ΤΕΙ ΜΕΣΟΛΟΓΓΙΟΥ
ΤΜΗΜΑ ΤΗΛΕΠ/ΚΩΝ ΣΥΣΤΗΜΑΤΩΝ ΚΑΙ ΔΙΚΤΥΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΣΥΓΚΡΙΤΙΚΗ ΜΕΛΕΤΗ ΚΑΙ ΑΝΑΛΥΣΗ ΤΕΧΝΙΚΩΝ ΣΧΕΔΙΑΣΜΟΥ ΤΗΛΕΠΙΚΟΙΝΩΝΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ ΜΕ ΕΜΦΑΣΗ ΣΤΑ ΜΟΝΤΕΛΑ ΕΠΙΚΟΙΝΩΝΙΑΣ ΚΙΝΗΤΗΣ ΤΗΛΕΦΩΝΙΑΣ

ΚΟΥΤΣΙΚΟΥ ΕΥΘΥΜΙΑ
ΑΜ: 0103

Επιβλέπων Καθηγητής: Δρ. ΠΡΑΓΙΑΤΗ ΑΓΓΕΛΙΚΗ

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

1 ΕΙΣΑΓΩΓΗ 6

2 ΣΗΜΑΝΤΙΚΕΣ ΤΕΧΝΙΚΕΣ ΣΧΕΔΙΑΣΜΟΥ ΛΟΓΙΣΜΙΚΟΥ 8

2.1 Εισαγωγή 8

2.2 Ο σχεδιασμός top-down και bottom-up 8

2.3 Σταδιακή βελτίωση (Stepwise Refinement) 10

2.4 Δομημένος σχεδιασμός (Structure Design) 11

2.5 Τεχνική δομημένης ανάλυσης και σχεδιασμού 15

2.6 Δομημένη ανάπτυξη συστημάτων 16

2.7 Αντικειμενοστραφής σχεδιασμός 16

3 ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΤΗΛΕΠΙΚΟΙΝΩΝΙΑΚΟΥ ΣΥΣΤΗΜΑΤΟΣ: ΤΥΠΟΠΟΙΗΜΕΝΕΣ ΓΛΩΣΣΕΣ 18

3.1 Απαιτήσεις του συστήματος τηλεπικοινωνιών 18

3.2 Χρήστες και συστήματα 18

3.2.1 Παράδειγμα 18

3.2.2 Τυποποίηση του παραδείγματος 19

3.3 Συστήματα και συστήματα 21

3.3.1 Παράδειγμα 22

3.3.2 Τυποποίηση του παραδείγματος 22

3.4 Προβλήματα 23

3.5 Χαρακτηριστικά των τηλεπικοινωνιακών συστημάτων 23

4 ΜΈΘΟΔΟΙ ΜΟΝΤΕΛΟΠΟΙΗΣΗΣ ΣΥΣΤΗΜΑΤΩΝ 24

4.1 Estelle 25

4.1.1 Γλωσσική επισκόπηση 25

4.1.2 Ανάλυση της Estelle 27

4.2 LOTOS 28

4.2.1 Γλωσσική επισκόπηση 28

4.2.2 Περιγραφή 29

4.3 SDL 29

4.3.1 Γλωσσική επισκόπηση 29

4.3.2 Ανάλυση της SDL 32

4.4 UML 33

4.4.1 Φάσεις ανάπτυξης ενός συστήματος 33

4.4.2 Διαγράμματα της UML 34

4.5 Απαιτήσεις των τηλεπικοινωνιακών συστημάτων από το σχεδιαστικό μοντέλο . 37
5 ΜΕΘΟΔΟΛΟΓΙΑ ΠΡΟΣ ΤΟΝ ΣΧΕΔΙΑΣΜΟ ΓΙΑ ΤΑ ΑΦΙΕΡΩΜΕΝΑ ΣΥΣΤΗΜΑΤΑ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΣΕ ΕΠΙΠΕΔΟ ΣΥΣΤΗΜΑΤΟΣ 39
5.1 Εξέλιξη των υπηρεσιών τηλεπικοινωνίας 39
5.2 Μέθοδοι σχεδιασμού συστημάτων 39
5.2.1 Κύκλος σχεδιασμού συστημάτων 39
5.2.2 Ροή σχεδιασμού 40
5.2.3 Βιομηχανικά εργαλεία 41
5.2.4 Ερευνητικές δραστηριότητες και πρόσφατη ανάπτυξη 42
5.2.5 Δραστηριότητες Lester 43
5.2.6 Έρευνα για τις τεχνικές σχεδιασμού για τη διαχείριση δυναμικής ισχύος σε επίπεδο-συστήματος 45
5.3 Σχεδιασμός ενσωματωμένων συστημάτων υλικού-λογισμικού για εφαρμογές δικτύων τηλεπικοινωνιών 58
5.3.1 Το πρότυπο προγραμματισμού 59
5.3.2 Η μεθοδολογία σχεδιασμού 61
5.3.3 Προσομοίωση και διόρθωση 62
5.3.4 Διαχωρισμός υλικού / λογισμικού 62
5.3.5 Εφαρμογή λογισμικού 62
5.3.6 Εφαρμογή υλικού 63
5.3.7 Σύνθεση επικοινωνίας υλικού / λογισμικού 64
5.3.8 Ετερογενής συσχεδιασμός και συμπροσομοίωση 64
6 Σύγκριση των Επίσημων Γλωσσών 66
6.1 Θεμελιώδη κριτήρια επιλογής 66
6.2 Σημαντικά κριτήρια επιλογής 67
7 Γενικά Συμπεράσματα 71
8 ΒΙΒΛΙΟΓΡΑΦΙΑ – ΑΝΑΦΟΡΕΣ 73

ΣΧΗΜΑΤΑ

Σχήμα 1: Διάγραμμα μηχανής μιας πεπερασμένης κατάστασης κλήσης 19
Σχήμα 2: Διάγραμμα μηχανής ενός πεπερασμένης κατάστασης πράκτορα συστήματος μεταγωγής 20
Σχήμα 3: Διάγραμμα μηχανής μιας πεπερασμένης κατάστασης κλήσης 21
Σχήμα 4: Ένας πίνακας προδιαγραφών της αυτόματης μεταγωγής προστασίας 23
Σχήμα 5: Παράδειγμα ενός τύπου καναλιών και περιπτώσεων σε Estelle 26
Σχήμα 6: SDL Block διάγραμμα επιπέδου συστήματος 31
Σχήμα 7: SDL Block διαγράμματα 32
Σχήμα 8: Διάγραμμα UML περίπτωση χρήσης 35
Σχήμα 9: Διάγραμμα UML κλάσεων 35
Σχήμα 10: Διάγραμμα UML αντικειμένων 36
Σχήμα 11: Διάγραμμα UML καταστάσεων 36
Σχήμα 12: Διάγραμμα UML δραστηριοτήτων 37
Σχήμα 13: Κύκλος Top-Down σχεδιασμού 40
Σχήμα 14: Ροή σχεδιασμού συστήματος 40
Σχήμα 15: Εργαλεία διαμόρφωσης πρωτοτύπου και σχεδιασμού 41
Σχήμα 16: Ερευνητική εστίαση 42
Σχήμα 17: Έρευνα ενδιαφερόντων και περιοχών εργασίας του εργαστηρίου Lester 43
Σχήμα 18: Διεπαφή ACPI και πλατφόρμα PC 52
Σχήμα 19: Ορισμοί κατάστασης για την ACPI 53
Σχήμα 20: Η DPM χρησιμοποιώντας τους οδηγούς φίλτρων 54
Σχήμα 21: Η PSM για IBM DTTA HDD 55
Σχήμα 22: Στατιστική ανάλυση για τον χρόνο αφίξεως 56
Σχήμα 23: Ένα απλό παράδειγμα μοντέλου 60
Σχήμα 24: Ροή σχεδιασμού συστήματος Matisse σε περιβάλλον CoWare 61

ΠΙΝΑΚΕΣ

Πίνακας 1: Παράμετροι δίσκων 55
Πίνακας 2: Σύγκριση γλωσσών 67

ΠΕΡΙΛΗΨΗ

Στόχος της πτυχιακής αυτής εργασίας είναι η σε βάθος κατανόηση των διαφορών των τεχνικών σχεδιασμού συστημάτων και η εξοικείωση με την διαδικασία μοντελοποίησης επικοινωνιακών πρωτοκόλλων για συστήματα κινητής τηλεφωνίας. Τα σύγχρονα συστήματα τηλεπικοινωνιών είναι τόσο περίπλοκα που απαιτούν συστηματική μελέτη, ανάλυση και σχεδίαση με γλώσσες μοντελοποίησης ικανές να εκφράσουν όλα τα χαρακτηριστικά των συστημάτων αυτών. Στα πλαίσια της πτυχιακής αυτής εργασίας, αναλύονται τα χαρακτηριστικά και οι σχεδιαστικές απαιτήσεις των συστημάτων τηλεπικοινωνιών. Στη συνέχεια, περιγράφονται οι σχεδιαστικές προσεγγίσεις και οι τυποποιημένες γλώσσες μοντελοποίησης. Ακολουθούν διάφορες σχεδιαστικές προσεγγίσεις που εστιάζουν σε μερικές πτυχές των απαιτήσεων όπως η διαχείριση δυναμικής ισχύος σε επίπεδο συστήματος, η ανάλυση των απαιτήσεων του χρήστη (υψηλό επίπεδο) και η συ-σχεδίαση υλικού-λογισμικού (χαμηλό επίπεδο), Τέλος, μελετάται η καταλληλότητα των προσεγγίσεων αυτών μέσα από μια συγκριτική μελέτη και ανάλυση τόσο των απαιτήσεων των συστημάτων όσο και των ιδιοτήτων των σχεδιαστικών τεχνικών.

SUMMARY

Objective of final is in-depth comprehension of differences of techniques of design of systems and the familiarization with the process of modeling of communication protocols on systems of mobile telephony. Modern telecommunications systems are so complicated that require systematic study, analysis and designing with modeling languages which are sufficient to express all characteristics of these systems. In frames of this final work, the characteristics and designing requirements of telecommunication systems are analyzed. Afterwards, the designing approaches and the standardized modeling languages are described. Follow several approaches that focus in certain aspects of requirements as the dynamic power management, the analysis user's requirements (high level) and the co-design of software/hardware (low level). Finally, is studied the appropriateness of these approaches through a comparative study and analysis of so much requirements of systems of what attributes of designing techniques

1 ΕΙΣΑΓΩΓΗ

Τα συστήματα τηλεπικοινωνιών γίνονται όλο και περισσότερο σύνθετα. Χρειαζόμαστε λοιπόν γλώσσες που να επιτρέπουν τον έλεγχο των προδιαγραφών του συστήματος για λάθη, με εργαλεία αυτοματοποίησης. Ο σχεδιασμός συστημάτων τηλεπικοινωνιών παρουσιάζει ένα ευρύ φάσμα λειτουργιών, όπως η γρήγορη εξέλιξη και η αναβάθμιση προϊόντων, οι ενσωματωμένοι περιορισμοί σχεδιασμού (χαμηλή ισχύς, κ.λπ.), η αυξανόμενη πολυπλοκότητα των κανόνων και μια πραγματική ανάγκη της ευελιξίας για τα ανοικτά συστήματα. Έτσι, οι σχεδιαστές πρέπει να έχουν νέες δεξιότητες μεθοδολογιών σχεδιασμού για να εκτελέσουν συγχρόνως την επικύρωση σχεδιασμού συστημάτων και την ολοκλήρωση των τεχνολογικών περιορισμών.

Η μεθοδολογία σχεδιασμού ορίζεται ως ένα σύνολο διαδικασιών που αυτό προκύπτει από την αρχή στην ολοκλήρωση της διαδικασίας ανάπτυξης λογισμικού. Η φύση της μεθοδολογίας εξαρτάται από διάφορους παράγοντες, συμπεριλαμβανομένου του περιβάλλοντος ανάπτυξης λογισμικού, τις πρακτικές της οργάνωσης, τη φύση ή τον τύπο του λογισμικού που αναπτύσσεται, τις απαιτήσεις των χρηστών, τα προσόντα και την κατάρτιση της ομάδας ανάπτυξης λογισμικού, τους διαθέσιμους πόρους υλικού και λογισμικού, τις υπάρχουσες ενότητες σχεδιασμού διαθεσιμότητας, ακόμη και τον προϋπολογισμό και το χρονικό σχεδιάγραμμα. Υπάρχουν δύο κατηγορίες μεθοδολογιών σχεδιασμού: α) οι συστηματικοί τύποι οι οποίοι είναι λιγότερο μαθηματικοί και αποτελούνται από το διαδικαστικό μέρος, το οποίο ορίζει ποιά ενέργεια ή διεργασία πρέπει να εκτελεστεί καθώς και το στοιχείο αναπαράστασης που καθορίζει τη δομή λογισμικού. β) οι επίσημες μεθοδολογίες που χρησιμοποιούν μαθηματικές δομές αναπαράστασης για τους μετασχηματισμούς των αντικειμένων και τον έλεγχο συμβατότητας.

Γενικά, οι τεχνικές από τις μεθοδολογίες συστηματικού σχεδιασμού μπορούν να ενσωματωθούν και μπορούν να χρησιμοποιήσουν τα σχέδια απεικόνισης από άλλες τεχνικές και ανάλογα με την περίπτωση. Εξαιτίας του γεγονότος ότι οι μεθοδολογίες έχουν αναπτυχθεί σε διαφορετικό περιβάλλον για να απευθυνθούν συγκεκριμένα σε ορισμένες ομάδες προβλημάτων ή ενός προβλήματος, δεν υπάρχει καμία κοινή βασική γραμμή στην οποία να αξιολογήσει ή να συγκρίνει τις μεθοδολογίες η μια ενάντια στην άλλη.

Η βιομηχανία τηλεπικοινωνιών έχει προσπαθήσει να λύσει αυτό το πρόβλημα με την πρόταση των γλωσσών μοντελοποίησης και των τεχνικών σχεδιασμού που προηγείται της υλοποίησης προσπαθώντας να δώσει λύση στην αυξημένη πολυπλοκότητα των τηλεπικοινωνιακών συστημάτων. Χαρακτηριστικές γλώσσες που έχουν προκύψει από αυτές τις προσεγγίσεις και οι οποίες έχουν προτυποποιηθεί είναι οι Estelle, LOTOS, SDL και UML.

Ανάλυση γίνεται και στη διαχείριση δυναμικής ισχύος (DPM) η οποία είναι μια μεθοδολογία σχεδιασμού για συστήματα δυναμικής μετατροπής, για να παρέχει τις ερωτηθείσες υπηρεσίες και τα επίπεδα απόδοσης με έναν ελάχιστο αριθμό ενεργών στοιχείων ή ένα ελάχιστο φορτίο σε τέτοια στοιχεία. Η DPM καλύπτει ένα σύνολο τεχνικών που επιτυγχάνει τον υπολογισμό της απόδοσης της ενέργειας με επιλογή τον τερματισμό ή την μείωση της απόδοσης των τμημάτων του συστήματος όταν είναι σε κατάσταση IDLE. Εξετάζεται πώς οι διαφορετικοί σχεδιασμοί της DPM έχουν εφαρμοστεί στα κυκλώματα και τα συστήματα.

Εξετάζεται επίσης ο σχεδιασμός σε επίπεδο συστήματος, και περιγράφεται πώς η διαχείριση ισχύος εφαρμόζεται στα συστήματα υλικού/λογισμικού, ειδικότερα σχετικά με τη λειτουργία διαχείριση ισχύος βασισμένη στη μελέτη των συστημάτων. Απ' την άλλη, τα σύγχρονα συστήματα τηλεπικοινωνιών αυξάνονται γρήγορα όσον

αφορά στην πολυπλοκότητα σχεδιασμού. Η διαμόρφωση στη διαχείριση δικτύων απαιτείται για να υποστηρίξει μια ευρεία ποικιλία των ευρυζωνικών υπηρεσιών και πολυμέσων. Ο σχεδιασμός αυτών των συστημάτων είναι συχνά μια προσπάθεια που πραγματοποιείται σε πολλαπλάσια sites.
Τέτοια σύνθετα συστήματα απαιτούν έναν συνδυασμό τμημάτων υλικού (hardware) και τμημάτων λογισμικού (software) προκειμένου να φτάσουν οι απαραίτητες λειτουργίες στο επιθυμητό επίπεδο απόδοσης και προγραμματικής επεξεργασίας. Αυτά τα τμήματα υλικού και λογισμικού πρέπει να σχεδιαστούν ταυτόχρονα για να ελαχιστοποιήσουν το χρόνο στην αγορά. Οι εφαρμογές δικτύων τηλεπικοινωνιών περιλαμβάνουν τα τμήματα συστημάτων για τα ευρυζωνικά δίκτυα βασισμένα στο ATM, τις κινητές υποδομές δικτύων για να υποστηρίξουν την κυψελοειδή επικοινωνία βασισμένη στο GSM, τα δίκτυα βασισμένα στα SONET και SDH, και τους διαλογικούς κεντρικούς υπολογιστές.
Τέλος με την βοήθεια πινάκων γίνεται η σύγκριση των επίσημων γλωσσών κάνοντας επίσης αναφορά και στα βασικά κριτήρια αυτών, τα οποία χωρίζονται σε δυο κατηγορίες: τα θεμελιώδη και τα σημαντικά κριτήρια, ο στόχος των οποίων είναι να βοηθήσουν στην αξιολόγηση της καταλληλότητας των γλωσσών για την βιομηχανική ανάπτυξη λογισμικού.

2 ΣΗΜΑΝΤΙΚΕΣ ΤΕΧΝΙΚΕΣ ΣΧΕΔΙΑΣΜΟΥ ΛΟΓΙΣΜΙΚΟΥ

2.1 Εισαγωγή

Μια μεθοδολογία μπορεί να οριστεί απλά ως ένα σύνολο διαδικασιών που η καθεμία προκύπτει από την αρχή προς την ολοκλήρωση της διαδικασίας ανάπτυξης λογισμικού. Η φύση της μεθοδολογίας εξαρτάται από διάφορους παράγοντες, συμπεριλαμβανομένου του περιβάλλοντος ανάπτυξης λογισμικού, τις πρακτικές της οργάνωσης, τη φύση ή τον τύπο του λογισμικού που αναπτύσσεται, τις απαιτήσεις των χρηστών, τα προσόντα και την κατάρτιση της ομάδας ανάπτυξης λογισμικού, τους διαθέσιμους πόρους υλικού και λογισμικού, τις υπάρχουσες ενότητες σχεδιασμού διαθεσιμότητας, ακόμη και τον προϋπολογισμό και το χρονικό σχεδιάγραμμα. Από τη δεκαετία του '70, έχει υπάρξει ένας πολλαπλασιασμός των μεθοδολογιών σχεδιασμού λογισμικού. Οι διαφορετικές μεθοδολογίες έχουν αναπτυχθεί για να επιλύσουν τους διαφορετικούς τύπους προβλημάτων. Στην περιγραφή αυτών των προβλημάτων, είναι συχνά δυνατό ή απαραίτητο να συγκεντρωθούν τα προβλήματα με τα παρόμοια χαρακτηριστικά. Αυτό καλείται περιοχή προβλήματος. Πολλές από αυτές τις μεθοδολογίες έχουν εξελιχθεί από το συγκεκριμένο περιβάλλον όπου το λογισμικό αναπτύχθηκε. Αυτό σημαίνει ότι οι συγκεκριμένες μεθοδολογίες αναπτύσσονται συχνά (για να εφαρμοστούν) για να επιλύσουν ορισμένες «κατηγορίες» προβλημάτων, ακόμα κι αν οι μηχανισμοί σχεδιασμού είναι διαφορετικοί σε κάθε μεθοδολογία.

Υπάρχουν δύο ευρείες κατηγορίες μεθοδολογιών σχεδιασμού: οι συστηματικοί και επίσημοι τύποι. Όπως το όνομα αφήνει να εννοηθεί, ο επίσημος τύπος κάνει την εκτενή χρήση των μαθηματικών σημειώσεων για τους μετασχηματισμούς αντικειμένου και για τον έλεγχο των συνεπειών. Ο συστηματικός τύπος είναι λιγότερο μαθηματικός και αποτελείται από το διαδικαστικό συστατικό, το οποίο ορίζει ποιά δράση ή ποιος στόχος θα εκτελεστεί καθώς και το τμήμα απεικόνισης, το οποίο ορίζει πώς η δομή λογισμικού πρέπει να αντιπροσωπευθεί. Γενικά, οι τεχνικές από τις μεθοδολογίες συστηματικού σχεδιασμού μπορούν να ενσωματωθούν και μπορούν να χρησιμοποιήσουν τα σχέδια απεικόνισης. Εξαιτίας του γεγονότος ότι οι μεθοδολογίες έχουν αναπτυχθεί σε διαφορετικό περιβάλλον για να απευθυνθούν συγκεκριμένα σε ορισμένες ομάδες προβλημάτων ή ενός προβλήματος, δεν υπάρχει καμία κοινή βασική γραμμή η οποία να αξιολογεί ή να συγκρίνει τις μεθοδολογίες τη μια ενάντια στην άλλη. Εντούτοις, οι αρχές των μεθοδολογιών μπορούν να αναλυθούν και να εξεταστούν για μια καλύτερη κατανόηση της βάσης για κάθε μεθοδολογία. Με μια καλύτερη κατανόηση της μεθοδολογίας, η περιοχή εφαρμογής της, μπορεί να εφαρμοστεί αποτελεσματικότερα ή να καθοριστεί ακριβέστερα.

Γενικά, ο εναλλακτικός σχεδιασμός επιτρέπει τη σημαντική ανάλυση ανταλλαγής πριν να κωδικοποιηθεί το λογισμικό. Κατά συνέπεια, η οικειότητα με διάφορες μεθοδολογίες κάνει τη δημιουργία των ανταγωνιστικών σχεδίων λογικότερη και πιο συστηματική με τη λιγότερη εμπιστοσύνη στην έμπνευση.

2.2 Ο σχεδιασμός top-down και bottom-up

Ο Top-down σχεδιασμός κατευθύνει τους σχεδιαστές να αρχίσουν με μια από την κορυφή περιγραφή ενός συστήματος. Με κάθε βελτίωση, το σύστημα αποσυντίθεται,

στο επίπεδο και τις μικρότερες ενότητες. Η Top-down αποσύνθεση απαιτεί τις σημαντικότερες υψηλότατου επιπέδου απαιτήσεις και λειτουργίες συστημάτων, και έπειτα τις σπάει διαδοχικά έως ότου μπορούν να σχεδιαστούν οι συγκεκριμένες λειτουργικές ενότητες. Κατά συνέπεια, ο Top-down σχεδιασμός είναι μια προσανατολισμένη ισόπεδη προσέγγιση σχεδιασμού.

Οι Top-down στρατηγικές είναι το θέμα μερικών προηγούμενων μελετών (Miller & Lindamood 1973, Budgen 1989, Yourdon 1979). Ο Top-down σχεδιασμός μειώνει το πεδίο και το μέγεθος κάθε ενότητας, και εστιάζει περισσότερο στα συγκεκριμένα ζητήματα. Είναι από τη φύση μια επαναληπτική διαδικασία όπου κάθε βελτίωση θα αποσυνθέσει μια ενότητα και πιο συγκεκριμένα μια λεπτομερή υπο-ενότητα έως ότου φθάσει σε ένα σημείο όπου το «ατομικό» επίπεδο επιτυγχάνεται. Μέσω αυτής της επαναληπτικής διαδικασίας, οι αποφάσεις κάνουν τα ανώτερα επίπεδα να έχουν μια σημαντική επίδραση στην επόμενη αποσύνθεση και στα χαμηλά επίπεδα. Κατά συνέπεια, υπάρχει μια δυνατότητα ότι οι αποφάσεις που λαμβάνονται στο ανώτερο επίπεδο θα οδηγήσουν σε μια αστήρικτη ή δύσχρηστη ή ανεπαρκή αποσύνθεση στα χαμηλά επίπεδα. Για να γίνει αυτό, πρέπει να υπάρξει ένα σημαντικό ποσό οπισθοδρόμησης όπου οι αποφάσεις των πιο υψηλών επιπέδων πρέπει να επαναξιολογηθούν και να αναδομηθούν έπειτα αναλόγως. Προκειμένου να ελαχιστοποιηθεί η οπισθοδρόμηση, συχνά οι σχεδιαστές αρχίζουν την αποσύνθεση σε ένα μέσο επίπεδο περίπου, παρά στην κορυφή, ή χρησιμοποιούν έναν Top-down σχεδιασμό αλλά πρώτα καθορίζουν τις χαμηλές-ισόπεδες ενότητες. Αυτό είναι, σε αντίθεση με την bottom-up προσέγγιση, όπου οι χαμηλότερες ενότητες ή τα βασικά λειτουργίες καθορίζονται πρώτες και έπειτα προστίθενται οι πρόσθετες ενότητες ή οι λειτουργίες..

Η bottom-up προσέγγιση έχει μελετηθεί επίσης από τους (Freeman & Wasserman 1983, Von Mayrhauser 1990, Jalote 1991). Στην bottom-up προσέγγιση, οι σχεδιαστές πρέπει να προσδιορίσουν ένα βασικό σύνολο ενοτήτων και αλληλεξαρτήσεών τους, που μπορεί να χρησιμοποιηθούν ως θεμέλιο για τη λύση του προβλήματος. Οι υψηλότερου επιπέδου έννοιες διατυπώνονται βασισμένες στα βασικά. Ο bottom-up σχεδιασμός είναι επίσης μια επαναληπτική διαδικασία, και μπορεί να καταλήξει στη σημαντική οπισθοδρόμηση εάν τα βασικά δεν κατασκευάζονται κατάλληλα. Το όφελος του bottom-up σχεδιασμού είναι ότι επιτρέπει την αξιολόγηση των υποενοτήτων κατά τη διάρκεια της διαδικασίας ανάπτυξης των συστημάτων. Εκτιμάται δε, ότι στον top-down σχεδιασμό η αξιολόγηση απόδοσης μπορεί να γίνει μόνο όταν ενσωματώνεται στο πλήρες σύστημα. Εντούτοις, ο top-down σχεδιασμός επιτρέπει την πρόωρη αξιολόγηση των λειτουργικών ικανοτήτων σε επίπεδο χρηστών με τη χρησιμοποίηση των παλαιών συνηθειών για τις χαμηλότερες ενότητες. Κατά συνέπεια, στην αρχή του προγράμματος, οι σημαντικότερες διεπαφές (interfaces) μπορούν να εξεταστούν, να ελεγχθούν ή να ασκηθούν. Το όφελος με την χρησιμοποίηση του top-down σχεδιασμού είναι ότι η κύρια εστίαση είναι στην απαίτηση των πελατών και τη γενική φύση του προβλήματος που πρέπει να λυθεί. Επίσης η διόρθωση είναι ευκολότερη από μερικές άλλες μεθόδους.

Στην πραγματικότητα, η καθεαυτού top-down ή η bottom-up προσέγγιση χρησιμοποιείται σπάνια. Η top-down προσέγγιση είναι καταλληλότερη όταν το πρόβλημα και το περιβάλλον του είναι καθορισμένα με σαφήνεια, παραδείγματος χάριν, στο σχεδιασμό ενός μεταγλωττιστή. Όταν το πρόβλημα είναι λάθος

καθορισμένο, η προσέγγιση πρέπει κυρίως να είναι bottom-up ή μικτή. Η top-down προσέγγιση έχει οδηγήσει στην εξέλιξη μιας πολύ δημοφιλούς μεθοδολογίας σχεδιασμού η οποία καλείται δομημένος σχεδιασμός.

2.3 Σταδιακή βελτίωση (Stepwise Refinement)

Η επαναληπτική διαδικασία όπου κάθε σύστημα αποσυντίθεται και βελτιώνεται βαθμιαία καλείται σταδιακή βελτίωση. Είναι μια ισόπεδη-προσανατολισμένη προσέγγιση σχεδιασμού. Η σταδιακή βελτίωση προτάθηκε αρχικά από τον Wirth (1974). Καθορίζει μια σειρά βημάτων βελτίωσης. Σε κάθε βήμα, το σύστημα ή η ενότητα αποσυντίθεται στα υποσυστήματα ή τις υποεννότητες. Κάθε βελτίωση της ενότητας πρέπει να συνοδευθεί από μια βελτίωση της δομής και της σχέσης μεταξύ των ενοτήτων. Επιπλέον, κάθε διαδοχικό βήμα στη διαδικασία βελτίωσης πρέπει να είναι μια πιστή επέκταση του προηγούμενου βήματος. Ο βαθμός συναρμολογησιμότητας που προέρχεται από την βελτίωση θα καθορίσει το εύρος της ενότητας. Δεδομένου ότι το σύστημα ή η ενότητα βελτιώνεται βαθμιαία, χρειάζεται μια δομή για την αναπαράσταση των αλλαγών στην διαδικασία. Η φύση αυτής της σημείωσης πρέπει να είναι τέτοια που να μπορεί να αναπαραστήσει τη δομή του συστήματος και των στοιχείων της. Πρέπει επίσης να έχει μια στενή σχέση με γλώσσα προγραμματισμού που χρησιμοποιείται για την κωδικοποίηση. Κάθε βελτίωση είναι βασισμένη σε κάποιο αριθμό σχεδιαστικών αποφάσεων και σε ένα ορισμένο σύνολο σχεδιαστικών κριτηρίων. Με κάθε βήμα βελτίωσης, οι σχεδιαστικές αποφάσεις αποσυντίθενται σε ένα χαμηλότερο επίπεδο, έως ένα στοιχειώδες επίπεδο.

Η σταδιακή βελτίωση αρχίζει με τις προδιαγραφές που λαμβάνονται από την απαιτούμενη ανάλυση. Η λύση στο πρόβλημα αναλύεται αρχικά σε μερικές σημαντικές ενότητες ή διαδικασίες που θα λύσουν ευαπόδεικτα το πρόβλημα. Κατόπιν μέσω της διαδοχικής βελτίωσης, κάθε ενότητα αποσυντίθεται έως ότου υπάρχουν ικανοποιητικές λεπτομέρειες έτσι ώστε η εφαρμογή σε μια γλώσσα προγραμματισμού να είναι απλή. Κατά αυτόν τον τρόπο, σε ένα πρόβλημα που κόβεται σε μικρότερα μέρη, οι εύχρηστες μονάδες και το ποσό των λεπτομερειών που πρέπει να προσεχθούν σε οποιοδήποτε σημείο, ελαχιστοποιούνται. Αυτό επιτρέπει στο σχεδιαστή να διοχετεύσει τις πηγές του σε ένα συγκεκριμένο ζήτημα όταν έρθει η στιγμή. Δεδομένου ότι η σταδιακή βελτίωση αρχίζει στο επίπεδο κορυφής, η επιτυχία εξαρτάται ιδιαίτερα από την εννοιολογική κατανόηση του σχεδιαστή, του πλήρη προβλήματος και της επιθυμητής λύσης.

Ο Parnas (1972) έχει προτείνει μερικές οδηγίες για την αποσύνθεση των συστημάτων στις ενότητες, αποκαλούμενη επίσης συναρμολογησιμότητα. Για να επιτευχθεί καλή διαμόρφωση, κάθε ενότητα πρέπει να εκτελέσει μόνο έναν συγκεκριμένο, ευδιάκριτο στόχο. Οι εισαγωγές και τα αποτελέσματά της είναι καθορισμένες με σαφήνεια. Κατά συνέπεια, τα λάθη και οι ανεπάρκειες μπορούν να επισημανθούν εύκολα στις συγκεκριμένες ή ιδιαίτερες ενότητες ώστε να ελαχιστοποιούν τη διόρθωση. Επίσης, κατ' αυτόν τον τρόπο, η συντήρηση είναι τμηματικής φύσης. Η συναρμολογησιμότητα επιτρέπει σε μια ενότητα να κωδικοποιηθεί χωρίς οποιαδήποτε γνώση του κώδικα στις άλλες ενότητες. Επιτρέπει επίσης στις ενότητες να συγκεντρωθούν εκ νέου και να αντικατασταθούν χωρίς επανασυναρμολόγηση ολόκληρου του συστήματος. Το λογισμικό έχει οριστεί ως μια οικογένεια των προγραμμάτων σύμφωνα με τον Parnas (1979). Υπάρχουν κύρια προγράμματα, θυγατρικά προγράμματα, καθώς επίσης και διαφορετικές γενεές προγραμμάτων.

Μερικά από τα μέλη μιας οικογένειας προγράμματος μπορούν να διαφέρουν: η διαμόρφωση του hardware (υλικού) με την οποία τρέχουν είναι διαφορετική, η εισαγωγή και τα δεδομένα εξόδου είναι διαφορετικά ακόμα κι αν η διενεργηθείσα λειτουργία είναι η ίδια, ο αλγόριθμος και οι βάσεις δεδομένων διαφοροποιούνται όσον αφορά τους διαθέσιμους πόρους και στο μέγεθος των συνόλων εισαγωγής ή τη σχετική συχνότητα ορισμένων γεγονότων, μερικοί χρήστες μπορεί να χρειαστούν μόνο ένα υποσύνολο των χαρακτηριστικών γνωρισμάτων, που άλλοι χρήστες χρησιμοποιούν.

Υπάρχουν βασικά τέσσερις κατηγορίες εμποδίων που παρεμβαίνουν στην επέκταση ή στην συστολή ενός προγράμματος: α) η διανομή υπερβολικών πληροφοριών όπου γράφονται πολλά προγράμματα που επιτρέπονται για την απουσία ή την παρουσία κάποιων λειτουργιών, β) η αλυσίδα δεδομένων που μετασχηματίζουν τα επιμέρους στοιχεία του συστήματος, όπου τα δεδομένα μετασχηματίζονται διαδοχικά από συστατικό σε συστατικό, γ) τα συστατικά που εκτελούν περισσότερες από μια λειτουργίες, δ) οι βρόχοι (loops) στην κλήση άλλων συστατικών όπου το λογισμικό λειτουργεί μόνο όταν λειτουργούν όλα τα άλλα συστατικά. Υπάρχουν δύο βασικά μέτρα για τη διαμόρφωση της συνοχής και της σύζευξης ενός συστήματος. Η συνοχή ενδιαφέρεται για τις αλληλεξαρτήσεις μεταξύ των στοιχείων μέσα σε μια ενότητα, ενώ η σύζευξη ενδιαφέρεται για τις αλληλεξαρτήσεις μεταξύ των διαφορετικών ενοτήτων.

Το πεδίο εφαρμογής της σταδιακής βελτίωσης είναι ευρύ. Περιλαμβάνει ένα οποιοδήποτε μη τετριμμένο προβληματικό σύστημα που μπορεί λογικά ή και λειτουργικά, να αποσυντεθεί. Και επειδή δεν είναι οριακό από οποιεσδήποτε συγκεκριμένες τεχνικές απεικόνισης ή σχεδιασμού, χρησιμοποιείται συχνά από κοινού με άλλες μεθοδολογίες σχεδιασμού.

2.4 Δομημένος σχεδιασμός (Structure Design)

Ο δομημένος σχεδιασμός (SD) αναπτύχθηκε αρχικά από τους Stevens, Myers and Constantine (1974). Είναι μια βάση δεδομένων σχεδιαστικής προσέγγισης προσανατολισμένης-ροής. Έχει γίνει πιθανώς η δημοφιλέστερη μεθοδολογία του σχεδιασμού του λογισμικού. Είναι εύχρηστο και υπάρχουν κριτήρια αξιολόγησης που μπορούν να χρησιμεύσουν ως οδηγός στο σχεδιασμό λογισμικού. Ο πιο διαδεδομένος σχεδιασμός που χρησιμοποιούν οι χρήσεις του SD είναι το διάγραμμα ροής στοιχείων (DFD). Ο SD εξαρτάται εννοιολογικά από τρεις λογικές (Peters, 1981): σύνθεση και βελτίωση του σχεδιασμού, διαχωρισμός των ζητημάτων στην αφαίρεση, και εφαρμογή αξιολόγησης των αποτελεσμάτων. Από τη συνθετική λογική (compositional rationale), ο SD βλέπει τα συστήματα από δύο προοπτικές: σαν τη ροή δεδομένων και σαν τους μετασχηματισμούς όπου η ροή δεδομένων υποβάλλεται μέσω ενός συστήματος. Δεδομένου ότι οι ροές δεδομένων και οι ενέργειες μετατροπής είναι τα μόνα χαρακτηριστικά που απεικονίζονται στα DFDs, το στοιχείο του χρόνου δεν είναι παρόν. Κατά συνέπεια, ο σχεδιαστής μπορεί ακριβώς να εστιάσει στους μετασχηματισμούς των ροών δεδομένων μέσω ενός συστήματος. Μέσω της αντίληψης για το σύστημα ως ροές δεδομένων και μετατροπές, υπάρχει ελάχιστη παραλλαγή στην κατάρτιση του προτύπου συστημάτων, κατά συνέπεια η μορφή ή η δομή του συστήματος διατηρείται. Αυτό έρχεται σε αντίθεση με τον top-down σχεδιασμό όπου οι αποφάσεις που λαμβάνονται στο υψηλό επίπεδο έχουν

επιπτώσεις στην αποσύνθεση των κατώτερων επιπέδων. Επιπλέον, η αλληλεξάρτηση αυτών των ροών δεδομένων και των μετασχηματισμών, θα οδηγήσει στον προσδιορισμό και στην οργάνωση των ενοτήτων που απαιτούνται στην οικοδόμηση του προτύπου συστήματος.

Από τη λογική των περιλήψεων ή της εφαρμογής, η διαδικασία του SD προτείνει μια διαφοροποίηση μεταξύ του λογικού σχεδιασμού (περίληψη) και του φυσικού σχεδιασμού (εφαρμογή). Ένα κοινό πρόβλημα που αντιμετωπίζουν οι σχεδιαστές, είναι το μεγάλο δίλημμα να καταλάβουν το πρόβλημα και να βρουν τη λύση του συγχρόνως. Μέσω της ανάλυσης των ροών δεδομένων και των μετασχηματισμών, ο σχεδιαστής μπορεί να παράγει μια λογική λύση απαλλαγμένη των εκτιμήσεων της εφαρμογής. Αυτή η πρόωρη λογική λύση δεν θα έχει λεπτομέρειες, δεν θα είναι ακριβής και δεν μπορεί να εφαρμοστεί αμέσως. Μόλις η λογική λύση είναι σε θέση να ικανοποιήσει τις απαιτήσεις ή να επιτύχει τους στόχους, ο σχεδιαστής θα κάνει έπειτα τις απαραίτητες αλλαγές έτσι ώστε η λύση να μπορεί να εφαρμοστεί. Με άλλα λόγια, ο λογικός σχεδιασμός (περίληψη) πρέπει να μετατραπεί σε φυσικό σχεδιασμό (εφαρμογή).

Από τη λογική της αξιολόγησης του σχεδιασμού, ο SD προσφέρει ένα σύνολο καθοδηγητικών κριτηρίων για την αξιολόγηση του σχεδιασμού του λογισμικού. Αυτά τα κριτήρια είναι ανεξάρτητα από τη μεθοδολογία και μπορούν να εφαρμοστούν σε άλλες μεθοδολογίες σχεδιασμού. Δύο κατηγορίες κριτηρίων αξιολόγησης: οι συνδέσεις με άλλες ενότητες (σύζευξη) και η ομοιογένεια μεταξύ ενοτήτων (συνοχή). Το κριτήριο επιπέδου συστήματος της σύζευξης παρέχει έναν τρόπο αξιολόγησης των αλληλεξαρτήσεων μεταξύ των ενοτήτων. Δεδομένου ότι οι ενότητες είναι οι δομικές μονάδες ενός λογισμικού συστήματος, οι σχέσεις τους θα καθορίσουν το πόσο καλά το σύστημα μπορεί να διατηρηθεί ή να αλλάξει. Εάν οι ενότητες είναι ιδιαίτερα αλληλοεξαρτώμενες από μια άλλη, θα είναι δυσκολότερο να γίνουν οι αλλαγές σε μια ενότητα χωρίς να επηρεάσουν η μια την άλλη.

Αντιθέτως, εάν οι ενότητες είναι ιδιαίτερα ανεξάρτητες οι μεν από τις δε, θα είναι εύκολο να διατηρήσει το σύστημα και οι αλλαγές μπορούν να γίνουν σε μια ενότητα χωρίς την επιρροή στις άλλες. Η ενιαία συνοχή κριτηρίου ενότητας παρέχει έναν τρόπο λειτουργικών συνδέσεων μεταξύ των στοιχείων επεξεργασίας της. Η πιο επιθυμητή συνοχή είναι αυτή όπου μια ενότητα εκτελεί έναν ενιαίο στόχο με τα μεμονωμένα στοιχεία δεδομένων. Η λιγότερη επιθυμητή συνοχή, αφ' ενός, είναι αυτή όπου μια ενότητα εκτελεί διαφορετικούς στόχους με τις ανεξάρτητες βάσεις δεδομένων. Ένας καλός σχεδιασμός συστημάτων θα έχει την ισχυρή συνοχή και την αδύνατη σύζευξη. Το φάσμα δύο ταξινομήσεων μπορεί να παραχθεί, βασισμένο στα χαρακτηριστικά συζεύξεων και συνοχής.

Κατηγορίες σύζευξης

1. Δεδομένα (Data): όλες οι επικοινωνίες μεταξύ των ενοτήτων είναι μέσω των επιχειρημάτων των στοιχείων δεδομένων.
2. Σφράγιση (Stamp): η επικοινωνία περιλαμβάνει τα επιχειρήματα των βάσεων δεδομένων (μερικοί τομείς δεν απαιτούνται).
3. Έλεγχος (Control): ένα επιχείρημα από μια ενότητα μπορεί να ελέγξει τη ροή των άλλων, παραδείγματος χάριν, μια σημαία.
4. Εξωτερικός (External): παραπέμπουν σε ένα εξωτερικά δηλωμένο στοιχείο δεδομένων.

5. Κοινός (Common): παραπέμπουν σε μια εξωτερικά δηλωμένη δομή δεδομένων.
6. Περιεχόμενο (Content): ο ένας αναφέρει το περιεχόμενο στον άλλο.

Κατηγορίες συνοχής

1. Λειτουργικό: εκτελούν μια ενιαία και συγκεκριμένη λειτουργία.
2. Συγκεντρωμένο: είναι μια ομάδα λειτουργιών που μοιράζεται μια βάση δεδομένων για να κρύψει συνήθως την αντιπροσώπευσή του από το υπόλοιπο σύστημα. Μόνο μια λειτουργία εκτελείται σε κάθε κλήση, παραδείγματος χάριν, ο πίνακας συμβόλων με τις λειτουργίες insert και look-up.
3. Διαδοχικό: αποτελείται από διάφορες λειτουργίες που περνούν τα στοιχεία από εμπρός, παραδείγματος χάριν, ενημερώνουν και γράφουν ένα αρχείο.
4. Επικοινωνιακό: αποτελείται από διάφορες λογικές λειτουργίες που λειτουργούν στα ίδια δεδομένα, παραδείγματος χάριν, τυπώνει ένα αρχείο.
5. Διαδικαστικό: τα στοιχεία του ομαδοποιούνται σε έναν αλγόριθμο, παραδείγματος χάριν, το σώμα ενός κόμβου.
6. Χρονικό: οι λειτουργίες που αφορούν το χρόνο, παραδείγματος χάριν, την έναρξη.
7. Λογικό: μπορεί να εκτελέσει μια γενική λειτουργία όπου μια αξία παραμέτρου καθορίζει τη συγκεκριμένη λειτουργία, παραδείγματος χάριν, ένα συνηθισμένο λάθος, που καλείται με έναν κώδικα λάθους.
8. Συμπτωματικό (Coincidental): δεν υπάρχει καμία σχέση μεταξύ των στοιχείων ενότητας που ομαδοποιούνται, για λόγους συσκευασίας.

Η πρόθεση του SD είναι να μετρηθούν οι ενότητες του προγράμματος από άποψη της συνοχής και της σύζευξής του. Ο στόχος του είναι για κάθε ενότητα να εκτελέσει μια ενιαία και συγκεκριμένη λειτουργία και ότι όλο το επιχείρημά του, να είναι μεμονωμένα στοιχεία δεδομένων. Χρησιμοποιώντας τα κριτήρια του SD, η ποιότητα του σχεδιασμού μπορεί να «θυσιαστεί» ως αποτέλεσμα των αναλύσεων της ανταλλαγής του σχεδιασμού.

Η βασική προσέγγιση του SD είναι να αρχίσει με προδιαγραφές των συστημάτων που προσδιορίζουν τις εισαγωγές, τα επιθυμητά αποτελέσματα και μια περιγραφή των λειτουργικών πτυχών των συστημάτων που μετασχηματίζει τα στοιχεία. Αυτές οι προδιαγραφές χρησιμοποιούνται για τη γραφική απεικόνιση του συστήματος, το διάγραμμα ροής στοιχείων. Έπειτα προσδιορίζονται οι γενικές αλληλεξαρτήσεις των μετασχηματισμών και των ροών στοιχείων. Αυτό οδηγεί στον καθορισμό και την απεικόνιση των ενοτήτων και της σχέσης τους σε ένα και περισσότερα άλλα στοιχεία συστημάτων με τη μορφή διαγράμματος δομών.

Η συχνή ανάλυση δομής (Often Structure Analysis) συνδυάζεται με το SD για να διαμορφώσει μια ενσωματωμένη και δομημένη τεχνική σχεδιασμού. Η τεχνική αρχίζει με τη δομημένη ανάλυση όπου ένα πρότυπο του συστήματος χτίζεται χρησιμοποιώντας τα διαγράμματα ροής στοιχείων (DFD). Το DFD προέρχεται από τις υπάρχουσες πραγματικές και επαναλαμβανόμενες διαδικασίες στο σύστημα που βελτιώνονται και αποσυντίθενται σε μια τελική λογική λύση. Το DFD πρέπει να ελεγχθεί για τη συντήρηση των στοιχείων μέσω κάθε επανάληψης. Ένα λεξικό στοιχείων πρέπει έπειτα να χτιστεί σε μια λογική μορφή που να διευκρινίζει το

περιεχόμενο των στοιχείων, των ροών στοιχείων και όλων των μορφών στοιχείων στο DFD. Το λεξικό στοιχείων πρέπει να συσχετιστεί καλά με το DFD συμπεριλαμβανομένου του χειρισμού των ψευδωνύμων, του καθορισμού των δομών δεδομένων και της εφαρμογής του λεξικού στοιχείων.

Έπειτα, οι διαδικασίες στο DFD πρέπει να καθοριστούν με την ανάλυση της λογικής διαδικασίας και έπειτα την απεικόνισή τους στον ψευδοκώδικα. Τέλος, ένας λογικός, ιεραρχικός σχεδιασμός πρέπει να προέλθει από τα επίπεδα DFD. Αυτό το στάδιο αρχίζει με την αξιολόγηση του μηχανισμού του ελέγχου, την προσαρμοστικότητα των ενοτήτων, της τμηματικής (modular) συνοχής και σύζευξης, της ανάλυσης της μετατροπής, της βελτίωσης του σχεδιασμού, της εξαίρεσης του λάθους και του χειρισμού, και της ανάλυσης της συναλλαγής. Ο σχεδιασμός ξεκινά με μια αξιολόγηση του υψηλότερου επιπέδου των DFD, όπου ο τύπος της ροής των πληροφοριών εξακριβώνονται και η ροή των ορίων τους καθορίζονται χωριστά από τα κέντρα μετατροπής ή συναλλαγής. Οι μετατροπές χαρτογραφούνται έπειτα στη δομή προγράμματος ως ενότητες.

Η εξωτερική ροή πληροφοριών σε ένα σύστημα πρέπει να μετασχηματιστεί στις εσωτερικές πληροφορίες για την επεξεργασία. Οι πληροφορίες εισάγουν ένα σύστημα λογισμικού κατά μήκος των πορειών, αποκαλούμενων εισερχόμενες ροές, που μετασχηματίζει τα εξωτερικά στοιχεία στα εσωτερικά στοιχεία. Η εισερχόμενη ροή περνά μέσω του πυρήνα του συστήματος του λογισμικού, αποκαλούμενου κέντρο μετατροπής, και κατά μήκος των πορειών που ρέουν από το σύστημα του λογισμικού, αποκαλούμενο εξερχόμενη ροή. Όταν η γενική ροή των στοιχείων είναι διαδοχική και ακολουθεί μια ενιαία ή «ευθεία» πορεία, καλείται ροή μετατροπής. Η ανάλυση μετατροπής είναι ένα σύνολο βημάτων σχεδιασμού που χαρτογραφούν το DFD με τα χαρακτηριστικά μετατροπής σε ένα διάγραμμα δομών σχεδιασμού. Η ανάλυση μετατροπής είναι μια στρατηγική.

Ακόμα κι αν το βασικό πρότυπο σύστημα είναι κυρίως μετατροπέας ροής, η ροή πληροφοριών χαρακτηρίζεται συχνά από ένα ενιαίο στοιχείο δεδομένων, αποκαλούμενο συναλλαγή, η οποία προκαλεί άλλη ροή στοιχείων κατά μήκος μιας από τις πολλές πορείες. Όταν ένα DFD εκθέτει ένα τέτοιο χαρακτηριστικό, καλείται μια ροή συναλλαγής. Η ροή συναλλαγής προέρχεται από μια εξωτερική ροή στοιχείων κατά μήκος μιας εισερχόμενης πορείας, αποκαλούμενη πορεία υποδοχής, η οποία αλλάζει τα στοιχεία σε μια συναλλαγή. Η συναλλαγή αξιολογείται και ξεκινά σε ένα από τα μονοπάτια των λειτουργιών που είναι βασισμένα στην τιμή της. Το κέντρο ροής των πληροφοριών όπου πολλά μονοπάτια λειτουργιών διακλαδίζονται καλείται κέντρο συναλλαγής. Είναι δυνατό για τα μεγάλα σύνθετα συστήματα να υπάρξουν και οι ροές μετατροπής και συναλλαγής. Η διαδικασία για την ανάλυση συναλλαγής είναι βασικά η ίδια όπως για την ανάλυση μετατροπής εκτός από τη χαρτογράφηση του DFD στη δομή προγράμματος.

Ο SD είναι δημοφιλής επειδή χρησιμοποιεί τη σημείωση (ροή στοιχείων και μετασχηματισμός) που ένας σχεδιαστής μπορεί να προσδιορίσει και είναι εύχρηστη. Επίσης, ο SD παρέχει το σχεδιαστή ως μέσο αξιολόγησης του σχεδιασμού του λογισμικού. Εντούτοις, η μετατροπή του σχεδιασμού και των προδιαγραφών ενότητας στον προγραμματισμό της γλώσσας, δεν εξετάζεται. Επιπλέον, ακόμα κι αν οι οδηγίες για την αποσύνθεση παρέχονται, οι διαδικασίες μέσα στις ενότητες δεν εξετάζονται ρητά. Επίσης, η παραγωγή των ροών στοιχείων και οι μετασχηματισμοί

ποικίλλουν μεταξύ των σχεδιαστών, κατά συνέπεια, για ένα ενιαίο σύστημα οι διαφορετικοί σχεδιαστές θα έχουν διαφορετικές ροές στοιχείων και μετασχηματισμούς. Οι ακριβείς οδηγίες για την παραγωγή των ροών στοιχείων και των μετασχηματισμών από τις προδιαγραφές των συστημάτων δεν είναι διαθέσιμες. Αν και ο SD έχει τους περιορισμούς του, όταν χρησιμοποιείται με άλλα εργαλεία όπως το λεξικό στοιχείων, τα διαγράμματα δομών, το δέντρο των αποφάσεων και τα δομημένα Αγγλικά, παρέχει την αρχική δομική μονάδα στην οποία ο σχεδιασμός μπορεί να βασιστεί. Αυτή η τεχνική, είναι ιδανικά ταιριασμένη για το λογισμικό που απαιτεί την εκτέλεση κατά τρόπο διαδοχικό. Χρησιμοποιείται ευρέως στα στοιχεία επεξεργασίας που οφείλονται κατά ένα μεγάλο μέρος στο γεγονός ότι η έμφαση είναι στη ροή πληροφοριών. Το DFD είναι εύχρηστο και εύκολο για τους τελικούς χρήστες να το καταλάβουν, οι οποίοι μπορούν έπειτα να παρέχουν την εισαγωγή στη διαδικασία σχεδιασμού. Αυτή η τεχνική είναι καθιερωμένη και τεκμηριωμένη.

2.5 Τεχνική δομημένης ανάλυσης και σχεδιασμού

Η Structured Analysis and Design Technique (SADT) είναι μια προσανατολισμένη προσέγγιση σχεδιασμού. Δημιουργήθηκε και προάγεται από την SofTech Corporation (1976). Η SADT προέρχεται αρχικά από τις μελέτες πάνω στην βοήθεια των υπολογιστών από τον S.Hori (1972). Η τελική μορφή της αναπτύχθηκε από τον D.T.Ross και τους συναδέλφους (Ross & Schoman, 1977) στην εταιρία SofTech.
Η SADT χρησιμοποιεί μια τεχνική γραφική «γλώσσα» και ένα σύνολο διαδικασιών και διοικητικών οδηγιών για να εφαρμόσει τη γλώσσα. Αυτή η γλώσσα καλείται γλώσσα της δομημένης ανάλυσης (Ross, 1977) (ή γλώσσα SA). Οι διαδικασίες για τη γλώσσα SA είναι παρόμοιες με τις οδηγίες που χρησιμοποιούνται για τα συστήματα σχεδιαγραμμάτων εφαρμοσμένης μηχανικής. Κάθε διάγραμμα SA επισύρεται την προσοχή σε μια ενιαία σελίδα και περιέχει τρεις έως έξι κόμβους με τη διασύνδεση των τόξων. Υπάρχουν δύο τύποι διαγραμμάτων SA - το διάγραμμα δραστηριότητας (αποκαλούμενο actigram) και το διάγραμμα DATA (αποκαλούμενο διάγραμμα δεδομένων).

Η SADT αναπτύχθηκε βασισμένη στις αρχές ότι:

- τα ακριβή πρότυπα που περιγράφουν ένα σύνθετο πρόβλημα θα παράσχουν τα καλύτερα μέσα σε μια αποτελεσματική λύση
- η ανάλυση πρέπει να είναι top-down, ιεραρχική και δομημένη ως ενότητες
- τα πρότυπα πρέπει να είναι σε θέση να παρουσιάσουν τα αντικείμενα (παραδείγματος χάριν στοιχεία, και ενότητες) και τις διαδικασίες του συστήματος καθώς επίσης και των σχέσεών τους
- το πρότυπο μπορεί να αντιπροσωπεύσει γραφικά τις διεπαφές μεταξύ των ενοτήτων και της ιεραρχικής δομής του
- οι λειτουργίες του συστήματος (το “what”) σαφώς διακριτές από τα μέσα (το “how”) του συστήματος
- αυτή η μέθοδος πρέπει να παρέχει μια συνεπή πειθαρχία μεταξύ των σχεδιαστών
- η αναθεώρηση και η τεκμηρίωση όλων των αποφάσεων και των ανατροφοδοτήσεων στη διαδικασία σχεδιασμού είναι σημαντικές

Η μεθοδολογία SADT παρέχει έναν ακριβή και συνοπτικό σχεδιασμό απεικόνισης και ένα σύνολο τεχνικών, για να καθοριστούν γραφικά οι σύνθετες απαιτήσεις συστημάτων. Υπάρχει η αποσύνθεση top-down με τη σαφή αποσύνθεση για την

εισαγωγή, την παραγωγή, τον έλεγχο και το μηχανισμό για κάθε κόμβο. Είναι ευεργετικό να διαχωριστούν τα στοιχεία και οι δραστηριότητες σε δύο διαγράμματα έτσι ώστε να μην είναι σκορπισμένα. Η τεχνική διαχείρισης της σημείωσης της ανάπτυξης, της αναθεώρησης και του συντονισμού ενός προτύπου της SADT είναι μάλλον αποδοτική. Εντούτοις, στα συστήματα όπου πολλά διαγράμματα περιλαμβάνονται (τακτοποιημένα σε μια ιεραρχική διαταγή), οι πρόσθετες πληροφορίες ελέγχου για τα διαγράμματα, μπορούν να το καταστήσουν δύσκολο να το καταλάβουν και να το ακολουθήσουν. Επίσης, δεδομένου ότι κάθε σχεδιαστής αναπτύσσει ανεξάρτητα τα διαγράμματά του, θα ήταν δύσκολο στην αναθεώρηση να ενσωματωθεί η μερίδα του σχεδιαστή και το υπόλοιπο του συστήματος. Κατά συνέπεια, λόγω της πολυπλοκότητάς του στο σχεδιασμό, δεν χρησιμοποιείται συχνά αλλά η χρησιμότητά του κοντά στο πραγματικού χρόνου σύστημα, είναι προφανής.

2.6 Δομημένη ανάπτυξη συστημάτων

Η *δομημένη ανάπτυξη συστημάτων* (Structured Systems Development-SSD), αποκαλούμενη επίσης ως *δεδομένα ανάπτυξης συστημάτων* (Data Structured Systems-DSSD), είναι βασισμένη στη στρατηγική σχεδιασμού που αναπτύσσεται αρχικά από τον Warnier (1976). Η SSD είναι βασισμένη στη oriented-δομή ως προς τα στοιχεία. Το κεντρικό θέμα της SSD είναι ότι η δομή των στοιχείων θα καθορίσει τη δομή του προγράμματος. Κατά συνέπεια, ο ακριβής προσδιορισμός των δομών δεδομένων θα οδηγήσει σε ένα καλά δομημένο πρόγραμμα. Οι σημειώσεις σχεδιασμού που χρησιμοποιούνται, είναι τα διαγράμματα Warnier-Orr σε SSD.

Από την ανάλυση των συστημάτων, η εστίαση είναι στην παραγωγή και τις διαδικασίες που μετασχηματίζουν τις βάσεις δεδομένων εισόδου στις βάσεις δεδομένων εξόδου. Ο Pressman (1992) τις καλεί, *λογικές δομές παραγωγής* (logical output structures -LOS) και *λογικές δομές διαδικασίας* (logical process structures-LPS). Η ανάλυση πρέπει επίσης να παρέχει τις πληροφορίες απαιτήσεων όπως το πλαίσιο εφαρμογής, που είναι το πώς τα στοιχεία ρέουν μεταξύ των παραγωγών και καταναλωτών, οι λειτουργίες εφαρμογής που είναι οι διαδικασίες στις οποίες τα στοιχεία υποβάλλονται, και τα αποτελέσματα εφαρμογής που είναι η επιθυμητή παραγωγή, αφού περάσει μέσω του συστήματος.

2.7 Αντικειμενοστραφής σχεδιασμός

Ο αντικειμενοστραφής σχεδιασμός (Object Oriented Design-OOD) παρέχει έναν μηχανισμό που καλύπτει τρεις σημαντικές έννοιες στο σχεδιασμό του λογισμικού: διαμόρφωση, αφαίρεση, και ενθυλάκωση. Ο OOD είναι βασικά μια προσέγγιση που διαμορφώνει το πρόβλημα από την μεριά των αντικειμένων και τις διενεργηθείσες διαδικασίες σύμφωνα με αυτά. Ο OOD αποσυνθέτει το σύστημα στις ενότητες. Τα αντικείμενα αντιπροσωπεύουν τις συγκεκριμένες οντότητες που είναι περιπτώσεις μιας ή περισσότερων κατηγοριών. Τα αντικείμενα τοποθετούνται στις ιδιότητες στοιχείων, που μπορούν να είναι βάσεις δεδομένων ή ακριβής ιδιότητες. Οι διαδικασίες περιέχουν τις μεθόδους, οι οποίες είναι ο κώδικας του προγράμματος, ο οποίος λειτουργεί αυτές τις ιδιότητες.

Μια κατηγορία είναι ένα σύνολο αντικειμένων που μοιράζονται ένα κοινό σύνολο δομής και συμπεριφοράς. Μια κατηγορία αναπαριστά έναν τύπο και ένα αντικείμενο είναι μια περίπτωση μιας κατηγορίας. Μια κατηγορία περιέχει τρία στοιχεία: όνομα

κατηγορίας, κατάλογος ιδιοτήτων, και κατάλογος διαδικασιών. Κατά συνέπεια, μια κατηγορία αντιπροσωπεύει ένα σύνολο αντικειμένων με τις παρόμοιες ιδιότητες, την κοινή συμπεριφορά και τις κοινές σχέσεις με άλλα αντικείμενα. Η παραγωγή των υποκατηγοριών από μια κατηγορία καλείται κληρονομιά (inheritance). Μια υποκατηγορία μπορεί να έχει μερικά στοιχεία από την κύρια κατηγορία, κατά συνέπεια πολλαπλάσια κληρονομιά. Η δυνατότητα οποιωνδήποτε αντικειμένων να αποκριθούν στο ίδιο μήνυμα και η δυνατότητα κάθε αντικειμένου να το εφαρμόζει αυτό κατάλληλα, καλείται πολυμορφισμός. Οι σχέσεις περιγράφουν τις εξαρτήσεις μεταξύ των κατηγοριών και των αντικειμένων.

Η αντικειμενοστραφής ανάλυση (Object Oriented Analysis – OOA), μια τεχνική ανάλυσης απαίτησης, ξεκινά στο επίπεδο κορυφής προσδιορίζοντας τα αντικείμενα και τις κατηγορίες, τις σχέσεις τους σε άλλες κατηγορίες, τις σημαντικές ιδιότητές τους, και οι σχέσεις κληρονομιάς τους αντλούν έπειτα μια ιεραρχία κατηγορίας από αυτά. Αφ' ενός, ο OOD εξάγει τα αντικείμενα, που είναι διαθέσιμα από κάθε κατηγορία και τη σχέση τους από το ένα στο άλλο, για να παράγει μια λεπτομερή απεικόνιση του σχεδιασμού. Οι βασικές δομικές μονάδες για να ολοκληρωθούν από τον OOD, είναι να καθιερωθεί ένας μηχανισμός για: απεικόνιση της δομής δεδομένων, διευκρίνιση της λειτουργίας και κατ' επέκταση όλη τη λειτουργία. Η αφαίρεση των στοιχείων δημιουργείται: με τον προσδιορισμό των κατηγοριών και των αντικειμένων, τις ενότητες που καθορίζονται και τη δομή για το λογισμικό που καθιερώνεται με τη σύνταξη των διαδικασιών στα στοιχεία, και τις διεπαφές (interfaces) που περιγράφονται με την ανάπτυξη ενός μηχανισμού για τη χρήση των αντικειμένων.

Κατά συνέπεια, ο προσδιορισμός των αντικειμένων είναι ένας αρχικός στόχος της OOA και ενεργεί ως αφετηρία για τον OOD. Μόλις προσδιοριστούν τα αντικείμενα, το σύνολο διαδικασιών που ενεργούν στα αντικείμενα εξετάζεται. Υπάρχουν βασικά τρεις τύποι διαδικασιών: εκείνοι που χειρίζονται τα στοιχεία, εκείνοι που εκτελούν τον υπολογισμό και εκείνοι που ελέγχουν ένα αντικείμενο. Ο καθορισμός του αντικειμένου και των διαδικασιών του, δεν είναι αρκετός να παράγει τη δομή προγράμματος. Οι διεπαφές (interfaces) που υπάρχουν μεταξύ της γενικής δομής και των αντικειμένων, πρέπει να προσδιοριστούν και να καθοριστούν. Όλοι αυτοί πρέπει έπειτα να ενσωματωθούν σε ένα πρόγραμμα όπως το κατασκεύασμα που πολύ μοιάζει με τη γλώσσα προγραμματισμού.

Ο OOD δημιουργεί ένα πρότυπο του πραγματικού κόσμου και το χαρτογραφεί στη δομή λογισμικού. Ακόμα κι αν ο OOD παρέχει το μηχανισμό για να χωρίσει τα στοιχεία και τις διαδικασίες του, οι αντιπροσωπεύσεις του είναι επιρρεπείς και έχουν τη γλωσσική εξάρτηση του προγραμματισμού. Υπάρχει μια απουσία οδηγιών για να διαμορφώσει τα αρχικά αντικείμενα ή τις κατηγορίες, κατά συνέπεια θα εξαρτηθεί από τις τεχνικές ανάλυσης και από άλλες μεθοδολογίες. Η μεθοδολογία του OOD είναι μια πρόσφατη ανάπτυξη, υπό αυτήν τη μορφή και είναι ακόμα δυναμική και εξελίξιμη. Σε μερικούς μήνες ή έτη η δύναμη της μεθοδολογίας έχει εξελιχθεί σε μια διαφορετική μεθοδολογία από τι έχουμε σήμερα.

3 ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΤΗΛΕΠΙΚΟΙΝΩΝΙΑΚΟΥ ΣΥΣΤΗΜΑΤΟΣ: ΤΥΠΟΠΟΙΗΜΕΝΕΣ ΓΛΩΣΣΕΣ

3.1 Απαιτήσεις του συστήματος τηλεπικοινωνιών

Τα συστήματα τηλεπικοινωνιών είναι μια πλούσια περιοχή για τις απαιτήσεις των προβλημάτων των προδιαγραφών. Υπάρχουν τουλάχιστον δύο τύποι σχέσεων που πρέπει να περιγραφούν σε αυτά τα συστήματα:
Χρήστες και συστήματα: η αλληλεπίδραση μεταξύ των χρηστών και των συσκευών
Συστήματα και συστήματα: η αλληλεπίδραση των συστημάτων τηλεπικοινωνιών με ένα άλλο

3.2 Χρήστες και συστήματα

Πριν από πολλά χρόνια ένα τηλέφωνο ήταν μια απλή συσκευή, για να λειτουργήσει. Ακόμη κι αν κάποιος χρησιμοποιεί μια απλή συσκευή μπορεί να χρησιμοποιεί λειτουργίες, όπως η διαβίβαση κλήσης, η αναμονή κλήσης, η αυτόματη επανάκληση, η αυτόματη ανάκληση, κλπ. Η εκρηκτική αύξηση όλων αυτών των λειτουργιών έχει δημιουργήσει ένα πρόβλημα για τους προμηθευτές τους, δηλαδή πώς να τις περιγράψει.

Μια τυποποιημένη μέθοδος για τις λειτουργίες τηλεφώνου είναι από τα παραδείγματα, ή τα σενάρια. Δυστυχώς, αυτό είναι όπως η περιγραφή ενός συστήματος με την απαρίθμηση των περιπτώσεων δοκιμής (test cases).

3.2.1 Παράδειγμα

Η αυτόματη επανάκληση (AC) είναι μια λειτουργία που παρέχεται από τις περισσότερες τοπικές τηλεφωνικές επιχειρήσεις, σήμερα. Εν συντομία, αυτή η λειτουργία επιτρέπει στο χρήστη (αυτόματα) να συνεχίσει την λειτουργία του τηλεφωνήματος ώσπου να γίνει διαθέσιμο. Όταν αυτό συμβεί, επιχειρείται μια κλήση. Από την άποψη του χρήστη:

1. Καλεί κάποιους, αλλά το τηλέφωνό τους είναι απασχολημένο
2. Κλείνει το τηλέφωνο, κατόπιν σηκώνει το τηλέφωνό του και σχηματίζει έναν κωδικό πρόσβασης για αυτόματη επανάκληση (AC)
3. Ακούει έναν τόνο επιβεβαίωσης και έπειτα κλείνει το τηλέφωνο
4. Τελικά το άλλο συμβαλλόμενο μέρος γίνεται διαθέσιμο
5. Το τηλέφωνό του χτυπά με ένα ειδικό σήμα
6. Σηκώνει το τηλέφωνό σας
7. Μια κλήση στο άλλο τηλέφωνο επιχειρείται αυτόματα

Αυτό το απλό σενάριο είναι μόνο ένα από τα πολλά που θα εμφανιστούν. Προκειμένου να περιγραφεί εντελώς η λειτουργία, κάποιος πρέπει να θεωρήσει διάφορες άλλες δυνατότητες, όπως:

- Τι συμβαίνει όταν περισσότερα από ένα συμβαλλόμενα μέρη προσπαθούν να χρησιμοποιήσουν αυτή τη λειτουργία για τον ίδιο προορισμό;

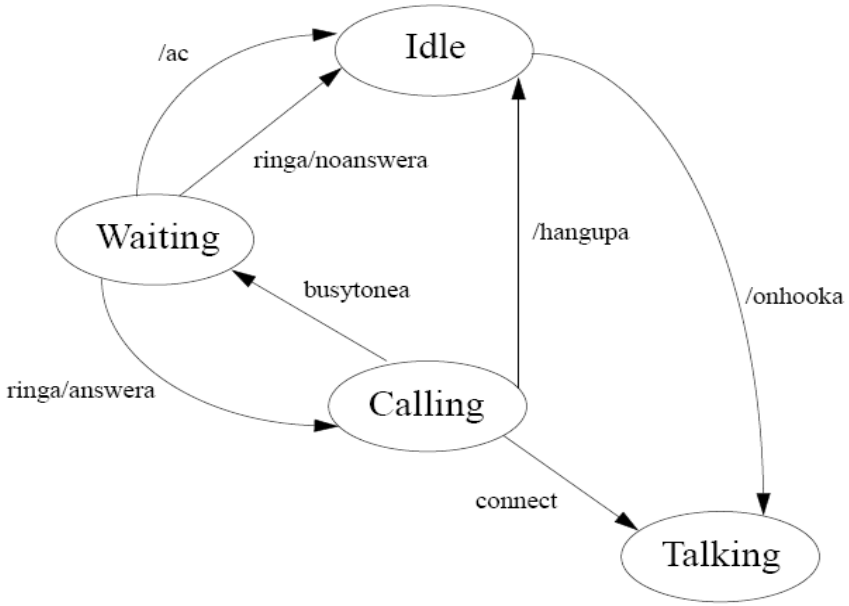
- Τι συμβαίνει εάν ο χρήστης της λειτουργίας δεν το σηκώνει όταν επισημαίνεται ότι το άλλο συμβαλλόμενο μέρος είναι ελεύθερο;
- Τι συμβαίνει εάν το άλλο συμβαλλόμενο μέρος γίνεται απασχολημένο ενώ ο δημιουργός-συμβαλλόμενο μέρος προειδοποιείται;
- Τι συμβαίνει εάν ο δημιουργός-συμβαλλόμενο μέρος θέλει να ακυρώσει την προσπάθεια αυτόματης επανάκλησης προτού αυτό τελειώσει-ολοκληρωθεί;

3.2.2 Τυποποίηση του παραδείγματος

Τα σενάρια έχουν το πλεονέκτημα ότι παρουσιάζουν πραγματικά παραδείγματα χρήσης. Αυτό που απαιτείται, εντούτοις, είναι πληρέστερες περιγραφές της συμπεριφοράς των λειτουργιών του συστήματος. Εάν τα σενάρια θα μπορούσαν να προέλθουν από αυτές τις περιγραφές, τότε οι χρήστες θα ήταν σε θέση να εξετάσουν την αντίληψή τους με το να θέσουν τις περιπτώσεις δοκιμής. Εκτός από την εξέταση τέτοιων περιγραφών, είναι σημαντικό να εκτελεσθεί η πληρέστερη ανάλυση για να εξασφαλίσει ότι οι ιδιότητες ασφάλειας και μακροζωίας (liveness) ικανοποιούνται.

Για το παράδειγμα αυτόματης επανάκλησης, αυτό που απαιτείται είναι ένα πλήρες πρότυπο της συμπεριφοράς κάθε μιας από τις διαδικασίες και τις αλληλεπιδράσεις τους. Τα σχήματα 1, 2 και 3 παρουσιάζουν μέρος ενός τέτοιου προτύπου χρησιμοποιώντας ένα παράδειγμα μηχανών πεπερασμένης κατάστασης. Σε κάθε σχήμα, οι καταστάσεις αντιπροσωπεύονται από ovals. Κάθε μετάβαση μπορεί να χαρακτηρισθεί με μια ώθηση εισαγωγής, και μια δράση απόδοσης (output), που χωρίζεται από μια κάθετο. Είτε η εισαγωγή είτε η απόδοση μπορεί να είναι απούσα, αλλά όχι και οι δύο.

Το σχήμα 1 παρουσιάζει το αυτόματο για το μέρος της κλήσης. Η διαδικασία αρχίζει στην κατάσταση Idle. Έπειτα ο πελάτης αποφασίζει να αρχίσει τη λειτουργία της αυτόματης επανάκλησης (ac), η διαδικασία μπαίνει στην κατάσταση Waiting. Εκεί μόλις, το μόνο ερέθισμα είναι το χτύπημα του τηλεφώνου, παρουσιαζόμενο από το γεγονός ringa.

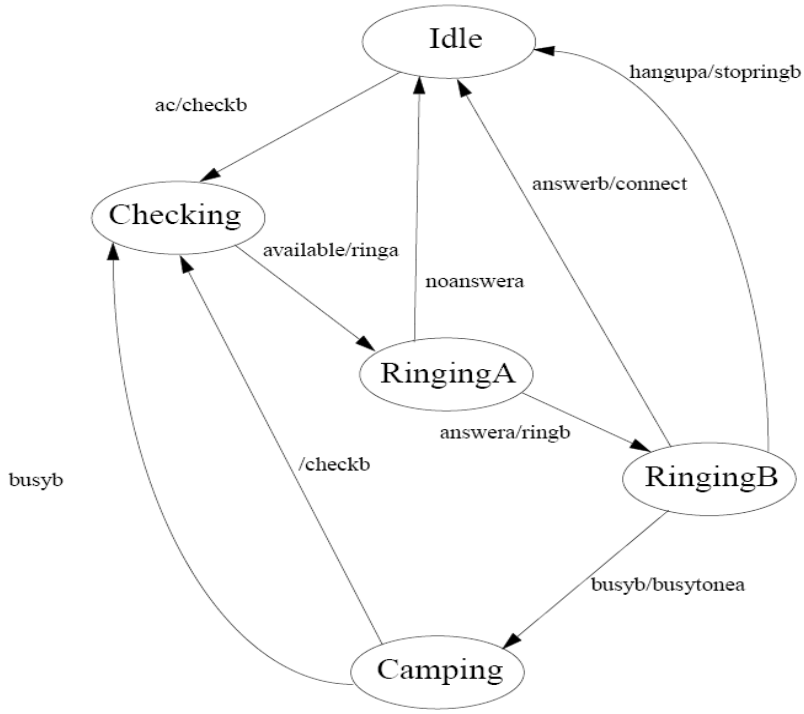


Σχήμα 1: Διάγραμμα μηχανής μιας πεπερασμένης κατάστασης κλήσης

Εάν ο πελάτης απαντήσει στο τηλέφωνο (answera) η διαδικασία κινείται στην κατάσταση Calling. Εάν ο πελάτης δεν απαντήσει (noanswera), η διαδικασία επιστρέφει στην κατάσταση Idle. Από την Calling κατάσταση ο πελάτης μπορεί να κλείσει (hangupa) προτού ολοκληρωθεί η κλήση, να λάβει ένα σήμα απασχόλησης (busytonea), ή να συνδεθεί με το άλλο συμβαλλόμενο μέρος (connect).

Εάν ο πελάτης κλείσει το τηλέφωνο η διαδικασία επιστρέφει στην κατάσταση Idle. Εάν το γεγονός busytonea ληφθεί η διαδικασία επιστρέφει στην κατάσταση Waiting για να δοκιμάσει πάλι. Εάν μια σύνδεση γίνει, η διαδικασία μπαίνει στη κατάσταση Talking. Η μόνη επιτρεπόμενη μετάβαση από την κατάσταση Talking είναι η Idle.

Το σχήμα 2 παρουσιάζει το αυτόματο για τον agent (πράκτορα) συστήματος μεταγωγής. Αρχικά το σύστημα είναι στην κατάσταση Idle. Όταν λαμβάνει ένα αίτημα αυτόματης επανάκλησης (ac) ελέγχει για να δει εάν το άλλο συμβαλλόμενο μέρος είναι διαθέσιμο (checkb). Εάν το συμβαλλόμενο μέρος είναι διαθέσιμο (available) ο agent "χτυπά" το καλούμενο συμβαλλόμενο μέρος (ringa) και πηγαίνει στη κατάσταση RingingA. Εάν το καλούμενο συμβαλλόμενο μέρος δεν είναι διαθέσιμο (busyb), ο πράκτορας πηγαίνει στην κατάσταση Camping. Από εκεί ελέγχει το καλούμενο συμβαλλόμενο μέρος (checkb) και καταγράφει εκ νέου την κατάσταση Checking.

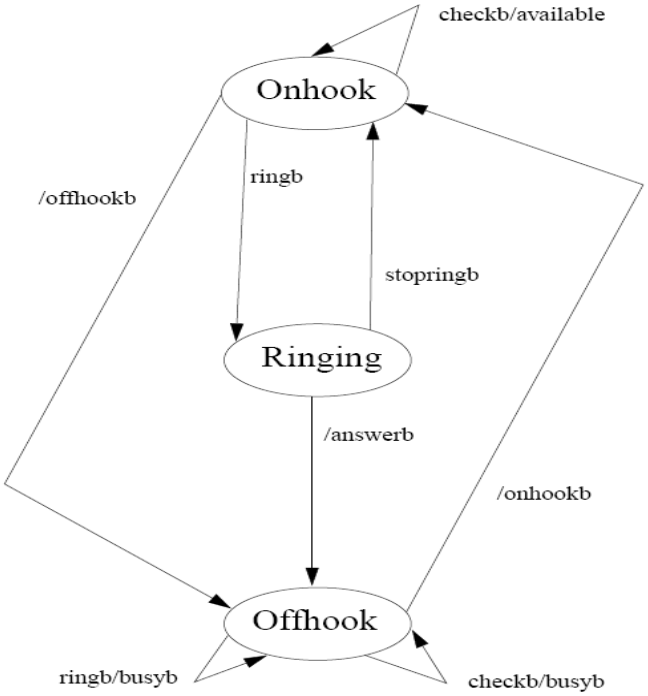


Σχήμα 2: Διάγραμμα μηχανής ενός πεπερασμένης κατάστασης πράκτορα συστήματος μεταγωγής

Μόλις η διαδικασία μπει στην κατάσταση RingingA, μπορεί είτε να ανιχνεύσει μια απάντηση (answera) είτε όχι (noanswera). Στην πρώτη περίπτωση "χτυπά" το καλούμενο συμβαλλόμενο μέρος (ringb) και μπαίνει στην κατάσταση RingingB. Στη δεύτερη περίπτωση επιστρέφει απλά στην κατάσταση Idle.

Από την κατάσταση RingingB υπάρχουν τρεις δυνατότητες: οι απαντήσεις των καλούμενων συμβαλλόμενων μερών (answerb), ή το καλούμενο συμβαλλόμενο μέρος επιστρέφει απασχολημένο (busyb), ή το καλούμενο συμβαλλόμενο μέρος κλείνει το τηλέφωνο πρόωρα (hangupa). Εάν το καλούμενο συμβαλλόμενο μέρος απαντήσει ο agent επισημαίνει μια σύνδεση (connect) και επιστρέφει στην κατάσταση Idle. Εάν το καλούμενο συμβαλλόμενο μέρος είναι απασχολημένο ο agent στέλνει ένα σήμα απασχόλησης (busytonea) στο καλούμενο συμβαλλόμενο μέρος και επιστρέφει στην κατάσταση Camping για να δοκιμάσει πάλι. Εάν το καλούμενο συμβαλλόμενο μέρος κλείσει το τηλέφωνο προτού να γίνει η σύνδεση ο agent σταματά να "χτυπά" το καλούμενο συμβαλλόμενο μέρος (stopringb) και επιστρέφει στην κατάσταση Idle.

Το σχήμα 3 παρουσιάζει το αυτόματο για το καλούμενο συμβαλλόμενο μέρος. Αυτό το αυτόματο έχει μόνο 3 καταστάσεις: Onhook, Offhook, και Ringing. Από την κατάσταση Onhook η διαδικασία μπορεί αυθόρμητα να εκπέμψει την παραγωγή offhook και να μπει στην κατάσταση Offhook. Ομοίως, μπορεί αυθόρμητα να κινηθεί από Offhook στην Onhook. Εάν η διαδικασία λάβει το σήμα checkb όταν είναι στην κατάσταση Onhook εκπέμπει το available και μένει στην Onhook. Επιπλέον, εάν λάβει το checkb ή το ringb από την κατάσταση Offhook παραμένει σε εκείνη την κατάσταση και επιστρέφει το busyb. Αλλά, εάν λάβει το ringb όταν είναι στην κατάσταση Onhook, κινείται προς την κατάσταση Ringing . Από εκεί μπορεί να λάβει το stopringb, δείχνοντας ότι το άλλο συμβαλλόμενο μέρος έκλεισε το τηλέφωνο νωρίς, οπότε σ' αυτή την περίπτωση επιστρέφει σε κατάσταση Onhook. Διαφορετικά, το τηλέφωνο απαντιέται (answerb), κινούμενο προς την κατάσταση Offhook.



Σχήμα 3: Διάγραμμα μηχανής μιας πεπερασμένης κατάστασης κλήσης

3.3 Συστήματα και συστήματα

Τα σύγχρονα συστήματα τηλεπικοινωνιών προσδιορίζουν την αξία τους από τη σύνδεσή τους σε ένα δίκτυο άλλων συστημάτων. Παραδείγματος χάριν, κάποιος δεν

θα ήταν ευχαριστημένος από ένα τηλέφωνο που θα μπορούσε μόνο να επικοινωνήσει με τα τηλέφωνα που συνδέθηκαν με τον ίδιο τοπικό διακόπτη γραφείου.
Η βιομηχανία τηλεπικοινωνιών έχει αναπτύξει μια μεγάλη συλλογή των προτύπων για την αλληλεπίδραση συστημάτων. Τα περισσότερα από αυτά τα πρότυπα περιγράφουν τα πρωτόκολλα.

3.3.1 Παράδειγμα

Η αυτόματη μετατροπή προστασίας (**APS**) [Bellcore 1991] είναι ένα πρωτόκολλο σχεδιασμένο για την αύξηση της αξιοπιστίας μιας σύνδεσης επικοινωνίας. Αυτό που πρέπει να επιτευχθεί είναι να παραχθούν περισσότερες από μια γραμμές για κάθε κανάλι επικοινωνίας. Εάν μια γραμμή αποτύχει, μια εφεδρική γραμμή καλούμενη ως γραμμή προστασίας (protection line) χρησιμοποιείται αντί αυτής. Σε αυτήν την στρατηγική, μια γραμμή προστασίας διανέμεται για κάθε γραμμή απασχόλησης (1+1).
Ένα σήμα γραμμής θεωρείται αποτυχημένο όταν έχει ένα ποσοστό λάθους bits μέσα σε μια επικίνδυνη σειρά, χαρακτηριστικά μεταξύ 10^{-5} και $10^{-9.}$ Η αναμενόμενη απάντηση σε ένα υποβιβασμένο ή αποτυχημένο σήμα στη γραμμή απασχόλησης είναι (αυτόματα) να μεταπηδήσει στη γραμμή προστασίας, εάν εκείνη η γραμμή είναι σε καλύτερη κατάσταση.
Μια τυποποιημένη μέθοδος πλεονασμού χρησιμοποιείται για να ελέγξει την ακρίβεια της μετάδοσης των μηνυμάτων. Μπορούμε να υποθέσουμε ότι ο αριθμός των λανθασμένων bits που παραλήφθηκαν στη γραμμή απασχόλησης καταγράφονται συνεχώς.

3.3.2 Τυποποίηση του παραδείγματος

Ένας τρόπος να παρουσιαστεί ένα πρωτόκολλο είναι να απαριθμηθούν όλες οι πιθανές καταστάσεις και οι μεταβάσεις του συστήματος σε έναν πίνακα. Το σχήμα 4 παρουσιάζει έναν τέτοιο πίνακα.

	RMW	RMP	RSW	RSP	FSW	FSP	CSW	CSP	DGW	DGP	FLW	FLP	CLW	CLP
WNN	PON	WNO	—	—	—	PNN	—	PNN	PDN	WND	—	—	—	—
WDN	PON	WDO	—	—	—	PDN	—	PDN	—	WDD	PFN	—	WNN	—
WFN	PON	WFO	—	—	—	PFN	—	PFN	—	WFD	—	—	WNN	—
WND	POD	WNO	—	—	—	PND	—	—	WDD	—	—	WNF	—	WNN
WDD	POD	WDO	—	—	—	PDD	—	PDD	—	—	PFD	WDF	WND	PDN
WFD	POD	WFO	—	—	—	PFD	—	PFD	—	—	—	WFF	WND	PFN
WNF	POF	WNO	—	—	—	PNF	—	—	WDF	—	—	—	—	WNN
WDF	POF	WDO	—	—	—	PDF	—	—	—	—	WFF	—	WNF	PDN
WFF	POF	WFO	—	—	—	PFF	—	PFF	—	—	—	—	WNF	PFN
WNO	WOO	—	—	WNN	—	—	—	—	WDO	—	—	—	—	—
WDO	WOO	—	—	PDN	—	—	—	—	—	—	WFO	—	WNO	—
WFO	WOO	—	—	PFN	—	—	—	—	—	—	—	—	WNO	—
WOO	—	—	WNO	PON	—	—	—	—	—	—	—	—	—	—
PNN	PON	WNO	—	—	WNN	—	WNN	—	PDN	WND	—	—	—	—
PDN	PON	WDO	—	—	WDN	—	—	—	—	PDD	PFN	—	PNN	—
PFN	PON	WFO	—	—	WFN	—	—	—	—	PFD	—	—	PNN	—
PON	—	POO	PNN	—	—	—	—	—	—	POD	—	—	—	—
PND	POD	WNO	—	—	WND	—	WND	—	PDD	—	—	WNF	—	PNN
PDD	POD	WDO	—	—	WDD	—	WDD	—	—	—	PFD	WDF	WND	PDN
PFD	POD	WFO	—	—	WFD	—	—	—	—	—	—	PFF	WND	PFN
POD	—	POO	WND	—	—	—	—	—	—	—	—	POF	—	PON
PNF	POF	WNO	—	—	WNF	—	WNF	—	PDF	—	—	—	—	PNN
PDF	POF	WDO	—	—	WDF	—	WDF	—	—	—	PFF	—	WNF	PDN
PFF	POF	WFO	—	—	WFF	—	WFF	—	—	—	—	—	WNF	PFN
POF	—	POO	WNF	—	—	—	—	—	—	—	—	—	—	PON
POO	—	—	WNO	PON	—	—	—	—	—	—	—	—	—	—

Σχήμα 4: Ένας πίνακας προδιαγραφών της αυτόματης μεταγωγής προστασίας

Οι καταστάσεις του συστήματος αντιπροσωπεύονται από έναν κώδικα 3-ψηφίων. Το πρώτο ψηφίο προσδιορίζει ποια γραμμή επιλέγεται (W για τη γραμμή απασχόλησης, P για τη γραμμή προστασίας). Το δεύτερο ψηφίο παρουσιάζει τη θέση της γραμμής απασχόλησης και το τρίτο ψηφίο παρουσιάζει τη θέση της γραμμής προστασίας. Η θέση μιας γραμμής μπορεί να είναι κανονική (N), υποβιβασμένη (D), αποτυχημένη (F), ή Out of Service (O).
Τα γεγονότα εισαγωγής παρουσιάζονται από κώδικες 3-ψηφίων, επίσης. Τα πρώτα δύο ψηφία παρουσιάζουν τον τύπο του γεγονότος και το τρίτο ψηφίο προσδιορίζει ποια γραμμή (η W ή η P) επηρεάζεται. Οι τύποι γεγονότος είναι: αφαίρεση (RM), αποκατάσταση (RS), υποχρεωτικός διακόπτης (FS), υποθετικός διακόπτης (CS), υποβιβασμός (DG), αποτυχία (FL), ή απαλλαγή (CL).

3.4 Προβλήματα

Κατά την περιγραφή των συστημάτων που αλληλεπιδρούν με τους χρήστες είναι λογικό να σχεδιαστούν τα διαγράμματα πεπερασμένης κατάστασης για τα μικρά συστήματα, αλλά αυτή η τεχνική γίνεται γρήγορα δύσχρηστη όπως τα συστήματα αυξάνονται σε μέγεθος και πολυπλοκότητα. Οι σύγχρονες λειτουργίες τηλεπικοινωνιών έχουν απενεργοποιημένες (Off) πάρα πολλές συμπεριφορές για να περιγράψουν αυτόν τον τρόπο. Αυτό που απαιτείται είναι γλώσσες και μέθοδοι που επιτρέπουν την αποσύνθεση του προβλήματος στα εύχρηστα κομμάτια.
Το ίδιο ισχύει για τις προδιαγραφές των συστημάτων που αλληλεπιδρούν με άλλα συστήματα. Αν και είναι δυνατό να διευκρινιστεί η απλούστερη έκδοση της APS με έναν πίνακα μετάβασης των καταστάσεων, οι πιο περίπλοκες εκδόσεις δεν μπορούν να αντιμετωπιστούν εύλογα με αυτόν τον τρόπο. Ακόμη και αυτό το απλό παράδειγμα θα ήταν δύσχρηστο εάν δεν υπήρχε συμμετρία στη συμπεριφορά των δύο γραμμών. Αυτό επιτρέπει μια συμπαγή αναπαράσταση για τις καταστάσεις και τα γεγονότα. Η αμφίδρομη έκδοση του πρωτοκόλλου απαιτεί αρκετές εκατοντάδες καταστάσεις. Τα πρότυπα APS δεν περιλαμβάνουν ακόμη τον πίνακα. Αντί αυτού, μερικά από τα ενδιαφέροντα σενάρια περιγράφονται, και ένα σύνολο κανόνων παρέχεται για να ερμηνεύσει το υπόλοιπο των περιπτώσεων. Δυστυχώς, οι κανόνες δεν είναι επαρκείς για να αποφασίσουν σαφώς σε όλες τις περιπτώσεις.
Οι υπάρχουσες μέθοδοι μπορούν να προσφέρουν την ελπίδα σε εκείνους που θέλουν απλά να καταλάβουν τι υποτίθεται ένα δεδομένο σύστημα θέλει να κάνει. Αλλά, υπάρχουν άλλοι λόγοι για την ακρίβεια σε αυτές τις προδιαγραφές απαιτήσεων. Η διαλειτουργικότητα είναι μια σημαντική πλευρά του προβλήματος. Δεδομένου ότι οι διαφορετικοί προμηθευτές μπορούν να εφοδιάσουν τον εξοπλισμό στις διαφορετικές άκρες ενός πρωτοκόλλου, πώς μπορεί κάποιος να είναι βέβαιος ότι και οι δύο έχουν ερμηνεύσει τα πρότυπα με έναν συμβατό τρόπο;

3.5 Χαρακτηριστικά των τηλεπικοινωνιακών συστημάτων

Τα βασικά χαρακτηριστικά των τηλεπικοινωνιακών συστημάτων είναι τα εξής:

- Ακολουθιακή λογική: οι λειτουργίες εκτελούνται η μια μετά την άλλη (υπάρχει μια λογική σειρά)
- Ανάγκη αναπαράστασης δυναμικής συμπεριφοράς: αναπαράσταση μεταβολής στο χρόνο
- Αυξημένη αλγοριθμική πολυπλοκότητα: το πόσο γρήγορος και αποτελεσματικός είναι ο αλγόριθμος

- Ανάγκη ιεραρχικής σχεδίασης: υπάρχει δηλαδή μια ιεραρχία, ξεκινώντας από τα γενικά (system level) και καταλήγοντας στην λεπτομέρεια.
- Αναπαράσταση περιορισμών της εφαρμογής: η εφαρμογή να έχει περιορισμούς (για παράδειγμα, η γραμμή να δίνει συγκεκριμένο bandwidth)
- Ανάγκη για εφαρμογή τεχνικών διαχείρισης ενέργειας → δυνατότητα αναπαράστασης στο σχέδιο: βελτιστοποίηση κατανάλωσης ενέργειας

4 ΜΈΘΟΔΟΙ ΜΟΝΤΕΛΟΠΟΙΗΣΗΣ ΣΥΣΤΗΜΑΤΩΝ

Οι επίσημες μέθοδοι προσφέρουν κάποια βοήθεια για τα προβλήματα που αναφέρθηκαν. Συγκεκριμένα, οι επίσημες μέθοδοι υποστηρίζουν ότι παρέχουν:

- *γλώσσες* για να περιγράψουν τη συμπεριφορά ακριβώς και σαφώς,
- *μεθόδους* και *εργαλεία* ανάλυσης για να αποκαλύψουν τα λάθη

Μερικές μέθοδοι παρέχουν επίσης ένα τρόπο για να εκτελεστεί, ή να μιμηθεί η συμπεριφορά των συστημάτων που περιγράφονται.

Ιστορικά, η βιομηχανία τηλεπικοινωνιών έχει υιοθετήσει τα πρότυπα και τις γλώσσες της, χωριστά από το υπόλοιπο της τεχνολογίας λογισμικού. Η *CCITT* (διεθνής συμβουλευτική Επιτροπή της τηλεγραφίας και της τηλεφωνίας), τώρα το *ITU-T* (διεθνής ένωση τηλεπικοινωνιών), καθιέρωσε τις γλώσσες της για την διευκρίνιση και την εφαρμογή των συστημάτων. Οι απαιτήσεις για πολλά διεθνή πρωτόκολλα διευκρινίζονται σε αυτές τις γλώσσες.

Αργότερα, το ISO (διεθνής οργανισμός για την τυποποίηση) ενίσχυσε την εργασία για να εφεύρει τις νέες γλώσσες ώστε διευκρινίσει τα πρωτόκολλα του OSI (διασύνδεση ανοικτών συστημάτων). Το ITU-T και οι ISO τυποποιημένες γλώσσες είναι γνωστές ως επίσημες τεχνικές περιγραφής. Υπάρχουν τέσσερις από αυτές τις γλώσσες που τυποποιήθηκαν : Estelle [ISO 1989a], LOTOS [ISO 1989b], και SDL [ITU 1993] και UML.

4.1 Estelle

Η Estelle αναπτύχθηκε στη δεκαετία του '80 από την ομάδα του ISO FDT. Είναι βασισμένη στην πεπερασμένη κατάσταση αυτόματα, και εξαρτάται πολύ από τη σύνταξη PASCAL. Εκτός από τη διευκρίνιση της συμπεριφοράς της επικοινωνίας των διαδικασιών, κάποια μπορεί να περιγράψει τις διασυνδέσεις μεταξύ των διαδικασιών, οι οποίες μπορούν να αλλάξουν κατά την διαδικασία τρεξίματος των διαδικασιών.

4.1.1 Γλωσσική επισκόπηση

Κάθε διαδικασία στην Estelle είναι ένας πεπερασμένος μετατροπέας [Aho και Ullman 1972]. Δηλαδή κάθε μετάβαση προκαλείται από κάποια εισαγωγή, και μπορεί επίσης να παράγει την απόδοση. Οι εισαγωγές αντιστοιχούν στα μηνύματα που λαμβάνονται με τη διαδικασία, και τα αποτελέσματα αντιστοιχούν στα μηνύματα που στέλνονται σε άλλες διαδικασίες. Στην πραγματικότητα, τα μηνύματα μπορούν να σταλούν από μια διαδικασία σε αυτά από μόνα τους, αν και αυτή δεν είναι μια κοινή τεχνική.

Η βασική μονάδα της περιγραφικής διαδικασίας είναι η ενότητα. Οι ενότητες επικοινωνούν η μια με την άλλη μέσω των καναλιών, τα οποία κανάλια συνδέουν την απόδοση μιας ενότητας με μια απεριόριστη σειρά αναμονής εισαγωγής μιας άλλης ενότητας. Αυτές οι συνδέσεις καλούνται σημεία αλληλεπίδρασης. Τα μηνύματα που περνούν μεταξύ των ενοτήτων καλούνται αλληλεπιδράσεις.

4.1.1.1 Κανάλια

Οι διαδικασίες της Estelle επικοινωνούν με τη διάβαση των μηνυμάτων μέσω των καναλιών. Κάθε τύπος καναλιών διευκρινίζεται με την απαρίθμηση των μηνυμάτων που μπορούν να περάσουν μέσω αυτού. Τα μηνύματα μπορούν να έχουν τις παραμέτρους, έτσι οι προδιαγραφές περιγράφουν μόνο τη δομή των μηνυμάτων. Ένα παράδειγμα ενός καναλιού δημιουργείται με τη διευκρίνιση των σημείων αλληλεπίδρασής της. Αυτό είτε γίνεται σε μια κεφαλίδα ενότητας είτε σ' ένα σώμα ενότητας.

Στο παράδειγμα στο σχήμα 5, ένας τύπος καναλιών Buy περιγράφει ότι επιτρέπει στα γεγονότα coin να περάσουν από έναν πελάτη σε έναν προμηθευτή, και ότι επιτρέπει στα γεγονότα trinket να περάσουν από τον προμηθευτή στον πελάτη. Ένα παράδειγμα στο κανάλι δημιουργείται με τη διευκρίνιση των σημείων αλληλεπίδρασης p1 και p2.

```
channel Buy(Customer, Vendor);

by Customer:
coin;
```

by Vendor:
trinket;
p1: Buy(Customer);
p2: Buy(Vendor);

Σχήμα 5: Παράδειγμα ενός τύπου καναλιών και περιπτώσεων σε Estelle

4.1.1.2 Ενότητες

Στην Estelle ένα σύστημα αποτελείται από μια ιεραρχία των ενοτήτων αλληλεπίδρασης. Κάθε ενότητα γονέων ελέγχει τη δημιουργία και την καταστροφή των ενοτήτων 'παιδιών' της. Μια ενότητα μπορεί να αλληλεπιδράσει με μια άλλη ενότητα στο ίδιο επίπεδο στην ιεραρχία, ή με τα παιδιά της. (Υπάρχει επίσης ένας μηχανισμός για τις αλληλεπιδράσεις κατευθείαν από ένα επίπεδο σε άλλο).

Οι προδιαγραφές μιας ενότητας αποτελούνται από δύο μέρη: η κεφαλίδα ενότητας και ένας κατάλογος πιθανών οργανισμών ενότητας. Στην κεφαλίδα κάποιος απαριθμεί τα εξωτερικά σημεία αλληλεπίδρασης και τις εξαγόμενες (κοινές) μεταβλητές. Η κεφαλίδα επίσης δηλώνει την ενότητα που είναι είτε μια διαδικασία (που εκτελούν ταυτόχρονα με άλλες διαδικασίες) είτε μια δραστηριότητα (που μπορούν μόνο να τρέξουν στη σειρά με τις άλλες δραστηριότητες σε επίπεδό της στην ιεραρχία ενότητας).

Ένα σώμα ενότητας αποτελείται από τρεις υποτομείς: δήλωση, έναρξη, και μετάβαση. Το μέρος δήλωσης περιέχει τις δηλώσεις του στυλ PASCAL των μεταβλητών και των διαδικασιών και τις δηλώσεις των αντικειμένων της Estelle (κανάλια, ενότητες, μεταβλητές ενότητας, καταστάσεις και εσωτερικά σημεία αλληλεπίδρασης). Το μέρος έναρξης δημιουργεί τις νέες περιπτώσεις submodules και εγκαθιστά τις συνδέσεις μεταξύ τους. Το μέρος μετάβασης διευκρινίζει τη συμπεριφορά της ενότητας σε απάντηση στις αλληλεπιδράσεις του.

Η δημιουργία μιας περίπτωσης ενότητας ολοκληρώνεται με τη πράξη init. Οι περιπτώσεις μπορούν να καταστραφούν με τη πράξη της release.

Οι πληροφορίες για τα σημεία αλληλεπίδρασης που περιλαμβάνονται σε μια κεφαλίδα ενότητας περιγράφουν τον τύπο αλληλεπιδράσεων που επιτρέπονται. Αλλά, οι συνδέσεις μεταξύ των σημείων αλληλεπίδρασης ολοκληρώνονται με την σύνδεση και την προσέγγιση των δηλώσεων. Η διαφορά μεταξύ αυτών των δύο δηλώσεων είναι ότι η σύνδεση καθιερώνει ένα νέο κανάλι επικοινωνίας, ενώ η επαφή συνεχίζει ένα κανάλι από έναν γονέα σε ένα παιδί. Ένα συνημμένο κανάλι περνά τις αλληλεπιδράσεις κατευθείαν από το γονέα στο παιδί (ή αντίστροφα), έτσι ώστε η άλλη ενότητα να μπορεί να τις επεξεργαστεί. Για να σπάσει τη σύνδεση κάποιος χρησιμοποιεί αποσύνδεση ή απόσπαση, αντίστοιχα.

4.1.1.3 Οι μεταβάσεις

Κάθε μετάβαση στην Estelle έχει έναν όρο και μια πράξη. Μπορεί να περιγραφεί από την άποψη ενός ή περισσότερων από τους ακόλουθους τύπους προτάσεων:

1. από (from) Ο τύπος προτάσεων **from** διευκρινίζει ένα σύνολο καταστάσεων στο οποίο η μετάβαση επιτρέπεται.

2. πότε (when). Η πρόταση **when** διευκρινίζει μια αλληλεπίδραση (μήνυμα εισαγωγής) που πρέπει να είναι παρούσα στο μέτωπο της σειράς αναμονής της εισαγωγής.

3. δεδομένου (provided). Η πρόταση **provided** διευκρινίζει μια έκφραση του Boolean που πρέπει να είναι αληθινή.

4. προτεραιότητα (priority). Η πρόταση **priority** διευκρινίζει μια μη αρνητική προτεραιότητα ακέραιων αριθμών σε συνεργασία με τη μετάβαση.

5. καθυστέρηση (delay). Η πρόταση **delay** διευκρινίζει τα χαμηλότερα/ανώτερα όρια των χρονικών μονάδων που πρέπει/μπορούν να παρέλθουν προτού να επιτραπεί η μετάβαση.

Οι προτάσεις μπορούν να παραλειφθούν, και το περισσότερο μια πρόταση κάθε τύπου επιτρέπεται. Η μετάβαση επιτρέπεται εάν όλες οι προτάσεις της είναι ικανοποιημένες, και έχει τη χαμηλότερη εκτιμημένη προτεραιότητα όλων των μεταβάσεων που επιτρέπονται αυτήν την περίοδο. Η μη-ντετερμινιστική συμπεριφορά επιτρέπεται, και οι εφαρμογές είναι ελεύθερες να επιλέξουν οποιαδήποτε επιτρεπόμενη μετάβαση σε εκείνη την περίπτωση.
Η πράξη μετάβασης έχει δύο μέρη: το όνομα της επόμενης κατάστασης, και ένα προαιρετικό block των δηλώσεων PASCAL για να εκτελούν αφού γίνει η μετάβαση. Το block των δηλώσεων επιτρέπει τη δημιουργία και την καταστροφή των περιπτώσεων ενότητας (διαδικασίες), τη σύνδεση και την αποσύνδεση των καναλιών επικοινωνίας, και την αποστολή των αλληλεπιδράσεων (μηνύματα).
Οι τοπικές μεταβλητές που χρησιμοποιούνται στην Estelle μειώνουν τον αριθμό των καταστάσεων που πρέπει να απαριθμηθεί ρητά στις προδιαγραφές, αλλά δεν μειώνουν το αποτελεσματικό κενό διάστημα του συστήματος. Παραδείγματος χάριν, μια μεταβλητή Boolean διπλασιάζει το μέγεθος του κενού διαστήματος, επειδή η μεταβλητή μπορεί να είναι είτε αληθινή είτε ψεύτικη σε κάθε ονομαζόμενη κατάσταση της διαδικασίας. Αυτές οι μεταβλητές μπορούν επίσης να χρησιμοποιηθούν για να περιγράψουν τους εσωτερικούς υπολογισμούς και για να απλοποιήσουν τις δηλώσεις ελέγχου που συνδέονται με τις μεταβάσεις.

4.1.2 Ανάλυση της Estelle

Η δήλωση των τύπων μηνυμάτων στην Estelle είναι χρήσιμη. Παρέχει τον πλεονασμό που βοηθά στον έλεγχο πιθανών λαθών που γίνονται στη χρήση εκείνων των τύπων μηνυμάτων. Συνοψίζει επίσης τις χρήσιμες πληροφορίες στο κατάλληλο επίπεδο. Τα ίδια οφέλη μπορούν να απαιτηθούν για τη δήλωση των καταστάσεων σε κάθε διαδικασία.
Το υπόλοιπο της σύνταξης για την Estelle δεν είναι τόσο χρήσιμο. Η σύνταξη του στυλ PASCAL μπορεί να είναι γνωστή στους προγραμματιστές, αλλά δεν είναι τόσο εξοικειωμένη στους συντάκτες απαιτήσεων. Η χρήση της εγκοπής και των λέξεων κλειδιών δίνει πάρα πολλή έμφαση στον τρόπο με τον οποίο η λογική μπορεί να εφαρμοστεί, παρά σε αυτό που πρέπει να είναι.
Τα περισσότερα από τα εργαλεία για την Estelle υποστηρίζουν αυτήν την προκατάληψη προς την εφαρμογή. Υπάρχουν εργαλεία για την σύνταξη της Estelle σε **C** και για την προσομοίωση των προδιαγραφών της Estelle. Αλλά, υπάρχουν λίγα εργαλεία για την ανάλυση των προδιαγραφών της Estelle.

4.2 LOTOS

Η γλώσσα του Temporal Ordering Specifications (LOTOS) αναπτύχθηκε για να διευκρινίσει τα πρωτόκολλα. Έχει ένα μέρος διαδικαστικής συμπεριφοράς, ονομαζόμενο Basic LOTOS, το οποίο ακολουθεί τη διαδικασία του παραδείγματος της άλγεβρας, παρόμοιο με το CCS [Milner 1980] και CSP [Hoare 1985]. Η πλήρης γλώσσα περιλαμβάνει επίσης ένα μέρος για την περιγραφή του στοιχείου που περνά μεταξύ των agents.

4.2.1 Γλωσσική επισκόπηση

Οι προδιαγραφές LOTOS αποτελούνται από ένα σύνολο ορισμών διαδικασίας. Υπάρχει συνήθως μια διαδικασία που καθορίζει ολόκληρο το σύστημα ενδιαφέροντος.

4.2.1.1 Δομή

Αντίθετα από την Estelle και την SDL, η LOTOS δεν έχει κανένα γλωσσικό στοιχείο για την περιγραφή των τμημάτων επεξεργασίας και των καναλιών ενός συστήματος. Αντιθέτως, αυτό υιοθετεί μια σύμβαση της ονομασίας των διαδικασιών και των τμημάτων των στοιχείων έτσι ώστε να μπορούν να διακριθούν όπως ανήκουν στις διαφορετικές κατηγορίες/κατατάξεις αντικειμένων.

4.2.1.2 Διαδικασίες

Κάθε διαδικασία περιγράφεται από μια ενιαία έκφραση συμπεριφοράς. Τα βασικά στοιχεία των εκφράσεων είναι:

η εναλλαγή ([])

οποιοιδήποτε από τους τελεστέους υπο-συμπεριφοράς (sub-behavior) μπορεί να επιλεχτεί ως η συμπεριφορά της έκφρασης. Αυτό είναι παρόμοιο με μια δήλωση περίπτωσης, εκτός από το ότι η επιλογή γίνεται μη ντετερμινιστικά

η παράλληλη σύνθεση ([])

αυτός ο χειριστής παίρνει τρία επιχειρήματα: δύο υπο-συμπεριφορές και μία λίστα γεγονότων. Τα γεγονότα μοιράζονται μεταξύ των υπο-συμπεριφορών. Δηλαδή ένα απαριθμημένο γεγονός μπορεί μόνο να εμφανιστεί εάν και οι δύο υπο-συμπεριφορές είναι έτοιμες να προσφέρουν εκείνο το γεγονός. Εάν μόνο η μια είναι έτοιμη, τότε η άλλη συμπεριφορά πρέπει να περιμένει ή να εκτελέσει ένα άλλο γεγονός (εάν μια επιλογή είναι διαθέσιμη).

διαδικασία παραδειγματισμού (Instantiation)

είναι όπως μια λειτουργία κλήσης -- η ονομαζόμενη διαδικασία περιγράφει τη συμπεριφορά της έκφρασης.

πρόθεμα

ένα γεγονός (ο πρώτος τελεστέος) πρέπει να εμφανιστεί έπειτα, ακολουθούμενο από τη συμπεριφορά του δεύτερου τελεστέου. Αυτό είναι παρόμοιο με μια ακολουθία δηλώσεων, όπου η πρώτη δήλωση είναι το περιστατικό του γεγονότος, και η δεύτερη δήλωση είναι ο τελεστέος της υπο-συμπεριφοράς.

Εντούτοις, έχει επιλεγεί να χρησιμοποιηθεί το εκτεταμένο αυτόματο ύφος Basic LOTOS, δεδομένου ότι ταιριάζει περισσότερο με τα στυλ που χρησιμοποιούνται από τις άλλες γλώσσες.

4.2.2 Περιγραφή

Η διαδικασία του παραδείγματος της άλγεβρας του Basic LOTOS είναι το πιο αφηρημένο όλων των παραστατικών γραφών που αναθεωρούνται εδώ. Αυτό είναι και ένα όφελος και ένα μειονέκτημα. Το όφελος είναι στην έμφαση του what παρά στο how στις προδιαγραφές απαιτήσεων. Το μειονέκτημα είναι στην παρουσίαση ενός ασυνήθιστου στυλ της γραφικής παράστασης που πρέπει να διδαχθεί. (Π.χ., οι επαναλαμβανόμενοι ορισμοί διαδικασίας είναι ιδιαίτερα δύσκολο να διδαχτούν στους αφελείς χρήστες.)

Όπως παρατηρήθηκε στις προηγούμενες μελέτες [Ardis 1994], μερικές από τις συντάξεις της LOTOS είναι 'ελαττωματικές'. Οι δηλώσεις των τύπων του γεγονότος λείπουν, στερώντας κατά συνέπεια στον αναλυτή τις χρήσιμες περιττές πληροφορίες που θα μπορούσαν να χρησιμοποιηθούν για να βρουν τα απρόσεκτα λάθη. Άλλες δηλώσεις, όπως τα επιχειρήματα να υποβάλουν σε επεξεργασία τα στιγμιότυπα (instantiations), είναι λίγο σημαντικές στις προδιαγραφές, αλλά είναι μια πηγή περιττών λαθών. Αυτά τα ελαττώματα μπορούν να θεραπευθούν εύκολα, έτσι ώστε να μην είναι σημαντικό εμπόδιο στην υιοθέτηση.

Υπάρχουν καλά εργαλεία για τον έλεγχο της συνέπειας των προδιαγραφών της LOTOS με τις φόρμουλες χρονικής λογικής (temporal logic formula) (π.χ., ο συναγωνισμός Workbench). Εναλλακτικά, είναι δυνατό να κωδικοποιηθούν μερικοί τέτοιοι τύποι όπως οι ορισμοί διαδικασίας σε LOTOS. Αυτό λαμβάνει ένα μεγάλο μέρος της ίδιας αξίας αποφεύγοντας την ανάγκη να εισαχθεί ακόμα μια γλώσσα και σχετική τεχνολογία.

4.3 SDL

Η SDL (γλώσσα περιγραφής προδιαγραφών) αναπτύχθηκε από τη βιομηχανία συστημάτων μεταγωγής και τυποποιήθηκε αρχικά από την CCITT το 1976. Όπως η Estelle, έτσι και αυτή είναι βασισμένη στις πεπερασμένες καταστάσεις αυτόματα, αλλά χρησιμοποιεί μια γραφική απεικόνιση των διαγραμμάτων ροής για να παρουσιάσει μεταβάσεις. Οι διασυνδέσεις μεταξύ των διαδικασιών περιγράφονται επίσης γραφικά. Υπάρχουν εναλλακτικές επιτρεπόμενες γραφικές παραστάσεις που δεν χρησιμοποιούν εικόνες (και που είναι, επομένως, πιο επιδεκτικές στην επεξεργασία από τα εργαλεία ανάλυσης), αλλά η γραφική μορφή είναι πιο διαδεδομένη.

4.3.1 Γλωσσική επισκόπηση

Κάθε διαδικασία στην SDL είναι ένας πεπερασμένος μετατροπέας, επιτρέποντας την εξαγωγή των αποτελεσμάτων σε κάθε μετάβαση. Οι εισαγωγές αντιστοιχούν στα μηνύματα που λαμβάνονται με τη διαδικασία, και τα αποτελέσματα αντιστοιχούν στα μηνύματα που στέλνονται σε άλλες διαδικασίες.
Η τοπολογία επικοινωνίας των διαδικασιών περιγράφεται από μια δομή blocks που παρουσιάζει κανάλια που συνδέουν τα blocks. Κάθε block μπορεί να είναι μια διαδικασία ή μπορεί να περιέχει μια υποδομή. Η επικοινωνία πέρα από τα κανάλια είναι ασύγχρονη, με μια απεριόριστη ουρά αναμονής στο λαμβάνον τέλος κάθε καναλιού.

4.3.1.1 Blocks και κανάλια

Ένα σύστημα περιγράφεται στην SDL από ένα διάγραμμα Blocks. Σε αυτό το επίπεδο κάποιος προσδιορίζει τα κανάλια και τα υπό-blocks, καθώς επίσης διευκρινίζονται οι αλληλεπιδράσεις με το περιβάλλον. Τα Subblocks μπορούν να αναλυθούν στα υπό-blocks και τα κανάλια από τα περαιτέρω διαγράμματα Blocks. Τελικά κάποιος φθάνει σε ένα επίπεδο όπου ένα Block περιγράφεται ως ένα σύνολο διαδικασιών επικοινωνίας.
Το γλωσσικό στοιχείο σημάτων χρησιμοποιείται για να προσδιορίσει τα μηνύματα που διαβιβάζονται σε ένα κανάλι. Το κανάλι καθορίζεται με το μαρκάρισμα ενός τόξου (arc) μεταξύ δύο Blocks. Στο τόξο μπορούν να αναπαρασταθούν τα σήματα που στέλνονται και λαμβάνονται. Είναι επίσης δυνατό να περιγραφεί ότι ένα κανάλι δεν έχει καμία καθυστέρηση, που δείχνει ότι τα μηνύματα λαμβάνονται αμέσως αφότου στέλνονται.

4.3.1.2 Διαδικασίες

Μια διαδικασία καθορίζεται από μια γραφική παράσταση διαδικασίας που αποτελείται συνήθως από διάφορες σελίδες μεταβάσεων των καταστάσεων. Κάθε σελίδα απεικονίζει ένα δέντρο, με αρχική κατάσταση στη ρίζα, και τις πιθανές μεταβάσεις από εκείνη την κατάσταση που παρουσιάζεται ως κλαδιά (branches), κάθε κλαδί τελειώνοντας με ένα σύμβολο παρουσιάζει νέα κατάσταση. Εάν υπάρχουν πάρα πολλές πιθανές μεταβάσεις για να εγκαταστήσουν στη σελίδα το δέντρο συνεχίζεται επάνω σε μια άλλη σελίδα με την ίδια αρχική κατάσταση στην κορυφή.
Υπάρχουν διάφορα ειδικά γραφικά σύμβολα στην SDL, αλλά το πιο κοινό είναι:

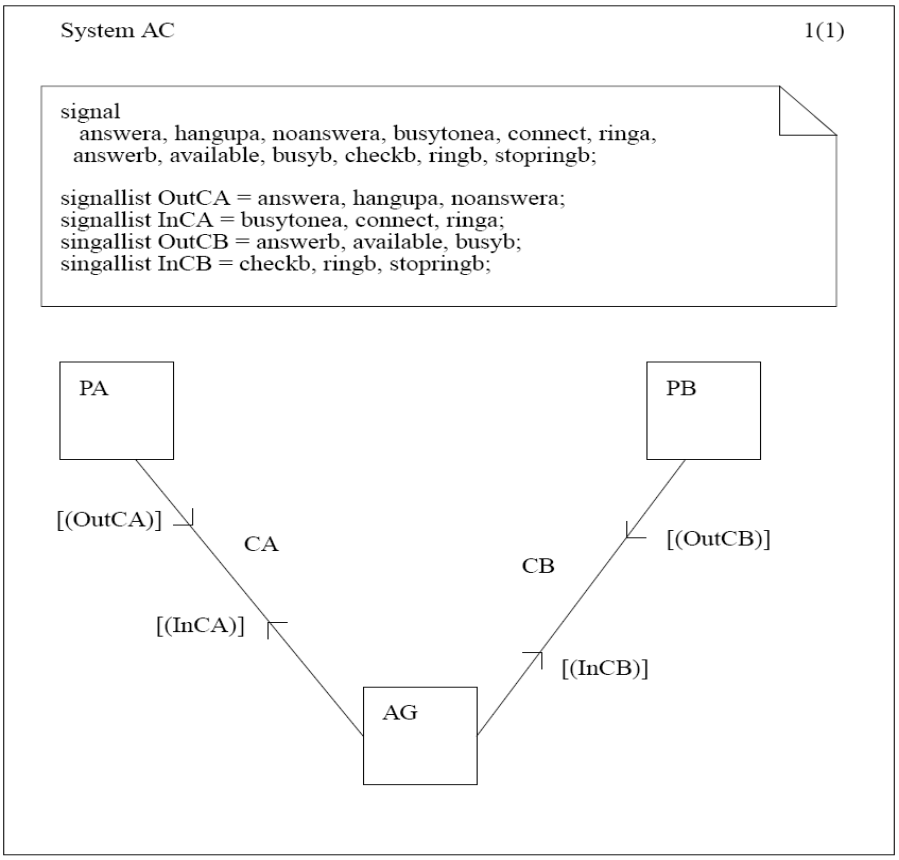
1. Το σύμβολο κατάσταση(state) χρησιμοποιείται για να καθορίσει μια κατάσταση, ή για να δείξει την επόμενη κατάσταση μιας μετάβασης.

2. Ένα σύμβολο εισαγωγής(input) χρησιμοποιείται για να δείξει την λήψη ενός μηνύματος ή το firing ενός χρονομέτρου. Δεν χρησιμοποιείται επίσης με την ειδική αξία (special value) στο να υποδείξει την μη αποφασιστικότητα της επιλογής σε καμία εισαγωγή.

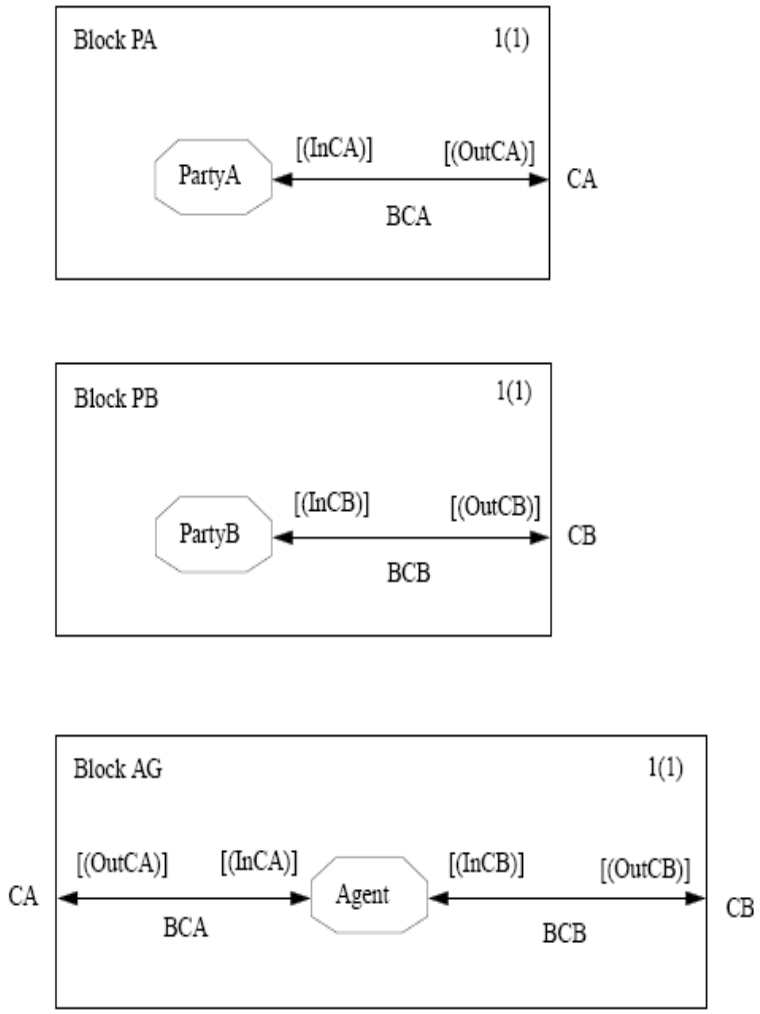
3. Το σύμβολο εξαγωγής(output) χρησιμοποιείται για να παρουσιάσει την δημιουργία (generation) ενός μηνύματος λήψης.

4. Ένα σύμβολο διεργασίας(task) χρησιμοποιείται για να περιγράψει τον υπολογισμό. Η περιγραφή μπορεί να είναι επίσημη ή ανεπίσημη. Παραδείγματος χάριν, κάποιος μπορεί να κινήσει ή να επανεκκινήσει ένα χρονόμετρο σε μια διεργασία.

Τα χρονόμετρα είναι τοπικά στις διαδικασίες, έτσι δεν μπορούν να χρησιμοποιηθούν για να επικοινωνήσουν μεταξύ των διαδικασιών. Είναι, φυσικά, δυνατό να σταλεί ένα μήνυμα σε μια άλλη διαδικασία που οδηγεί στη ρύθμιση ενός χρονομέτρου εκεί. Αλλά, η λήξη εκείνου του χρονομέτρου θα είναι τοπική σε εκείνη την διαδικασία. Υπάρχει μια πρωτόγονη μορφή διανομής των μεταβλητών, αλλά αυτό είναι ακριβώς μια στενογραφία για την αποστολή των μηνυμάτων που περνούν τις τιμές των τοπικών μεταβλητών από μια διαδικασία σε μια άλλη. Μόνο η διαδικασία που είναι κύρια μιας μεταβλητής μπορεί να ενημερώσει την αξία ή την τιμή της.
Το σχήμα 6 και 7 μας δείχνουν τη μορφή που έχουν τα Block διαγράμματα σε SDL.



Σχήμα 6: SDL block διάγραμμα επιπέδου συστήματος



Σχήμα 7: SDL block διαγράμματα

4.3.2 Ανάλυση της SDL

Το στυλ διαγράμματος ροής (flowchart) των προδιαγραφών είναι αρκετά φυσικό, αλλά και εύκολο να κάνει κακή χρήση. Γίνεται εξοικείωση των διαγραμμάτων ροής που μεταβιβάζουν λίγες πληροφορίες πέρα από την άχρηστη πολυπλοκότητα των προδιαγραφών. Ένα αμφισβητούμενο σημείο για την SDL είναι ότι επιτρέπει το άτυπο κείμενο μέσα στα κουτιά των διαγραμμάτων ροής. Αυτό θεωρείται ότι είναι ένα όφελος για τη βαθμιαία βελτίωση των προδιαγραφών. Δυστυχώς, είναι επίσης μια δικαιολογία για να αναβάλει τις αποφάσεις, νικώντας κατά συνέπεια έναν από τους κύριους σκοπούς των απαιτήσεων των προδιαγραφών.

Η χρονική λογική μπορεί να χρησιμοποιηθεί για να αυξήσει τις προδιαγραφές της SDL ώστε να ελέγξει την ακρίβειά τους. Αυτό έχει γίνει επιτυχώς στα εργαστήρια Bell [Chaves 1992], αν και η χρήση της δεν είναι διαδεδομένη στη βιομηχανία. Αυτοί οι τύποι μπορούν να ελεγχθούν με την τεχνολογία αποδοτικού προτύπου-ελέγχου.
Τα εργαλεία CASE που υποστηρίζουν την SDL είναι αρκετά ώριμα, αν και ακριβά. Το μεγαλύτερο μέρος της έμφασης αυτών των εργαλείων είναι στον χαμηλού επιπέδου σχεδιασμό και στην εφαρμογή. Αυτό είναι πιθανώς αρμόζον για να χτίσει μια καθιερωμένη βάση των πεπειραμένων χρηστών, δεδομένου ότι υπάρχουν

περισσότεροι εφαρμοστές από τους συγγραφείς απαιτήσεων. Τελικά, θα πρέπει να προστεθούν περισσότερα εργαλεία ανάλυσης.

4.4 UML

Η ενοποιημένη γλώσσα μοντελοποίησης (UML) είναι μια γραφική γλώσσα για την οπτική παράσταση, τη διαμόρφωση προδιαγραφών και την τεκμηρίωση συστημάτων που βασίζονται σε λογισμικό. Πρακτικά η UML είναι μια γλώσσα μοντελοποίησης δηλαδή ένα σύνολο από διαγράμματα. Δεν είναι γλώσσα προγραμματισμού. Η UML δημιουργήθηκε από την ανάγκη να λυθούν τα προβλήματα τα οποία εμφανίστηκαν κατά τον σχεδιασμό και τη μοντελοποίηση κάποιων συστημάτων. Η UML είναι μία γλώσσα η οποία αναπαριστά όλη την πληροφορία που περιέχει μία μεγάλη επιχείρηση ή ένα μεγάλο σύστημα. Σκοπός της είναι να διευκολύνει την δουλειά αυτών που ασχολούνται με την ανάπτυξη μεγάλων συστημάτων και επιχειρήσεων, και να βοηθά στην δημιουργία «καλών» συστημάτων, ώστε η δουλειά του προγραμματιστή να είναι εύκολη αργότερα όταν αυτός θα περάσει από το σχεδιαστικό στάδιο στο προγραμματιστικό.
Η UML στοχεύει στο σχεδιασμό αντικειμενοστραφών συστημάτων. Ο σχεδιασμός είναι μια απλοποιημένη παράσταση της πραγματικότητας. Με την χρήση ενός σχεδίου επιτυγχάνεται:

1. παριστάνεται οπτικά το σύστημα το οποίο επρόκειτο να κατασκευαστεί,
2. προσδιορίζεται η δομή και η συμπεριφορά του συστήματος,
3. δημιουργείται ένα πρότυπο στο οποίο θα βασιστεί ην κατασκευή του συστήματος

Η UML χρησιμοποιείται για την αναπαράσταση των επιμέρους στοιχείων ενός συστήματος. Η ιεραρχική δομή της UML επιτρέπει διαφορετικές φάσεις ανάπτυξης του μοντέλου, από τις απαιτήσεις προδιαγραφών ως και τον έλεγχο ενός τελικού συστήματος. Το γεγονός ότι χρησιμοποιούνται πολλά διαγράμματα είναι αυτό που την κάνει πιο προσιτή από τους ανθρώπους που σχεδιάζουν συστήματα λογισμικού και επιχειρήσεις. Σήμερα, η γλώσσα UML είναι ευρύτατα χρησιμοποιούμενη, εξαιτίας των μεγάλων αναγκών για την δημιουργία όλο και περισσότερων μοντέλων συστημάτων ή επιχειρήσεων.
Με τη UML ουσιαστικά προβλέπονται οι τυχόν δυσκολίες που θα εμφανιστούν στο μέλλον, κατά την υλοποίηση του συστήματος. Μπορεί να ακούγεται ασήμαντο όμως η σημασία της είναι τεράστια στο τομέα ανάπτυξης λογισμικού. Επίσης, η UML χρησιμοποιείται κυρίως για την μοντελοποίηση συστημάτων τα οποία στη συνέχεια θα περιγραφούν σε μία αντικειμενοστρεφή γλώσσα προγραμματισμού.

4.4.1 Φάσεις ανάπτυξης ενός συστήματος

Υπάρχουν πέντε φάσεις ανάπτυξης ενός συστήματος:

1. Ανάλυση απαιτήσεων: Η UML έχει use-cases τα οποία συλλαμβάνουν τις απαιτήσεις του πελάτη. Διαμέσου του σχεδιασμού με την χρήση του use-case οι εξωτερικοί αλληλεπιδρώντες με το σύστημα μοντελοποιούνται μαζί με την λειτουργικότητα που απαιτούν από το σύστημα.. Τα άτομα τα οποία αλληλεπιδρούν με το σύστημα και τα use-cases μοντελοποιούνται με σχέσεις, και επικοινωνούν μεταξύ τους συνδετικά ή είναι χωρισμένα σε ιεραρχίες. Τα άτομα αυτά, ονομάζονται actors και οι σχέσεις μεταξύ τους περιγράφονται με τα διαγράμματα use-cases της UML. Κάθε use-case περιγράφεται με κείμενο και

αναλύει τις απαιτήσεις του πελάτη, δηλαδή τι ακριβώς περιμένει από το σύστημα χωρίς να λάβει υπόψη του πως θα υλοποιηθεί η λειτουργικότητά του..

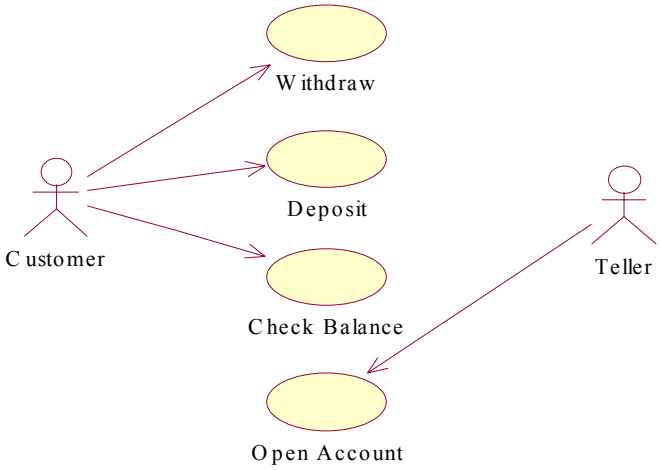
2. Ανάλυση του μοντέλου: Η φάση της ανάλυσης απαιτήσεων αφορά την κύρια αφηρημένη έννοια (κλάσεων και αντικειμένων) και μηχανισμούς οι οποίοι είναι παρόντες στην περιοχή του προβλήματος. Στην ανάλυση, μόνο οι κλάσεις οι οποίες ανήκουν στον τομέα του προβλήματος μοντελοποιούνται με τη χρήση μη τεχνικών κλάσεων οι οποίες καθορίζουν τις λεπτομέρειες και τις λύσεις στα συστήματα λογισμικού, όπως είναι οι κλάσεις για διεπαφές, βάσεις δεδομένων, τηλεπικοινωνίες κ.τ.λ.
3. Σχεδιασμός: Στη φάση της σχεδίασης τα αποτελέσματα της ανάλυσης μπορούν να δοθούν με τεχνικό τρόπο. Νέες κλάσεις προστίθενται για να βελτιώσουν την τεχνική υποδομή, την διεπαφή με το χρήστη, τη βάση χειρισμού για την αποθήκευση αντικειμένων σε αυτή, τις επικοινωνίες με άλλα συστήματα, την αλληλεπίδραση με τις συσκευές του συστήματος και άλλα. Η ανάλυση του προβλήματος των κλάσεων βασίζεται σε αυτή την τεχνική υποδομή, κάνοντας δυνατή την αλλαγή και την τροποποίηση του συστήματος.
4. Προγραμματισμός: Στη φάση της υλοποίησης οι κλάσεις από την σχεδιαστική φάση μετατρέπονται σε πηγαίο κώδικα τις περισσότερες φορές με κάποια αντικειμενοστρεφή γλώσσα προγραμματισμού. Κατά την ανάλυση απαιτήσεων ή τον σχεδιασμό με UML, είναι καλύτερα να αποφεύγεται η νοηματική ερμηνεία των μοντέλων σε κώδικα. Στις αρχικές φάσεις, η σημασία των μοντέλων είναι η κατανόηση και η δόμηση του συστήματος. Επιπλέον βγάζοντας εύκολα συμπεράσματα όσον αφορά τον κώδικα μπορεί να είναι αντιπαραγωγικό στη δημιουργία απλού και σωστού κώδικα. Ο προγραμματισμός είναι μία ξεχωριστή φάση κατά την διάρκεια του οποίου τα μοντέλα μετατρέπονται σε πηγαίο κώδικα.
5. Έλεγχος: Ένα σύστημα είναι λογικά δοκιμασμένο και διαχωρίσιμο σε μονάδες ελέγχου, σε μονάδες ελέγχου ενοποίησης και μονάδες ελέγχου αποδοχής. Οι μονάδες ελέγχου είναι ατομικές κλάσεις και είναι τυπικά εκτελούμενες από τον προγραμματιστή. Οι μονάδες ελέγχου ενοποίησης συνδέουν τις κλάσεις και τα συστατικά σε διάταξη η οποία επαληθεύει ότι επικοινωνούν όπως ήταν αναμενόμενο. Οι μονάδες ελέγχου αποδοχής του συστήματος βλέπουν το σύστημα ως ένα μαύρο κουτί και ελέγχουν αν το σύστημα έχει την αξιοπιστία που αναμένεται από έναν τελικό χρήστη. Ο έλεγχος αποδοχής γίνεται από τον πελάτη για να επιβεβαιώσει ότι το σύστημα ικανοποιεί τις απαιτήσεις οι οποίες είναι παραπλήσιες με τους ελέγχους του συστήματος. Διαφορετικές ομάδες ελέγχου, βασίζονται σε διαφορετικά διαγράμματα UML. Οι μονάδες ελέγχου χρησιμοποιούν διαγράμματα κλάσεων και λεπτομερείς κλάσεις, οι μονάδες ελέγχου ενοποίησης χρησιμοποιούν τυπικά διαγράμματα συνιστωσών και διαγράμματα συνεργασίας, και οι μονάδες ελέγχου αποδοχής του συστήματος εφαρμόζουν τα διαγράμματα use-case για να επικυρώσουν ότι το σύστημα συμπεριφέρεται όπως αρχικά είχε καθοριστεί.

4.4.2 Διαγράμματα της UML

Τα διαγράμματα δείχνουν όλα τα στοιχεία σχεδιασμού τοποθετημένα κατάλληλα για να επεξηγήσουν ένα συγκεκριμένο τμήμα της επιχείρησης. Ένα μοντέλο έχει τυπικά διάφορα διαγράμματα κάθε τύπου και είναι ένα κομμάτι μίας συγκεκριμένης όψης η οποία όταν απεικονίζεται, προσδιορίζεται από μία όψη. Μερικοί τύποι διαγραμμάτων μπορεί να είναι τμήματα διαφόρων όψεων, ανάλογα με τα περιεχόμενα του διαγράμματος.

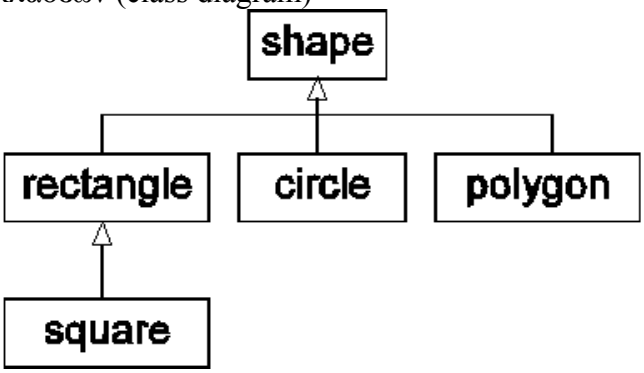
Η UML ορίζει τα παρακάτω διαγράμματα:

1. Διάγραμμα περιπτώσεων χρήσης (use case diagram)



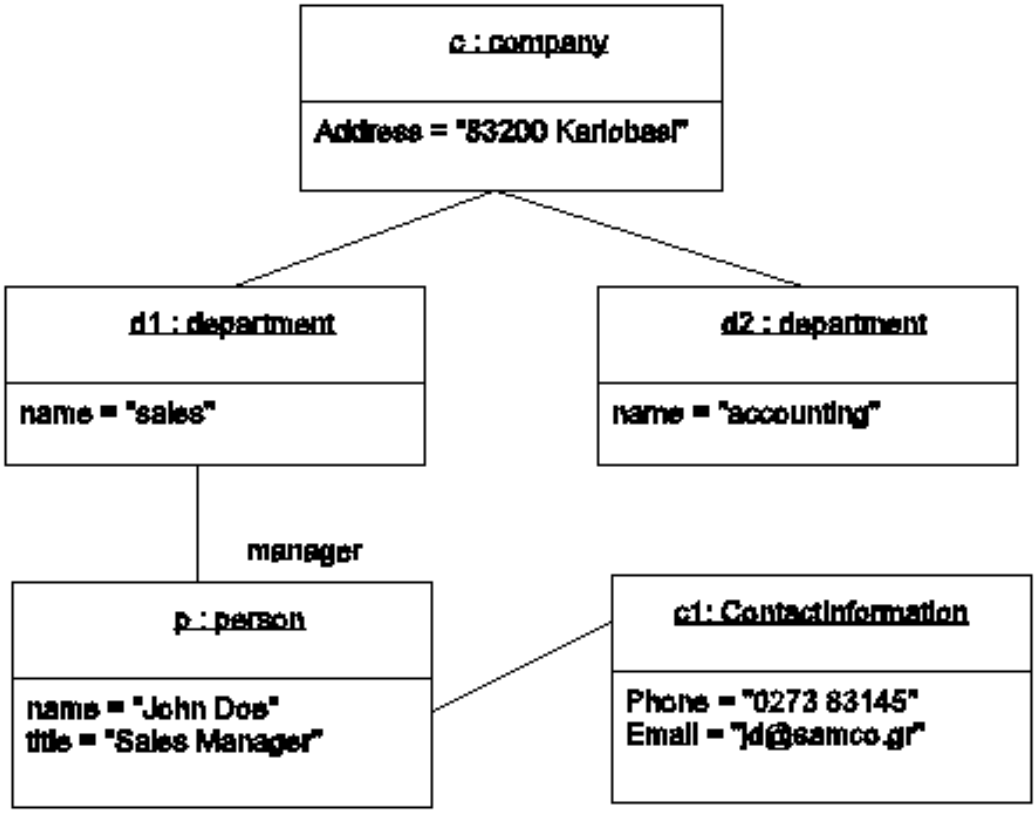
Σχήμα 8: Διάγραμμα Περίπτωσης Χρήσης

2. Διαγράμματα δομής
   - Διάγραμμα κλάσεων (class diagram)



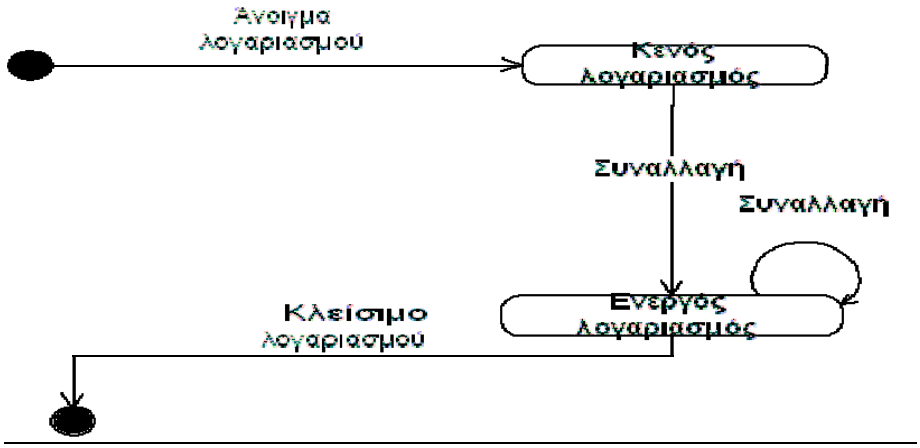
Σχήμα 9: Διάγραμμα κλάσεων

   - Διάγραμμα αντικειμένων (object diagram)



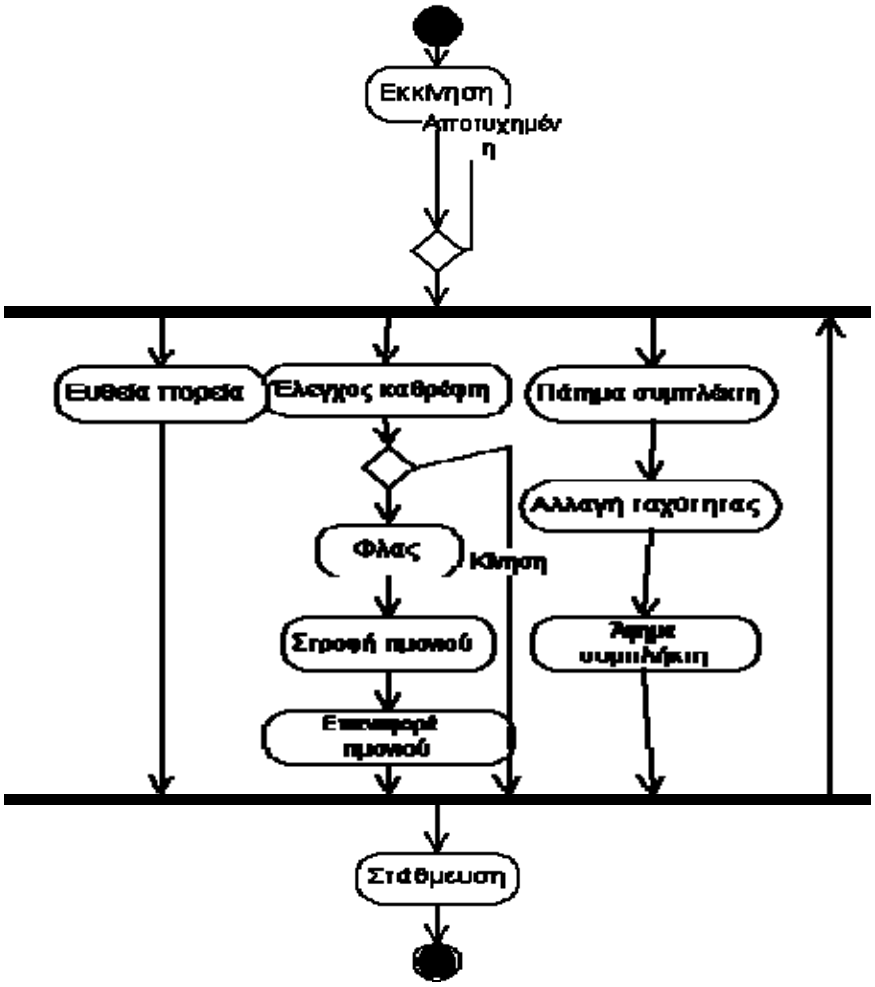
Σχήμα 10: Διάγραμμα αντικειμένων

3. Διαγράμματα συμπεριφοράς
   - Διάγραμμα καταστάσεων (statechart diagram)



Σχήμα 11: Διάγραμμα καταστάσεων

   - Διάγραμμα δραστηριοτήτων (activity diagram)



Σχήμα 12: Διάγραμμα δραστηριοτήτων

4. Διαγράμματα αλληλεπίδρασης
   - Διάγραμμα ακολουθίας (sequence diagram)
   - Διάγραμμα συνεργασίας (collaboration diagram)
5. Διαγράμματα δομής υλοποίησης
   - Διάγραμμα εξαρτημάτων (component diagram)
   - Διάγραμμα ανάπτυξης (deployment diagram)

4.5 Απαιτήσεις των τηλεπικοινωνιακών συστημάτων από το σχεδιαστικό μοντέλο

Εδώ είναι μερικά προβλήματα που αποτρέπουν την διαδεδομένη χρήση των επίσημων μεθόδων για να περιγράψουν τις απαιτήσεις για τα συστήματα τηλεπικοινωνιών:

Αφαίρεση (abstraction)

Αν και υπάρχουν πολλοί μηχανισμοί αφαίρεσης διαθέσιμοι, μερικοί σημαντικοί από αυτούς λείπουν ακόμα. Ειδικότερα, τα γεγονότα πρέπει να συλλεχθούν στις έννοιες αφαίρεσης των στοιχείων. Οι μόνοι μηχανισμοί αφαίρεσης στοιχείων διαθέσιμοι σήμερα είναι για τις τιμές των στοιχείων που ανταλλάσσονται στα πρωτόκολλα, όχι για τη συλλογή των αφηρημένων γεγονότων.

Ιδιομορφία περιοχών

Λαμβάνοντας υπόψη την πλούσια παράδοση τηλεπικοινωνιών, είναι εκπληκτικό ότι οι μέθοδοι δεν είναι πιο εξειδικευμένες για αυτήν την περιοχή.

Φιλικότητα χρηστών

Είναι απίθανο να δούμε τους πελάτες να εκφράζουν τις απαιτήσεις τους για τα συστήματα χρησιμοποιώντας τις επίσημες μεθόδους.

Ένα μεγάλο μέρος της εργασίας στις επίσημες μεθόδους για τα συστήματα τηλεπικοινωνιών έχει εστιάσει στην πολυπλοκότητα των αλγορίθμων και της αλληλεπίδρασης των διαδικασιών. Αυτό είναι παρόμοιο με την πρόοδο των γλωσσών προγραμματισμού, όπου η αρχική εστίαση ήταν στον δομημένο προγραμματισμό για να αναπαραστήσει τις σύνθετες δομές ελέγχου.

Τα σύγχρονα πρωτόκολλα έχουν μεγάλους αριθμούς γεγονότων. Είναι πάρα πολύ εύκολο να γίνει λάθος στον ορισμό του τύπου και να δημιουργηθεί τυχαία άλλο. Οι γλώσσες προγραμματισμού έλυσαν αυτό το πρόβλημα με την απαίτηση των δηλώσεων των μεταβλητών.

Τέλος, χρειαζόμαστε μια εναλλακτική λύση στη χρονική λογική. Εκείνοι που έχουν προσπαθήσει να ισχύσουν οι επίσημες μέθοδοι στα βιομηχανικά προβλήματα έχουν βρει δημιουργικούς τρόπους να κρύψουν τους σύνθετους υπολογισμούς. Οι πίνακες, παραδείγματος χάριν, είναι ένας κατάλληλος τρόπος να κρυφτεί η κλίση και η αποσύνδεση.

5 ΜΕΘΟΔΟΛΟΓΙΑ ΠΡΟΣ ΤΟΝ ΣΧΕΔΙΑΣΜΟ ΓΙΑ ΤΑ ΑΦΙΕΡΩΜΕΝΑ ΣΥΣΤΗΜΑΤΑ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΣΕ ΕΠΙΠΕΔΟ ΣΥΣΤΗΜΑΤΟΣ

5.1 Εξέλιξη των υπηρεσιών τηλεπικοινωνίας

Η ανάπτυξη των τηλεπικοινωνιών προσφέρει τις όλο και περισσότερο νέες υπηρεσίες για τους χρήστες. Η εφαρμογή πολυμέσων υποβάλλεται προς το παρόν σε σημαντικές εξελίξεις στην τηλεφωνία και την ψηφιακή τηλεοπτική ραδιο-αναμετάδοση. Τα φορητά συστήματα τηλεπικοινωνιών παρουσιάζουν μια πραγματική επανάσταση: Το European Cellular standard GSM ανοίγει στις εφαρμογές πολυμέσων από τον μεσάζοντα του κανόνα UMTS. Το χαμηλό ποσοστό λεκτικής κωδικοποίησης, στο G729 πρότυπο, γίνεται όλο και περισσότερο σύνθετο. Η ψηφιακή τηλεόραση (HDTV) περιλαμβάνει ψηφιακές διαμορφώσεις όπως ODFM και πολλές εφαρμογές πολυμέσων ανοικτές στο πρότυπο MPEG 4.

Ο σχεδιασμός των συστημάτων τηλεπικοινωνιών πρέπει να ικανοποιήσει μερικούς ιδιαίτερους περιορισμούς: (ι) απαιτήσεις υπολογισμού που επιβάλλουν τη χρήση των αφιερωμένων συνεπεξεργαστών (ASIC), (ii) ευελιξία που μπορεί να εξασφαλιστεί με τη χρησιμοποίηση των ειδικευμένων επεξεργαστών (με την πιθανή επέκταση εκπαίδευσης πολυμέσων), αφιερωμένος ψηφιακός επεξεργαστής επεξεργασίας (VelociTI processor) ή επεξεργαστές oriented βίντεο (MPACT).

Ο σχεδιασμός ενσωματωμένων συστημάτων και οι χαμηλοί περιορισμοί ισχύος οδηγούν στη χρήση των λύσεων MCM (Multi Chip Module) και SOC (System on Chip). Τέλος, οι απαιτήσεις ευελιξίας οδηγούν στη χρήση των βασισμένων συστημάτων PGA. Η απόδοση υπολογισμού τέτοιων συστημάτων εξελίσσεται αναλογικά με την πυκνότητα ολοκλήρωσης (π.χ. Xilinx FPGA CX4062 με 250 000 πύλες). Όλοι οι περιορισμοί που επιβάλλονται στη διαδικασία σχεδιασμού απαιτούν τη χρήση νέων τεχνικών συσχεδίασης και συνεπαλήθευσης (co-verification).

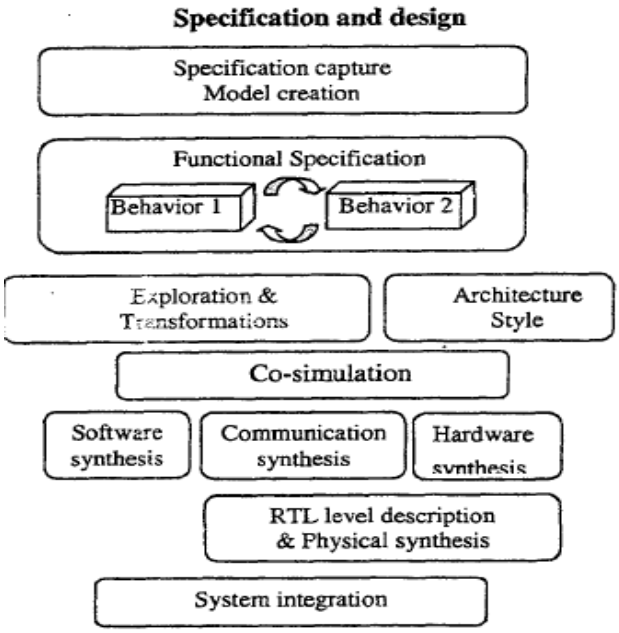
5.2 Μέθοδοι σχεδιασμού συστημάτων

5.2.1 Κύκλος σχεδιασμού συστημάτων

Ο κύκλος σχεδιασμού στοχεύει στην εξασφάλιση, πέρα της ικανοποίησης περιορισμού εφαρμογής, των πρόσφατων βελτιώσεων σχεδιασμού και της βελτιστοποίησης της απόδοσης (Design costs, Time-To-Market κ.λπ.). Ένας σημαντικός στόχος είναι να προταθεί μια προσέγγιση σχεδιασμού που να περιλαμβάνει την πλήρη ιεραρχία σχεδιασμού, από τις απαιτήσεις στη χαμηλού επιπέδου περιγραφή λεπτομερειών και στοιχείων κατασκευής προϊόντων.

Για να καταστήσουν τη διαδικασία σχεδιασμού αποδοτικότερη, οι σχεδιαστές πρέπει να χρησιμοποιήσουν τα κατάλληλα εργαλεία για κάτι το διαφορετικό, πέρα από τους εξής στόχους: οι προδιαγραφές και η διαμόρφωση συστημάτων, η προσομοίωση και η επικύρωση αλγορίθμων, και ο στόχος διαμόρφωσης υλικού /λογισμικού. Σε αυτό το πλαίσιο, μια επιπλέον διαδικασία σχεδιασμού παρουσιάζεται στο σχήμα 13. Κάθε στάδιο στη διαδικασία χρησιμεύει ως μια αφετηρία για το επόμενο στάδιο. Η διαδικασία σχεδιασμού αρχίζει με τον καθορισμό απαιτήσεων, ο οποίος περιλαμβάνει μια περιγραφή των αλγορίθμων που εφαρμόζονται στο σύστημα στόχων. Ένα βασικό ζήτημα στη ροή σχεδιασμού, συνίσταται στις αρχικές προδιαγραφές που βελτιώνονται σε κάθε στάδιο σχεδιασμού, ενώ επικυρώνεται, από επίσημες αποδείξεις ή προσομοιώσεις, αποφάσεις σχεδιασμού. Επιπλέον αυτή η ροή

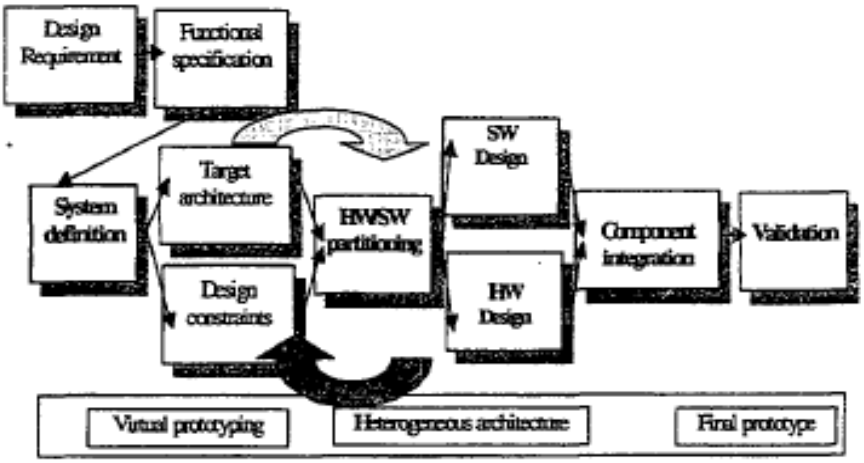
σχεδιασμού επιτρέπει την επαναχρησιμοποίηση των υπαρχουσών ενοτήτων όπως η ASIC [1] που διευκρινίστηκε ήδη σε VHDL.



Σχήμα 13: Κύκλος Top-Down σχεδιασμού

5.2.2 Ροή σχεδιασμού

Σήμερα, ουσιαστικά τα χαρακτηριστικά σχεδιασμού (δείτε σχήμα 14) στηρίζονται στη συσχεδίαση (co-synthesis) [1, 2], η οποία περιλαμβάνει τις προδιαγραφές συστημάτων, τον αρχιτεκτονικό σχεδιασμό, τον χωρισμό υλικού-λογισμικού και την συνεπαλήθευση για την ολοκλήρωση και τη δοκιμή.



Σχήμα 14: Ροή σχεδιασμού συστήματος

Κατάκτηση των προδιαγραφών: Για να διευκρινίσει τη λειτουργία συστημάτων, τα εννοιολογικά πρότυπα χρησιμοποιούνται για κάθε συμπεριφορά των λειτουργικών προδιαγραφών: πρότυπο ελέγχου, πρότυπο στοιχείων, και πρότυπο συγχρονισμού.

Εξερεύνηση σχεδιασμού: Η αρχική περιγραφή μετασχηματίζεται σε μια ή περισσότερες κατάλληλες εφαρμογές. Η εκτίμηση της απόδοσης είναι ένα βασικό ζήτημα για την καθοδήγηση στις εξερευνήσεις των εναλλακτικών προβλημάτων.

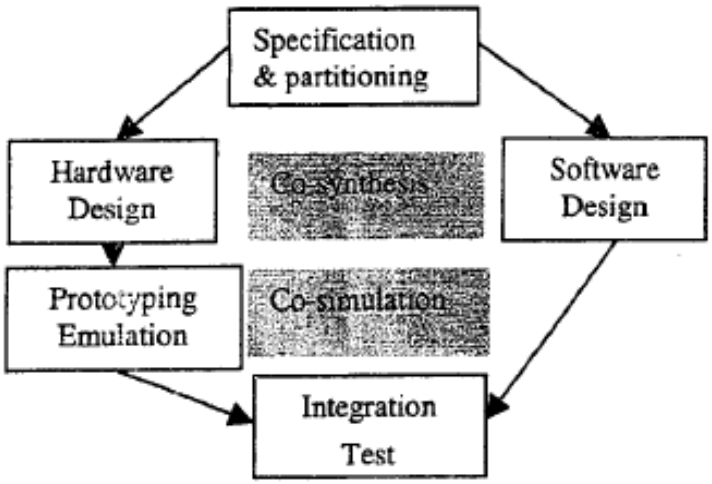
Συμπροσομοίωση: Μετά το διαχωρισμό, τα πρότυπα μετασχηματίζονται στη γλωσσική περιγραφή που αφιερώνεται για την εφαρμογή λογισμικού ή υλικού. Και οι δύο περιγραφές προσομοιώνονται ταυτόχρονα.

Σύνθεση λογισμικού και υλικού: Ένας τυποποιημένος επεξεργαστής απαιτεί τη σύνθεση λογισμικού, η οποία καθορίζει μια διαταγή εφαρμογής λογισμικού που ικανοποιεί τον περιορισμό και την εκτέλεση των πόρων. Η ASIC μπορεί να αποκτηθεί μέσω της υψηλού επιπέδου σύνθεσης. Ιδιαίτερη προσοχή πρέπει να κρατηθεί στη σύνθεση επικοινωνίας.

Φυσικός σχεδιασμός: Αυτό το βήμα παράγει στοιχεία κατασκευής για κάθε στοιχείο, ως στοιχεία σχεδιαγράμματος για την gate-arrays, FPGAs, ή τη συνήθεια των ASICs χρησιμοποιώντας φυσικά εργαλεία για την τοποθέτηση και δρομολόγηση συγχρονισμού.

5.2.3 Βιομηχανικά εργαλεία

Η συσχεδίαση (co-synthesis) μπορεί να χωριστεί σε τρεις σημαντικές κατηγορίες (σχήμα 15): συμπροδιαγραφές, co-synthesis και συμπροσομοίωση.



Σχήμα 15: Εργαλεία διαμόρφωσης πρωτοτύπου και σχεδιασμού

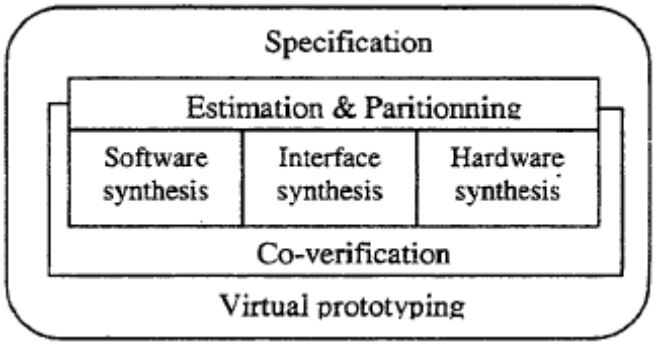
Μερικά υπάρχοντα εργαλεία επιτρέπουν τους ακόλουθους στόχους:

- Οι προδιαγραφές επιπέδων συστημάτων των εφαρμογών τηλεπικοινωνιών μπορούν να πραγματοποιηθούν από κάποιο από τα ακόλουθα εργαλεία όπως ο σταθμός DSP (από Mentor Graphics), SPW (Cadence)) ή COSSAP (Synopsis).
- Ο σχεδιασμός υλικού στηρίζεται σε υψηλού επιπέδου εργαλεία σύνθεσης που αρχίζει από τις προδιαγραφές συμπεριφοράς σε VHDL ή Verilog και παράγει μια περιγραφή RTL VHDL και για την αρχιτεκτονική και για τους ελέγχους του υλικού.

- Η συμπροσομοίωση του υλικού / λογισμικού είναι πολύ συνδεμένη με την "άμιλλα". Αποτελείται από τον έλεγχο της καλής εκτέλεσης του υλικού και του λογισμικού πριν από τη σύνθεση υλικού. Οι διαθέσιμες διαφορετικές επιλογές παρουσιάζουν μια ανταλλαγή μεταξύ του γρήγορου και δαπανηρού πρωτοτύπου του υλικού και του χαμηλού προσομοιωτή απόδοσης EDA.
- Ο σχεδιασμός και η παραγωγή λογισμικού στηρίζονται κυρίως στον μεταγλωττιστή και τον μεταγλωττιστή, τα λειτουργικά συστήματα πραγματικού χρόνου, και τους προσομοιωτές.

5.2.4 Ερευνητικές δραστηριότητες και πρόσφατη ανάπτυξη

Περισσότερα από 13 προγράμματα στις ΗΠΑ [1], 11 προγράμματα στην Ευρώπη για την συσχεδίαση συζητούνται. Τα περισσότερα από τα πανεπιστημιακά προγράμματα έχουν καθιερώσει ένα καλό ίδρυμα που κινείται στην προδιαγραφές συστημάτων, την αρχιτεκτονική εξερεύνησης σχεδιασμού, το διαχωρισμό, την co-σύνθεση και την εικονική διαμόρφωση πρωτοτύπου. (Σχήμα 16)



Σχήμα 16: Ερευνητική εστίαση

Οι βασικές τεχνικές και μεθοδολογίες είναι λεπτομερής παρακάτω:

Προδιαγραφές: Τα διαφορετικά προγράμματα και μελέτες για τις προδιαγραφές συστημάτων [1] παρήγαγαν μια μεγάλη ποικιλία διαφορετικών γλωσσών και βαθύτερων σημασιολογικών προτύπων. Οι προσεγγίσεις των συσχεδιάσεων για την επεξεργασία σήματος, παρουσιάζουν θεμελιώδεις διαφορές στον έλεγχο και τις προσανατολισμένες προσεγγίσεις επικοινωνίας. Το πρώτο επαναλαμβάνεται περιοδικά και μπορεί επομένως να ταξινομηθεί στα συστήματα cycle-static. Ο έλεγχος και τα προσανατολισμένα συστήματα επικοινωνίας έχουν αντιδραστικές ιδιότητες.

Αρχιτεκτονική εξερεύνησης και εκτίμηση απόδοσης: Οι σχεδιαστές συστημάτων πρέπει να παρέχονται με την δυνατότητα να υπολογίσουν εύκολα και γρήγορα τον αντίκτυπο (trade-off) εναλλακτικών αρχιτεκτονικών σχεδιασμών. Ο χρήστης καθοδηγείται στην εξερεύνηση του χρονικού διαστήματος του σχεδιασμού από τις μεταβαλλόμενες παραμέτρους σχεδιασμού, οι οποίες μπορούν να είναι περιορισμοί του στόχου της αρχιτεκτονικής ή της εφαρμογής. Τα εργαλεία εκτίμησης προσπαθούν να διαμορφώσουν τα πιθανά αποτελέσματα σχεδιασμού. Από την πλευρά υλικού αρκετή προσοχή έχει δοθεί στη χαμηλή ισχύ ή τους submicronic σχεδιασμούς. Από την πλευρά λογισμικού οι περισσότερες εκτιμήσεις απόδοσης προέρχονται από τους προσομοιωτές ή τον υπολογισμό κύκλων.

Διαχωρισμός: Συνδέθηκε για να αφαιρέσει τις βιβλιοθήκες και τα εργαλεία εκτίμησης, ο αυτόματος διαχωρισμός πρέπει να συνδεθεί στην co-synthesis: σύνθεση λογισμικού, σύνθεση υλικού και σύνθεση διασυνδέσεων.

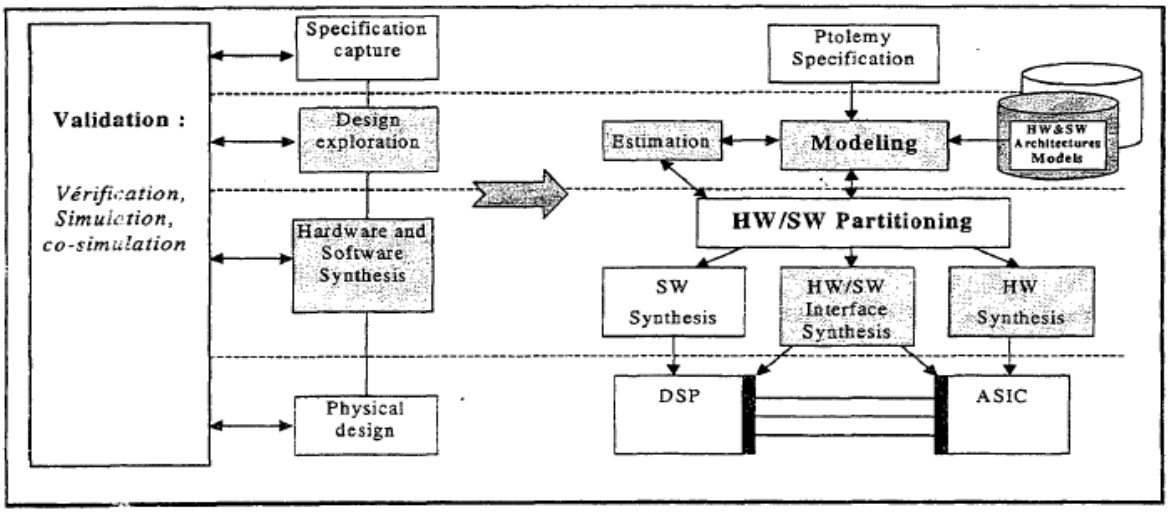
Co-synthesis: Λόγω της ποικιλίας των αρχιτεκτονικών πιθανών στόχων, είναι δύσκολο να αναζητήσει ένα ανομοιογενές διάστημα σχεδιασμού αυτόματα. Υπάρχουν τρεις κύριες περιοχές: (1) μικρά προσανατολισμένα συστήματα ελέγχου, (2) μεγαλύτερα συστήματα ελέγχου και επικοινωνιών και (3) προσανατολισμένα συστήματα στοιχείων και επεξεργασίας σήματος. Η σύνθεση λογισμικού μπορεί να διαιρεθεί σε τέσσερις διαφορετικές περιοχές: σύνολο οδηγιών (για ASIP), σύνταξη, παραγωγή κώδικα και λειτουργικά συστήματα. Η σύνθεση υλικού έχει αρχίσει να έρχεται στην αγορά, εντούτοις για τις σύνθετες αρχιτεκτονικές (συμπεριλαμβανομένης της σωλήνωσης, της ιεραρχίας της μνήμης, κλπ).

Εικονική διαμόρφωση πρωτοτύπου: Δημιουργώντας ένα περιβάλλον για να κάνει τη διόρθωση υλικού/λογισμικού, η αποδοτικότητα της συνεπαλήθευσης μπορεί να βελτιωθεί πολύ εδώ. Η εικονική διαμόρφωση πρωτοτύπου μπορεί να κρατηθεί χάρη στις υψηλές ικανότητες του FPGA (250.000 πύλες για το Xilinx 40250).

Επαναχρησιμοποίηση σχεδιασμού: Η επαναχρησιμοποίηση σχεδιασμού υλικού και λογισμικού κυρίως οδηγείται από τον αυξανόμενο αριθμό των προμηθευτών του σχεδιασμού πυρήνων (π.χ. DSP, επεξεργαστές RISC, κ.λπ.).

5.2.5 Δραστηριότητες Lester

Σε αυτό το τμήμα, παρέχουμε μια επισκόπηση των δραστηριοτήτων εργαστηριακής (LESTER LAB) έρευνάς μας. Η περιοχή λειτουργίας μας είναι η εφαρμογή πραγματικού χρόνου των εφαρμογών επεξεργασίας ψηφιακού σήματος και εικόνας. Το σχήμα 17 αντιπροσωπεύει την κορυφή της ροής της διαδικασίας του σχεδιασμού όπου οι σκιασμένες περιοχές αντιπροσωπεύουν τα ερευνητικά ενδιαφέροντα μας και τις περιοχές εργασίας μας: Εξερεύνηση σχεδιασμού και σύνθεση Hardware/Software.



Σχήμα 17: Έρευνα ενδιαφερόντων και περιοχών εργασίας του εργαστηρίου Lester

5.2.5.1 Η εξερεύνηση σχεδιασμού

Η διερεύνηση σχεδιασμού αρχίζει με τις προδιαγραφές εφαρμογής με το Ptolemy Tool [8] το οποίο μετατρέπεται έπειτα σε CDFG. Μετά από ένα βήμα του κόμβου ταξινομώντας μέσα στο CDFG προκειμένου να μπορεί η εκτίμηση να γίνει εφικτή, μια στρατηγική έρευνας [9] εκτελείται. Ανιχνεύει ένα σύνολο κρίσιμων στόχων από την άποψη του χρονικού υπολογισμού, των δαπανών μνήμης ή των εξαρτήσεων και των δαπανών των βημάτων ελέγχου.

Κατόπιν, μια άλλη στρατηγική σχεδιασμού ορίζει σε κάθε κρίσιμο στόχο μια προτεραιότητα σχεδιασμού για τη σχεδίαση αρχιτεκτονικής. Με βάση ένα σύνολο εκτελέσεων παραμετροποιημένων αρχιτεκτονικών (FPGA, ASIC, DSP, κλπ) ένα βήμα εξερεύνησης προσπαθεί: (i) να αναλύσει την αλληλοεξάρτηση μεταξύ των κρίσιμων στόχων και (ii) να επιλέξει για κρίσιμους στόχους ένα πιο υψηλής προτεραιότητας σύνολο κατάλληλων αρχιτεκτονικών. Ως εκ τούτου ένα μειωμένο σύνολο λύσεων αρχιτεκτονικής παρέχεται στο επόμενο βήμα εκτίμησης βασισμένο στα ακριβή πρότυπα αρχιτεκτονικής. Τέλος, οι πρόσθετες τεχνικές εκτιμήσεων σχετικά με κάθε αρχιτεκτονική θα παρείχαν: (i) ακριβή αποτελέσματα για την εφαρμογή εκτέλεσης και (ii) συστατική επιλογή και ταξινόμηση αρχιτεκτονικής στόχων.

5.2.5.2 Σύνθεση HW/SW

Η σύνθεση υλικού πραγματοποιείται με ένα υψηλού επιπέδου εργαλείο GAUT [3] σύνθεσης αρχιτεκτονικής αγωγών. Οι προδιαγραφές υλικού που παράγεται μετά από το βήμα διαχωρισμού περιγράφεται σε VHDL σε προσεκτικό επίπεδο και αποτελεί την είσοδο για την GAUT. Από αυτήν την περιγραφή, οι περιορισμοί συγχρονισμού (καθυστερήσεις υπολογισμού) επιβάλλονται μετά από το διαχωρισμό, και μια τεχνολογική driven βιβλιοθήκη (τυποποιημένα κύτταρα, FPGA), η αρχιτεκτονική υλικού θα συντεθεί από την GAUT. Αυτό το εργαλείο δέχεται την περιγραφή μιας επίσημης βιβλιοθήκης των χειριστών.
Μετά από μια φάση σύνταξης, που επιτρέπει την εξαγωγή του παραλληλισμού εφαρμογής, η γραφική παράσταση ροής στοιχείων που αποκτήθηκε θα παραχθεί σύμφωνα με ένα πρότυπο πυρήνων DSP. Έχουμε αναπτύξει έναν στοχαστικό εκτιμητή που, λαμβάνοντας υπόψη μια εφαρμογή γραφικής παράστασης ροής στοιχείων, παράγει τον πιθανό σχεδιασμό κάθε χειριστή υλικού. Ο εκτιμητής δέχεται ένα βήμα χρονισμού για τη λανθάνουσα κατάσταση αλγορίθμου (περιορισμοί εκτέλεσης) και παράγει και την περιοχή υλικού και την κατανάλωση ισχύος.
Η αρχιτεκτονική υλικού είναι βασισμένη σε τέσσερις λειτουργικές μονάδες: την μονάδα επεξεργασίας (Processing Unit), την μονάδα ελέγχου (Control Unit), την μονάδα μνήμης (Memory Unit) και την μονάδα επικοινωνίας (Unit Communication) που εφαρμόζει τη διασύνδεση του H/S στην πλευρά υλικού. Το γενικό δομικό πρότυπο της μονάδας επεξεργασίας είναι βασισμένο σε στοιχειώδη κύτταρα συμπεριλαμβανομένου ενός χειριστή, καταχωρητών και στοιχείων διασύνδεσης. Η μονάδα μνήμης εφαρμόζει τους καταχωρητές, τις τράπεζες μνήμης και τις σχετικές γεννήτριες διευθύνσεών τους, καθώς επίσης και το κανάλι για τη μεταφορά στοιχείων. Η διασύνδεση υλικού περιλαμβάνει τους I/O buffers αποθήκευσης στοιχείων (Fifos, Registers) και έναν βασισμένο στο FSM ελεγκτή που εφαρμόζει το πρωτόκολλο bus. Ο σχεδιασμός της διασύνδεσης υλικού θα περιγραφεί αργότερα. Το GAW αρχίζει τον κύκλο σχεδιασμού ASIC από τη μονάδα επεξεργασίας και παράγει

έπειτα τη μονάδα αποθήκευσης και τη μονάδα επικοινωνίας. Η μονάδα ελέγχου περιγράφεται απλά προκειμένου να συντεθεί από ένα εργαλείο σχεδιασμού μηχανών πεπερασμένης κατάστασης. Τέλος, η σύνθεση οδηγεί στην παραγωγή μιας δομικής και λειτουργικής περιγραφής VHDL για τις σχεδιασμένες αρχιτεκτονικές.

Έχουμε αναπτύξει τη μεθοδολογία σχεδιασμού [5] για τη διασύνδεση των ενοτήτων υλικού όπως τα ASIC με τις ενότητες λογισμικού (DSPs, από σημείο σε σημείο συνδέσεις). Στην προτεινόμενη μεθοδολογία, χρησιμοποιούμε έναν "τοπικά ασύγχρονο" τρόπο επικοινωνίας του Hardware / Software. Αυτός ο τρόπος επιτρέπει μια υψηλή επικάλυψη μεταξύ του υπολογισμού και της I/O επικοινωνίας της ενότητας υλικού, η οποία οδηγεί στις καλύτερες εκτελέσεις στο χρόνο εφαρμογής και τη μείωση δαπανών υλικού.

Η γενική αρχιτεκτονική της διασύνδεσης υλικού περιλαμβάνει τους I/O buffers αποθήκευσης στοιχείων (Fifos, Lifos, Registers) και έναν ελεγκτή FSM που εφαρμόζει το πρωτόκολλο bus. Η διαδικασία σχεδιασμού χρησιμοποιεί μια νέα μέθοδο της I/O σύνθεσης buffers που είναι βασισμένη σε επίσημους μετασχηματισμούς που επιτρέπουν την ταξινόμηση και τη βελτιστοποίηση των I/O buffers. Η επίσημη πτυχή αυτών των μετασχηματισμών εξασφαλίζει την καλή εκτέλεση της παραχθείσας αρχιτεκτονικής χωρίς τη χρησιμοποίηση μερικών δαπανηρών βημάτων επαλήθευσης και επικύρωσης.

5.2.6 Έρευνα για τις τεχνικές σχεδιασμού για τη διαχείριση δυναμικής ισχύος σε επίπεδο-συστήματος

Η διαχείριση δυναμικής ισχύος (DPM) είναι μια μεθοδολογία σχεδιασμού που μετατρέπει δυναμικά ένα ηλεκτρονικό σύστημα ώστε να παρέχει τις υπηρεσίες με ζήτηση και τα επίπεδα απόδοσης με έναν ελάχιστο αριθμό ενεργών στοιχείων ή ένα ελάχιστο φορτίο σε τέτοια στοιχεία. Η DPM καλύπτει ένα σύνολο τεχνικών που επιτυγχάνουν τον υπολογισμό απόδοσης της ενέργειας με το να κλείσουν επιλεκτικά (ή να μειώσουν την απόδοση) τα στοιχεία συστημάτων όταν είναι σε κατάσταση αναμονής (ή μερικώς ανεκμετάλλευτα). Η DPM χρησιμοποιείται στις διάφορες μορφές στους πιο φορητούς (και σε μερικά στάσιμα) ηλεκτρονικούς σχεδιασμούς, ακόμα η αίτησή της είναι μερικές φορές πρωτόγονη επειδή οι πλήρεις δυνατότητές της είναι ακόμα ανεξερεύνητες και επειδή η πολυπλοκότητα της διασύνδεσης των ετερογενών στοιχείων έχει περιορίσει τους σχεδιαστές στις απλές λύσεις.

Τα περισσότερα ηλεκτρονικά κυκλώματα και σχέδια συστημάτων έρχονται αντιμέτωπα με το πρόβλημα της παράδοσης της υψηλής απόδοσης με μια περιορισμένη κατανάλωση της ηλεκτρικής ενέργειας. Η υψηλή απόδοση απαιτείται από τις όλο και περισσότερο σύνθετες εφαρμογές (π.χ., πολυμέσα) που τρέχουν ακόμη και στις φορητές συσκευές. Η χαμηλής ισχύος κατανάλωση απαιτείται για να επιτύχει την αποδεκτή αυτονομία στα συστήματα με μπαταρίες, καθώς επίσης και να μειώσει την περιβαλλοντική επίδραση (π.χ., διασκεδασμός θερμότητας, ψύξη προκληθείσα από θόρυβο) και το κόστος λειτουργίας των στάσιμων συστημάτων. Με άλλα λόγια, η επίτευξη του ιδιαίτερα αποδοτικής ενέργειας υπολογισμού είναι μια σημαντική πρόκληση στον ηλεκτρονικό σχεδιασμό.

Μερικά στοιχεία μπορεί να έχουν μηχανικά μέρη, π.χ., κινήσεις σκληρών δίσκων (HDD), ή οπτικά μέρη, π.χ., επιδείξεις. Παραδείγματος χάριν, ένα κυψελοειδές

τηλέφωνο έχει ένα στοιχείο ψηφιακής ολοκλήρωσης πολύ μεγάλης κλίμακας (VLSI), ένα στοιχείο αναλογικής ραδιοσυχνότητας (RF), και μια απεικόνιση. Τέτοια στοιχεία μπορούν να είναι ενεργά σε διαφορετικούς χρόνους, και να καταναλώσουν αντίστοιχα τα διαφορετικά μέρη του προϋπολογισμού τηλεφωνικής ισχύος. Ομοίως, τα κύρια στοιχεία των φορητών υπολογιστών είναι τα τσιπ VLSI, HDD, και η απεικόνιση. Είναι συχνά η περίπτωση που το HDD και η απεικόνιση είναι τα πιο power-hungry [10] στοιχεία, και έτσι η αποτελεσματική χρήση τους είναι βασική στην επίτευξη των μακροχρόνιων χρόνων λειτουργίας μεταξύ των επαναφορτίσεων των μπαταριών.

5.2.6.1 Εφαρμογή της Διαχείρισης της Δυναμικής Ισχύος

Α. Η διαχείριση ισχύος στα στοιχεία συστημάτων

Εδώ, μεγάλη σημασία έχει η εσωτερική δομή των στοιχείων, και περιγράφονται διάφορες τεχνικές που μπορούν να χρησιμοποιηθούν για να σχεδιάσουν τα εύχρηστα σε ισχύς στοιχεία (PMC).

1) Clock Gating: Εξετάζονται τα πρώτα ψηφιακά στοιχεία που χρονομετρούνται. Αυτή η κατηγορία στοιχείων είναι ευρεία, και περιλαμβάνει περισσότερους επεξεργαστές, ελεγκτές και μνήμες. Η κατανάλωση ισχύος στα χρονικά ψηφιακά στοιχεία (στην τεχνολογία CMOS) είναι κατά προσέγγιση ανάλογη προς τη συχνότητα ρολογιού και προς το τετράγωνο του δυναμικού τροφοδοσίας. Η ισχύς μπορεί να σωθεί με τη μείωση της συχνότητας ρολογιού (και στο όριο με την παύση του ρολογιού), ή με τη μείωση του δυναμικού τροφοδοσίας (και στο όριο με την τροφοδότηση ενός στοιχείου). Οι δύο περιπτώσεις περιορισμού (ρολόι που παγώνει και τροφοδοτεί) ισχύουν μόνο στα Idle στοιχεία. Για τα στοιχεία που είναι σε μια ενεργή κατάσταση αλλά των οποίων η απάντηση δεν είναι η κριτική απόδοσης, η κατανάλωση ισχύος μπορεί να ανταλλαχτεί για την απόδοση με τη μείωση της συχνότητας ρολογιού ή του δυναμικού τροφοδοσίας. Η τελευταία λύση προτιμάται συνήθως λόγω της τετραγωνικής εξάρτησης της κατανάλωσης ισχύος στο δυναμικό τροφοδοσίας, και συνδυάζεται συχνά με την κλιμάκωση συχνότητας.

Κατά την εξέταση των ενδεχομένως ψηφιακών στοιχείων Idle, το gating ρολογιών (ή πάγωμα) είναι η πιο κοινή τεχνική για τη διαχείριση ισχύος. Δηλαδή, το ρολόι ενός Idle στοιχείου μπορεί να σταματήσει κατά τη διάρκεια της περιόδου αδράνειας. Η αποθήκευση ισχύς επιτυγχάνεται στους registers (των οποίων το ρολόι σταματά) και στις συνδυαστικές λογικές πύλες όπου τα σήματα δεν διαδίδουν λόγω του παγώματος των δεδομένων στους registers.

Παράδειγμα 5.3.1: Το Gating ρολογιών έχει εφαρμοστεί σε διάφορους επεξεργαστές [23-26]. Ο μικροεπεξεργαστής Alpha 21 264 χρησιμοποιεί ένα ιεραρχικό σχεδιασμός χρονομέτρησης με τα περιορισμένα ρολόγια [26]. Ειδικότερα, ο 21 264 Floating Point Unit (=μονάδα κινητής υποδιαστολής) έχει έναν ελεγκτή που μπορεί να παγώσει το ρολόι στα στοιχεία του, όπως ο αθροιστής, ο πολλαπλασιαστής, ο διαιρέτης, κ.λπ., σύμφωνα με τις οδηγίες που εκτελούνται, έτσι ώστε τα Idle στοιχεία δεν σπαταλούν ισχύ.

Ο επεξεργαστής PowerPC 603 [23] έχει και τον τοπικό και σφαιρικό έλεγχο ρολογιών. Δίνεται έμφαση εδώ σε ένα λειτουργία του σφαιρικού ελέγχου ρολογιών. Όταν ο επεξεργαστής είναι σε ένα κατάσταση Sleep, το ρολόι σε όλες τις μονάδες μπορεί να τεθεί εκτός λειτουργίας. Αφ' ετέρου, ο PLL δεν είναι απαραιτήτως εκτός

λειτουργίας στην κατάσταση Sleep, έτσι ώστε ο ελεγκτής συστήματος να μπορεί να επιλέξει από διαφορετικά επίπεδα της αποθήκευσης ισχύς, ανάλογα με τις απαιτήσεις χρόνου απόκρισης wake-up.

Παραδείγματος χάριν, εάν ένα γρήγορο wake-up απαιτείται, ο επεξεργαστής μπορεί να "ξυπνήσει" από την κατάσταση Sleep σε δέκα χρονικούς κύκλους συστημάτων, εάν το PLL είναι ενεργό. Αφ' ετέρου, για τη μέγιστη αποθήκευση ισχύς, το PLL μπορεί να αποκλειστεί στην κατάσταση Sleep. Σε αυτήν την περίπτωση, ο wake-up χρόνος μπορεί να είναι 100 μs, για να επιτρέψει στο PLL να ξανά-κλειδώσει στο εξωτερικό ρολόι.

Ο Gating ρολογιών έχει μικρά έξοδα απόδοσης γενικά: το ρολόι μπορεί να ξανά-ξεκινήσει απλά απενεργοποιώντας το σήμα παγώματος-ρολογιού. Ως εκ τούτου, το gating ρολογιών είναι ιδανικά ταιριαγμένο για την εφαρμογή των αυτορυθμισμένων στοιχείων. Σε αυτήν την περίπτωση, το ρολόι είναι πάντα σταματημένο μόλις μερικά λογικά σήματα ανίχνευσης αδράνειας που έχουν σχεδιαστεί για συγκεκριμένο σκοπό ότι το στοιχείο (ή μερικές από τις υπομονάδες του) είναι σε κατάσταση Idle.

Διάφορα εργαλεία CAD έχουν αναπτυχθεί για να υποστηρίξουν το σχεδιασμός με τοπικό gating ρολογιών (ή σημάτων) [17]–[21], [30]. Αυτά τα εργαλεία στοχεύουν να παραγάγουν αυτόματα το κύκλωμα που ανιχνεύει την αδράνεια και που εκδίδει το σήμα για να παγώσει το ρολόι. Τα εργαλεία εφαρμόζουν τις διάφορες μεθόδους με την πραγματοποίηση gating ρολογιών, οι οποίες διαφέρουν σύμφωνα με τον τύπο μονάδας που ελέγχεται (π.χ., διαδοχικός έλεγχος, μονοπάτι δεδομένων, διοχετευμένο κύκλωμα) και με τον τύπο της αδράνειας που ελέγχεται (π.χ., ζευγάρι κατάστασης/ παραγωγής ενός διαδοχικού κυκλώματος, εξωτερικό παρατηρητικότητα μερικών σημάτων).

Ο Gating ρολογιών χρησιμοποιείται ευρέως επειδή είναι εννοιολογικά απλός, έχει μικρά γενικά έξοδα από την άποψη των πρόσθετων κυκλωμάτων και συχνά μηδαμινά έξοδα απόδοσης επειδή το στοιχείο μπορεί να κάνει μετάβαση από μια Idle σε μια ενεργή κατάσταση σε έναν (ή λίγους) κύκλους.

Οι κύριες προκλήσεις σχεδιασμού στην εφαρμογή του gating ρολογιών είναι: 1) να κατασκευάσει ένα κύκλωμα ανίχνευσης-αδράνειας που είναι μικρό (και καταναλώνοντας έτσι λίγη ενέργεια) και ακριβές (δηλ., ικανό να σταματήσει το ρολόι όποτε το στοιχείο είναι σε κατάσταση Idle) και 2) να σχεδιάσει τα στοιχεία κυκλώματος διανομής ρολόι που εισάγει το ελάχιστο επιπλέον φορτίο δρομολόγησης και κρατά τη λοξή κίνηση ρολογιών υπό αυστηρό έλεγχο [22].

Σε μερικές περιπτώσεις λοιπόν, ο διασκεδασμός ισχύος μπορεί να μειωθεί περαιτέρω από την παύση όχι μόνο της διανομής ρολογιών, αλλά και από την παραγωγή ρολογιών (δηλ., με την παύση του κύριου ρολογιού PLL ή του εσωτερικού ταλαντωτή). Αυτή η επιλογή υπονοεί τις μη αμελητέες καθυστερήσεις τερματισμού και επανεκκίνησης και γενικά δεν είναι αυτοματοποιημένη. Οι καταστάσεις Sleep, όπου η σφαιρική παραγωγή ρολογιών σταματά, μπορεί μόνο να εισαχθεί με την έκδοση των εξωτερικών εντολών. Για τους επεξεργαστές, ο τερματισμός μπορεί να αρχίσει είτε από μια αφιερωμένη εντολή είτε από τη βεβαίωση ενός αφοσιωμένου σήματος.

2) Τερματισμός τροφοδοσίας: Είναι σημαντικό να τονιστεί ότι το clock-gating δεν αποβάλλει το διασκεδασμό ισχύος. Πρώτον, εάν ο gating ρολογιών είναι τοπικός, ή εάν η γεννήτρια ρολογιών είναι ενεργή, υπάρχει ακόμα δυναμικός διασκεδασμός ισχύος στα ενεργά στοιχεία κυκλώματος ρολογιών. Δεύτερον, τα ρεύματα διαρροής διαλύουν τη ισχύς ακόμα και όταν όλα τα ρολόγια σταματούν. Κατά συνέπεια, το αντικείμενο της επίτευξης ελάχιστου διασκεδασμού ισχύος, όπως απαιτείται από μερικές με μπαταρίες φορητές συσκευές, δεν μπορεί να επιτευχθεί από το gating ρολογιών.

Η κατανάλωση ισχύος των Idle στοιχείων μπορεί να αποφευχθεί με την τροφοδότηση της μονάδας. Αυτή η ριζική λύση απαιτεί ελέγξιμους διακόπτες στη γραμμή τροφοδοσίας στοιχείου. Ένα πλεονέκτημα αυτής της προσέγγισης είναι η ευρεία δυνατότητα εφαρμογής σε όλο το είδος ηλεκτρονικών στοιχείων, δηλ., ψηφιακές και αναλογικές μονάδες, αισθητήρες, και μετατροπείς. Ένα σημαντικό μειονέκτημα είναι ο wake-up χρόνος αποκατάστασης, ο οποίος είναι χαρακτηριστικά υψηλότερος απ' ότι στην περίπτωση του gating ρολογιών επειδή η λειτουργία των στοιχείων πρέπει να επανεκκινήσει.

Κατά το συλλογισμό ενός μικρο-ηλεκτρονικού κυκλώματος (π.χ., επεξεργαστής, ελεγκτής), ένα τέτοιο στοιχείο είναι χαρακτηριστικά δομημένο ως ιεραρχικές συνθέσεις των μικρών στοιχείων. Κατά συνέπεια, ο τερματισμός ισχύς εφαρμόζεται σε έναν επιλεγμένο αριθμό μικρών στοιχείων. Στην περίπτωση των σύνθετων κυκλωμάτων, συνήθως μια μερίδα του κυκλώματος δεν τροφοδοτείται, έτσι ώστε μπορεί να τρέξει ένα σύνολο ελάχιστων λειτουργιών παρακολούθησης και ελέγχου, και να ξυπνά τα στοιχεία τροφοδοτημένα όταν απαιτούνται.

Παράδειγμα 5.3.2: Το τσιπ StrongARM SA-1100 [12] έχει δύο παροχές ηλεκτρικού ρεύματος: μια VDDI 1.5-V εσωτερική παροχή ηλεκτρικού ρεύματος και μια VDDX 3.3-V διεπαφή δυναμικού τροφοδοσίας. Η VDDI τροφοδοτεί τον πυρήνα CPU και την πλειοψηφία των λειτουργικών μονάδων στο τσιπ (ελεγκτής DMA, MMU, LCD, κ.λπ.). Η VDDX τροφοδοτεί τους εισόδου-εξόδου drivers, έναν εσωτερικό 32-KHz ταλαντωτή κρυστάλλου, τη μονάδα ελέγχου συστημάτων, και μερικά κρίσιμα κυκλώματα.

Η Sleep κατάσταση δηλώνει ότι ο SA - 1100 είναι ένα παράδειγμα του τερματισμού παροχής ηλεκτρικού ρεύματος. Το ηλεκτρικό ρεύμα στην κατάσταση Sleep μειώνεται στα 0,16 mW (σε αντίθεση με τα 400 mW στην κατάσταση Run) με την παύση της τροφοδοσίας του VDDI. Η ακολουθία τερματισμού για την εισαγωγή σε κατάσταση Sleep περνά από τρεις φάσεις: 1) βελτίωση μνήμης από όλες τις πληροφορίες κατάστασης που πρέπει να συντηρηθούν καθ' όλη τη διάρκεια της περιόδου Sleep, 2) επαννεκίνηση όλων των εσωτερικά καταστάσεων επεξεργαστών και του προγράμματος των wakeup γεγονότων και 3) τερματισμός εσωτερικής γεννήτριας ρολογιών. Κάθε φάση παίρνει περίπου 30 μs. Κατά τη διάρκεια της κατάστασης Sleep, ο SA - 1100 παρακολουθεί μόνο για τα προγραμματισμένα εκ των πρότερων wake-up γεγονότα.
Ο Wake-up επεξεργαστής περνά από τρεις φάσεις: 1) ramp-up VDDX και έναρξη ρολογιών και επεξεργαστών, 2) αναμονή χρόνου για τη σταθεροποίηση του ρολογιού του επεξεργαστή και 3) ακολουθία εκκίνησης της CPU. Οι πρώτες δύο φάσεις παίρνουν, αντίστοιχα, 10 και 150 ms. Η τρίτη φάση έχει την αμελητέα διάρκεια έναντι των δύο πρώτων. Η κατάσταση Sleep μπορεί να εισαχθεί είτε αυξάνοντας μια

αφιερωμένη "καρφίτσα" (αποκαλούμενη BATT_FAULT) είτε από μια διαδικασία λογισμικού που γράφει στον καταχωρητή ελέγχου διαχείρισης ισχύος PMCR της CPU.

Το κλείσιμο του υπολογιστή ισχύει για τα ηλεκτροπτικά και ηλεκτρομηχανικά τμήματα συστημάτων, όπως οι επιδείξεις και HDDs. Για τα συστήματα με τα μηχανικά κινούμενα μέρη, όπως των HDD, οι χρονικές σταθερές που περιλαμβάνονται στην επιτάχυνση και την επιβράδυνση των κινούμενων μερών είναι συνήθως πολύ μεγαλύτερες από εκείνες που περιλαμβάνονται στο άνοιγμα και κλείσιμο των ηλεκτρονικών στοιχείων. Επιπλέον, η επιτάχυνση και η επιβράδυνση τείνουν να μειώσουν την αναμενόμενη διάρκεια ζωής του στοιχείου [43]. Η μείωση διάρκειας ζωής μπορεί να φανεί ως ένα άλλο κόστος που συνδέεται με τις μεταβάσεις καταστάσεων.

Παράδειγμα 5.3.3: Εξετάζεται πάλι ο μηχανισμός δίσκου IBM Travelstar 14GS [13]. Σε αυτό το στοιχείο, μπορεί να τονιστεί ως κύριες υπομονάδες: η μηχανή αξόνων, η τοποθέτηση κεφαλής του υποσυστήματος, και η κεντρική διασύνδεση.

Ο IBM Travelstar HDD έχει εννέα καταστάσεις ισχύς: μια κατάσταση απότομης επιτάχυνσης (spin-up) για να εκκινήσει το μηχανισμό από το κλείσιμο, τρείς λειτουργικές καταστάσεις (αναζήτηση, γράψιμο και διάβασμα), και πέντε ανενεργές καταστάσεις (Performance Idle, Active Idle, Low power idle, Standby, and Sleep). Οι διαφορετικοί φυσικοί μηχανισμοί χρησιμοποιούνται για να μειώσουν τη ισχύς στις ανενεργές καταστάσεις.

Στην κατάσταση Performance Idle, όλα τα ηλεκτρονικά στοιχεία τροφοδοτούνται ενώ στην κατάσταση Active Idle, κάποια στοιχεία κυκλώματος είναι στον τρόπο αποθήκευσης ισχύς, και στην κατάσταση Low power idle η κεφαλή είναι μη-φορτωμένη. Εκτιμώντας ότι η μηχανή αξόνων περιστρέφεται στις τρείς Idle καταστάσεις, η μηχανή περιστρέφεται στις καταστάσεις Standby και Sleep. Στην κατάσταση Standby η κεντρική διασύνδεση είναι ενεργή, ενώ στην Sleep είναι αδρανή.

Η κατανάλωση ισχύος στις ενεργές καταστάσεις (μέσο όρο 2.6 W) μειώνεται στις ανενεργές καταστάσεις στις τιμές 2, 1.3, 0.85, 0.25, και 0.1 W, αντίστοιχα. Η επαννεκίνηση του HDD απαιτεί μια μέγιστη ισχύς των 5 W, λόγω της επιτάχυνσης των δίσκων. Τέλος, σημειώστε ότι όσο χαμηλότερη είναι η κατανάλωση ισχύος, τόσο περισσότερος ο αντίστοιχος χρόνος wake up είναι. Κατά συνέπεια, οι στρατηγικές DPM πρέπει να εκμεταλλευθούν τις χαμηλής ισχύος καταστάσεις ελαχιστοποιώντας τον αντίκτυπο στην απόδοση.

3) Πολλαπλάσιες και μεταβλητές παροχές ηλεκτρικού ρεύματος: η DPM ισχύει επίσης στα στοιχεία που δεν είναι σε κατάσταση Idle, αλλά των οποίων οι απαιτήσεις απόδοσης (π.χ., οι I/O καθυστερήσεις) ποικίλλουν με το χρόνο. Η τεχνολογία εφαρμογής μπορεί έπειτα να βασιστεί στην επιβράδυνση των μη κρίσιμων στοιχείων. Η επιβράδυνση επιτυγχάνεται με το χαμήλωμα του δυναμικού τροφοδοσίας, έτσι ώστε το στοιχείο να γίνεται ως απόδοση κρίσιμη. Οι πρόωρες εφαρμογές των τσιπ πολυδυναμικού χρησιμοποίησαν έναν στατικό κατευθυντήριας-ισχύος διαμοιρασμό στις υπομονάδες, κάθε ένας τροφοδοτημένος από ένα διαφορετικό δυναμικό τροφοδοσίας. Τα συχνότερα δύο επίπεδα δυναμικού χρησιμοποιήθηκαν, και οι μοχλοί μετατόπισης επιπέδων χρησιμοποιήθηκαν στα σύνορα των υπομονάδων που τρέχουν

σε διαφορετικές τάσεις [53]. Η επέκταση αυτής της προσέγγισης στη σφαίρα της DPM είναι να επιτραπεί η δυναμική ρύθμιση του δυναμικού τροφοδοσίας ηλεκτρικού ρεύματος κατά τη διάρκεια της λειτουργίας του συστήματος. Μια από τις κύριες προκλήσεις στην εφαρμογή αυτής της επέκτασης είναι να εγγυηθεί ότι η συχνότητα ρολογιών ακολουθεί τις αλλαγές ταχύτητας που προκαλούνται από τις δυναμικές ρυθμίσεις του δυναμικού τροφοδοσίας.

Στην πρωτοποριακή εργασία από την Nielsen [54], τα αυτο-χρονομετρούμενα κυκλώματα χρησιμοποιήθηκαν από κοινού με το μεταβλητό δυναμικό τροφοδοσίας. Τα αυτο-χρονομετρούμενα κυκλώματα συγχρονίζουν χρησιμοποιώντας τα τοπικά σήματα χειραψίας, ως εκ τούτου, αυτοί δεν χρειάζονται τα διευθετήσιμα ρολόγια. Δυστυχώς, τα αυτο-χρονομετρούμενα κυκλώματα δεν είναι η επικρατούσα τεχνολογία. Οι εναλλακτικές προσεγγίσεις υιοθετούν την τυποποιημένη σύγχρονη λογική [55, 57, 58] που συνδέονται με τα διευθετήσιμα ρολόγια που προσαρμόζουν τη συχνότητά τους στην ταχύτητα της κρίσιμης πορείας κάτω από τα διαφορετικά δυναμικά τροφοδοσίας.

Ένα άλλο ζήτημα στα συστήματα με το δυναμικά μεταβλητό δυναμικό τροφοδοσίας είναι ότι απαιτούν τους υψηλής απόδοσης μετατροπείς συνεχούς τάσης που μπορούν να προγραμματιστούν πέρα από ένα ευρύ φάσμα των τάσεων παραγωγής. Διάφοροι διευθετήσιμοι μετατροπείς συνεχούς τάσης έχουν περιγραφεί στη λογοτεχνία [59-62]. Η προσέγγιση μεταβλητού δυναμικού τροφοδοσίας μπορεί να συμπληρωθεί από τη δυναμική ρύθμιση ορίου-τάσης, που επιτυγχάνεται με τον έλεγχο της πόλωσης των σωμάτων [57,58].

Οι δυναμικά ποικίλες τάσεις τροφοδοσίας μπορούν να κβαντοποιηθούν [55] και να περιοριστούν έτσι σε έναν πεπερασμένο αριθμό τιμών, ή μπορούν να πάρουν τις τιμές σε μια συνεχή σειρά. Στην προηγούμενη περίπτωση είναι δυνατό να προσδιοριστεί ένας πεπερασμένος αριθμός καταστάσεων ισχύος για το σύστημα, στην τελευταία η έννοια της πεπερασμένης κατάστασης δεν ισχύει. Η μετάβαση κατάστασης παίρνει έναν πεπερασμένο χρόνο επειδή οι μετατροπείς συνεχούς τάσης δεν μπορούν να υποστηρίξουν τις αυθαίρετα γρήγορες αλλαγές στην τάση τροφοδοσίας.

Β. Εφαρμογή/Υλοποίηση της διαχείρισης ισχύος σε επίπεδο συστήματος

Σε αυτό το κεφάλαιο, εξετάζεται η DPM σε επίπεδο συστημάτων, και τα αντίστοιχα ζητήματα εφαρμογής. Ο σχεδιασμός της DPM σε επίπεδο συστημάτων μπορεί να συνυπάρξει με την τοπική διαχείριση ισχύος των στοιχείων. Μελετώντας τα ηλεκτρονικά συστήματα που εφαρμόζονται στο υλικό, ο διαχειριστής ισχύος είναι μια εξειδικευμένη μονάδα ελέγχου που ενεργεί παράλληλα και σε συντονισμό με τη μονάδα ελέγχου συστημάτων. Με άλλα λόγια, ο διαχειριστής ισχύος μπορεί να είναι ένας hardwired ή microprogrammed ελεγκτής, και ενδεχομένως συγχωνευμένος με τον ελεγκτή συστημάτων. Οι πολιτικές βασισμένες στα timeouts εφαρμόζονται εύκολα από τους χρονιστές. Οι στοχαστικές πολιτικές μπορούν να εφαρμοστούν από τους πίνακες εμφάνισης (όταν είναι στάσιμες) ή από τα διαδοχικά κυκλώματα. Οι τυχαίες πολιτικές απαιτούν τη χρήση των ψευδοτυχαίων γεννητριών αριθμών, οι οποίες μπορούν να εφαρμοστούν από τους γραμμικούς καταχωρητές μετατόπισης ανατροφοδότησης (LFSR).

Τα χαρακτηριστικά ηλεκτρονικά συστήματα είναι λογισμικό προγραμματίσημο, και μια πλειοψηφία έχει ένα λειτουργικό σύστημα που κυμαίνεται από έναν απλό

σχεδιαστή χρόνου εκτέλεσης ή το λειτουργικό σύστημα πραγματικού χρόνου (RTOS) (για τις ενσωματωμένες εφαρμογές) σε ένα ολοκληρωμένο λειτουργικό σύστημα (όπως στην περίπτωση των προσωπικών υπολογιστών ή των τερματικών σταθμών).

Υπάρχουν διάφοροι λόγοι για τη μετανάστευση τον διαχειριστή ισχύος στο λογισμικό. Οι διαχειριστές ισχύος λογισμικού είναι εύκολο να γράψουν και να μετατρέψουν. Στις περισσότερες περιπτώσεις, ο σχεδιαστής δεν μπορεί, ή δε θέλει, να παρεμποδίσει και να τροποποιήσει την underlying πλατφόρμα υλικού. Οι εφαρμογές DPM είναι ακόμα μια νέα τέχνη, και ο πειραματισμός με το λογισμικό είναι ευκολότερος απ' ότι με το υλικό.

Γενικά, το λειτουργικό σύστημα είναι το επίπεδο λογισμικού όπου η πολιτική της DPM μπορεί να εφαρμοστεί καλύτερα. Η διαχείριση ισχύος βασισμένη στο OS (OSPM) έχει το πλεονέκτημα ότι ο δυναμικός έλεγχος ισχύος/απόδοσης εκτελείται από το επίπεδο λογισμικού (το OS) που διαχειρίζεται τον υπολογιστικό, την αποθήκευση και τους I/O στόχους του συστήματος. Η εφαρμογή OSPM είναι ένα πρόβλημα υλικού/λογισμικού συσχεδίασης επειδή οι πόροι υλικού πρέπει να διασυνδεθούν με τον διαχειριστή ισχύος λογισμικού βασισμένη στο OS, και επειδή και οι πόροι υλικού και τα προγράμματα εφαρμογής λογισμικού πρέπει να σχεδιαστούν έτσι ώστε συνεργάζονται με τον OSPM.

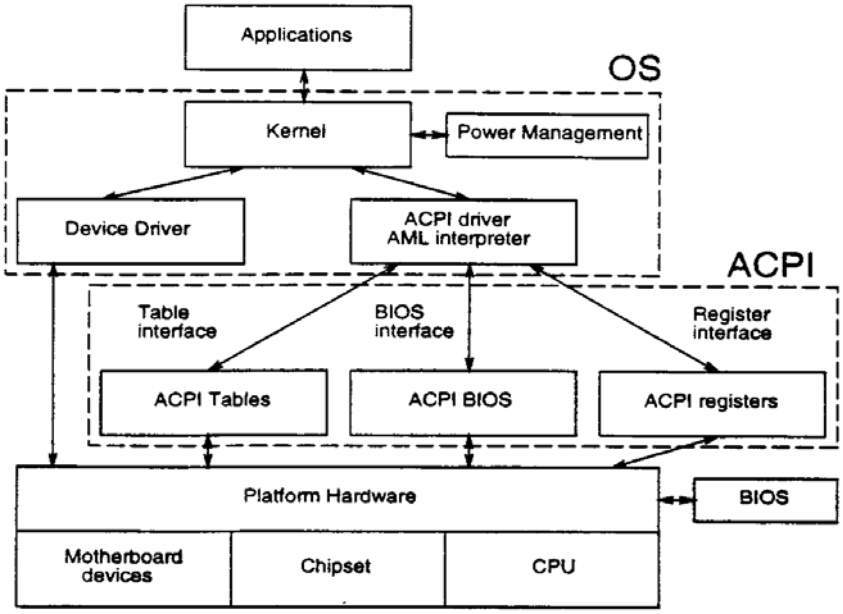
Οι πρόσφατες πρωτοβουλίες για να αντιμετωπιστεί η διαχείριση ισχύος σε επίπεδο συστήματος περιλαμβάνουν την πρωτοβουλία του OnNow της Microsoft [29] και τα πρότυπα διεπαφών προηγμένης διαμόρφωσης και ισχύος (ACPI) που προτείνονται από την Intel, τη Microsoft, και Toshiba [30]. Το πρώτο υποστηρίζει την εφαρμογή OSPM και στοχεύει στο σχεδιασμό των προσωπικών υπολογιστών με τη βελτιωμένη δυνατότητα χρησιμοποίησης μέσω του καινοτόμου σχεδιασμού του OS. Το τελευταίο απλοποιεί την συσχεδίαση του OSPM με την παροχή προτύπων διεπαφών στους πόρους συστημάτων ελέγχου. Αφ' ετέρου, τα προαναφερθέντα πρότυπα δεν παρέχουν τις διαδικασίες για το βέλτιστο έλεγχο του συστήματος ρυθμισμένο σε ισχύς.

1) Βιομηχανικά πρότυπα σχεδιασμού: Τα βιομηχανικά πρότυπα έχουν προταθεί για να διευκολύνουν την ανάπτυξη της διαχείρισης ισχύος βασισμένης στο λειτουργικό σύστημα. Η Intel, η Microsoft και Toshiba πρότειναν τα ανοικτά πρότυπα που ονομάστηκαν, προηγμένη διασύνδεσης διαμόρφωσης και ισχύος (ACPI) [30]. Η ACPI παρέχει το πρότυπο διαχείρισης και διαμόρφωσης ισχύος ανεξάρτητες από το OS. Επιτρέπει μια τακτική μετάβαση από το υλικό κληρονομικότητας στο ACPI συμβατό υλικό. Αν και αυτή η πρωτοβουλία στοχεύει στους προσωπικούς υπολογιστές (PC), περιέχει χρήσιμες οδηγίες για μια γενικότερη κατηγορία συστημάτων. Οι κύριοι στόχοι ACPI είναι: 1) να επιτρέψει σε όλα τα PC να εφαρμόσουν τη δυναμική διαμόρφωση και τη διαχείριση ισχύος των μητρικών καρτών, 2) να ενισχύσει τα λειτουργίες της διαχείρισης ισχύος και την ευρωστία των συστημάτων ρυθμισμένων σε ισχύς, και 3) να επιταχύνει την εφαρμογή των υπολογιστών ρυθμισμένων σε ισχύς, να μειώνει τις δαπάνες και το χρόνο στην αγορά.

Η προδιαγραφές ACPI καθορίζει τις περισσότερες διασύνδεσης μεταξύ του λογισμικού OS και του υλικού. Τα στοιχεία λογισμικού και υλικού σχετικά με την ACPI παρουσιάζονται στο σχήμα 18. Οι εφαρμογές αλληλεπιδρούν με τον πυρήνα OS μέσω των διασυνδέσεων προγραμματισμού εφαρμογής (API). Μια ενότητα του

OS εφαρμόζει τις πολιτικές διαχείρισης ισχύος. Η ενότητα διαχείρισης ισχύος αλληλεπιδρά με το υλικό μέσω των πυρήνων των υπηρεσιών (κλήσεις συστημάτων). Ο πυρήνας αλληλεπιδρά με το υλικό χρησιμοποιώντας τους οδηγούς συσκευών. Πριν τη διασύνδεσης ACPI είναι ο οδηγός ACPI. Ο οδηγός που είναι ο συγκεκριμένος OS, απεικονίζει τα αιτήματα πυρήνων στις εντολές ACPI, και οι απαντήσεις / μηνύματα ACPI στα σήματα /διακοπές των πυρήνων. Σημειώστε ότι ο πυρήνας μπορεί επίσης να αλληλεπιδράσει με το μη-υπάκουο ACPI υλικό μέσω άλλων οδηγών συσκευών.

Στο κατώτατο σημείο του σχήματος 18 παρουσιάζεται η πλατφόρμα υλικού. Αν και αντιπροσωπεύεται ως μονολιθικό μπλοκ, είναι χρήσιμο να διακριθούν τρεις τύποι στοιχείων υλικού. Κατ' αρχάς, οι πόροι υλικού (ή συσκευές) είναι τα τμήματα συστημάτων που παρέχουν κάποιο είδος εξειδικευμένης λειτουργίας (π.χ., τηλεοπτικοί ελεγκτές, συσκευές τηλεπικοινωνίας, ελεγκτές αρτηριών). Δεύτερον, η CPU μπορεί να φανεί ως εξειδικευμένος πόρος που πρέπει να είναι ενεργός για το OS (και το στρώμα διασύνδεσης ACPI) που τρέχει.



Σχήμα 18: Διεπαφή ACPI και πλατφόρμα PC

Τέλος, το chipset (επίσης αποκαλούμενο ως λογικός πυρήνας) είναι η λογική μητρική κάρτα που ελέγχει τις πιο βασικές λειτουργίες υλικού (όπως τους χρονόμετρα πραγματικού χρόνου, τη διακοπή σημάτων, τους επεξεργαστές αρτηριών) και διασυνδέει την CPU με όλες τις άλλες συσκευές. Αν και η CPU τρέχει το OS, καμία δραστηριότητα συστημάτων δεν θα μπορούσε να εκτελεσθεί χωρίς το chipset. Από την άποψη της διαχείρισης ισχύος, το chipset, ή ένα κρίσιμο μέρος από αυτό, πρέπει πάντα να είναι ενεργό επειδή το σύστημα στηρίζεται σε αυτό για να βγει από την κατάσταση Sleep.

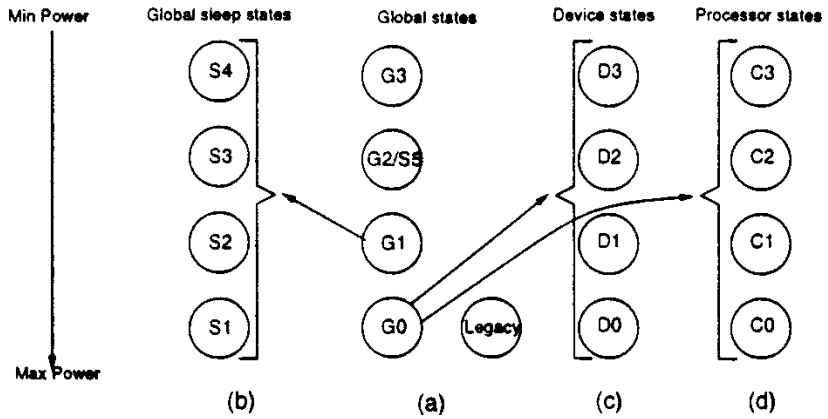
Είναι σημαντικό να παρατηρηθεί ότι το ACPI δεν διευκρινίζει ούτε το πώς να εφαρμόσει τις συσκευές υλικού ούτε το πώς να πραγματοποιήσει τη διαχείριση ισχύος στο λειτουργικό σύστημα. Κανένας περιορισμός δεν επιβάλλεται στις μορφές εφαρμογής για το υλικό και στις πολιτικές διαχείρισης ισχύος. Η εφαρμογή του ACPI - υπάκουου υλικού μπορεί να «ενισχύσει» οποιαδήποτε τεχνολογία ή αρχιτεκτονική

βελτιστοποίησης εφ' όσον η συσκευή ρυθμισμένη σε ισχύς είναι ελέγξιμη από την πρότυπη διασύνδεση που διευκρινίζεται από την ACPI.

Στην ACPI, το σύστημα έχει πέντε συνολικές καταστάσεις ισχύος:

- ***Mechanical off state G3***, χωρίς την κατανάλωση ισχύος.
- ***Soft off state G2*** (επίσης αποκαλούμενο ως *S5*). Μια πλήρης επανεκκίνηση OS απαιτείται για να αποκαταστήσει την λειτουργική κατάσταση.
- ***Sleep state G1***. Το σύστημα εμφανίζεται να είναι κλειστό και η κατανάλωση ισχύος μειώνεται. Το σύστημα επιστρέφει στη λειτουργική κατάσταση σε ένα χρονικό διάστημα που αυξάνεται με το αντίστροφο της κατανάλωσης ισχύος.
- ***Working state G0***, όπου το σύστημα είναι ανοικτό και πλήρως χρησιμοποιήσιμος.
- ***Legacy state***, η οποία εισέρχεται όταν το σύστημα δεν ‘’υπακούει’’ με την ACPI.

Οι συνολικές καταστάσεις παρουσιάζονται στο σχήμα 19(α). Ταξινομούνται από πάνω έως κάτω με την αύξηση του σκεδασμού ισχύος.
Οι προδιαγραφές της ACPI βελτιώνουν την ταξινόμηση των συνολικών καταστάσεων των συστημάτων με τον καθορισμό τεσσάρων καταστάσεων Sleep μέσα στην κατάσταση *G1*, όπως φαίνεται στο σχήμα 19(β).



Σχήμα 19: Ορισμοί κατάστασης για την ACPI

- Η ***S1*** είναι μια κατάσταση Sleep με την χαμηλή wake-up λανθάνουσα κατάσταση. Κανένα πλαίσιο συστημάτων δεν χάνεται στην CPU ή στο chipset.
- Η ***S2*** είναι μια χαμηλή wake-up λανθάνουσας κατάστασης της κατάστασης Sleep. Αυτή η κατάσταση είναι παρόμοια με την ***S1*** κατάσταση Sleep με την εξαίρεση ότι το πλαίσιο της CPU και της cache (=βοηθητική μνήμη) του συστήματος χάνεται.
- Η ***S3*** είναι μια άλλη χαμηλή wake-up λανθάνουσας κατάστασης της κατάστασης Sleep όπου όλο το πλαίσιο συστήματος χάνεται εκτός από τη μνήμη συστήματος.
- Η ***S4*** είναι η κατάσταση Sleep με τη χαμηλότερη ισχύς και την πιο μακροχρόνια wake-up λανθάνουσα κατάσταση. Για να μειώσουν την ισχύ στο ελάχιστο, όλες οι συσκευές πρέπει να κλείσουν.

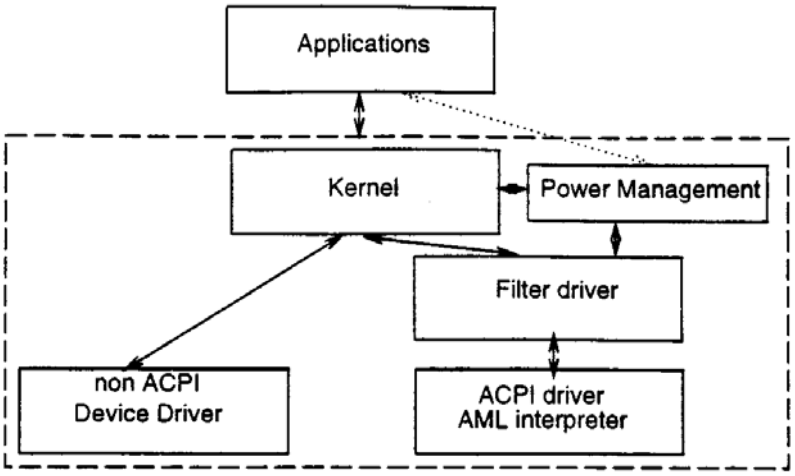
Επιπλέον, οι προδιαγραφές ACPI καθορίζουν τις καταστάσεις για τα τμήματα συστημάτων. Υπάρχουν δύο τύποι τμημάτων συστημάτων, οι συσκευές και ο επεξεργαστής, για τα οποία οι καταστάσεις ισχύος καθορίζονται. Οι συσκευές είναι αφηρημένες αντιπροσωπεύσεις των πόρων υλικού στο σύστημα. Τέσσερις

καταστάσεις καθορίζονται για τις συσκευές, όπως φαίνεται στο σχήμα 19(γ). Σε αντίθεση με τις συνολικές καταστάσεις ισχύος, οι καταστάσεις ισχύος των συσκευών δεν είναι ορατές στο χρήστη.

Παραδείγματος χάριν, μερικές συσκευές μπορεί να είναι σε μια αδρανή κατάσταση, αλλά το σύστημα να φαίνεται ότι είναι σε μια λειτουργική κατάσταση. Επιπλέον, οι μεταβάσεις καταστάσεων για τις διαφορετικές συσκευές μπορούν να ελεγχθούν από τα διαφορετικά σχέδια διαχείρισης ισχύος. Ο επεξεργαστής είναι η μονάδα κεντρικής επεξεργασίας που ελέγχει ολόκληρη την πλατφόρμα του PC. Ο επεξεργαστής έχει δικές του καταστάσεις ισχύος, όπως φαίνεται στο σχήμα 19(δ). Παρατηρήστε την ενδογενή ασυμμετρία του προτύπου ACPI. Ο κεντρικός ρόλος της CPU αναγνωρίζεται, και ο επεξεργαστής δεν συμπεριφέρεται ως απλός πόρος.

2) Εφαρμογές DPM βασισμένες στην ACPI: Ένα σύνολο πειραμάτων πραγματοποιήθηκε από την LU [44, 45] για να μετρήσει την αποτελεσματικότητα των διαφορετικών πολιτικών DPM. Η LU χρησιμοποίησε δύο υπάκουους στην ACPI υπολογιστές, που τρέχουν μια βήτα έκδοση των Windows NT V5, η οποία είναι επίσης υπάκουο στην ACPI. Ο πρώτος υπολογιστής είναι ένας υπολογιστής γραφείου VarStation 2861A, που χρησιμοποιεί έναν Pentium II επεξεργαστή και μια IBM DTTA 350-640 HDD. Ο δεύτερος είναι ένα laptop Sony VAIO PCG F-150, με ένα Pentium II και ένα Fujitsu MHF 2043AT HDD. Τα πειράματα στοχεύουν στον έλεγχο της μονάδας HDD χρησιμοποιώντας διαφορετικές πολιτικές.

Για αυτόν το λόγο, η LU εφάρμοσε τους οδηγούς φίλτρων (σχήμα 20) για να ελέγξει τις καταστάσεις ισχύος του HDD, για να καταγράψει τις προσβάσεις δίσκων και για να αναλύσει τον αντίκτυπο απόδοσης της διαχείρισης ισχύος πάνω από κάθε αλγόριθμο. Τα ηλεκτροφόρα καλώδια των δίσκων ελέγχθηκαν από τα ψηφιακά πολύμετρα, που συνδέθηκαν με ένα PC μέσω μιας θύρας RS-232 για να καταγράψουν τις μετρήσεις.



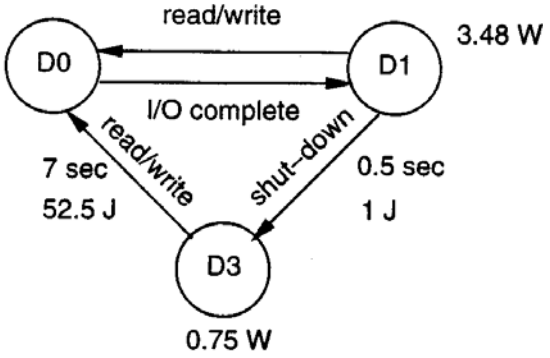
Σχήμα 20: Η DPM χρησιμοποιώντας τους οδηγούς φίλτρων

Η IBM HDD μπορεί να είναι μια από τις τρεις καταστάσεις: *PowerDeviceD0* όταν διαβάζει ή γράφει, *PowerDeviceD1* όταν οι πλάκες περιστρέφονται και *PowerDeviceD3* όταν οι πλάκες σταματούν. Τα I/O αιτήματα περιμένουν μόνο για καθυστερήσεις αναζήτησης και περιστροφής όταν ο δίσκος είναι σε κατάσταση

PowerDeviceD1 (σχήμα 21). Εάν ένα αίτημα φθάνει όταν είναι ο σκληρός δίσκος είναι σε κατάσταση *PowerDeviceD3*, πρέπει να περιμένει για τη wake-up διαδικασία εκτός από τις καθυστερήσεις αναζήτησης και περιστροφής. Ο δίσκος καταναλώνει 3,48 και 0,75 W στις καταστάσεις D1 και D3, αντίστοιχα. Παίρνει περίπου 7 s και 52,5 J για να πλησιάσει από την κατάσταση D3 στην D0. Παίρνει (κατά μέσο όρο) 0,5 s για να εισαγάγει την D3 από την D1. Η συμπεριφορά της Fujitsu HDD είναι παρόμοια, αλλά με διαφορετικές παραμέτρους (δείτε τον πίνακα 3). Οι ισορροπημένου προϋπολογισμού χρόνοι της IBM και της Fujitsu HDD είναι 17,6 και 5,43 s, αντίστοιχα.

Model	P_{Off}	P_{On}	T_{sd}	E_{sd}	T_{wu}	E_{wu}
	Watt	Watt	sec	J	sec	J
IBM	0.75	3.48	0.51	1.08	6.97	52.5
Fujitsu	0.13	0.95	0.67	0.36	1.61	4.39

Πίνακας 3: Παράμετροι δίσκων: Οι subscripts SD και WU δείχνουν το κλείσιμο και ξυπνήστε, αντίστοιχα



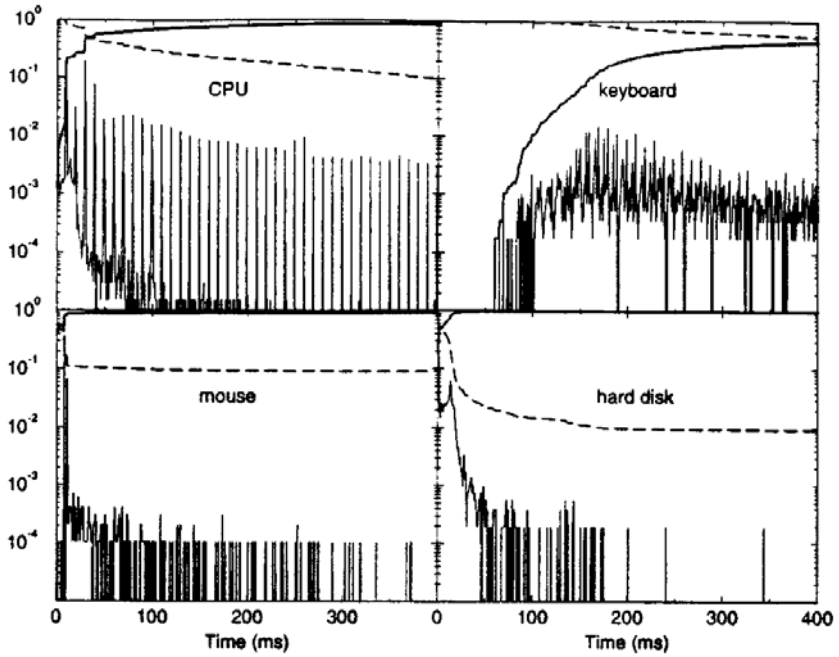
Σχήμα 21: Η PSM για IBM DTTA HDD

Για λόγους σύγκρισης, και οι δύο υπολογιστές εκτελούν το ίδιο ίχνος των εισερχόμενων δεδομένων (ένα 11-h μακρύ ίχνος εκτέλεσης 11 βημάτων). Τα αποτελέσματα δείχνουν ότι όλοι οι αλγόριθμοι ξοδεύουν λιγότερο από 1% του υπολογισμού στη διαχείριση της ίδιας της ισχύος τους, επικυρώνοντας κατά συνέπεια μια θεμελιώδη προϋπόθεση αυτού του κύριου τμήματος της εργασίας. Για τον υπολογιστή laptop (υπολογιστής γραφείου), οι μειώσεις ισχύος έχουν μετρηθεί μέχρι 55% (43%) και μέχρι 34% (23%) (σε σύγκριση με την προεπιλογή της πολιτικής timeout των 3 min των Windows OS). Η μεγαλύτερη αποθήκευση ισχύος επιτυγχάνεται στον υπολογιστή laptop λόγω του πιο σύντομου ισορροπημένου προϋπολογισμού χρόνου του δίσκου της.

3) *Εφαρμογή παρατηρητών:* Η διαχείριση ισχύος απαιτεί τις πληροφορίες για τη χρήση κάθε πόρου υλικού, όπως: 1) διανομή των χρόνων μεταξύ αφίξεων του αιτήματος στους πόρους και 2) διανομή των χρόνων υπηρεσιών για τα αιτήματα. Η

ενότητα παρατηρητών (το σχήμα 22) του PM φροντίζει για την συλλογή δεδομένων. Στα PC υπάκουα στην ACPI, ο παρατηρητής μπορεί να στηριχθεί στα μηνύματα ACPI για να λάβει τα δεδομένα που απαιτούνται για να οδηγήσουν τις πολιτικές. Εντούτοις, δεν είναι όλοι οι υπολογιστές υπάκουοι στην ACPI. Σε αυτό το τμήμα, αναλύεται η εφαρμογή μιας ενότητας παρατηρητών διαχειριστών ισχύος που δεν εκμεταλλεύεται την ACPI, ούτε είναι βασισμένη σε ένα ιδιόκτητο λειτουργικό σύστημα της Microsoft. Οι βασικές προϋποθέσεις για την εφαρμογή του παρατηρητή είναι οι ακόλουθες:

- *Χαμηλή διαταραχή της κανονικής δραστηριότητας συστημάτων:* Ο έλεγχος πρέπει να είναι διαφανής στον τελικό χρήστη και πρέπει να τροποποιήσει τα σχέδια χρήσης των πόρων υλικού όσο το δυνατόν ελάχιστα.
- *Ευελιξία*: Πρέπει να είναι εύκολο να ελεγχθούν οι πολλαπλοί τύποι πόρων. Επιπλέον, ο αριθμός και οι τύποι των παρατηρηθέντων πόρων πρέπει να είναι δυναμικά ελέγξιμοι. Αυτή τη λειτουργία είναι ιδιαίτερα χρήσιμη για τους υπολογιστές laptop όπου οι νέες συσκευές μπορούν να εγκατασταθούν κατά τη διάρκεια της λειτουργίας των συστημάτων (δηλ., έτοιμη προς χρήση ικανότητα).
- *Ακρίβεια:* Οι καλά-γνωστές χρησιμότητες συστημάτων δίνουν την πρόσβαση στις συσσωρευτικές αριθμήσεις των προσβάσεων στους πόρους συστημάτων. Αυτή η λειτουργία δεν είναι επαρκής για να λάβει τις ακριβείς στατιστικές των χρόνων μεταξύ των αιτήσεων και των χρόνων υπηρεσιών. Μια σημαντική λειτουργία του παρατηρητή είναι η ικανότητα της χρόνο-σφράγισης των γεγονότων με υψηλό ψήφισμα.



Σχήμα 22: Στατιστική ανάλυση για τον χρόνο αφίξεως. Για κάθε συσκευή, σχεδιάζονται τρεις καμπύλες στην κλίμακα lin-log: η πιθανότητα πυκνότητας (συμπαγής γραμμή), η πιθανότητα κατανομής (έντονη γραμμή), και το συμπλήρωμά του καθενός (ορμούμενη (dashed) γραμμή). Η πληροφορία αναφέρεται στη ανάπτυξη λογισμικού.

Η βασισμένη στο λογισμικό αρχιτεκτονική παρατηρητών που αναλύεται σε αυτό το τμήμα καλείται IPM [15], και έχει εφαρμοστεί ως επέκταση του λειτουργικού συστήματος Linux [63]. Ο παρατηρητής ελέγχει τις προσβάσεις στους πόρους

συστημάτων και τις αποθηκεύει με μορφή χρόνο-σφραγισμένων γεγονότων. Η δομή δεδομένων πυρήνων βρίσκεται στο χώρο μνήμης πυρήνων, το οποίο αναγκάζεται να ‘’ανήκει’’ στο χώρο φυσικής-διεύθυνσης. Ως εκ τούτου, η αποθήκευση των γεγονότων στο χώρο πυρήνων αποτρέπει τη χρήση της σελιδοποίησης της μνήμης, αποφεύγοντας κατά συνέπεια την αυστηρή ποινή απόδοσης που προκαλείται ενδεχομένως από τις αποτυχίες TLB.

Αφ' ετέρου, η αποθήκευση του καταλόγου γεγονότος στο χώρο πυρήνων επιβάλλει έναν σφιχτό περιορισμό στο μέγιστο μέγεθός της. Ο κατάλογος δεν μπορεί να γίνει μεγαλύτερος από 64 KB, το οποίο αντιστοιχεί στα $L_{max}=4096$ γεγονότα. Ο κατάλογος γεγονότος εφαρμόζεται ως κυκλικός buffer και διατίθεται μια για πάντα (για λόγους απόδοσης). Η κυκλική δομή προστατεύει από τις παραβιάσεις μνήμης. Εάν ο αριθμός μη επεξεργασμένων γεγονότων που αποθηκεύονται στον κατάλογο γίνεται μεγαλύτερος από τον αριθμό σχισμών, τα παλαιότερα γεγονότα είναι επανεγγράψιμα. Η απώλεια γεγονότος προκαλεί μια μείωση στην ακρίβεια στον έλεγχο αλλά δεν βλάπτει την κανονική λειτουργία συστημάτων.

Ο περιορισμός μεγέθους του καταλόγου γεγονότος στη μνήμη πυρήνων δεν είναι ένα ενδιαφέρον για το αν τα γεγονότα υποβάλλονται σε επεξεργασία και απορρίπτονται μόλις καταχωρούνται (on-line έλεγχος). Εντούτοις, η απώλεια γεγονότος πρέπει να αποφευχθεί εάν ο παρατηρητής συλλέγει τα μακροχρόνια ίχνη γεγονότος για την εκτός λειτουργίας επεξεργασία. Ο παρατηρητής υποστηρίζει τον εκτός λειτουργίας έλεγχο μέσω ενός απλού μηχανισμού αποτυπώματος που μπορεί να συνοψιστεί ως εξής.
Όποτε ο αριθμός των μη επεξεργασμένων γεγονότων φθάνει σε μια αξία $L_{low}<L_{max}$, ένα wake-up σήμα στέλνεται σε μια αφιερωμένη διαδικασία. Η διαδικασία είναι κανονικά αδρανή, περιμένοντας το wake-up σήμα, κατά συνέπεια δεν αλλάζει την κανονική δραστηριότητα συστημάτων. Όποτε το wake-up σήμα βεβαιώνεται, η διαδικασία γίνεται ενεργή και μπορεί να σχεδιαστεί. Σαφώς, η εκτέλεση αυτής της διαδικασίας αλλάζει την κανονική δραστηριότητα συστημάτων. Εντούτοις, η διαταραχή περιορίζεται από το γεγονός ότι ο κατάλογος υποβάλλεται σε επεξεργασία μόνο όταν είναι σχεδόν πλήρης.

Οι συσκευές που ελέγχονται από το OS μέσω των οδηγών συσκευών ελέγχονται με την παρεμβολή των βασικών κλήσεων λειτουργίας που ενημερώνουν τον κατάλογο γεγονότος στις ρουτίνες οδηγών συσκευών που τρέχουν όποτε το στοιχείο προσεγγίζεται. Ο έλεγχος δεν αλλάζει τη ροή της εκτέλεσης του οδηγού συσκευών, και ασκεί ελάχιστη επίδραση στο χρόνο εκτέλεσης. Στο χρόνο εκκίνησης, ο παρατηρητής εκκινείτε με τη διευκρίνιση ποιοι πόροι πρέπει να ελεγχθούν.

Η CPU και όλα τα τμήματα υλικού που απαιτούνται για τη λειτουργία του (chipset, RAM, ελεγκτές αρτηριών, κλπ.) δεν ελέγχεται μέσω των οδηγών συσκευών. Ευτυχώς, είναι δυνατό να ελεγχθεί η CPU και τα βοηθητικά τμήματά του με την παρατήρηση ότι ο ίδιος ο πυρήνας OS δεν είναι τίποτα άλλο από έναν εκτελέσιμο κώδικα που τρέχει στην CPU. Όποτε ο πυρήνας τρέχει, η CPU είναι ενεργή. Όταν δεν υπάρχει τίποτα να κάνει, ο πυρήνας προγραμματίζει μια πλαστή διαδικασία, ονομαζόμενη Idle στόχος. Ως εκ τούτου, για να ανιχνεύσει την αδράνεια της CPU, είναι ικανοποιητικό να ελεγχθεί ο σχεδιασμός του Idle στόχου.

Η εγκατάσταση οργάνων ελέγχου απαιτεί την ανά-μεταγλώττιση πυρήνων, και υποστηρίζει τον έλεγχο της CPU, του πληκτρολογίου, των σειριακών και παράλληλων θυρών, του PS2 ποντίκι, του σκληρού δίσκου IDE, και του CD-ROM. Κατά τη διάρκεια της εκκίνησης συστημάτων, μια δομή δεδομένων δημιουργείται για κάθε IPM πόρο υπάκουο στο IPM, που περιέχει το όνομα της, τον τύπο της, τις σημαίες διαμόρφωσης της, το μοναδικό προσδιορισμό της, και τις πληροφορίες συγκεκριμένου πόρου (όπως ο τύπος γεγονότων που ελέγχονται). Ο έλεγχος μπορεί να επιτραπεί επιλεκτικά για κάθε πόρο με τη ρύθμιση των αντίστοιχων σημαιών.

Διάφορα πειράματα [14] (τρέξιμο σε ένα HP Omnibook 5500 CT με επεξεργαστή 133-MHz Pentium και 48 MB της RAM) έδειξαν ότι η λειτουργία συστημάτων επιβραδύνεται κατά λιγότερο από 0,38% κατά μέσο όρο, ακόμα και όταν ελέγχονται όλα τα διαθέσιμα τμήματα συστημάτων, παρουσιάζοντας κατά συνέπεια πειστικά στοιχεία για την δικαιοσύνη του ελεγκτή.

5.3 Σχεδιασμός ενσωματωμένων συστημάτων υλικού-λογισμικού για εφαρμογές δικτύων τηλεπικοινωνιών

Ο σχεδιασμός των συστημάτων (System Design), πριν από το χωρισμό του υλικού/λογισμικού, προχωρά ακόμα σε μεγάλο βαθμό κατά τρόπο άτυπο. Οι προδιαγραφές των συστημάτων που παραδίδονται στους σχεδιαστές υλικού και λογισμικού είναι συνήθως μια άτυπη περιγραφή υπό μορφή εγγράφου εκθέσεων
Μόλις προσδιοριστούν τα μέρη υλικού και λογισμικού, τα επίσημα πρότυπα προδιαγραφών και οι τεχνικές σχεδιασμού, υιοθετούνται. Στην περίπτωση του λογισμικού, χρησιμοποιείται η C ή η C++. Στην περίπτωση του υλικού, χρησιμοποιείται μια γλώσσα μεταφοράς καταλόγων όπως η VHDL ή η Verilog. Τα διαφορετικά τμήματα υλικού και λογισμικού ενός πλήρους συστήματος διευκρινίζονται και συντίθενται χωριστά, που εισάγει συχνά τους κακούς συνδυασμούς εφαρμογής, εκτός από τους κακούς συνδυασμούς προδιαγραφών. Οι μικρές αλλαγές στο επίπεδο συστημάτων απαιτούν συχνά τις πολύ ουσιαστικές αλλαγές στα πρότυπα υλικού ή λογισμικού των συστατικών. Ενώ οι αυτοματοποιημένες γεννήτριες του κώδικα υπάρχουν για την παραγωγή του κώδικα μηχανών από τη C ή C++, και τα αυτοματοποιημένα εργαλεία σύνθεσης υλικού υπάρχουν για την παραγωγή των εφαρμογών υλικού από την VHDL ή Verilog. Υπάρχει μια γενική έλλειψη εργαλείων για να γεφυρώσει το χάσμα από τις προδιαγραφές σχεδιασμού των επιπέδων συστημάτων, σε αυτές τις παραδοσιακές μεθόδους σχεδιασμού υλικού και λογισμικού.
Επιπλέον, υπάρχει μια γενική έλλειψη μιας συνεπούς μεθοδολογίας σχεδιασμού για τη δόμηση της ροής σχεδιασμού συστημάτων.

Περιγράφεται μια μεθοδολογία σχεδιασμού και μια ροή σχεδιασμού συστημάτων για τον σχεδιασμό των υβριδικών συστημάτων λογισμικού/υλικού που βρίσκονται χαρακτηριστικά στις εφαρμογές δικτύων τηλεπικοινωνιών. Αυτή η μεθοδολογία είναι βασισμένη στα αποτελέσματα μιας έρευνας και αξιολόγησης μιας πραγματικής βιομηχανικής εφαρμογής συστημάτων για τα ευρυζωνικά δίκτυα βασισμένα στο ATM (Asynchronous Transfer Mode) του Alcatel Bell [89]. Συγκεκριμένα, έχει ερευνηθεί ένας χωρίς σύνδεση διακομιστής (router) για την υπερσύνδεση σε ένα γεωγραφικά διανεμημένο υψηλής ταχύτητας δίκτυο, σε μια σύνδεση του ATM.

Ως συνέπεια αυτής της έρευνας, έχει αναπτυχθεί μια μεθοδολογία σχεδιασμού συστημάτων βασισμένη σε ένα ταυτόχρονο αντικειμενοστραφές πρότυπο προγραμματισμού ως φορμαλιστική συμπεριφορά προδιαγραφών των συστημάτων. Τα εργαλεία προσομοίωσης και διόρθωσης παρέχονται στο ευρύ σώμα των υπαρχόντων εργαλείων ανάπτυξης λογισμικού, για την πρόωρη εννοιολογική επικύρωση των προδιαγραφών των συστημάτων πριν προχωρήσουν στη ροή του σχεδιασμού εφαρμογής.

Για την εφαρμογή, παρέχονται τα αυτοματοποιημένα εργαλεία σχεδιασμού για το πρότυπο επιπέδων συστημάτων, στα παραδοσιακά επίπεδα καταχωρήσεων σχεδιασμού, για την εφαρμογή υλικού και λογισμικού. Η συμβατική C++ [87] και τα πρότυπα VHDL [79], παράγονται για τα μέρη του συστήματος που εφαρμόζονται στο λογισμικό και το υλικό, αντίστοιχα. Οι νέες λειτουργίες σύνθεσης επιπέδων συστημάτων, απαιτούνται για να επιτύχουν αυτές τις αυτοματοποιήσεις. Αυτές οι νέες λειτουργίες είναι μέρος μιας ροής σχεδιασμού συστημάτων. Είναι σημαντικό να σημειωθεί ότι οι παραδοσιακές μέθοδοι σχεδιασμού υλικού και λογισμικού, είναι συνήθως διαθέσιμες στο εμπόριο, χρησιμοποιούνται στη ροή σχεδιασμού μας, με αυτόν τον τρόπο, αποφεύγεται η «εφεύρεση» των νέων λύσεων όπου υπάρχουν ήδη και είναι επαρκείς.

5.3.1 Το πρότυπο προγραμματισμού

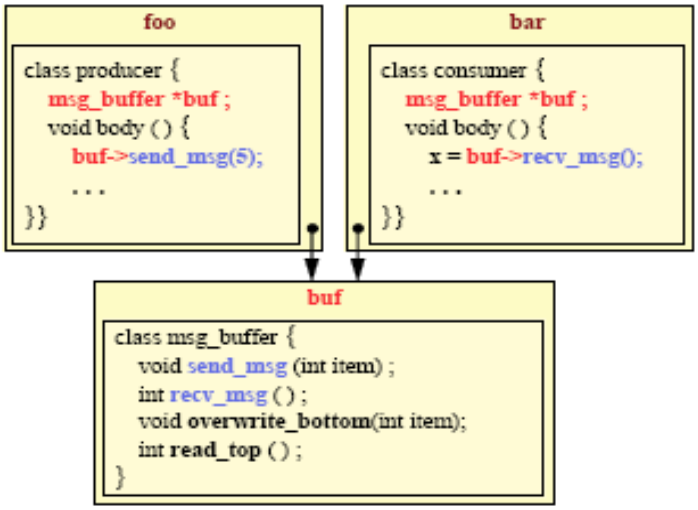
Από την εμπειρία της εφαρμογής μας σε διάφορες βιομηχανικές ευρυζωνικές εφαρμογές των δικτύων τηλεπικοινωνιών, καταλήγει κανείς στο συμπέρασμα ότι η συμπεριφορά αυτών των εφαρμογών χαρακτηρίζεται καλύτερα από τα ελεγχόμενης ροής στοιχεία αλγορίθμων επεξεργασίας. Η ροή ελέγχου από άποψη εκφραστικότητας είναι ουσιαστική. Οι αλγόριθμοι και οι βάσεις δεδομένων πάνω στις οποίες λειτουργούν, είναι συνήθως στενά συνδεδεμένα. Σε πολλές περιπτώσεις, το στοιχείο αντιπροσωπεύει τον πραγματικό εννοιολογικό πυρήνα της εφαρμογής παρά οι αλγόριθμοι.
Λαμβάνοντας υπόψη τις ανωτέρω εκτιμήσεις, έχει ακολουθηθεί μια αντικειμενοστραφή προσέγγιση στον παραλληλισμό ως βάση του προτύπου προγραμματισμού μας, όπως είναι πιστευτό ότι παρέχει ένα φυσικό πρότυπο του συναγωνισμού για τις εξεταζόμενες εφαρμογές: τα αντικείμενα τοποθετούν τις διαδικασίες και (οι μακρινές) κλήσεις λειτουργίας των μελών τοποθετούν τη διά επικοινωνία διαδικασία. Επιπλέον, οι αντικειμενοστραφής λειτουργίες όπως η ενθυλάκωση, η κληρονομιά, και ο πολυμορφισμός είναι ανεκτίμητες σε οποιαδήποτε ανάπτυξη μεγάλης κλίμακας.

Συγκεκριμένα, χρησιμοποιείται ένα ενεργό σημασιολογικό πρότυπο αντικειμένου που συνδυάζει τον παραλληλισμό με τις αντικειμενοστραφείς αρχές. Έχει εφαρμοστεί το πρότυπο προγραμματισμού πάνω από τον ευρέως χρησιμοποιημένο αντικειμενοστραφή προγραμματισμό, γλώσσα C++ [87] με την εισαγωγή των ελάχιστων συντακτικών επεκτάσεων σε αυτό.

Ένα ενεργό αντικείμενο διαφέρει από ένα ενεργητικό αντικείμενο (όπως βρίσκεται στη γλώσσα C++) δεδομένου ότι τοποθετεί πρόσθετα μια διαδικασία, στην κατάσταση και τις διαδικασίες. Η διαδικασία επιτρέπει σε ένα ενεργό αντικείμενο, να εκτελέσει τις λειτουργίες μελών της παράλληλα με τη δραστηριότητα του υπολοίπου του προγράμματος. Σε ένα τυπικό πρόγραμμα, μόνο ένας μικρός αριθμός

αντικειμένων θα είναι ενεργός, παρέχοντας την παράλληλη δομή του προγράμματος. Η πλειοψηφία των αντικειμένων θα είναι τυποποιημένα ενεργητικά αντικείμενα, που υπάρχουν ως μέλη των ενεργών αντικειμένων ή στις βάσεις δεδομένων, διοικούμενες από τα ενεργά αντικείμενα. Αυτό παρέχει την υποστήριξη σε έναν άτεχνο παραλληλισμό έως έναν παραλληλισμό μιας μέσης κλίμακας.

Η επικοινωνία μεταξύ των ενεργών αντικειμένων είναι μέσω της χρήσης (των μακρινών) μελών κλήσεων λειτουργίας. Ένα μέλος κλήσης λειτουργίας σε ένα ενεργό αντικείμενο συμπεριφέρεται με τον ίδιο τρόπο όπως ένα τυποποιημένο μέλος κλήσης λειτουργίας της C++, δεδομένου ότι ο αυτός που καλεί αναστέλλεται μέχρι να τερματιστεί το αποκαλούμενο λειτουργικό μέλος. Αυτή η σημασιολογία του ραντεβού παρέχει το συγχρονισμό μεταξύ των ενεργών αντικειμένων που λειτουργούν ταυτόχρονα. Το επιχείρημα που περνά (και στις δύο κατευθύνσεις) είναι ο μηχανισμός αρχής των μεταφορών στοιχείων. Εκτός από τις βασικές έννοιες των ενεργών αντικειμένων, και την επικοινωνία και το συγχρονισμό μέσω (των μακρινών) κλήσεων λειτουργίας μελών, το πρότυπο προγραμματισμού μας παρέχει επίσης τους μηχανισμούς για τον αμοιβαίο αποκλεισμό βασισμένο στο παράδειγμα οργάνων ελέγχου Hoare [78]. Ένα απλό παράδειγμα του προτύπου είναι διευκρινισμένο στο σχήμα 23.



Σχήμα 23: Ένα απλό παράδειγμα μοντέλου

Δεδομένου ότι έχει εφαρμοστεί το πρότυπό μας πάνω στη C++, η γλώσσα υποστηρίζει επίσης τους μηχανισμούς για (τους αφηρημένους) τύπους στοιχείων, την ανάθεση και τις αριθμητικές διαδικασίες, και τις δηλώσεις ροής ελέγχου όπως if-then-else, for-and while loops.

Επίσης, είναι δυνατό να ασκηθεί δύναμη επάνω στο ευρύ σώμα των υπαρχόντων εργαλείων υποστήριξης σύνταξης και χρόνου εκτέλεσης λογισμικού για την πορεία εφαρμογής λογισμικού, και στα υπάρχοντα περιβάλλοντα εκτέλεσης [73, 75] και τα εργαλεία διόρθωσης [76] για την πρόωρη εννοιολογική επικύρωση της προδιαγραφής συστημάτων.

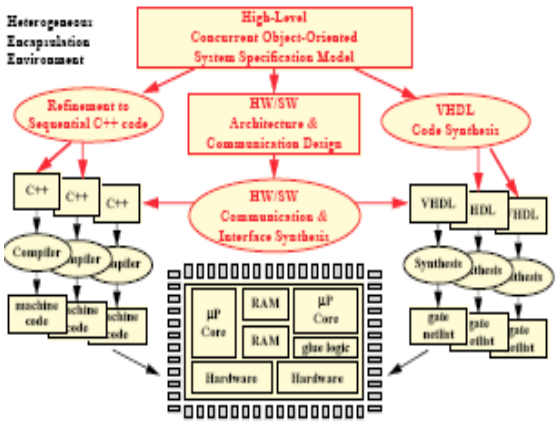
Στην ανάπτυξη του προτύπου, ο στόχος για την όσο το δυνατόν περισσότερη συμβατότητα με την C++ είναι ένα κεντρικό ζήτημα. Δίνεται βαρύτητα στο ευρύ σώμα των υπαρχόντων εργαλείων υποστήριξης σύνταξης και χρόνου εκτέλεσης

λογισμικού για την πορεία εφαρμογής του λογισμικού μας. Αυτό είναι σημαντικό δεδομένου ότι η εφαρμογή λογισμικού αντιπροσωπεύει ένα ουσιαστικό μέρος με πολλές από τις απαιτήσεις των στόχων μας. Επιπλέον, η συμβατότητα με την C++ επιτρέπει στους νέους χρήστες ήδη εξοικειωμένους με την C++ να είναι παραγωγικοί σε ένα πολύ σύντομο χρονικό διάστημα. Επίσης, τα υπάρχοντα περιβάλλοντα εκτέλεσης και τα εργαλεία διόρθωσης μπορούν να προσαρμοστούν εύκολα για την πρόωρη εννοιολογική επικύρωση της προδιαγραφής συστημάτων.

Ενώ τα ταυτόχρονα πρότυπα προγραμματισμού βασισμένα στις αντικειμενοστραφείς αρχές δεν είναι νέα [68, 86, 64, 66, 70, 84], ο στόχος μας εδώ είναι διαφορετικός. Αυτά τα γλωσσικά περιβάλλοντα υποθέτουν τυπικά τα χρόνου εκτέλεσης περιβάλλοντα που μπορούν να υποστηρίξουν τη δυναμική διαχείριση της διαδικασίας, την μετανάστευση-μεταφορά του αντικειμένου μεταξύ των επεξεργαστών, διαφάνεια δικτύων, μεταξύ άλλων χαρακτηριστικών γνωρισμάτων του χρόνου εκτέλεσης. Αντί αυτού, το γλωσσικό περιβάλλον μας, πρέπει να είναι κατάλληλο για τις ενσωματωμένες εφαρμογές όπου η εφαρμογή μπορεί να είναι εν μέρει στο υλικό, και όπου οι εφαρμογές λογισμικού υποτίθεται ότι υποστηρίχθηκαν από τους εξαιρετικά μικρούς σε πραγματικό χρόνο πυρήνες, που προσφέρουν μόνο την ελάχιστη υποστήριξη για το σχεδιασμό της διεργασίας (task) και για την επικοινωνία. Αυτά τα microkernels μπορούν να είναι λιγότερο από 3K byte σε μέγεθος κώδικα, σε αντίθεση με τον διανεμημένο περιβάλλοντα χρόνο εκτέλεσης που είναι πολλά μεγέθη μεγαλύτερος.

5.3.2 Η μεθοδολογία σχεδιασμού

Το πρότυπο προγραμματισμού που περιγράφεται στο προηγούμενο τμήμα παρέχει στους σχεδιαστές συστημάτων έναν φορμαλισμό προδιαγραφών που μπορεί να χρησιμοποιηθεί για να διευκρινίσει το επίπεδο συμπεριφοράς των συστημάτων και των χαρακτηριστικών εφαρμογών δικτύων τηλεπικοινωνιών. Αυτό επιτρέπει στους σχεδιαστές να χρησιμοποιήσουν έναν επίσημο φορμαλισμό προδιαγραφών αντί των άτυπων εγγράφων αναφοράς ως μέσο ανταλλαγής στα χαμηλότερα επίπεδα εφαρμογής. Σε αυτό το τμήμα, περιγράφουμε τα εργαλεία σχεδιασμού που είναι ένα μέρος της ροής σχεδιασμού μας, από τις προδιαγραφές των συστημάτων στην εφαρμογή. Η ροή σχεδιασμού παρουσιάζεται στο 24.



Σχήμα 24: Ροή σχεδιασμού συστήματος Matisse σε περιβάλλον CoWare

5.3.3 Προσομοίωση και διόρθωση

Δεδομένου ότι ο φορμαλισμός των προδιαγραφών μας είναι βασισμένος στις ελάχιστες συντακτικές επεκτάσεις της C++, ένα ευρύ σώμα των υπαρχόντων εργαλείων ανάπτυξης λογισμικού μπορεί να είναι ο κύριος μοχλός λειτουργίας. Για την προσομοίωση, χρησιμοποιούμε τα υπάρχοντα περιβάλλοντα χρόνου εκτέλεσης όπως το PVM [73] και το Nexus [75] που παρέχουν ένα διαφανές στρώμα χρόνου εκτέλεσης επάνω από τα συμβατικά λειτουργικά συστήματα όπως η UNIX για την εκτέλεση των διανεμημένων προγραμμάτων μέσω ενός δικτύου, των ενδεχομένως ετερογενών τερματικών σταθμών. Η υποστήριξη για την προσομοίωση και την πρόωρη επικύρωση στα υψηλά επίπεδα των αφαιρέσεων, είναι ουσιαστική προκειμένου να επιτραπεί η αυξανόμενη παραγωγικότητα και η ανταγωνιστικότητα. Αυξάνοντας το περιβάλλον εκτέλεσης, διορθώνει τα εργαλεία. Τα προγράμματα που περιγράφονται στο μοντέλο μας μπορούν να διορθωθούν από τους ίδιους τους διορθωτές που χρησιμοποιούνται για να διορθώσουν τα τυποποιημένα σε C++ προγράμματα. Αυτοί οι διορθωτές μπορούν να χρησιμοποιηθούν από κοινού με τα χρονικά επίπεδα (layers) εκτέλεσης όπως το PVM και το Nexus με την απαραίτητη προεργασία.

5.3.4 Διαχωρισμός υλικού / λογισμικού

Ο χωρισμός υλικού / λογισμικού οδηγείται από το σχεδιαστή με τη βοήθεια των οδηγιών. Το σύστημα που χωρίζει τις αποφάσεις οδηγείται από τους παράγοντες που δεν είναι συχνά εύκολα ποσοτικά προσδιορισμένοι. Αυτοί οι παράγοντες δεν μπορούν συχνά να διατυπωθούν εύκολα στις απλές εξισώσεις δαπανών που ένα αυτοματοποιημένο εργαλείο μπορεί να βελτιστοποιήσει. Επομένως, πιστεύεται ότι τέτοιες αποφάσεις πρέπει να αφεθούν καλύτερα στο σχεδιαστή που είναι σε καλύτερη θέση να κάνει τέτοιες κρίσεις. Προσπαθείτε αντί αυτού, να αυτοματοποιηθούν τα βήματα καθαρισμού σε χαμηλότερων επιπέδων εργαλεία υλικού και λογισμικού δεδομένου ότι αυτά τα βήματα τείνουν να είναι τα πιο προβληματικά για τους σχεδιαστές.

5.3.5 Εφαρμογή λογισμικού

Για την εφαρμογή λογισμικού, χρησιμοποιούνται οι υπάρχοντες σε C++ μεταγλωττιστές για να παραγάγουν τον κώδικα μηχανών για τον ενσωματωμένο στόχο επεξεργαστών. Εντούτοις, αυτοί οι μεταγλωττιστές στοχεύουν στη σύνταξη του συμβατικού σε C++ κώδικα, ο οποίος είναι διαδοχικός. Για τη διαχείριση των ταυτόχρονων στόχων που τρέχουν στον ίδιο επεξεργαστή, χρησιμοποιούνται οι εξαιρετικά ελαφριοί σε πραγματικό χρόνο πυρήνες μικροϋπολογιστών, οι οποίοι παρέχουν τις ελάχιστες υπηρεσίες για το σχεδιασμό στόχου και της διαδικασίας μέσω της επικοινωνίας. Αυτές οι υπηρεσίες παρέχονται στο επίπεδο κλήσεων λειτουργίας βιβλιοθηκών. Αυτήν την περίοδο, χρησιμοποιείται μια λύση βασισμένη σε έναν εμπορικό σε πραγματικό χρόνο πυρήνα αποκαλούμενο Virtuoso [90,83].

Για να χρησιμοποιήσουν τέτοιους σε πραγματικό χρόνο πυρήνες μικροϋπολογιστών και συμβατούς με την C++ μεταγλωττιστές, τα αυτοματοποιημένα εργαλεία σύνταξης παρέχονται στη ροή σχεδιασμού μας για να μετασχηματίσουν το ταυτόχρονο πρότυπο προδιαγραφών μας σε ένα επίπεδο εισόδου σχεδιασμού κατάλληλο για αυτά τα εργαλεία.

5.3.6 Εφαρμογή υλικού

Αρχίζοντας από το πρότυπο προδιαγραφών συστημάτων, ο βελτίωση στα χαμηλότερα πλαίσια σχεδιασμού υλικού και λογισμικού επιπέδων είναι αυτοματοποιημένος. Οι σχεδιαστές θα είναι σε θέση να αρχίσουν από το ίδιο πρότυπο προδιαγραφών συστημάτων αντί να είναι απαραίτητο να διευκρινιστούν ξανά και να επανερμηνευθούν οι προδιαγραφές των συστημάτων σε χαμηλότερες γλώσσες περιγραφής υλικού επιπέδων όπως η VHDL. Αυτό θα βοηθήσει να αποβάλει το χρόνο και την προσπάθεια που ξοδεύονται στην κατανόηση και την ερμηνεία των άτυπων προδιαγραφών, οι οποίες είναι πάλι συχνά ευαίσθητες στην ασάφεια και τους κακούς συνδυασμούς.

Συγκεκριμένα, χρησιμοποιούμε VHDL ως επίπεδο εισόδου του σχεδιασμού στα τρέχοντα εμπορικά εργαλεία σύνθεσης του υλικού. Αυτή η VHDL παράγεται αυτόματα από τα μέρη των προδιαγραφών των συστημάτων που ορίζονται στο υλικό συνήθειας. Διάφορα εμπορικά εργαλεία σύνθεσης πρώτης γενεάς εμφανίζονται στην αγορά. Τα παραδείγματα περιλαμβάνουν το μεταγλωττιστή από την Synopsys (Behavioral Compiler from Synopsis) και το μεταγλωττιστή Mistral-2 DSP (Compiler) από τον Mendor. Αυτοί οι μεταγλωττιστές περιλαμβάνουν πολλά από τα βασικά αποτελέσματα από την τελευταία δεκαετία υψηλού επιπέδου ερευνητικής σύνθεσης [69]. Πειράματα γίνονται με τον μεταγλωττιστή της Synopsys [88] ως οπίσθιο μέρος στη ροή σχεδιασμού μας, η οποία παρέχει ήδη τις βασικές λειτουργίες όπως την κατασκευή γραφικών παραστάσεων εξάρτησης, το σχεδιασμό, την ανάθεση των πόρων και καταλόγων, και τη σύνθεση πορειών στοιχείων και ελεγκτών.

Παρά τη διαθεσιμότητα αυτών των υψηλού επιπέδου λειτουργιών σύνθεσης, υπάρχει ακόμα ένα σημαντικό χάσμα από το πρότυπο επίπεδο συστημάτων μας, στο πρότυπο VHDL. Το πρότυπο αντικείμενο πρέπει να είναι μεταφρασμένο σε VHDL πρότυπο. Επίσης, οι διατάξεις για τις δυναμικές βάσεις δεδομένων είναι έμφυτες στο αντικειμενοστραφές πρότυπο προγραμματισμού και πρέπει να υποστηριχθούν. Τα εργαλεία σύνθεσης συμπεριφοράς μέχρι σήμερα υποστηρίζουν τους απλούς τύπους στοιχείων (data) όπως τα στατικά αρχεία και οι σειρές. Οι δυναμικές βάσεις δεδομένων δεν υποστηρίζονται. Αντί αυτού, ο σχεδιαστής πρέπει ρητά να τους χαρτογραφήσει από άποψη θέσεις μνήμης και διαδικασίες μνήμης.
Η συμπεριφορά υλικού πρέπει να παρασχεθεί ρητά για να διαχειριστεί τη συμπεριφορά του χρόνου εκτέλεσης της μνήμης. Για τις εφαρμογές, στην περιοχή δικτύων τηλεπικοινωνιών, αυτό μπορεί να είναι ένας σημαντικός περιορισμός. Στο βήμα καθαρισμού μας σε VHDL, αυτοματοποιείται η κατανομή της φυσικής μνήμης, η χαρτογράφηση των δυναμικών βάσεων δεδομένων στις θέσεις μνήμης, η βελτίωση των προσβάσεων των βάσεων δεδομένων στις πρωτόγονες διαδικασίες μνήμης, και η σύνθεση της διαχείρισης της δυναμικής συμπεριφοράς της μνήμης για τη διαχείριση του χρόνου εκτέλεσης σε αυτή.

Με αυτές τις πρόσθετες λειτουργίες της σύνθεσης του υλικού, ο σχεδιαστής μπορεί να αρχίσει από το ίδιο επίπεδο εισόδου σχεδιασμού με τις προδιαγραφές των συστημάτων. Αυτό σημαίνει ότι οι αλλαγές στις προδιαγραφές των σε επίπεδο συστημάτων προσαρμόζονται εύκολα, δεδομένου ότι η σύνταξη στα χαμηλότερα βήματα υλικού σε επίπεδο σχεδιασμού είναι αυτοματοποιημένη.

5.3.7 Σύνθεση επικοινωνίας υλικού / λογισμικού

Τα διαφορετικά μέρη των προδιαγραφών θα οριστούν στο λογισμικό ή το υλικό. Δεδομένου ότι χτίζεται μια οριζόμενη από εφαρμογή λύση, μπορούν να χρησιμοποιηθούν διαφορετικοί προγραμματίσημοι επεξεργαστές λογισμικού από διαφορετικούς προμηθευτές, με συνέπεια μια ετερογενή αρχιτεκτονική υλικού / λογισμικού. Η εφαρμογή τέτοιων ετερογενών ενσωματωμένων αρχιτεκτονικών, εξασφαλίζοντας ότι τα διαφορετικά τμήματα συστημάτων είναι σωστά ενσωματωμένα μαζί, είναι ένας εκπληκτικά δύσκολος στόχος. Οι σχεδιαστές ξοδεύουν ένα τεράστιο χρονικό διάστημα σε αυτόν τον στόχο και εν μέρει στην κατανόηση του πώς να διασυνδέσουν στους διαφορετικούς επεξεργαστές που χρησιμοποιούνται, του πώς να βάλουν τα μέρη υλικού και λογισμικού για να επικοινωνήσουν σωστά, και του πώς να συγχρονίσουν μεταξύ τους τα διαφορετικά συστατικά που λειτουργούν σε διαφορετικό χρόνο. Αυτός είναι ένας ιδιαίτερα λανθασμένος στόχος, που συχνά είναι αρμόδιος για πολλούς χαμηλού επιπέδου κακούς συνδυασμούς εφαρμογής, που οδηγούν σε μια φάση δοκιμής μήκους μετά από την εφαρμογή.

Η προσέγγισή σε αυτό το πρόβλημα είναι βασισμένη σε έναν ενορχηστρωμένο συνδυασμό αρχιτεκτονικών στρατηγικών, σε παραμετρικές βιβλιοθήκες και τα εργαλεία CAD για τους χαμηλού επιπέδου στόχους σχεδιασμού, που είναι λανθασμένοι και χρονοβόροι. Πιο συγκεκριμένα, τα εργαλεία παρέχονται για να εφαρμόσουν τη δομή επικοινωνίας μεταξύ των τμημάτων λογισμικού και υλικού, λύνοντας κατά συνέπεια τα προβλήματα της ολοκλήρωσης των συστημάτων και της εφαρμογής της αρχιτεκτονικής. Η δομή επικοινωνίας θα υποστηρίξει την από σημείο σε σημείο επικοινωνία, για να δείξει την κοινή βάση στη μεταξύ τους επικοινωνία, και την κοινή επικοινωνιακή μνήμη. Επιπλέον, η επικοινωνία και τα εργαλεία σύνθεσης της αρχιτεκτονικής θα υποστηρίξουν επίσης την ενσωμάτωση τους σε πραγματικού χρόνου πυρήνες για την πολλαπλών καθηκόντων υποστήριξη σε έναν επεξεργαστή λογισμικού. Αν και υπάρχουν πολλοί διαθέσιμοι, σε πραγματικό χρόνο, πυρήνες, δεν είναι εύκολα φορητοί στις οριζόμενες από εφαρμογή ενσωματωμένες αρχιτεκτονικές. Για να λύσουν αυτό το πρόβλημα, τα εργαλεία παρέχονται επίσης για να αυτοματοποιήσουν τη χαμηλού επιπέδου διαμόρφωση σε πραγματικό χρόνο πυρήνες, έτσι ώστε να μπορούν να χρησιμοποιηθούν με τις δομές επικοινωνίας και την ενσωματωμένη αρχιτεκτονική που παράγεται. Αυτά τα εργαλεία ενσωματώνονται σε ένα σύστημα αποκαλούμενο *συμφωνία* (Symphony) [82, 83].

5.3.8 Ετερογενής συσχεδιασμός και συμπροσομοίωση

Με βάση ένα κοινό πρότυπο προδιαγραφών συστημάτων για τη σύλληψη της ελεγχόμενης ροής, που εξουσιάζει τα στοιχεία και τις εφαρμογές επεξεργασίας, το χαμηλότερου επίπεδου υλικό και τα πρότυπα προδιαγραφών λογισμικού, παράγονται για την περαιτέρω σύνθεση. Αυτό συμπληρώνει τις ροές του σχεδιασμού λογισμικού υλικού από τα πρότυπα ροής στοιχείων συστημάτων για τις εφαρμογές DSP [80], τα οποία είναι επίσης βασισμένα σε μια κοινή κορυφαία απεικόνιση . Εντούτοις, υπάρχουν καταστάσεις σχεδιασμού όπου θα ήταν καλύτερο να αναμιχθούν τα διαφορετικά επίσημα πρότυπα σχεδιασμού συστημάτων. Αυτό είναι μια εκδήλωση της ετερογένειας.

Ακόμη και για τις ροές σχεδιασμού, όπου ένα κοινό επίσημο μοντέλο σχεδιασμού συστημάτων χρησιμοποιείται, η ετερογένεια θα παρουσιαστεί φυσικά και οι

διαφορετικές τεχνολογίες εφαρμογής θα χρησιμοποιηθούν. Παραδείγματος χάριν, τα πρότυπα C/C++ θα χρησιμοποιηθούν πιθανότατα για τους στόχους λογισμικού, και τα πρότυπα VHDL/Verilog θα χρησιμοποιηθούν πιθανότατα ως χαμηλότερη είσοδος σχεδιασμού επιπέδων για τους στόχους υλικού. Γενικά, υπάρχουν πολλά άμεσα βήματα από τις προδιαγραφές των συστημάτων στην τελική εφαρμογή, και τα διαφορετικά υπολογιστικά πρότυπα απαιτούνται σε διαφορετικά επίπεδα της τροχιάς καθαρισμού.

Κατά συνέπεια, ένα ετερογενές περιβάλλον σχεδιασμού απαιτείται και μπορεί να ενσωματώσει τις πολλαπλές υπολογιστικές περιοχές σε διαφορετικά επίπεδα αφαίρεσης μέσα σε ένα ενιαίο περιβάλλον. Αναπτύσσεται αυτήν την περίοδο ένα τέτοιο περιβάλλον αποκαλούμενο CoWare [71]. Το Ptolemy [68] είναι ένα άλλο τέτοιο περιβάλλον. Ένα βασικό πρόβλημα είναι η ενθυλάκωση των διαφορετικών εργαλείων σχεδιασμού. Παραδείγματος χάριν, το σύστημα Matisse που περιγράφεται σε αυτή την εργασία, το συμφωνικό σύστημα για την επικοινωνία υλικού/λογισμικού, ο μεταγλωττιστής Synopsys, και οι C++ μεταγλωττιστές για τους διαφορετικούς επεξεργαστές που χρησιμοποιούνται, είναι όλα ενσωματωμένα στο περιβάλλον CoWare.

6 Σύγκριση των Επίσημων Γλωσσών

Προκειμένου να συγκριθούν αυτές οι διαφορετικές επίσημες γλώσσες, είναι σημαντικό να περιγραφούν αρχικά τα κριτήρια που θα χρησιμοποιηθούν. Τα κριτήρια ομαδοποιούνται σε δύο κατηγορίες. Η σειρά των κριτήριων που εμφανίζονται παρακάτω μέσα σε κάθε ομάδα είναι αυθαίρετη. Τα θεμελιώδη κριτήρια επιλογής είναι εκείνα που πιστεύουμε ότι είναι πολύ σημαντικά για μια γλώσσα ώστε να είναι επιτυχής, ειδικά στην ανάπτυξη τηλεπικοινωνιακού λογισμικού. Τα σημαντικά κριτήρια επιλογής είναι εκείνα που πιστεύουμε ότι είναι γενικά πολύ σημαντικά, αλλά για τα οποία βρήκαμε τα λιγότερα στοιχεία. Ο στόχος αυτών των κριτηρίων είναι να βοηθήσει στην αξιολόγηση της καταλληλότητας των γλωσσών για την σχεδίαση τηλεπικοινωνιακών συστημάτων.

6.1 Θεμελιώδη κριτήρια επιλογής

Δυνατότητα εφαρμογής

Μπορεί η τεχνολογία να περιγράψει τα πραγματικά παγκόσμια προβλήματα και τις λύσεις κατά τρόπο φυσικό και απλό; Εάν η τεχνολογία κάνει υποθέσεις για το περιβάλλον, είναι λογικές ή παραπλανητικές; Είναι η τεχνολογία συμβατή με τη φυσική πραγματικότητα του προοριζόμενου συστήματος;

Υλοποιησιμότητα: **Το λογικό μονοπάτι στον κώδικα**

Μπορούν οι προδιαγραφές να βελτιωθούν εύκολα ή να μεταφραστούν σε μια εφαρμογή που είναι συμβατή με το υπόλοιπο του συστήματος; (Παραδείγματος χάριν, εάν ο κώδικας παράγεται αυτόματα, είναι αυτός στην ίδια γλώσσα/διάλεκτο με το υπόλοιπο του συστήματος;) Πόσο δύσκολη είναι αυτή η μετάφραση;

Δυνατότητα δοκιμής/προσομοίωση

Μπορούν οι προδιαγραφές να χρησιμοποιηθούν για να εξετάσουν την εγκυρότητα του σχεδιασμού; Υπάρχει δυνατότητα εξομοίωσης των προδιαγραφών με το σχεδιαστικό μοντέλο ώστε να ελεγχθεί ότι η συμπεριφορά του τελικού συστήματος και οι απαιτήσεις / προδιαγραφές είναι συμβατές;

Συντήρηση (maintainability)

Μπορούν οι προδιαγραφές να χρησιμοποιηθούν ως αφετηρία για τις δραστηριότητες συντήρησης; Είναι εύκολο να γίνουν τροποποιήσεις στις προδιαγραφές;

Συναρμολογησιμότητα (modularity)

Μπορεί το σχεδιαστικό μοντέλο να αποσυντεθεί σε μικρότερα μέρη; Είναι πιθανό να γίνουν εννοιολογικά μικρές προσθήκες, αλλαγές, και γενικεύσεις στις προδιαγραφές χωρίς επίσημη γραφή ξανά;

Επίπεδο αφαίρεσης (level of abstraction)

Πόσο στενά και εκφραστικά μπορούν να περιγραφούν τα αντικείμενα στις προδιαγραφές από την άποψη του χρήστη, τα αντικείμενα, τον εξοπλισμό και το περιβάλλον λειτουργίας;

Αξιοπιστία (soundness)

Η γλώσσα και τα εργαλεία παρέχουν την υποστήριξη για την ανίχνευση των ασυνεπειών και των ασαφειών στις προδιαγραφές; Η γλώσσα έχει μια ακριβώς καθορισμένη σημασιολογία;

Επαλήθευση (verifiability)
Είναι δυνατό να καταδειχθεί τυπικά ότι οι προδιαγραφές ή ένα πρόγραμμα ικανοποιεί τις ιδιότητες που δηλώνονται στο ίδιο ή πιο υψηλό επίπεδο της αφαίρεσης;

Ασφάλεια χρόνου εκτέλεσης (run-time safety)
Εάν η γλώσσα υποστηρίζει την αυτοματοποιημένη παραγωγή κώδικα, η προκύπτουσα εφαρμογή μειώνει υπό τους απροσδόκητους όρους του χρόνου εκτέλεσης;

Ωριμότητα εργαλείων (tools maturity)
Πόσο ώριμα είναι τα εργαλεία από την άποψη των διαθέσιμων εγκαταστάσεων, της ποιότητας, της ποσότητας και της ποιότητας της κατάρτισης που είναι διαθέσιμη, της υποστήριξης που είναι διαθέσιμη στον εξελίξιμο χρόνο, στο χρόνο δοκιμής και στον τρέχον χρόνο, του βασικού μεγέθους του χρήστη, της βιομηχανικής χρήσης, της ορμής και της ώθησης που κέρδισαν στις βιομηχανικές εγκαταστάσεις;

Πίνακας 2: Σύγκριση γλωσσών

Κριτήρια	Estelle	LOTOS	SDL	UML	Top-Down	Bottom-Up
Δυνατότητα Εφαρμογής	+	+	+	+	+	+
Δυνατότητα Δοκιμής / Προσομοίωσης	+	+	+	+	+	+
Δυνατότητα Ελέγχου	0	0	+	+	0	0
Δυνατότητα Συντήρησης	0	-	0	0	-	-
Δυνατότητα Διαμόρφωσης	0	0	0	+	-	-
Επίπεδο Αφαίρεσης	-	0	0	0	-	-
Ασφάλεια του χρόνου εκτέλεσης	0	0	0	0	0	0
Ωριμότητα εργαλείων	-	0	+	+	-	-
Καμπύλη μάθησης	0	0	+	+	-	-
Ωριμότητα γλώσσας	0	+	+	+	0	0
Διαμόρφωση στοιχείων	0	0	0	0	0	0
Πειθαρχία	0	0	0	0	-	-
Δυνατότητα Σαφήνειας	0	+	0	+	-	-
Δυνατότητα Επαλήθευσης	0	+	+	+	0	0

+ = δυνατό, - = αδύναμο, 0 = επαρκής

6.2 Σημαντικά κριτήρια επιλογής

Looseness
Η γλώσσα επιτρέπει την ημιτέλεια ή την μη-καθοριστικότητα στις προδιαγραφές;

Καμπύλη εκμάθησης (learning curve)

Μπορεί ένας απόλυτα νέος χρήστης να μάθει γρήγορα τις έννοιες και τις τεχνικές για να χρησιμοποιήσει παραγωγικά τη γλώσσα;

Μοντελοποίηση στοιχείων (data modeling)
Η γλώσσα παρέχει τις δομές για να περιγράψει την απεικόνιση , τις σχέσεις ή/και τις αφαιρέσεις στοιχείων; Τους παρέχει με έναν φυσικό, ενσωματωμένο τρόπο;

Πειθαρχία (discipline)
Η γλώσσα επιβάλλει αυστηρούς κανόνες στους χρήστες για να γράψουν προγράμματα;

Για την καλύτερη κατανόηση λοιπόν της συγκριτικής μελέτης των τεχνικών σχεδίασης με κριτήριο την απόδοση των χαρακτηριστικών των τηλεπικοινωνιακών συστημάτων, ακολουθούν δυο πίνακες κατάταξης των τεχνικών που έχουν αναλυθεί ως τώρα. Στον πρώτο πίνακα φαίνεται ποια τεχνική καλύπτει ποια χαρακτηριστικά και στον δεύτερο πίνακα φαίνεται ποια τεχνική καλύπτει ποιες απαιτήσεις.

Χαρακτηριστικά μοντέλων Τεχνικές	Εύχρηστο	Φιλικό προς το χρήστη περιβάλλον σχεδιασμού	Δυνατότητα γραφικής / λεκτικής αναπαράστασης των δομών : επίπεδο ευκολίας περιγραφής	Αυτόματη παραγωγή κώδικα	Υποστήριξη δομών για την αναπαράσταση δυναμικής συμπεριφοράς (εξέλιξη στο χρόνο)	Υποστήριξη δομών για την αναπαράσταση χρονικών περιορισμών	Υποστήριξη δομών για την αναπαράσταση αλγοριθμικής λογικής	Υποστήριξη δομών για την αναπαράσταση κατανεμημένου προγραμματισμού	Δυνατότητα ελέγχου αξιοπιστίας σχεδιασμού	Επίπεδο λεπτομέρειας σχεδιασμού (υψηλό / συστήματος, μέσο / SW, χαμηλό / HW)	Λογική σχεδίασης (αλγοριθμική / λειτουργίες ή αντικειμενοστραφής)	Δυνατότητα τροποποίησης του σχεδίου σε ενδιάμεσα στάδια
Estelle	-	-	Λεκτική	X	-	-	X	-	-	Χαμηλό/HW	Αλγοριθμική	-
LOTOS	-	-	Γραφικά	X	-	X	X	-	-	Χαμηλό/HW	Αλγοριθμική	-
SDL	X	X	Γραφική	-	X	X	X	-	-	Υψηλό/συστήματος	Αλγοριθμική / λειτουργίες	X
UML	X	X	Γραφική και λεκτική	X	X	X	X	X	X	Όλα	Αντικειμενοστραφής	X
Bottom up	-	-	Λεκτική	-	-	-	-	-	-	Μέσο/SW	Αλγοριθμική	-
Top down	-	-	Λεκτική	-	-	-	-	-	-	Υψηλό/συστήματος	Αλγοριθμική	-

Χαρακτηριστικά μοντέλων Τεχνικές	Ακολουθιακή Λογική	Ανάλυση αναπαράστασης δυναμικής συμπεριφοράς	Αναπαράσταση αυξημένης αλγοριθμικής πολυπλοκότητας	Ιεραρχική σχεδίαση	Αναπαράσταση περιορισμών της εφαρμογής	Δυνατότητα εφαρμογής τεχνικών διαχείρισης ενέργειας (δυνατότητα αναπαράστασης στο σχέδιο)
Estelle	X	-	-	-	-	-
LOTOS	-	-	-	-	-	-
SDL	X	-	X	X	X	X
UML	-	X	X	X	X	X
Bottom up	X	-	X	-	-	-
Top down	X	-	-	X	-	X

Κοιτώντας τους παραπάνω πίνακες παρατηρούμε ότι: η Estelle δεν είναι από τις εύχρηστες γλώσσες μιας και η σύνταξη του στυλ PASCAL μπορεί να είναι γνωστή στους προγραμματιστές, αλλά δεν είναι τόσο εξοικειωμένη στους συντάκτες απαιτήσεων. Επίσης, υπάρχουν λίγα εργαλεία για την ανάλυση των προδιαγραφών της Estelle. Η Estelle έχει τη δυνατότητα της λεκτικής αναπαράστασης των δομών και τη δυνατότητα της αυτόματης παραγωγής κώδικα, μπορεί δηλαδή να περιγράψει ένα τηλεπικοινωνιακό σύστημα με κώδικα χρησιμοποιώντας έναν ή περισσότερους τύπους προτάσεων (from, when, provided, priority, delay) και όχι γραφικά, τη δυνατότητα υποστήριξης των δομών για την αναπαράσταση αλγοριθμικής λογικής. . Τέλος, παρουσιάζει ακολουθιακή λογική γιατί οι λειτουργίες εκτελούνται η μια μετά την άλλη.
Η LOTOS δεν είναι εύχρηστη και φιλική προς το χρήστη διότι το στυλ της γραφικής παράστασης που πρέπει να διδαχθεί είναι ασυνήθιστο (π.χ., οι επαναλαμβανόμενοι ορισμοί διαδικασίας). Η LOTOS έχει τη δυνατότητα της γραφικής, και όχι της λεκτικής αναπαράστασης των δομών μιας και δεν έχει κανένα γλωσσικό στοιχείο για την περιγραφή των τμημάτων επεξεργασίας και των καναλιών ενός συστήματος αλλά μπορεί να αναπαραστήσει ένα τηλεπικοινωνιακό σύστημα με γραφική παράσταση. (Επίσης, έχει τη δυνατότητα αυτόματης παραγωγής κώδικα, τη δυνατότητα υποστήριξης των δομών για την αναπαράσταση αλγοριθμικής λογικής, της υποστήριξης δομών για την αναπαράσταση χρονικών περιορισμών
Η SDL είναι εύχρηστη και διαθέτει ένα φιλικό περιβάλλον σχεδιασμού προς τον χρήστη. Η SDL έχει τη δυνατότητα της γραφικής, και όχι της λεκτικής αναπαράστασης των δομών διότι χρησιμοποιεί γραφική απεικόνιση των διαγραμμάτων ροής προκειμένου να παρουσιάσει τις μεταβάσεις και τις διασυνδέσεις μεταξύ των διαδικασιών. Υπάρχουν διάφορα ειδικά γραφικά σύμβολα στην SDL. Επίσης, έχει τη δυνατότητα εξέλιξης στο χρόνο μιας και κάθε διάγραμμα περιέχει μια (timeline) υπόδειξη ως προς το χρόνο για κάθε συμμετέχουσα διαδικασία. Τέλος, έχει ακολουθιακή λογική γιατί οι λειτουργίες εκτελούνται η μια μετά την άλλη, παρουσιάζει την αναπαράσταση αυξημένης αλγοριθμικής πολυπλοκότητας μιας και ο αλγόριθμος είναι γρήγορος και πιο αποτελεσματικός, ιεραρχική σχεδίαση διότι ξεκινά από τα γενικά και καταλήγει στην λεπτομέρεια, αναπαράσταση περιορισμών εφαρμογής διότι η εφαρμογή έχει περιορισμούς (πχ μια γραμμή δίνει συγκεκριμένο bandwidth), και τη δυνατότητα εφαρμογής τεχνικών διαχείρισης ενέργειας καθώς βοηθά στην βελτιστοποίηση κατανάλωσης της ενέργειας.
Η UML είναι εύχρηστη και διαθέτει ένα φιλικό περιβάλλον σχεδιασμού προς τον χρήστη. Η UML έχει τη δυνατότητα γραφικής και λεκτικής αναπαράστασης των δομών, υψηλό, μέσο και χαμηλό επίπεδο λεπτομέρειας σχεδιασμού καθώς είναι μια γλώσσα μοντελοποίησης (ένα σύνολο από διαγράμματα, όπως: τα διαγράμματα περιπτώσεων χρήσης, τα διαγράμματα δομής, τα διαγράμματα συμπεριφοράς και τα διαγράμματα δομής υλοποίησης), για τη διαμόρφωση προδιαγραφών και την τεκμηρίωση συστημάτων που βασίζονται σε λογισμικό σε υψηλό επίπεδο αλλά και μια γλώσσα με τις απαραίτητες δομές για να αναπαράσταση πηγαίου κώδικα χαμηλού επιπέδου σχεδίασης. Η UML παρέχει αντικειμενοστραφή λογική σχεδίασης γιατί στοχεύει στο σχεδιασμό αντικειμενοστραφών συστημάτων. Έχει τη δυνατότητα της αυτόματης παραγωγής κώδικα καθότι στη φάση του προγραμματισμού οι κλάσεις από την σχεδιαστική φάση μετατρέπονται σε πραγματικό κώδικα στην αντικειμενοστραφή γλώσσα προγραμματισμού. Στην UML η ομαδοποίηση δεδομένων και λειτουργιών μέσα στις κλάσεις, καθιστούν τις αδυναμίες της

δομημένης ανάλυσης και σχεδίασης περισσότερο αντιστοίχισες με οντότητες του πραγματικού κόσμου. Υποστηρίζει τις δομές για την αναπαράσταση δυναμικής συμπεριφοράς γιατί υπάρχει εξέλιξη στο χρόνο. Τέλος, παρουσιάζει ανάλυση αναπαράστασης δυναμικής συμπεριφοράς καθότι παρουσιάζει μεταβολή στο χρόνο, παρουσιάζει την αναπαράσταση αυξημένης αλγοριθμικής πολυπλοκότητας μιας και ο αλγόριθμος είναι γρήγορος και πιο αποτελεσματικός, ιεραρχική σχεδίαση διότι ξεκινά από τα γενικά και καταλήγει στην λεπτομέρεια, αναπαράσταση περιορισμών εφαρμογής διότι η εφαρμογή έχει περιορισμούς (πχ μια γραμμή δίνει συγκεκριμένο bandwidth), και τη δυνατότητα εφαρμογής τεχνικών διαχείρισης ενέργειας καθώς βοηθά στην βελτιστοποίηση κατανάλωσης της ενέργειας.

Η bottom-up προσέγγιση έχει τη δυνατότητα της λεκτικής αναπαράστασης των δομών, έχει μέσο επίπεδο λεπτομέρειας σχεδιασμού και αλγοριθμική λογική σχεδίασης. Επίσης, έχει ακολουθιακή λογική γιατί οι λειτουργίες εκτελούνται η μια μετά την άλλη, και αναπαράσταση αυξημένης αλγοριθμικής πολυπλοκότητας μιας και ο αλγόριθμος είναι γρήγορος και πιο αποτελεσματικός.

Ο top-down σχεδιασμός έχει τη δυνατότητα της λεκτικής αναπαράστασης των δομών, έχει υψηλό επίπεδο λεπτομέρειας σχεδιασμού γιατί η Top-down αποσύνθεση απαιτεί τις σημαντικότερες υψηλότατου επιπέδου απαιτήσεις και λειτουργίες συστημάτων, και έπειτα τις σπάει διαδοχικά έως ότου μπορούν να σχεδιαστούν οι συγκεκριμένες λειτουργικές ενότητες, και αλγοριθμική λογική σχεδίασης. Επίσης, έχει ακολουθιακή λογική γιατί οι λειτουργίες εκτελούνται η μια μετά την άλλη, ιεραρχική σχεδίαση διότι ξεκινά από τα γενικό και καταλήγει στη λεπτομέρεια και τέλος έχει τη δυνατότητα εφαρμογής τεχνικών διαχείρισης ενέργειας καθώς βοηθά στην βελτιστοποίηση κατανάλωσης της ενέργειας.
Συνοψίζοντας όλα τα παραπάνω, οδηγούμαστε στο συμπέρασμα ότι η πιο πλήρης γλώσσα για τον σχεδιασμό ενός τηλεπικοινωνιακού συστήματος είναι η *UML*.

7 Γενικά Συμπεράσματα

Συνοψίζοντας, γίνεται αντιληπτό ότι λόγω των χαρακτηριστικών και των απαιτήσεων ενός συστήματος τηλεπικοινωνιών, μελετήθηκαν δομές μοντελοποίησης που θα διευκολύνουν την σχεδίαση και ανάπτυξή τους. Στην συνέχεια, αναλύθηκαν μεθοδολογίες για την επίλυση διαφόρων τύπων προβλημάτων κατά την ανάπτυξη του λογισμικού. Τον λόγο έχουν οι top-down & bottom-up τεχνικές σχεδιασμού με καταλληλότερο τελικά τον top-down σχεδιασμό, ο οποίος ξεκινά από το υψηλό επίπεδο συστήματος καταλήγοντας στις συγκεκριμένες λειτουργικές ενότητες, εστιάζει κατά κύριο λόγο στην απαίτηση των χρηστών και τη γενική φύση του προβλήματος, καθώς και έχει οδηγήσει στην εξέλιξη του δομημένου σχεδιασμού. Ενώ ο δομημένος σχεδιασμός είναι μια τεκμηριωμένη τεχνική, πιο εξελίξιμος και κατάλληλος είναι ο αντικειμενοστραφής προγραμματισμός, ο οποίος δημιουργεί ένα πρότυπο του πραγματικού κόσμου το οποίο και χαρτογραφεί στη δομή του λογισμικού.
Επίσης μια μεθοδολογία κατάλληλη για την μείωση της κατανάλωσης της ισχύος στα ηλεκτρονικά συστήματα είναι η μέθοδος DPM (η διαχείριση δυναμικής ισχύος), η οποία καλύπτει ένα σύνολο τεχνικών που επιτυγχάνουν τον υπολογισμό της

απόδοσης της ενέργειας, με το να μειώσουν την απόδοση των στοιχείων των συστημάτων κατά την παραμονή τους σε κατάσταση αναμονής (Idle).

Τέλος, όσον αφορά στις γλώσσες μοντελοποίησης, από τις οποίες μετά από την σύγκριση τους με βάση κάποια σημαντικά κριτήρια καταλήξαμε στο συμπέρασμα ότι η UML είναι η καταλληλότερη και η πιο πλήρης γλώσσα για τον σχεδιασμό ενός τηλεπικοινωνιακού συστήματος με την δυνατότητα λεκτικής και γραφικής αναπαράστασης δομών σε υψηλό, μέσο και χαμηλό επίπεδο λεπτομερειών σχεδιασμού.

8 ΒΙΒΛΙΟΓΡΑΦΙΑ – ΑΝΑΦΟΡΕΣ

1. Hardware/software Co-Design study report", MCC/OMI Report, 1997.
2. De Micheli Giovanni, M. Sami 'HardwardSoftware Codesign'. Ed Kluwer Academic Publisher 95.
3. E. Martin, 0. Sentieys, H. Dubois, J.L. Philippe, 'GAUT, An Architectural Synthesis Tool for Dedicated Signal Processors', In proceedings of EURO-DAC 93.
4. S. Gailhard , N. Julien, J.-Ph. Diguet, E. Martin: Methods to Transform Easily Classical Architectural Synthesis Tools to Low Power Ones. thIE EE-Great Lakes Symposium on V a l , Louisiana, USA, February 19-21 1998.
5. A. Baganne, J.L Philippe, E. Martin " A Formal Technique for Hardware Interface design." IEEE Transactions On Circuits And Systems-Il: Analog And Digital Signal Processing, 'Vol. 45, NO. 5, May 1998.
6. C. Jego, E. Casseau, E. Martin " MOCAT: Mod6lisation et Outils pour la Conception Architecturale en T616communications". ITR Study Report, UBS Umversity, Lorient, France, 1998.
7. A. Baganne, J.L Philippe E. Martin "A CO-design Methodology for Telecommunication Systems: A Case Study of an Acoustic Echo Canceller". Proceedings of the IEEE International Workshop on Signal Processing Systems . (SPS'97), Leicester, UK, Nov 97.
8. Almagest -Volume 1, " Ptolemy 0.7 Users manual", http://ptolemv.eecs.berkelev.edu
9. H. Thomas, J.Ph. diguet, J. L. Philippe "Estimation et mktriques pour l'ad6quation application systi!me" Internal Report, Lester Lab, Lorient, France
10. J. Lorch and A. Smith, “Software strategies for portable computer energy management,” IEEE Personal Commun., vol. 5, pp. 60–73, June 1998.
11. L. Benini and G. De Micheli, Dynamic Power Management: Design Techniques and CAD Tools. Norwell, MA: Kluwer, 1998.
12. SA-1100 Microprocessor Technical Reference Manual, Intel, 1998.
13. 2.5-Inch Travelstar Hard Disk Drive, IBM, 1998.
14. L. Benini, R. Hodgson, and P. Siegel, “System-Level power estimation and optimization,” in Int. Symp. Low Power Architecture and Design, Aug. 1998, pp. 173–178.
15. L. Benini, A. Bogliolo, S. Cavallucci, and B. Riccó, “Monitoring system activity for OS-directed dynamic power management,” in Int. Symp. Low Power Architecture and Design, Aug. 1998, pp. 185–190.
16. “Advanced micro devices,” in AM29SLxxx Low-Voltage Flash Memories, 1998.
17. M. Alidina, J. Monteiro, S. Devadas, A. Ghosh, and M. Papaefthymiou, “Precomputation-based sequential logic optimization for low power,” IEEE Trans. VLSI Syst., vol. 2, pp. 426–436, Dec. 1994.
18. S. Malik, V. Tiwari, and P. Ashar, “Guarded evaluation: Pushing power management to logic synthesis/design,” in Int. Symp. Low Power Design, Apr. 1995, pp. 221–226.
19. L. Benini and G. De Micheli, “Transformation and synthesis of FSM’s for low power gated clock implementation,” IEEE Trans. Computer-Aided Design, vol. 15, pp. 630–643, June 1996.

20. F. Theeuwen and E. Seelen, “Power reduction through clock gating by symbolic manipulation,” in Symp. Logic and Architecture Design, Dec. 1996, pp. 184–191.
21. M. Ohnishi et al., “A method of redundant clocking detection and power reduction at RT-level design,” in Int. Symp. Low Power Electronics and Design, Aug. 1997, pp. 131–136.
22. J. Oh and M. Pedram, “Gated clock routing minimizing the switched capacitance,” in Design Automation and Test in Europe Conf., Feb. 1998, pp. 692–697.
23. S. Gary et al., “PowerPC 603, a microprocessor for portable computers,” IEEE Design & Test of Computers, vol. 11, pp. 14–23, 1994.
24. G. Debnath, K. Debnath, and R. Fernando, “The pentium processor- 90/100, microarchitecture and low-power circuit design,” in Int. Conf. VLSI Design, Jan. 1995, pp. 185–190.
25. S. Furber, ARM System Architecture. Reading, MA: Addison-Wesley, 1997.
26. M. Gowan, L. Biro, and D. Jackson, “Power considerations in the design of the alpha 21 264 microprocessor,” in Design Automation Conf., June 1998, pp. 726–731.
27. E. Harris et al., “Technology directions for portable computers,” Proc. IEEE, vol. 83, pp. 636–657, Apr. 1996.
28. M. Stemm and R. Katz, “Measuring and reducing energy consumption of network interfaces in hand-held devices,” IEICE Trans. Commun., vol. E80-B, pp. 1125–1131, Aug. 1997.
29. Microsoft. (1997) On now: The evolution of the PC platform. [Online] http://www.microsoft.com/hwdev/pcfuture/OnNOW.HTM.
30. Intel, Microsoft, and Toshiba. (1996) Advanced configuration and power interface specification. [Online] http://www.intel.com/ial/powermgm/specs.html.
31. N. Bambos, “Toward power-sensitive network architectures in wireless communications: Concepts, issues and design aspects,” IEEE Personal Commun., vol. 5, pp. 50–59, June 1998.
32. J. Rulnick and N. Bambos, “Mobile power management for wireless communication networks,” Wireless Networks, vol. 3, no. 1, pp. 3–14, Jan. 1997.
33. K. Sivalingham et al., “Low-power access protocols based on scheduling for wireless and mobile ATM networks,” in Int. Conf. Universal Personal Communications, Oct. 1997, pp. 429–433.
34. M. Zorzi and R. Rao, “Energy-constrained error control for wireless channels,” IEEE Personal Commun., vol. 4, pp. 27–33, Dec. 1997.
35. B. Mangione-Smith, “Low-power communication protocols: Paging and beyond,” in IEEE Symp. Low-Power Electronics, Apr. 1995, pp. 8–11.
36. P. Krishnan, P. Long, and J. Vitter, “Adaptive disk spindown via optimal rent-to-buy in probabilistic environments,” in Int. Conf. Machine Learning, July 1995, pp. 322–330.
37. D. Helmbold, D. Long, and E. Sherrod, “Dynamic disk spin-down technique for mobile computing,” in IEEE Conf. Mobile Computing, Nov. 1996, pp. 130–142.
38. F. Douglis, P. Krishnan, and B. Bershad, “Adaptive disk spin-down policies for mobile computers,” in 2nd USENIX Symp. Mobile and Location-Independent Computing, Apr. 1995, pp. 121–137.
39. R. Golding, P. Bosh, and J. Wilkes, “Idleness is not sloth,” HP Laboratories Tech. Rep. HPL-96-140, 1996.

40. A. Karlin, M. Manasse, L. McGeoch, and S. Owicki, “Competitive randomized algorithms for nonuniform problems,” Algorithmica, vol. 11, no. 6, pp. 542–571, June 1994.
41. M. Srivastava, A. Chandrakasan, and R. Brodersen, “Predictive system shutdown and other architectural techniques for energy efficient programmable computation,” IEEE Trans. VLSI Syst., vol. 4, pp. 42–55, Mar. 1996.
42. C.-H. Hwang and A. Wu, “A predictive system shutdown method for energy saving of event-driven computation,” in Int. Conf. Computer-Aided Design, Nov. 1997, pp. 28–32.
43. Y. Lu and G. De Micheli, “Adaptive hard disk power management on personal computers,” in Great Lakes Symp. VLSI, Feb. 1999, pp. 50–53.
44. Y. Lu, T. ˇSimunic´, and G. De Micheli, “Software controlled power management,” in Hardware–Software Codesign Symp., May 1999, pp. 151–161.
45. Y. Lu, E. Y. Chung, T. ˇSimunic´, L. Benini, and G. De Micheli, “Quantitative comparison of power management algorithms,” in DATE, Proc. Design Automation and Test in Europe, Mar. 2000.
46. T. ˇSimunic´, L. Benini, and G. De Micheli, “Event-driven power management of portable systems,” in ISSS, Proc. Int. Symp. System Synthesis, Nov. 1999, pp. 18–23.
47. T. ˇSimunic´, L. Benini, P. Glynn, and G. De Micheli, “Dynamic power management of portable systems using semi-Markov decison processes,” in DATE, Proc. Design Automation and Test in Europe, Mar. 2000.
48. L. Benini, G. Paleologo, A. Bogliolo, and G. De Micheli, “Policy optimization for dynamic power management,” IEEE Trans. Computer-Aided Design, vol. 18, pp. 813–33, June 1999.
49. Q. Qiu and M. Pedram, “Dynamic power management based on continuous-time Markov decision processes,” in Design Automation Conf., June 1999, pp. 555–561.
50. E. Chung, L. Benini, A. Bogliolo, and G. De Micheli, “Dynamic power management for nonstationary service requests,” in Design and Test in Europe Conf., Mar. 1999, pp. 77–81.
51. S. Ross, Introduction to Probability Models, 6th ed. New York: Academic, 1997.
52. M. Puterman, Finite Markov Decision Processes. New York: Wiley, 1994.
53. K. Usami et al., “Automated low-power technique exploiting multiple supply voltages applied to a media processor,” IEEE J. Solid-State Circuits, vol. 33, pp. 463–472, Mar. 1998.
54. L. Nielsen, C. Niessen, J. Sparso, and K. van Berkel, “Low-power operation using self-timed circuits and adaptive scaling of supply voltage,” IEEE Trans. VLSI Syst., vol. 2, pp. 425–435, Dec. 1994.
55. A. Chandrakasan, V. Gutnik, and T. Xanthopoulos, “Data driven signal processing: An approach for energy efficient computing,” in Int. Symp. Low Power Electronics and Design, Aug. 1996, pp. 347–352.
56. H. Kapadia, G. De Micheli, and L. Benini, “Reducing switching activity on datapath buses with control-signal gating,” in Custom Integrated Circuit Conf., May 1998, pp. 589–592.
57. K. Suzuki et al., “A 300 MIPS/W RISC core processor with variable supply-voltage scheme in variable threshold-voltage CMOS,” in Custom Integrated Circuits Conf., May 1997, pp. 587–590.

58. K. Usami et al., “Design methodology of ultra low-power MPEG4 codec core exploiting voltage scaling techniques,” in Design Automation Conf., June 1998, pp. 483–488.
59. A. Stratakos, S. Sanders, and R. Brodersen, “A low-voltage CMOS dc–dc converter for a portable battery-operated system,” in Power Electronics Specialists Conf., June 1994, pp. 619–626.
60. G. Wei and M. Horowitz, “A low power switching power supply for self-clocked systems,” in Int. Symp. Low Power Electronics and Design, Aug. 1996, pp. 313–317.
61. W. Namgoong, M. Yu, and T. Meng, “A high-efficiency variable-voltage CMOS dynamic dc–dc switching regulator,” in Int. Solid-State Circuits Conf., Feb. 1997, pp. 380–381.
62. V. Gutnik and A. Chandrakasan, “Embedded power supply for lowpower DSP,” IEEE Trans. VLSI Syst., vol. 5, pp. 425–435, Dec. 1997.
63. L. Torvalds, “The Linux operating system,” Commun. ACM, vol. 42, no. 4, pp. 38–39, Apr. 1999.
64. H. E. Bal, M. F. Kaashock, and A. S. Tanenbaum. “Orca: A Language for Parallel Programming of Distributed Systems”, IEEE Transactions on Software Engineering, 18(3),March 1992.
65. Bellcore, “Generic System Requirements in Support of Switched Multi-megabit Data Service”. TR-TSV-0007772,no 1, May 1991.
66. A. Black, N. Hutchison, E. Jul, and H. Levy. “Object Structure in the Emerald System”, In Proceedings of the ACM Conference on Object-Oriented Programming Systems, Languages, and Applications, pages 78-86, 1986.
67. G. Booch, Object-Oriented Design with Applications, Menlo Park, CA, Benjamin/Cummings, 1991.
68. J. T. Buck et al. “Ptolemy: A framework for simulating and prototyping heterogeneous systems”, International Journal on Computer Simulation, January 1994.
69. R. Camposano andW.Wolf (editors), Trends in High-Level Synthesis, Kluwer Academic Publishers, 1993.
70. D. Caromel. “Toward a Method of Object-Oriented Concurrent Programming”, Communications of the ACM, September 1993.
71. H. De Man, I. Bolsens, B. Lin, K. Van Rompaey, S. Vercauteren, and D. Verkest. “Co-design of DSP systems”, NATO ASI Hardware/Software Co-Design, Tremezzo, June 1995.
72. M. De Prycker, “AsynchronousTransferMode, Solution forBroadbandISDN”, Ellis Horwood, 1991.
73. J. Dogarra,G. Geist, R. Manchek, and V. Sunderam. “Integrated PVM Framework Supports Heterogeneous Network Computing”, In Computers in Physics, April 1993.
74. G. de Jong, B. Lin, C. Verdonck, S. Wuytack, F. Catthoor, “BackgroundMemory Management for Dynamic Data Structure Intensive Processing Systems”, ICCAD, November 1995.
75. I. Foster,C. Kesselman, S. Tuecke, “Nexus: RuntimeSupportforTask-ParallelProgramming Languages”, Technical report, Mathematics and Computer Science Division, ArgonneNational Laboratory, 1994.
76. GNU distribution. Free Software Foundation. 1994.
77. D. Harel. “Statecharts: A Visual Formalism for Complex Systems”, Sci. Comput. Program, vol.8, pp. 231-274, 1987.

78. C.A.R. Hoare. “Monitors: An operating system structuring concept”,Communications of the ACM, 17(10):549-557,October 1974.
79. “IEEE Standard VHDL Language Reference Manual”, IEEE Std. 1076-1987, IEEE, New York, NY, 1988.
80. A. Kalavade, E. A. Lee. “A Hardware/Software Codesign Methodology for DSP Applications”, IEEE Design & Test, Sept. 1993.
81. R. Lauwereins et al. “Grape-II: A system level prototyping environment for DSP applications”, IEEE Computer, pp.35-43, February 1995.
82. S. Vercauteren, B. Lin, H. De Man. “Constructing Application-SpecificHeterogeneous Embedded Architectures from Custom HW/SW Applications”, ACM/IEEE Design Automation Conference, June, 1996.
83. S. Vercauteren, B. Lin, H. De Man. “A Strategy for Real-Time Kernel Support in Application-Specific HW/SW Embedded Architectures”, ACM/IEEE Design Automation Conference, June, 1996.
84. B. Meyer. “Systematic Concurrent Object-Oriented Programming”, Communications of the ACM, September 1993.
85. “Mistral-2 Architecture Compiler”, Mentor Graphics, Beaverton, OR.
86. J. Rumbaugh et al., Object-oriented modeling and design, Prentice-Hall, 1991.
87. B. Stroustrup, “The C++ ProgrammingLanguage”,Addison-Wesley, 1986.
88. “Synopsys BehavioralCompiler”, Synopsys,Mountain View, CA.
89. Y. Therasse, G. Petit, M. Delvaux, “VLSI architecture of a SMDS/ATMrouter”, in Annales des T´el´ecommunications, 48, no3-4, 1993.
90. E. Verhulst, “Virtuoso, DSP programming tools covering from distributed multitasking to synchronous dataflow”, Eonics Inc., 1994.
91. Douglass, Bruce Powel. Doing Hard Time: Developing Real-Time Systems with UML, Objects,Frameworks and Patterns, Reading, MA: Addison-Wesley, 1999.
92. Gamma, E., R. Helm, R. Johnson, and J. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software, Reading, MA: Addison-Wesley, 1995.
93. Buschmann, F., R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. A System of Patterns: Pattern-Oriented Software Architecture, New York: Wiley & Sons, 1996 .
94. Selic, B., G. Gullekson, and P. Ward. Real-Time Object-Oriented Software, New York: Wiley & Sons, 1994.
95. Selic, B., and J. Rumbaugh. Using UML to Model Complex Real-Time Systems Rational Corporation white paper, www.rational.com March, 1998.
96. Sickle, Ted. Reusable Software Components, Upper Saddle River, NJ: Prentice Hall, 1997.
97. Herzum, P., and O. Sims. Business Component Factory, New York: Wiley & Sons, 2000.
98. Brown, Alan. Large-Scale Component-Based Development, Upper Saddle River, NJ: Prentice Hall, 2000.
99. Szyperski, Clemens. Component Software: Beyond Object-Oriented Programming, Reading, MA: Addison-Wesley, 1998.