ορισμό των μονάδων λογισμικού. Μια μονάδα η οποία εκτελεί έναν υπολογισμό (εργασία επιχειρησιακής λογικής) δε θα πρέπει να τυπώνει η ίδια το αποτέλεσμά του στην οθόνη (εργασία παρουσίασης).

4.4.1 Η ΜΟΝΟΛΙΘΙΚΗ ΔΙΑΤΑΞΗ

Η απλούστερη διάταξη λογισμικού είναι η μονολιθική. Σε αυτήν ολόκληρη η εφαρμογή τρέχει σε ένα υπολογιστικό σύστημα. Η διάταξη αυτή είναι κατάλληλη για μικρές εφαρμογές με σχετικά

περιορισμένες απαιτήσεις και λειτουργίες και, όπως είναι αναμενόμενο, υπήρξε η πρώτη διάταξη που για μεγάλο χρονικό διάστημα χρησιμοποιήθηκε. Ο συμβολισμός που χρησιμοποιείται στο Σχήμα
αναφέρεται ως διάγραμμα διάταξης λογισμικού (deployment diagram) και περιγράφει την ανάθεση τμημάτων της εφαρμογής σε υπολογιστικούς πόρους. Τα τμήματα αυτά συμβολίζονται με ένα τρισδιάστατο παραλληλεπίπεδο σκιασμένο όπως στο σχήμα, στην μπροστινή πλευρά του οποίου αναγράφεται η ονομασία κάθε τμήματος

4.4.2 Η ΔΙΑΤΑΞΗ ΠΕΛΑΤΗ- ΕΞΥΠΗΡΕΤΗΤΗ

Η αύξηση των απαιτήσεων από το λογισμικό της πολυπλοκότητας αλλά και του όγκου των δεδομένων που διαχειρίζεται μια εφαρμογή, κατέστησαν τη μονολιθική διάταξη ανεπαρκή για την ικανοποίηση πολλών απαιτήσεων. Παράλληλα, η ανάπτυξη των συστημάτων διαχείρισης σχεσιακών βάσεων δεδομένων, αλλά και των δικτύων, επέτρεψαν την εξέλιξη της μονολιθικής διάταξης σε αυτή του πελάτη-εξυπηρετητή. Οι εργασίες που σχετίζονται με τη διαχείριση δεδομένων ανατίθενται σε ένα ξεχωριστό τμήμα της εφαρμογής, το οποίο τρέχει συνήθως σε ένα αφιερωμένο στη διαχείριση δεδομένων υπολογιστικό σύστημα. Οι εργασίες παρουσίασης και επιχειρησιακής λογικής τρέχουν σε ένα άλλο τμήμα, το οποίο επικοινωνεί μέσω δικτύου με τον εξυπηρετητή ζητώντας του την παροχή σχετικών με δεδομένα υπηρεσιών. Η ιδέα, πρωτοποριακή για την εποχή της, έλυσε το πρόβλημα των ολοένα και μεγαλύτερων υπολογιστικών απαιτήσεων από τα γιγαντωμένα μονολιθικά συστήματα, οι οποίες μεγάλωναν μαζί με τον αριθμό των χρηστών αλλά και την πολυπλοκότητα των εργασιών που εκτελούσαν. Με τον καιρό, διάφορα προβλήματα της διάταξης αυτής αναδείχτηκαν. Το σημαντικότερο εντοπίζεται στην ανάγκη συντήρησης όλων των συστημάτων πελάτη (τα οποία μπορούσαν να είναι πολυάριθμα), καθώς συνέβαιναν οποιεσδήποτε μεταβολές στο επίπεδο της επιχειρησιακής λογικής. Επίσης, όσο μεγάλωνε η πολυπλοκότητα των λειτουργιών, τόσο περισσότερο τα συστήματα όπου έτρεχαν τα συστήματα πελάτη αποδεικνύονταν ανεπαρκή από πλευράς υπολογιστικής ισχύος.

4.4.3 Η ΤΡΙΜΕΡΗΣ ΔΙΑΤΑΞΗ

Εμφανίστηκαν, λοιπόν, νέες εκδοχές της διάταξης πελάτη- εξυπηρετητή που διαχωρίζουν ακόμη περισσότερο τις «αρμοδιότητες» του λογισμικού. Ο πελάτης ελαφρύνεται και μένει μόνο με την ευθύνη της παρουσίασης, ενώ εμφανίζεται και ένας δεύτερος τύπος εξυπηρετητή, ο εξυπηρετητής εφαρμογών, ο οποίος κάνει τις εργασίες του επιπέδου της επιχειρησιακής λογικής. Η διάταξη αναφέρεται ως τριμερής. Στην περίπτωση αυτή, ο πελάτης χαρακτηρίζεται ως ελαφρής, ακριβώς διότι κάνει λιγότερα πράγματα απ’ ότι στην αρχική εκδοχή της διάταξης πελάτη-εξυπηρετητή. Τα συστήματα των εξυπηρετητών δεδομένων και εφαρμογών είναι συνήθως μεγάλα κεντρικά υπολογιστικά συστήματα. Τα προβλήματα της διάταξης πελάτη-εξυπηρετητή περιορίζονται, διότι οι απαιτήσεις συντήρησης των πελατών υφίστανται μόνο όταν συμβαίνουν μεταβολές στο επίπεδο της παρουσίασης. Πάντως, δεν παύουν να υπάρχουν.

4.4.4 Η ΠΟΛΥΜΕΡΗΣ ΔΙΑΤΑΞΗ

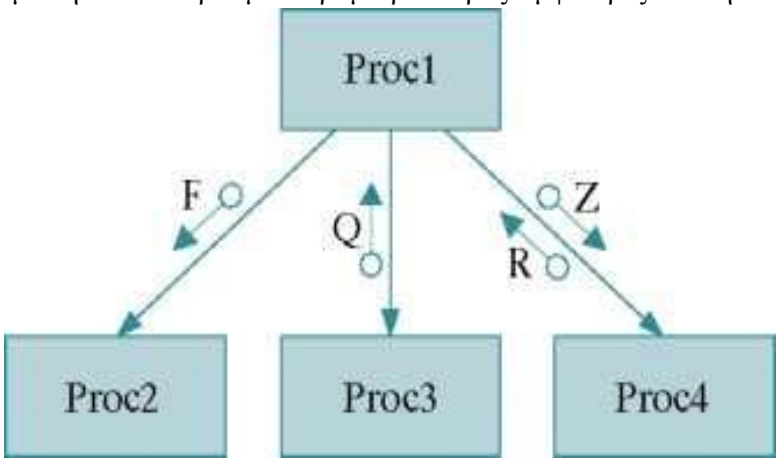
Το επόμενο αναμενόμενο βήμα είναι η αφαίρεση και της αρμοδιότητας της παρουσίασης από τον πελάτη και η ανάθεσή της σε έναν εξυπηρετητή παρουσίασης. Η ιδέα του εξυπηρετητή παρουσίασης γεννήθηκε με την εμφάνιση του παγκόσμιου ιστού (world wide web) και του Internet και έγινε δυνατή με την ανάπτυξη τεχνολογιών που επιτρέπουν την αλληλεπίδραση μεταξύ του web server και του συνδεδεμένου σε αυτόν πελάτη (browser). Τέτοιες τεχνολογίες είναι τα scripts, η Java, καθώς και το μοντέλο DCOM, και σήμερα είναι ήδη αρκετά διαδεδομένες. Η διάταξη ονομάστηκε πολυμερής (multi-tier) ή βασισμένη στο web (web based).
Τα τμήματα της εφαρμογής λογισμικού διατάσσονται με τρόπο ώστε όλες οι εργασίες των τριών κατηγοριών (παρουσίασης, διαχείρισης δεδομένων και επιχειρησιακής λογικής) να εκτελούνται σε έναν ή περισσότερους για κάθε κατηγορία εξυπηρετητές. Ο εξυπηρετητής παρουσίασης δεν είναι παρά ένας εξυπηρετητής υπηρεσιών web (web server). Ο πελάτης δεν απαιτείται να διαθέτει κανένα τμήμα της εφαρμογής, παρά μόνο τη δυνατότητα επικοινωνίας με τον web server, δηλαδή μια δικτυακή σύνδεση και ένα πρόγραμμα πλοήγησης στο web (browser). Για το λόγο αυτό, ο πελάτης αναφέρεται και ως «web client». Τα προβλήματα ανάγκης συντήρησης των συστημάτων των πελατών εκμηδενίζονται, γεννιούνται όμως άλλα, αυτά της ταχύτητας και της ασφάλειας των δικτυακών συνδέσεων.

4.5ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΣΧΕΔΙΑΣΗ

Στην ενότητα αυτή θα περιγραφεί η διαδικασία κατασκευής του *αρχιτεκτονικού σχεδίου συστήματος* με βάση το διάγραμμα ροής δεδομένων, η οποία ακολουθείται στη δομημένη σχεδίαση.

4.5.1 ΟΡΙΣΜΟΙ

Η μέθοδος της δομημένης σχεδίασης βασίζεται στα διαγράμματα ροής δεδομένων, από τα οποία, με κάποιο συστηματικό αλλά όχι αυτόματο τρόπο, παράγεται το αρχιτεκτονικό σχέδιο του λογισμικού. Το σχέδιο αυτό αποτυπώνεται με τη βοήθεια ενός διαγράμματος που ονομάζεται διάγραμμα δομής προγράμματος. Στο διάγραμμα δομής προγράμματος οι μονάδες λογισμικού παριστάνονται με ένα παραλληλόγραμμο. Η κλήση της μονάδας Β από τη μονάδα Α υποδηλώνεται με ένα βέλος που συνδέει την Α με τη Β. Το πέρασμα παραμέτρων προς αμφότερες υποδηλώνεται με ένα βέλος με έναν κύκλο στο άκρο της αρχής του, σημειωμένο σε διεύθυνση παράλληλη με τη διεύθυνση του βέλους που περιγράφει την κλήση, το οποίο φέρει το όνομα της παραμέτρου. Η διαδικασία εκτελείται για καθένα από τα διαγράμματα ροής δεδομένων του συστήματος. Στα

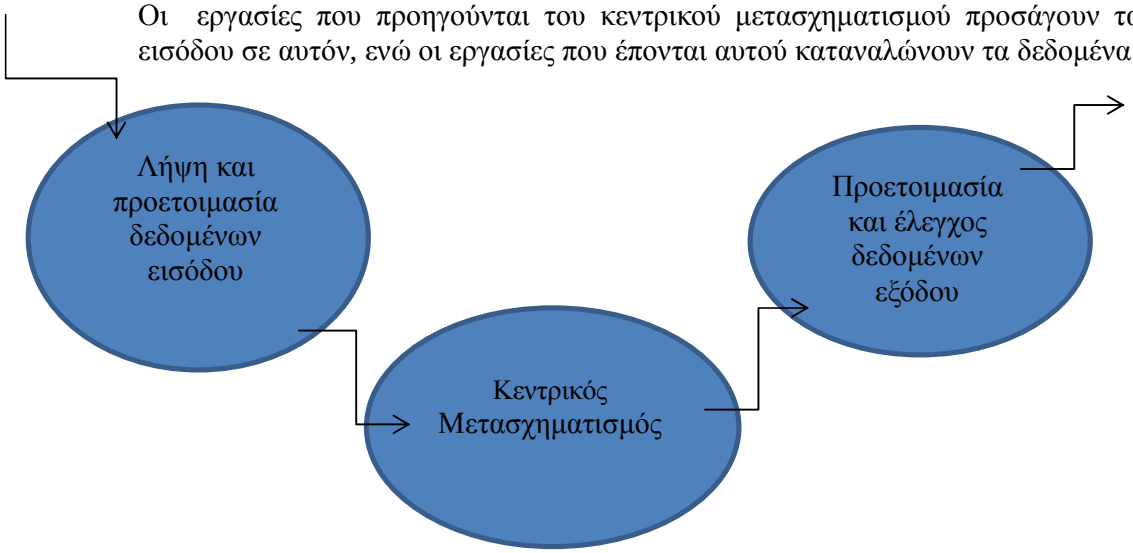


διαγράμματα αυτά εντοπίζονται δύο τύποι χαρακτηριστικών περιοχών, οι *κεντρικοί μετασχηματισμοί* και τα *κέντρα δοσοληψιών*. Όταν εντοπιστεί μια τέτοια περιοχή, δημιουργεί- ται ή συμπληρώνεται ένα επίπεδο του διαγράμματος δομής προγράμματος και η διαδικασία επαναλαμβάνεται μέχρις ότου να εξαντληθεί το διάγραμμα ροής δεδομένων.

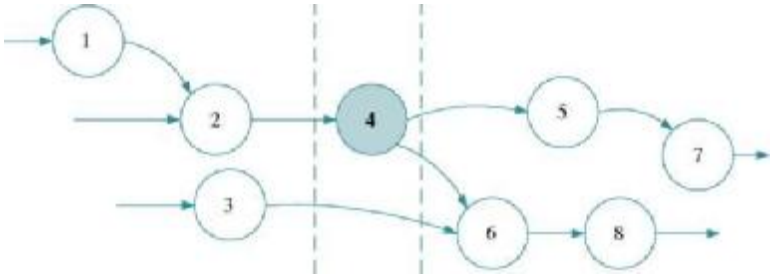
- **ΚΕΝΤΡΙΚΟΣ ΜΕΤΑΣΧΗΜΑΤΙΣΜΟΣ**

Ως κεντρικός μετασχηματισμός ορίζεται μια περιοχή ενός διαγράμματος ροής δεδομένων, οι μετασχηματισμοί που ανήκουν στην οποία μπορεί να θεωρηθεί ότι μετατρέπουν δεδομένα εισόδου προκειμένου να δημιουργήσουν νέα δεδομένα εξόδου.

Οι εργασίες που προηγούνται του κεντρικού μετασχηματισμού προσάγουν τα δεδομένα εισόδου σε αυτόν, ενώ οι εργασίες που έπονται αυτού καταναλώνουν τα δεδομένα εξόδου.

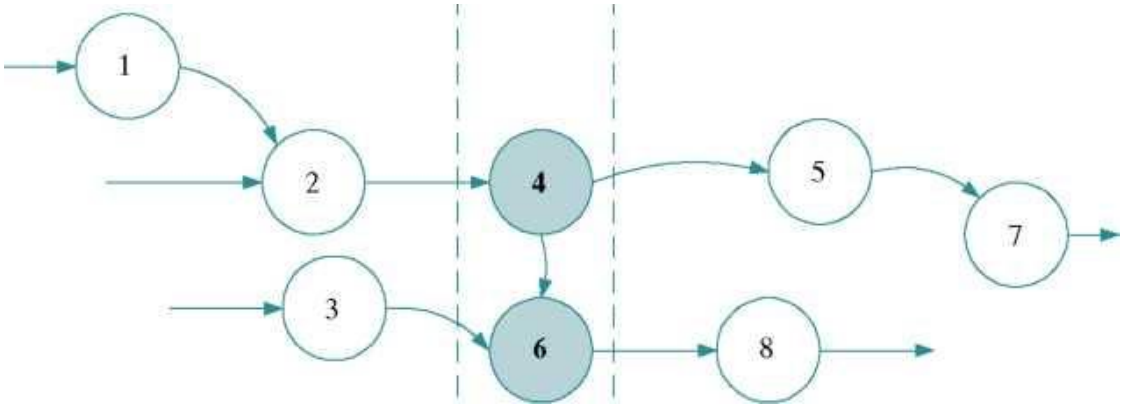


Στο παραπάνω σχήμα φαίνεται ένας κεντρικός μετασχηματισμός ως εντοπισμός χαρακτηριστικών περιοχών σε ένα υποσύνολο ενός διαγράμματος ροής δεδομένων. Στον κεντρικό μετασχηματισμό μπορεί να ανήκουν περισσότεροι του ενός μετασχηματισμοί και το ίδιο ισχύει για τα αριστερά και δεξιά τμήματα του διαγράμματος. Αυτό σημαίνει ότι ο κεντρικός μετασχηματισμός, η προετοιμασία δεδομένων εισόδου και η προετοιμασία δεδομένων εξόδου μπορούν να αντιστοιχούν σε σύνθετα τμήματα του διαγράμματος ροής δεδομένων και όχι σε απλούς μετασχηματισμούς. Δεν υπάρχει αυτόματος τρόπος για τον εντοπισμό των κεντρικών μετασχηματισμών. Η εμπειρία, ο αυτοσχεδιασμός και η διαίσθηση του μηχανικού λογισμικού έχουν και εδώ τον πρώτο λόγο. Επίσης, η επιλογή ενός κεντρικού μετασχηματισμού δεν είναι μοναδική. Αυτό σημαίνει ότι στο ίδιο διάγραμμα ροής δεδομένων δύο ή περισσότερες περιοχές μπορούν να χαρακτηριστούν ως κεντρικός μετασχηματισμός, χωρίς να είναι απαραίτητα λάθος ο ένας από τους δύο χαρακτηρι- σμούς. Η σύλληψη και η λεπτομέρεια του διαγράμματος ροής δεδομένων παίζουν, όπως είναι φανερό, καθοριστικό ρόλο στο σημείο αυτό.

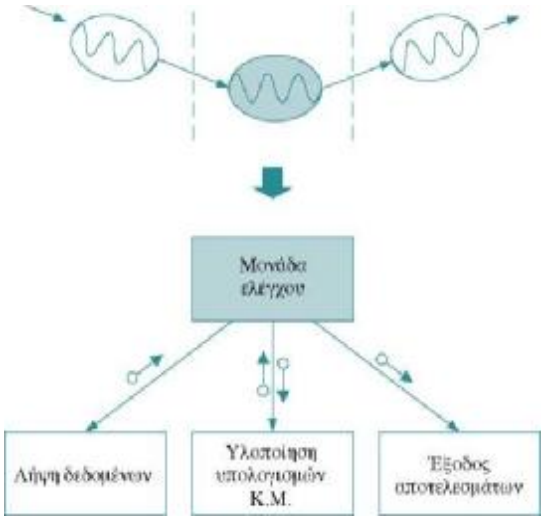


Στο παραπάνω σχήμα φαίνεται ένα διάγραμμα ροής δεδομένων στο οποίο θεωρείται ότι οι μετασχηματισμοί 1, 2 και 3 λαμβάνουν και προετοιμάζουν τα δεδομένα εισόδου. Ο μετασχηματισμός 4 κάνει την κύρια δουλειά της δημιουργίας νέων δεδομένων και χαρακτηρίζεται ως κεντρικός μετασχηματισμός, ενώ οι μετασχηματισμοί 5, 6, 7 και 8 προετοιμάζουν τα δεδομένα εξόδου.

Μια διαφορετική εκδοχή επί του ιδίου διαγράμματος ροής δεδομένων φαίνεται στο παρακάτω σχήμα. Η ιδέα είναι ότι και ο μετασχηματισμός 6 ανήκει στον κεντρικό μετασχηματισμό.



Καμία από τις δύο εκδοχές δεν μπορεί να χαρακτηριστεί ως καλύτερη από την άλλη χωρίς να είναι γνωστό το συγκεκριμένο πρόβλημα και ο ορισμός εκάστου των μετασχηματισμών του διαγράμματος ροής δεδομέν



Δίπλα έχουμε ένα κεντρικό μετασχηματισμό σε απεικόνισης διαγράμματος ροής. Η μονάδα ελέγχου εκτέλεσης καλεί τις μονάδες που απαιτείται για να λάβει τα δεδομένα εισόδου, τις μονάδες που απαιτείται να εκτελέσουν τον κεντρικό μετασχηματισμό και, τέλος, τις μονάδες στις οποίες θα διαθέσει τα δεδομένα εξόδου. Εδώ μπορούν να υπάρχουν περισσότερες από μια μονάδες λήψης και εξόδου αποτελεσμάτων, ανάλογα με το πλήθος των εισερχόμενων και εξερχόμενων ροών στον κεντρικό μετασχηματισμό, αντίστοιχα. Το σχέδιο που αντιστοιχεί στον κεντρικό μετασχηματισμό περιέχει τόσες μονάδες όσοι και οι απλοί μετασχηματισμοί που περιλαμβάνονται στον κεντρικό μετασχηματισμό.

Μια δεύτερη χαρακτηριστική περιοχή ενός διαγράμματος ροής δεδομένων είναι το κέντρο δοσοληψιών.

- Κέντρο δοσοληψιών

Ως κέντρο δοσοληψιών (transaction center) ορίζεται ένας μετασχηματισμός του διαγράμματος ροής δεδομένων, ο οποίος δέχεται κάποια δεδομένα εισόδου και παράγει ένα σύνολο δεδομένων εξόδου ανάλογα με την τιμή των δεδομένων εισόδου. Θ πρέπει να αναφερθεί πως ενώ ένας κεντρικός μετασχηματισμός μπορεί να περιλαμβάνει περισσότερους του ενός απλούς μετασχηματισμούς του διαγράμματος ροής δεδομένων, ένα κέντρο δοσοληψιών περιλαμβάνει μόνο έναν.

4.5.2 ΒΗΜΑΤΑ ΚΑΤΑΣΚΕΥΗΣ ΔΙΑΓΡΑΜΜΑΤΩΝ ΔΟΜΗΣ

Η μετάβαση από το διάγραμμα ροής δεδομένων στο διάγραμμα δομής γίνεται με διαδοχική επανάληψη κάποιων βημάτων, μέχρι να έχουν προσδιοριστεί μονάδες για όλους τους μετασχηματισμούς που περιέχονται στα διαγράμματα ροής δεδομένων της εφαρμογής. Τα βήματα αυτά είναι:

1. **Εντοπισμός κεντρικού μετασχηματισμού.** Για κάθε τμήμα του διαγράμματος ροής δεδομένων εντοπίζεται ο κεντρικός μετασχηματισμός και διακρίνονται οι μετασχηματισμοί εισόδου και εξόδου σε αυτόν.
2. **Απεικόνιση του κεντρικού μετασχηματισμού σε διάγραμμα** δομής. Δημιουργείται ένα επίπεδο του διαγράμματος δομής που αντιστοιχεί στον κεντρικό μετασχηματισμό.
3. **Παραγοντοποίηση (factoring).** Για το αριστερό και το δεξί τμήμα του κεντρικού μετασχηματισμού (είσοδοι και έξοδοι) δημιουργού- νται τα διαγράμματα δομής, που αντιστοιχούν στους μετασχηματισμούς που περιέχονται σε αυτά. Κάθε τέτοιος μετασχηματισμός απεικονίζεται σε μια μονάδα ελέγχου και σε δύο άλλες μονάδες. Από αυτές, η πρώτη λαμβάνει τα δεδομένα εισόδου, ενώ η δεύτερη πραγματοποιεί τη μετατροπή. Η μονάδα ελέγχου διαθέτει τα δεδομένα στο παραπάνω επίπεδο. Κατά την παραγοντοποίηση ενδεχομένως να αναγνωριστούν

και άλλοι κεντρικοί μετασχηματισμοί ή κέντρα δοσοληψιών, τα οποία αντιμετωπίζονται όπως περιγράφηκε. Η διαδικασία επαναλαμβάνεται μέχρις ότου να φτάσου- με στις πηγές και τους αποδέκτες των δεδομένων, δηλαδή το χρήστη, εξωτερικά συστήματα, συσκευές ή αρχεία.

4. **Συνένωση.** Η τελευταία εργασία που πρέπει να γίνει είναι αυτή της συνένωσης. Για τις περιπτώσεις όπου τα δεδομένα δε λαμβάνονται από εξωτερική πηγή ή δεν καταλήγουν σε εξωτερικό αποδέκτη, η μονάδα που τα εισάγει στον κεντρικό μετασχηματισμό αντικαθίσταται από τη μονάδα ελέγχου του μετασχηματισμού που τα παρέχει.

Κατά τη διαδικασία αυτή ενδεχομένως να διαπιστωθεί ότι είναι χρήσιμο να πραγματοποιηθούν τροποποιήσεις στα διαγράμματα ροής δεδομένων, γεγονός που συνιστά οπισθοδρόμηση στη διαδικασία ανάπτυξης λογισμικού, η οποία όμως είναι χρήσιμο να πραγματοποιηθεί.

4.6 ΛΕΠΤΟΜΕΡΗΣ ΣΧΕΔΙΑΣΗ ΜΟΝΑΔΩΝ

Έχοντας διαθέσιμο το αρχιτεκτονικό σχέδιο, είναι δυνατή η λεπτομερής σχεδίαση των μονάδων λογισμικού που περιλαμβάνονται σε αυτό. Κατά τη λεπτομερή σχεδίαση θα προσδιοριστεί η εσωτερική δομή κάθε μονάδας, δηλαδή θα δοθεί μια περιγραφή του πηγαίου κώδικα. Μέχρι το σημείο αυτό είναι γνωστή μόνο η ονομασία κάθε μονάδας και οι παράμετροι εισόδου και εξόδου αυτής. Στοιχεία που σχετίζονται με την περιγραφή της συμπεριφοράς της είναι διαθέσιμα από τη φάση της προδιαγραφής των απαιτήσεων. Με το υλικό αυτό ο σχεδιαστής λογισμικού καλείται να κατασκευάσει το λεπτομερές σχέδιο.

Σε περίπτωση που το διάγραμμα δομής προγράμματος είναι επαρκώς λεπτομερές, καθεμία μονάδα του διαγράμματος αντιστοιχεί σε μια μονάδα κώδικα. Αν όμως αυτό δεν ισχύει, είναι ανάγκη κάθε μονάδα του διαγράμματος δομής να απεικονιστεί, ενδεχομένως, σε περισσότερες από μία μονάδες λογισμικού. Η εργασία αυτή γίνεται αναλύοντας κάθε πρόβλημα σχεδίασης σε διαδοχικά βήματα. Σε κάθε βήμα δημιουργούμε ένα σύνολο από μικρότερα προβλήματα σχεδίασης, καθένα εκ των οποίων επιλύουμε και πάλι αναλύοντάς το κ .ο .κ., μέχρις ότου φτάσουμε σε απλές δομικές μονάδες λογισμικού, όπως οι «διαδικασίες» και οι «συναρτήσεις». Η αντιμετώπιση αυτή ονομάζεται εκλέπτυνση σε διαδοχικά βήματα (stepwise refinement) ή συναρτησιακή αποσύνθεση (functional decomposition).

Υπάρχουν αρκετοί τρόποι για να περιγράφονται τα αποτελέσματα της λεπτομερούς σχεδίασης. Ο επικρατέστερος είναι με χρήση μιας γλώσσας σχεδίασης προγράμματος (PDL: program description language). Μια γλώσσα σχεδίασης προγράμματος μοιάζει με γλώσσα προγραμματισμού, χωρίς να έχει την αυστηρότητα στη σύνταξη και τη γραμματική που χαρακτηρίζει τις γλώσσες προγραμματισμού. Σκοπός της είναι να παρέχει μια εικόνα του λογισμικού η οποία να μπορεί να υλοποιηθεί εύκολα σε κάποια γλώσσα προγραμματισμού. Οι γλώσσες σχεδίασης προγράμματος περιέχουν τις βασικές δομές ελέγχου που απα- ντώνται στις γλώσσες προγραμματισμού. Μια απλή τέτοια γλώσσα που ομοιάζει με την Pascal και στην οποία θα βασιστούν τα παραδείγματα που θα ακολουθήσουν στο πιο κάτω σχέδιο. Οι γλώσσες σχεδίασης προγράμματος αναφέρονται συχνά και ως ψευδοκώδικας.

ΑΛΛΕΣ ΕΚΦΡΑΣΕΙΣ	ΕΠΑΝΑΛΗΠΤΙΚΗ ΕΚΤΕΛΕΣΗ
/♦ΣΧΟΛΙΟ */ ΜΕΤΑΒΛΗΤΗ:-ΤΙΜΗ /* ΑΝΑΘΕΣΗ */ ΦΡΑΣΤΙΚΗ ΠΕΡΙΓΡΑΦΗ ΕΝΕΡΓΕΙΑΣ + - * / Λ /* ΜΑΘΗΜΑΤΙΚΕΣ ΕΚΦΡΑΣΕΙΣ */	FOR ΜΤΒΛ FROM ΤΙΜΗΪ TO ΤΙΜΗ2 STEP ΤΙΜΗ3 DO (ΕΝΕΡΓΕΙΕΣ) END FOR
ΕΚΤΕΛΕΣΗ ΜΕ ΕΠΙΛΟΓΗ ΠΕΡΙΠΤΩΣΗΣ	**ΕΚΤΕΛΕΣΗ ΥΠΟ ΣΥΝΘΗΚΗ**
CASE ΕΚΦΡΑΣΗ OF (ΤΙΜΗ 1) : (ΕΝΕΡΓΕΙΕΣ) (ΤΙΜΗ 2) : (ΕΝΕΡΓΕΙΕΣ) (ΤΙΜΗ Ν) : (ΕΝΕΡΓΕΙΕΣ) OTHERWISE (ΕΝΤΟΛΕΣ ΑΝ Η ΕΚΦΡΑΣΗ ΕΧΕΙ ΑΛΛΗ ΤΙΜΗ) END CASE	IF ΣΥΝΘΗΚΗ THEN (ΕΝΕΡΓΕΙΕΣ ΑΝ Η ΣΥΝΘΗΚΗ ΕΙΝΑΙ ΑΛΗΘΗΣ) ELISE (ΕΝΤΟΛΗ ΑΝ Η ΣΥΝΘΗΚΗ ΕΙΝΑΙ ΨΕΥΔΗΣ) END IF
ΕΠΑΝΑΛΗΠΤΙΚΗ ΕΚΤΕΛΕΣΗ ΜΕ ΣΥΝΘΗΚΗ (1)	**ΕΠΑΝΑΛΗΠΤΙΚΗ ΕΚΤΕΛΕΣΗ ΜΕ ΣΥΝΘΗΚΗ (2)**
REPEAT (ΕΝΕΡΓΕΙΕΣ) UNTIL ΣΥΝΘΗΚΗ	WHILE ΣΥΝΘΗΚΗ DO (ΕΝΕΡΓΕΙΕΣ) END WHILE
ΟΡΙΣΜΟΣ ΔΙΑΔΙΚΑΣΙΩΝ	**ΟΡΙΣΜΟΣ ΣΥΝΑΡΤΗΣΕΩΝ**
PROCEDURE ΟΝΟΜΑ (ΠΑΡΑΜΕΤΡΟΣ: IN/OUT ...) GLOBAL VAR ΟΝΟΜΑ 1, ΟΝΟΜΑ 2, ... LOCAL VAR ΟΝΟΜΑ1, ΟΝΟΜΑ 2, ... (ΕΝΕΡΓΕΙΕΣ) CALL ΟΝΟΜΑ ΔΙΑΔΙΚΑΣΙΑΣ (ΠΑΡΑΜΙ, ΠΑΡΑΜ2, .) (ΕΝΕΡΓΕΙΕΣ) END PROCEDURE	FUNCTION ΟΝΟΜΑ ΣΥΝΑΡΤΗΣΗΣ (ΠΑΡΑΜΕΤΡΟΣ ...) GLOBAL VAR ΟΝΟΜΑ 1, ΟΝΟΜΑ 2, ... LOCAL VAR ΟΝΟΜΑ 1, ΟΝΟΜΑ 2, ... (ΕΝΕΡΓΕΙΕΣ) ΟΝΟΜΑ ΣΥΝΑΡΤΗΣΗΣ: = ΤΙΜΗ (ΕΝΕΡΓΕΙΕΣ) END FUNCTION

Προκειμένου ο σχεδιαστής να κατασκευάσει το παραπάνω ομοίωμα έρχεται αντιμέτωπος με αρκετά ζητήματα όπως τα ακόλουθα:

1. Ποιο είναι το καλύτερο από τα σχέδια τα οποία μπορεί να συλλάβει;
2. Ποιος είναι ο προσφορότερος τρόπος για την περιγραφή του λεπτομερούς σχεδίου μονάδων;
3. Πώς σχετίζεται το σχέδιο με τη γλώσσα προγραμματισμού και το περιβάλλον υλοποίησης γενικότερα;
4. Πώς μπορεί ένα λεπτομερές σχέδιο μονάδων να διατηρείται επίκαιρο;

Οι απαντήσεις στα ερωτήματα αυτά δεν είναι εύκολες και εξαρτώνται από πλήθος παραμέτρων και υποκειμενικών προσεγγίσεων.

4.7 ΣΧΕΔΙΑΣΗ ΔΕΔΟΜΕΝΩΝ

Σκοπός της σχεδίασης δεδομένων είναι ο προσδιορισμός των δομών δεδομένων, η ύπαρξη των οποίων εντοπίστηκε κατά τη φάση της προδιαγραφής των απαιτήσεων από το λογισμικό. Ο προσδιορισμός αυτός πρέπει να είναι επαρκής για την απεικόνιση των δεδομένων σε ένα περιβάλλον υλοποίησης. Η εργασία της σχεδίασης δεδομένων είναι στενά συνυφασμένη με την εργασία κατάστρωσης των υπόλοιπων τμημάτων του σχεδίου του λογισμικού, στις οποίες αναφερθήκαμε. Συχνά, η εργασία σχεδίασης δεδομένων υποτιμάται από τους κατασκευαστές σε σχέση με την υπόλοιπη σχεδίαση.

Η σχεδίαση δεδομένων μπορεί να μελετηθεί αυτοτελώς σε σημαντικό βάθος και πλάτος. Παραθέτουμε τις εργασίες που πρέπει οπωσδήποτε να εκτελούνται κατά τη λεπτομερή σχεδίαση δεδομένων:

- Τροποποίηση του μοντέλου οντοτήτων - συσχετίσεων, την αρχική μορφή του οποίου κατασκευάσαμε κατά τη συγγραφή των προδιαγραφών με σκοπό αυτό να περιέχει ακριβώς όση πληροφορία απαιτείται, κατανεμημένη σωστά μεταξύ των οντοτήτων. Η διαδικασία αυτή ονομάζεται «κανονικοποίηση».
- Καθορισμός των πεδίων που περιέχει κάθε πίνακας στο επίπεδο της φυσικής αποθήκευσης δεδομένων.
- Καθορισμός των εναλλακτικών μορφών εμφάνισης της οργάνωσης των δεδομένων πάνω από το φυσικό επίπεδο (ορισμός views).
- Βελτιστοποιήσεις με σκοπό τη βελτιστοποίηση των επιδόσεων και, γενικά, την επίτευξη κριτηρίων επάρκειας.

Η εργασία απεικόνισης οντοτήτων του πεδίου της εφαρμογής με τη βοήθεια ενός σχήματος δεδομένων είναι από τις πιο ενδιαφέρουσες, δημιουργικές και σημαντικές εργασίες κατά την ανάπτυξη του λογισμικού. Μερικές αρχές που είναι χρήσιμο να ακολουθούνται κατά τη διαδικασία αυτή είναι οι ακόλουθες:

- Η σχεδίαση δεδομένων δεν πρέπει να υποτιμάται. Η απόδοση χρόνου και σημασίας σε αυτή και η πειθαρχημένη εφαρμογή αρχών σχεδίασης ανάλογων με αυτές που χρησιμοποιούνται για την κατασκευή του αρχιτεκτονικού σχεδίου του λογισμικού ανταποδίδουν πάντα το χρόνο που καταναλώνουν.
- Θα πρέπει να εντοπιστούν όλες οι δομές δεδομένων πάνω στα οποία επιδρούν οι μονάδες λογισμικού. Μετά από τη σχεδίαση δε θα πρέπει να υπάρχουν δεδομένα των οποίων ο προσδιορισμός να αφήνεται για το μέλλον.
  - Το λεξικό δεδομένων θα πρέπει να τηρείται ενημερωμένο.
- Οι αποφάσεις σχεδίασης δεδομένων που σχετίζονται με κατασκευαστικές λεπτομέρειες θα πρέπει να λαμβάνονται όσο αργότερα γίνεται. Η επιδίωξη αποφυγής της εξάρτησης των αποτελεσμάτων της σχεδίασης δεδομένων από το περιβάλλον υλοποίησης (όπως ένα σύστημα διαχείρισης βάσεων δεδομένων) συνήθως απαιτεί σημαντική προσπάθεια.
- Δεν είναι καλή ιδέα οποιαδήποτε μονάδα λογισμικού να μπορεί να διαβάσει και να μεταβάλει δεδομένα. Είναι σκόπιμο κατά τη σχεδίαση να καθορίζεται ποιες μονάδες λογισμικού επιτρέπεται να επιδρούν κατευθείαν στα δεδομένα. Με τον τρόπο αυτό μειώνονται οι πιθανότητες παρενεργειών στο λογισμικό ,καθώς πραγματοποιούνται μεταβολές στην εσωτερική δομή των δεδομένων.

- Θα πρέπει να διερευνάται η δυνατότητα χρήσης έτοιμων και δοκιμασμένων δομών δεδομένων από βιβλιοθήκες. Αν και η γενικευμένη χρήση βιβλιοθηκών συστατικών λογισμικού δεν έχει γίνει μέχρι σήμερα δυνατή, η εφαρμογή της ιδέας σε μικρή κλίμακα μέσα στην εμβέλεια ενός κατασκευαστή λογισμικού συχνά αποδίδει.

Βιβλιογραφία

Βιβλία:

- Ι.ΒΟΓΙΑΤΖΗΣ- Η.ΑΝΤΩΝΟΠΟΥΛΟΥ *ΕΙΣΑΓΩΓΗ ΣΤΗΝ ΠΛΗΡΟΦΟΡΙΚΗ*
- ΒΑΣΙΛΕΙΟΣ ΒΕΣΚΟΥΚΗΣ *ΑΡΧΕΣ ΤΕΧΝΟΛΟΓΙΑΣ ΛΟΓΙΣΜΙΚΟΥ-ΤΟΜΟΣ Α΄-ΤΕΧΝΟΛΟΓΙΑ ΛΟΓΙΣΜΙΚΟΥ Ι*

Χρήσιμα Προγράμματα:

- Για τα σχεδιαγράμματα, σχήματα και γραφικές παραστάσεις χρησιμοποιήσαμε το πρόγραμμα *Photofiltre.*
- Επίσης για κάποια σχεδιαγράμματα χρησιμοποιήθηκε το πρόγραμμα *Microsoft Word*

Ηλεκτρονικές Σελίδες:

- *www.wikipedia.org*
- http://www.de.teipat.gr/index.php/genikesplirofories/52-2008-11-14-19-57-05/92-odigosspoudastes.html
- http://www.ct.aegean.gr/people/vkavakli/software_engineering/slides/2_requirements_engineering.pdf
- http://www.multilingualarchive.com/ma/enwiki/el/Software_crisis
- http://www.scribd.com/doc/68501162/93/%CE%9A%CF%81%CE%AF%CF%83%CE%B7-%CE%BB%CE%BF%CE%B3%CE%B9%CF%83%CE%BC%CE%B9%CE%BA%CE%BF%CF%8D
- http://users.softlab.ece.ntua.gr/~bxb/files/v.vescoukis_atm.pdf
- http://nemertes.lis.upatras.gr/jspui/bitstream/10889/268/1/150.pdf
- http://dmst.aueb.gr/dds/intro/softeng/indexw.htm
- http://moumoutzis.blogspot.com/2008/11/h.html
- http://users.softlab.ece.ntua.gr/~bxb/books/TLSlides_Chapter12.pdf
- http://techedu.unipi.gr/files/notes/2010-2011/eksamino_3/texnologia_logismikoy/tl_lecture01_eisagwgi_4site_compatibility_mode.pdf http://www.arnos.gr/dmdocuments/eap/plh/plh11/ergasiesplh11/2008-2009.1-1.on.line.ergasia.pli11.sol.eap.pdf
- http://www.arnos.gr/dmdocuments/yliko/plhroforiki/texnologialogismikou/shmeiwseis.texnologia.logismikou.eisagwgh.eap.pdf
- http://users.softlab.ece.ntua.gr/~bxb/pub1/gr_phd.htm
- http://aetos.it.teithe.gr/~dranidis/IS_Notes_1.pdf
- http://www.etpe.gr/files/proceedings/26/1286267956_94.pdf
- http://www.asda.gr/tee02per/manpap/educat16.html
- http://panayotiskoutsopinis.blogspot.com/2011/02/blog-post.html
- http://www.pi-schools.gr/programs/ktp/previous_version/book1/12.pdf

- http://www.musesnet.gr/ekp2000/VASIKA%20PC.htm
- http://iiu.teikav.edu.gr/iiw/courses/eksamino_05/tello_ii/ejamina/0910XE/pdf/TP_DELDP_12.pdf

- www.wikipedia.org
- http://www.oxagile.com/article/228-software-development-models
- http://www.ics.uci.edu/~wscacchi/Papers/SE-Encyc/Process-Models-SE-Encyc.pdf
- http://www.ba.duth.gr/files/Parousiase2.pdf
- http://dtps.unipi.gr/files/notes/2007-2008/eksamino_3/texnologia_logismikoy/22_1192185501.pdf
- http://www.dmst.aueb.gr/dds/intro/softeng/index.htm