ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΠΑΤΡΩΝ

ΣΧΟΛΗ ΔΙΟΙΚΗΣΗΣ ΚΑΙ ΟΙΚΟΝΟΜΙΑΣ
ΤΜΗΜΑ ΕΠΙΧΕΙΡΗΜΑΤΙΚΟΥ ΣΧΕΔΙΑΣΜΟΥ ΚΑΙ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

«ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ»

Περιβάλλοντα επίλυσης προβλημάτων: Case Study Προσομείωση Χρηματαγορών

Καστρίτου Αικατερίνη
Επιβλέπων Καθηγητής: Παπαϊωάννου Βάιος

2009
ΑΘΗΝΑ

1 Περίληψη

Σε αυτήν την εργασία κάνουμε μια ανασκόπηση της παρούσας κατάστασης του πεδίου των περιβαλλόντων επίλυσης προβλημάτων (ΠΕΠ). Αρχικά περιγράφουμε τα έμπειρα συστήματα, την user interface και τελικώς, τον ορισμό ενός ΠΕΠ και τους στόχους του. Γίνεται περίληψη της κατάστασης της επιστήμης μαζί με πηγές (βιβλία, βιβλιογραφία, ιστοσελίδες) για μια πιο λεπτομερειακή πληροφόρηση. Παρουσιάζονται, επίσης, τα βασικά στοιχεία και παραδείγματα για την κατασκευή των ΠΕΠ ενώ ολοκληρώνεται με μια αναφορά στις προσομειώσεις και ειδικότερα στην προσομείωση χρηματαγοράς.

Περιεχόμενα

1 Περίληψη 1
2 Εισαγωγή 5
3 Έμπειρα Συστήματα 7
3.1 Ορισμός 7
3.2 Τομείς δραστηριότητας 8
3.2.1 Παιχνίδια - απόδειξη μαθηματικών θεωρημάτων 8
3.2.2 Κωδικοποίηση λύσης προβλημάτων 8
3.3 Διαδικασία δημιουργίας Έμπειρου Συστήματος 8
3.3.1 Εκμέευση γνώσης 8
3.3.2 Κωδικοποίηση Γνώσης 9
3.3.3 Βάση Γνώσης (Knowledge Base) 9
3.3.4 Μηχανή Εξαγωγής Συμπερασμάτων (Inference Engine) 9
3.3.5 Χαρακτηριστικά Συμπερασμάτων 11
3.3.6 Μέθοδοι Επίλυσης Προβλημάτων 11
3.4 Διαδεδομένα Συστήματα Έμπειρων Συστημάτων 11
3.4.1 MYCIN 12
3.4.2 DENDRAL 12
3.4.3 PROSPECTOR 12
3.4.4 Κελύφη (Expert System Shells) και η εξέλιξη 12
3.5 Περιοχές Εφαρμογής 13
3.5.1 Διάγνωση 13
3.5.2 Πρόγνωση 13
3.5.3 Εκπαίδευση 13
3.5.4 Παρακολούθηση καταστάσεων 14
3.5.5 Επιδιόρθωση λαθών 14
3.5.6 Ερμηνεία 14
3.5.7 Έλεγχος 14
3.5.8 Σχεδιασμός 14
3.5.9 Παροχή Συμβουλών 14
3.6 Πλεονεκτήματα Ε.Σ 15
3.6.1 Ορθότητα απαντήσεων 15
3.6.2 Οργάνωση και αποθήκευση γνώσης 15
3.6.3 Ελαχιστοποίηση λάθους 15
3.6.4 Ευκολία χρήσης 15
3.6.5 Πολυ-χρηστικότητα 15
3.7 Μειονεκτήματα Ε.Σ 16
3.7.1 Απουσία κοινής λογικής 16
3.7.2 Αδυναμία συνθετικής λήψης αποφάσεων 16
3.7.3 Λάθη στο σχεδιασμό 16
3.7.4 Αδυναμία αυτόματης εξέλιξης 16
3.8 Έμπειρα Συστήματα για την επίλυση προβλημάτων 17

4 User Interface 18
4.1 Πλεονεκτήματα GUI 19
4.1.1 Ευκολία μάθησης 19
4.1.2 Multitasking 19
4.1.3 Προσανατολισμένο στο χρήστη 19
4.1.4 Συμβατότητα 19
4.1.5 Ανοχή σε λάθη 19
4.1.6 Βοήθεια στο χρήστη 19
4.2 Συστήματα Διεπαφής 20
4.2.1 Απευθείας χειρισμός 20
4.2.2 Μενού 20
4.2.3 Συστήματα Διεπαφής Εντολών 21
4.2.4 Οδηγίες Χρήσης 21
4.3 Χρηστικότητα (Usability) και Σχεδιασμός Προσανατολισμένος στον Χρήστη (User Centered Design) 22
4.3.1 Χρηστικότητα (Usability) 22
4.3.2 Σχεδίαση Προσανατολισμένη προς τον Χρήστης (User-Centered Design - UCD) 26
4.4 User Interface και Έμπειρα Συστήματα 28
5 Περιβάλλοντα Επίλυσης Προβλημάτων 29
5.1 Τι είναι ένα ΠΕΠ 29
5.2 Χαρακτηριστικά ΠΕΠ 31
5.2.1 Προσανατολισμένο στο πρόβλημα (Problem oriented). 31
5.2.2 Ενοποίηση (Integrated). 31
5.2.3 Διανεμημένο (Distributed). 31
5.2.4 Συνεχιζόμενο (Persistent). 31
5.2.5 Δυναμικά (Powerful). 31
5.2.6 Ανοιχτά, ελαστικά, προσαρμόσημα (Open, flexible, adaptive). 31
5.2.7 Γραφιστικά (Graphical, visual) 32
5.2.8 Έξυπνα (Intelligent). 32
5.3 Στάδια Επίλυσης Προβλημάτων 33
5.3.1 Έρευνα. 33
5.3.2 Σχεδιασμός και ανάπτυξη. 33
5.3.3 Διαδικασία Παραγωγής. 33
5.3.4 Εκπαίδευση. 33
5.4 Τεχνολογίες ΠΕΠ 34
5.4.1 Αλληλεπίδραση ανθρώπου μηχανής (Human-computer interaction – HCI). 34
5.4.2 Τεχνητή νοημοσύνη. 34
5.4.3 Παράλληλος προγραμματισμός. 34
5.4.4 Μαθηματικά μοντέλα. 34
5.4.5 Οπτικοποίηση αποτελεσμάτων. 35

6 Συστήματα Προσομοίωσης....36
6.1.1 Είδη προσομείωσης....36
6.2 Εφαρμογές προσομείωσης....36
6.3 Σύνδεση ενός Έμπειρου Συστήματος με ένα σύστημα Προσομοίωσης....37
6.4 Τα Πλεονεκτήματα και Μειονεκτήματα της προσομοίωσης....38
6.5 Προσομείωση Χρηματαγορών....39
6.5.1 Είδη Προσομείωσης Χρηματαγορών....39
6.5.2 Τεχνολογία και Εφαρμογές....39
6.5.3 Παιχνίδια Χρηματαγορών....40
7 Επίλογος....41
8 Βιβλιογραφία....43
9 Παράρτημα 1....45

2 Εισαγωγή

Τα τελευταία χρόνια αναπτύσσεται μια έντονη ερευνητική δρστηριότητα στο πεδίο σχεδιασμού και ανάπτυξης «Περιβάλλοντων Επίλυσης Προβλημάτων» (Problem Solving Environments) τα οποία προσφέρουν στο χρήστη ένα ολοκληρωμένο περιβάλλον που τον καθοδηγεί, με παραστατικό τρόπο μέχρι την τελική επίλυση του προβλήματός του. Η δυνατότητα αυτή, προκύπτει από το εξαιρετικά «φιλικό» περιβάλλον επικοινωνίας με το χρήστη και από το γεγονός ότι συχνά ενσωματώνουν ένα Έμπειρο Σύστημα που καλύπτει την αντίστοιχη γνωστική περιοχή που εξερευνάται τη δεδομένη στιγμή. Στην ουσία, τα περιβάλλοντα αυτά αποτελλούνται ακριβώς από ένα Έμπειρο Σύστημα και ένα πολύ «εκλεπτυσμένο» περιβάλλον επικοινωνίας με το χρήστη, έτσι ώστε να καθοδηγούν, βήμα προς βήμα, με παραστατικό τρόπο το χρήστη μέχρι την τελική επίλυση του προβλήματός του, δίνοντας ταυτόχρονα ανάλογες επεξηγήσεις, διευκρινήσεις και εναλλακτικές λύσεις.

Παρακάτω θα αρχίσουμε με μια διεξοδική ανάλυση των Έμπειρων Συστημάτων και της User Interface, τα βασικά στοιχεία υλοποίησης ενός Περιβάλλοντος Επίλυσης Προβλημάτων με σκοπό να καταλάβουμε τον τρόπο λειτουργίας των επιμέρους τμημάτων ενώ θα ακολουθήσει η εξερεύνηση των καθε αυτό Συστημάτων.

Αναλυτικότερα ξεκινούμε την εργασία μας, στο πρώτο κομμάτι, με μια εισαγωγή στα Έμπειρα Συστήματα εξηγώντας σε ποιο τομέα της επιστήμης των υπολογιστών υπάγονται και με τι πραγματεύονται δίνοντας και αναλύοντας τον ορισμό τους. Παραθέτουμε μερικούς από τους τομείς δραστηριότητας στα οποία έχουν εστιάσει οι επιστήμονες το ενδιαφέρον τους καθώς και το πώς λειτουργούν τα Έμπειρα Συστήματα ακολουθώντας βήμα προς βήμα τα στάδια δημιουργίας ενός τέτοιου συστήματος. Φυσικά αναφέρονται τα πιο διαδεδομένα συστήματα και περιοχές εφαρμογής ενώ δε λύπουν και οι αναφορές στα πλεονεκτήματα και μειονεκτήματά τους.

Τελειώνοντας την αναφορά στα Έμπειρα Συστήματα προχωρούμε στην ανάλυση της User Interface. Παραθέτουμε τον ορισμό της, τα πλεονεκτήματα των περιβαλλόντων διεπαφής με τον χρήστη καθώς και τα συστήματα διεπαφής. Δε θα μποσούσαν να λύπουν από αυτή την αναφορά η χρηστικότητα και η σχεδίαση προσανατολισμένη στο χρήστη καθώς και η μεθοδολογία σχεδίασής της.

Στο βασικό κομμάτι αναφερόμαστε στα Περιβάλλοντα Επίλυσης Περιβάλλοντος και στο συνδυασμό των ανωτέρω συστημάτων που οδηγούν στη δημιουργία τους. Εξαιρευνούμε τους ορισμούς που έχουν διαμορφωθεί για αυτή τη νέα σχετικά περιοχή έρευνας, τα χαρακτηριστικά τους όπως και τα στάδια επίλυσης προβλημάτων έτσι ώστε να καταλάβουμε καλύτερα για ποιο λόγο και πως χρησιμοποιείται η τεχνολογία στην οποία στηρίζονται τα Περιβάλλοντα Επίλυσης Προβλημάτων.

Τελικά φτάνουμε στο σημείο γνωριμίας με τα συστήματα προσομείωσης, μοντέλα τα οποία σχεδιάζονται σε ένα έμπειρο σύστημα με τον απαραίτητο συνδυασμό της user interface και αναπόφευκτα την αποτελεσματικότητα ενός Περιβάλλοντος Επίλυσης Προβλημάτων.

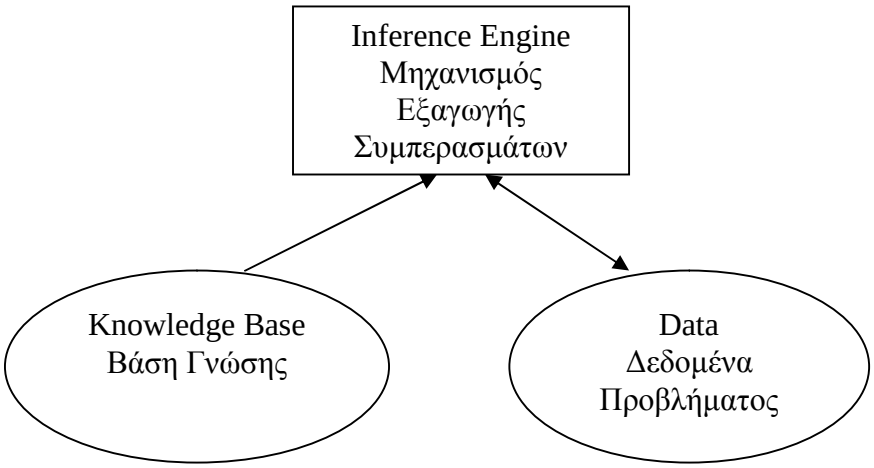
Το δεύτερο κομμάτι αφορά την προσομείωση χρηματαγορών. Μια ανάλυση στο τι ακριβώς είναι, ποια είδη υπάρχουν, και ποιές τεχνολογίες. Δεν είναι ευρύτερα αναπτυγμένο το τμήμα αυτό για αυτό και στο παράρτημα παραθέτουμε ένα δείγμα κατοχυρωμένης πατέντας συστήματος προσομείωσης χρηματαγοράς.

3 Έμπειρα Συστήματα

Τα έμπειρα συστήματα ανήκουν στον κλάδο της Πληροφορικής που ασχολείται με την Τεχνητή Νοημοσύνη. Ο κλάδος αυτός ασχολείται με την προσπάθεια των ερευνητών να κάνουν τους ηλεκτρονικούς υπολογιστές ικανούς να πραγματοποιούν εργασίες που μέχρι στιγμής οι άνθρωποι κατάφερναν καλύτερα λόγω της ευφυΐας τους. Ένα από τα σημαντικότερα θέματα που καλείται να επιλύσει η επιστήμη στο χώρο της Τεχνητής Νοημοσύνης είναι ο τρόπος με τον οποίο αποκτιέται, επεξεργάζεται και μεταδίδεται η γνώση για τη θεμελίωση μιας ευφυούς συμπεριφοράς.

3.1 Ορισμός

"Έμπειρο Σύστημα είναι ένα υπολογιστικό σύστημα το οποίο μπορεί να επιλύσει αποδοτικά και αποτελεσματικά ρεαλιστικά προβλήματα, η επίλυση των οποίων εκ μέρους του ανθρώπου συνεπάγεται την ύπαρξη κάποιας μορφής εμπειρογνωμοσύνης" [ΕΑΠ, 2000].



3.2 Τομείς δραστηριότητας

3.2.1 Παιχνίδια - απόδειξη μαθηματικών θεωρημάτων

Τα παιχνίδια και η αποδειξη μαθηματικών θεωρημάτων ήταν δύο από τους πρώτους χώρους στους οποίους δραστηριοποιήθηκαν οι ερευνητές. Στις περιπτώσεις αυτές ενώ οι άνθρωποι που τα καταφέρνουν καλά θεωρούνται ότι διαθέτουν αρκετή ευφυΐα, οι υπολογιστές μπορούν να είναι επίσης καλοί εξαιτίας του γεγονότος ότι μπορούν να κάνουν γρήγορους υπολογισμούς και να διερευνήσουν πολύ γρήγορα τις εναλλακτικές λύσεις ώστε να επιλέξουν την καλύτερη. Η διαδικασία αυτή απαιτεί λίγη γνώση και μπορεί εύκολα να προσομοιωθεί μέσα από ένα πρόγραμμα για Η/Υ. Σε ορισμένες περιπτώσεις κυρίως σε αδόμητα προβλήματα κανένας υπολογιστής δεν είναι αρκετά γρήγορος ώστε να μπορεί να ξεπεράσει την εκρηκτική παραγωγή πολλαπλών πιθανών συνδυασμών.

3.2.2 Κωδικοποίηση λύσης προβλημάτων

Η προσπάθεια κωδικοποίησης της λύσης προβλημάτων στην καθημερινή ζωή, που βασίζονται σε ένα σύνολο κανόνων που θα μπορούσαμε να ονομάσουμε «κοινή λογική». Το πεδίο αυτό ερεύνησε ο Newell στον «General Problem Solver» το 1963. [Morik, 1987] Εκτός όμως από τα προβλήματα που λύνονται με τη χρήση ενός γενικού συνόλου κανόνων, υπάρχουν δραστηριότητες οι οποίες είναι εξειδικευμένες και βασίζονται σε εξειδικευμένη γνώση γι' αυτό και οι άνθρωποι που τις διεκπεραιώνουν θεωρούνται «ευφυείς», όπως για παράδειγμα μια πρόβλεψη καιρού ή μια ιατρική διάγνωση. Η λύση στα προβλήματα αυτά δεν δίνεται μέσα από μια ιδιαίτερα πολύπλοκη νοητική διαδικασία αλλά στηρίζεται κυρίως στην ειδική γνώση που απαιτείται και το εξειδικευμένο σύνολο κανόνων για να ληφθεί μια απόφαση. Αρκετά από αυτά τα προβλήματα μπορούν να λυθούν από αντίστοιχα προγράμματα Η/Υ, που ονομάζονται «Έμπειρα Συστήματα» (Expert Systems). [D. Waterman, 1986]

3.3 Διαδικασία δημιουργίας Έμπειρου Συστήματος

Τα Έμπειρα Συστήματα (Ε.Σ.) βασίζονται στα αποτελέσματα μιας πολύχρονης έρευνας στο χώρο της Τεχνητής Νοημοσύνης όπου αποδεικνύεται ότι σε πολλές περιπτώσεις η συγκέντρωση εκτεταμένης και κατάλληλα διαμορφωμένης γνώσης για ένα εξειδικευμένο πεδίο ανθρώπινης δραστηριότητας και εμπειρίας, μπορεί να οδηγήσει σε ευφυή συμπεριφορά. Αρχικά πρέπει να συγκεντρωθεί η γνώση αυτή η οποία θα αποτελέσει τη βάση για τη δημιουργία ενός πλαισίου κανόνων για τη λήψη αποφάσεων.

3.3.1 Εκμέευση γνώσης

Η κύρια πηγή συγκέντρωσης της γνώσης είναι οι άνθρωποι που θεωρούνται ειδήμονες δηλαδή ειδικοί στο συγκεκριμένο πεδίο όπως για παράδειγμα ένα διακεκριμένος νομικός στην περίπτωση επίλυσης νομικών προβλημάτων ή ένας διακεκριμένος ιατρός εξειδικευμένος στην επεξεργασία ιατρικών δεδομένων για τη διάγνωση συγκεκριμένων ασθενειών. Το έργο της εκμαίευσης της γνώσης από ένα ειδικό είναι πολύ δύσκολο λόγω του λεγόμενου «παράδοξου της

εμπειρογνωμοσύνης» (knowledge acquisition), του φαινόμενου δηλαδή όπου όσο πιο ικανός είναι ένας εμπειρογνώμον στο πεδίο γνώσης του , τόσο λιγότερο ικανός στο να περιγράψει τους κανόνες επίλυσης των προβλημάτων της γνωστικής του περιοχής. Γι' αυτό και την εκμαίευση της γνώσης συνήθως αναλαμβάνει ένας μηχανικός γνώσης (knowledge engineer) μέσα από μια διαδικασία ερωτηματολογίων και συνεντεύξεων προς τους ειδικούς.

3.3.2 Κωδικοποίηση Γνώσης

Αφού συγκεντρωθούν οι απαραίτητες πληροφορίες από το μηχανικό γνώσης και οργανωθούν κατάλληλα, κωδικοποιούνται χρησιμοποιώντας κάποια γλώσσα προγραμματισμού που στην περίπτωση της Τεχνητής Νοημοσύνης ως πιο διαδεδομένες είναι η LISP και PROLOG, όμως μπορεί εναλλακτικά να χρησιμοποιηθεί κάποιο «εργαλείο κατασκευής Εμπείρων Συστημάτων», όπως το Expert System Shells.

3.3.3 Βάση Γνώσης (Knowledge Base)

Η συγκέντρωση και κωδικοποίηση της γνώσης σε ένα σύνολο πληροφοριών παράγει τη Βάση Γνώσης (Knowledge Base). Στη συνέχεια ένας προγραμματιστής αναλαμβάνει να κατασκευάσει το δεύτερο μέρος ενός Έμπειρου Συστήματος.

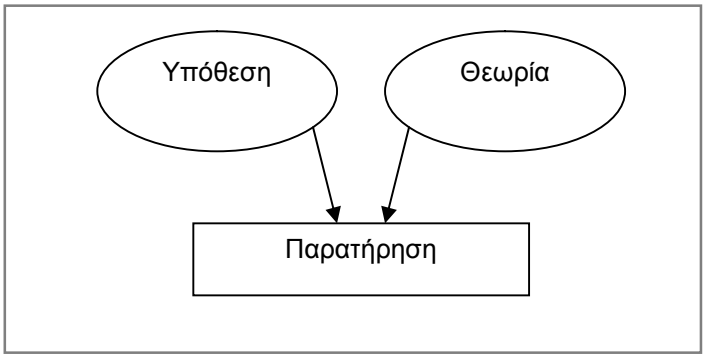
3.3.4 Μηχανή Εξαγωγής Συμπερασμάτων (Inference Engine)

Πρόκειται για ένα σύνολο γενικών μεθόδων και κανόνων που επιτρέπει την επίλυση προβλημάτων με τη χρήση της ειδικής γνώσης που περιέχεται στην Βάση Γνώσης.

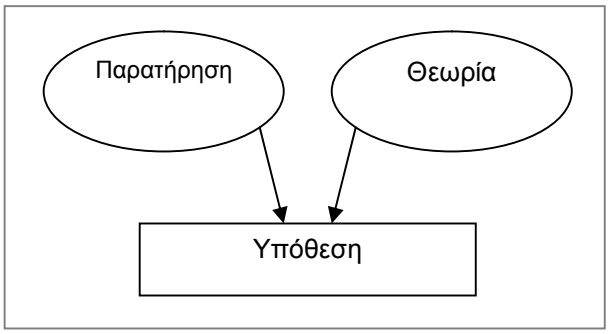
Σύμφωνα με τον φιλόσοφο C.S. Pierce [Pierce 1958] υπάρχουν τρεις βασικές μορφές συλλογισμού:

1. το συμπέρασμα (deduction)
2. η απαγωγή (abduction)
3. η επαγωγή (induction).

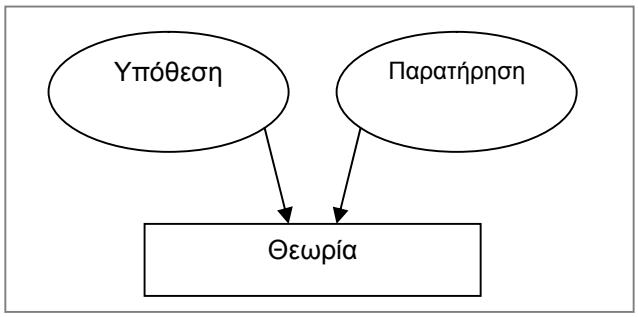
Έτσι χρησιμοποιώντας τρία βασικά συστατικά του ανθρώπινου συλλογισμού, τη θεωρία, την παρατήρηση και την υπόθεση, προκύπτουν οι παρακάτω ροές συλλογισμού:



Α. Συμπέρασμα: χρησιμοποιώντας μια υπόθεση ή ένα γεγονός και θεωρία μπορεί να προβλεφθεί μια παρατήρηση



Β. Απαγωγή: χρησιμοποιώντας μια τυχαία παρατήρηση και τη θεωρία μπορεί να παραχθεί μια υπόθεση που εξηγεί την παρατήρηση.



Γ. Επαγωγή: χρησιμοποιώντας πολλές παρατηρήσεις και κάποιες υποθέσεις που συσχετίζουν τις παρατηρήσεις μπορεί να παραχθεί θεωρία δηλαδή νέα γνώση

Η απαγωγή και το συμπέρασμα συμμετέχουν στη διαδικασία επίλυσης προβλημάτων από ένα έμπειρο σύστημα ενώ η επαγωγή χρησιμοποιείται για την εξέλιξη και ανάπτυξη του έμπειρου συστήματος. Υπάρχουν έμπειρα συστήματα με δυνατότητα αυτό-βελτίωσης. Η ικανότητα αυτή λέγεται μάθηση κατά προσαύξηση (incremental learning) και οδηγεί στη σταδιακή εκλέπτυνση της γνώσης του συστήματος.

3.3.5 Χαρακτηριστικά Συμπερασμάτων

Συνεπώς τα Έμπειρα Συστήματα είναι σε θέση να επεξεργαστούν την ειδική γνώση που έχει συγκεντρωθεί σε μια Βάση Γνώσης με τη χρήση κάποιου συνόλου κανόνων έτσι ώστε να καταλήξουν σε συμπεράσματα. Ο τρόπος με τον οποίο καταλήγουν σε αυτά τα συμπεράσματα συνήθως περιγράφεται από το ίδιο το Έμπειρο Σύστημα και είναι κατανοητός όχι μόνο σε προγραμματιστές έμπειρων συστημάτων αλλά και σε όσους γνωρίζουν τον τρόπο εξαγωγής αυτών των συμπερασμάτων και χωρίς τη χρήση Η/Υ.

Έτσι λοιπόν ένα Έμπειρο Σύστημα μπορεί να περιγράψει τον τρόπο εξαγωγής των συμπερασμάτων μέσα από ένα άλλο τμήμα του συστήματος που ονομάζεται Μονάδα Εξήγησης (Explanation Module) στα πλαίσια της αυτό-γνώσης του συστήματος (self-knowledge).

1. Μάθηση κατά προσαύξηση (incremental learning). Η γνώση που παράγεται από την εξαγωγή συμπερασμάτων από τη Βάση Γνώσης ονομάζεται μετα-γνώση (metaknowledge) η οποία μπορεί να δημιουργήσει κατόπιν επεξεργασίας νέα γνώση και έτσι. το σύστημα να εξελίσσεται εμφανίζοντας δυνατότητες Μάθησης.

2. Επεκτασιμότητα. Άλλο ένα χαρακτηριστικό των Έμπειρων Συστημάτων είναι ότι είναι επεκτάσιμα και η γνώση που παράγεται δεν είναι στατική αφού η Βάση Γνώσης μπορεί να εμπλουτίζεται διαρκώς με νέες πληροφορίες. Όπως ακριβώς και στην περίπτωση της ανθρώπινης γνώσης.

3. Λήψη Αποφάσεων. Τέλος τα Έμπειρα Συστήματα είναι έτσι προγραμματισμένα έτσι ώστε να έχουν τη δυνατότητα λήψης αποφάσεων σε συνθήκες αβεβαιότητας, όπως και στην περίπτωση των αποφάσεων που καλούνται να λάβουν οι εμπειρογνώμονες. Αυτό μπορεί να οφείλεται είτε σε έλλειψη δεδομένων ή σε αδυναμία των ίδιων των κανόνων και έτσι να οδεύουμε σε συμπεράσματα που έχουν μεγαλύτερη πιθανότητα να είναι ορθά.

3.3.6 Μέθοδοι Επίλυσης Προβλημάτων

Στα Έμπειρα Συστήματα χρησιμοποιούνται πολύ συχνά οι ευριστικές μέθοδοι επίλυσης προβλημάτων κατά τις οποίες δεν εξετάζονται όλες οι πιθανές λύσεις γιατί αυτό είναι επεξεργαστικά και χρονικά ασύμφορο. Σε προβλήματα όπου το εύρος των πιθανών λύσεων είναι τεράστιο αντί να εξαντλείται η εξέταση της κάθε πιθανής λύσης για να επιλεχθεί η βέλτιστη, εξαιρούνται οι συνδυασμοί οι οποίοι έχουν χαμηλή πιθανότητα να επιλύουν το πρόβλημα και προτιμώνται οι συνδυασμοί αυτοί που έχουν τη μεγαλύτερη πιθανότητα. Οι συνδυασμοί αυτοί χρησιμοποιούνται για να παράξουν νέους συνδυασμούς από τους οποίους πάλι θα επιλεχθούν οι πιο πιθανοί να οδηγήσουν σε λύση με βάση την εμπειρία και η γνώση.

3.4 Διαδεδομένα Συστήματα Έμπειρων Συστημάτων

Όπως ήδη αναφέραμε ένας βασικός χώρος στον οποίο έχουν εφαρμοστεί τα Έμπειρα Συστήματα είναι η Ιατρική Διάγνωση και Θεραπείας.

3.4.1 MYCIN

Το MYCIN είναι το πιο διαδεδομένο σύστημα διάγνωσης και επιλογής θεραπείας για ασθενείς με βακτηριακές μολύνσεις ενώ παρόμοια συστήματα είναι το PIP'3, το INTERNIST9 και το CASNET [Weiss 1978].

3.4.2 DENDRAL

Ένα από τα πρώτα διαδεδομένα Έμπειρα .Συστήματα. ήταν το DENDRAL το οποίο χρησιμοποιείται στην ανάλυση οργανικών ουσιών και τον προσδιορισμό της δομής τους. Το σύστημα ήταν τόσο επιτυχημένο που τα αποτελέσματα που εξήγαγε δημοσιεύτηκαν ως πρωτότυπα συμπεράσματα έρευνας.

3.4.3 PROSPECTOR

Επίσης το PROSPECTOR είναι ένα ακόμα σύστημα που χρησιμοποιήθηκε από γεωλόγους για την ανίχνευση μεταλλευμάτων. Το PROSPECTOR4 χρησιμοποιήθηκε για την απόδειξη ύπαρξης μολυβδαινίου και άλλων μεταλλευμάτων σε μια περιοχή της Washington, το οποίο επαληθεύθηκε από γεωτρήσεις που έγιναν μεταγενέστερα.

3.4.4 Κελύφη (Expert System Shells) και η εξέλιξη

Η εξέλιξη των συστημάτων αυτών ήταν να δημιουργηθούν εργαλεία παραμετροποίησης και κατασκευής έμπειρων συστημάτων ανεξάρτητα από την περιοχή εξειδίκευσης. Τα εργαλεία αυτά που ονομάστηκαν κελύφη (Expert System Shells) αποτελούνταν από μια κενή Βάση Γνώσης και μια Μηχανή Εξαγωγής Συμπερασμάτων που μπορούσε να παραμετροποιηθει έτσι ώστε να μπορούν οι χειριστές να δημιουργήσουν ένα νέο Έμπειρο Σύστημα σε μια νέα περιοχή ενδιαφέροντος. Χρησιμοποιώντας σαν βάση το μηχανισμό του MYCIN, δημιουργήθηκε το EMYCIN (EmptyMYCIN), ενώ από το CASNET δημιουργήθηκε το κέλυφος EXPERT και από το PROSPECTOR το σύστημα KAS. Ο τυπικός χρήστης ενός έμπειρου συστήματος συνήθως ασχολείται με τη γνωστική περιοχή του συστήματος χωρίς όμως να απαιτείται να είναι ο ίδιος έμπειρος. Το σύστημα μπορεί να τον βοηθήσει πολλές φορές και στον ίδιο βαθμό που θα τον ικανοποιούσε ένας εμπειρογνώμονας ενώ μπορεί να χρησιμοποιηθεί και ως βοηθητικό εργαλείο από τον ίδιο τον εμπειρογνώμονα.

Έτος	**Πλήθος ΕΣ που κατασκευάστηκαν**
1985	50
1986	86
1987	1100
1988	2200
1992	12000

3.5 Περιοχές Εφαρμογής

Τα Έμπειρα Συστήματα σχεδιάστηκαν για να διευκολύνουν την εξαγωγή συμπερασμάτων σε διάφορα πεδία γνώσης όπως στη λογιστική, την ιατρική, τον Έλεγχο Διαδικασιών, την παραγωγή ή τη διαχείριση ανθρώπινου δυναμικού. Ένα παράδειγμα εφαρμογής των Έμπειρων Συστημάτων είναι η περίπτωση υποθηκών. Οι τράπεζες ενδιαφέρονται να μειώσουν το κόστος λειτουργίας των τμημάτων που ασχολούνται με τη χορήγηση μικρών δανείων. Η χρήση ενός τέτοιου συστήματος στην έγκριση ενός δανείου με βάση τους ειδικούς κανόνες χορήγησης που ισχύουν, θεωρείται αποδοτική και διασφαλίζει την αντικειμενικότητα της απόφασης παρά το εύρος της εξειδικευμένης γνώσης που απαιτείται.

Παρόλα αυτά τα Ε.Σ. πολλές φορές ενέχουν τον κίνδυνο σφάλματος σε προβλήματα όπου η ανθρώπινη διαίσθηση εύκολα αντιλαμβάνεται. Επίσης η λογική των Έμπειρων Συστημάτων έχει σήμερα ενσωματωθεί σε πιο πολύπλοκα προγράμματα και δεν αποτελεί ξεχωριστή εφαρμογή. Ένα τέτοιο παράδειγμα είναι η δυνατότητα επίλυσης προβλημάτων που έχει συμπεριλάβει η Microsoft στο λειτουργικό σύστημα Windows είναι τμήμα της βοήθειας προς τον χρήστη (Microsoft Windows operating system troubleshooting).

Επίσης πολύ συχνά χρησιμοποιήθηκε η λογική των Έμπειρων Συστημάτων στο χώρο των παιχνιδιών. Ακόμα και σε παιχνίδια που δεν ανήκαν στο χώρο των Παιχνιδιών Στρατηγικής, ήταν ενσωματωμένη η δυνατότητα λήψης αποφάσεων εκ μέρους του υπολογιστή όπως στην περίπτωση του παιχνιδιού "Earl Weaver Baseball" όπου όταν ο παίκτης έπαιζε εναντίον του υπολογιστή, ο υπολογιστής χρησιμοποιούσε μια λογική Έμπειρου Συστήματος για να λάβει μια απόφαση. Οι πιο χαρακτηριστικές περιοχές εφαρμογής είναι λοιπόν οι εξής:

3.5.1 Διάγνωση

Διάγνωση δυσλειτουργιών ενός συστήματος βάσει παρατηρήσεων και μετρήσεων με πιο χαρακτηριστική εφαρμογή το MYCIN – διάγνωση μόλυνσης του αίματος [Shortliffe, 1974]

3.5.2 Πρόγνωση

Πρόγνωση μελλοντικών επιπτώσεων για δεδομένες καταστάσεις, χρησιμοποιώντας σαν βάση παρελθοντικές καταστάσεις με πιο χαρακτηριστική εφαρμογή την PRICE-STRAT για πρόβλεψη τιμών.

3.5.3 Εκπαίδευση

Εφαρμογές κατανόησης, αξιολόγησης και διόρθωσης απαντήσεων μαθητών σε εκπαιδευτικά προβλήματα με πιο χαρακτηριστική εφαρμογή το SOPHIE για εκπαίδευση φοιτητών σε θέματα ηλεκτρονικών. [Brown, Burton, 1982]

3.5.4 Παρακολούθηση καταστάσεων

Παρακολούθηση παραμέτρων και σύγκριση με αναμενόμενες καταστάσεις με χαρακτηριστική εφαρμογή: VENTILATOR MANAGEMENT ASSISTANT .

3.5.5 Επιδιόρθωση λαθών

Εφαρμογές ανάπτυξης και εκτέλεσης σχεδίων για τη διαχείριση βλαβών. Τέτοια εφαρμογή είναι το COOKER ADVISER που παράγει συμβουλές για επιδιόρθωση βλαβών σε εταιρίες τροφίμων.

3.5.6 Ερμηνεία

Εξαγωγή συμπερασμάτων βάσει δεδομένων από παρατηρήσεις με χαρακτηριστική εφαρμογή το PROSPECTOR [Duda & Hart, 1976].

3.5.7 Έλεγχος

Έλεγχος της συμπεριφοράς ενός συστήματος όπως το ARTIFACT που είναι ένα real-time shell για έλεγχο παραγωγής

3.5.8 Σχεδιασμός

Σύστημα δημιουργίας μιας αλληλουχίας ενεργειών για την επίτευξη συγκεκριμένων στόχων όπως το MOLGEN για το σχεδιασμό χημικών διεργασιών με σκοπό την ανάλυση και τη σύνθεση του [Stefik, 1981]

3.5.9 Παροχή Συμβουλών

Εναλλακτικές προτάσεις για την επίλυση προβλημάτων, λαμβάνοντας υπόψη συγκεκριμένες απαιτήσεις.

3.6 Πλεονεκτήματα Ε.Σ

3.6.1 Ορθότητα απαντήσεων

Ένα από τα πλεονεκτήματα των Έμπειρων Συστημάτων είναι ότι παρέχουν ορθές απαντήσεις σε ερωτήματα και διαδικασίες που επαναλαμβάνονται πολύ συχνά.

3.6.2 Οργάνωση και αποθήκευση γνώσης

Επίσης οργανώνουν και αποθηκεύουν τη γνώση πάνω σε διάφορα θέματα και προτρέπουν τους οργανισμούς και τις επιχειρήσεις να διαχωρίζουν αυτή τη γνώση από τους κανόνες λήψης αποφάσεων.

3.6.3 Ελαχιστοποίηση λάθους

Επίσης μειώνουν την πιθανότητα λάθους στη λήψη μιας απόφασης γιατί εξαλείφουν τον παράγοντα του ανθρώπινου λάθους, όπως είναι το να ξεχάσει κάποιος ειδικός να θέσει κάποια ερώτηση την οποία το σύστημα δεν πρόκειται να ξεχάσει να θέσει ποτέ.

3.6.4 Ευκολία χρήσης

Ανάλογα με την ευκολία χρήσης, μπορεί ένα έμπειρο σύστημα να γίνει βασικό συμβουλευτικό εργαλείο.

3.6.5 Πολυ-χρηστικότητα

Τέλος ένα πολύ-χρηστικό σύστημα μπορεί να εξυπηρετήσει ταυτόχρονα πολλούς χρήστες.

3.7 Μειονεκτήματα Ε.Σ

Πέρα από τα πλεονεκτήματα έχουν κάποια σημαντικά μειονεκτήματα σε σχέση με τη λήψη αποφάσεων.

3.7.1 Απουσία κοινής λογικής

Πρώτα από όλα τα Έμπειρα Συστήματα δεν έχουν αυτό που θα ονομάζαμε κοινή λογική και η οποία πολλές φορές αποτελεί βασικό παράγοντα στη λήψη αποφάσεων.

3.7.2 Αδυναμία συνθετικής λήψης αποφάσεων

Επίσης δεν είναι σε θέση να λειτουργήσουν συνθετικά και να απαντήσουν δημιουργικά σε ασυνήθιστες καταστάσεις σε αντίθεση με έναν ειδικό ο οποίος χρησιμοποιώντας τη διαίσθηση του μπορεί να δώσει απαντήσεις σε πιο σύνθετες καταστάσεις. Σε πολλές επίσης γνωστικές περιοχές, αν και υπάρχουν κανόνες λήψης αποφάσεων, οι ειδικοί είναι δύσκολο να εκφράσουν τον τρόπο συλλογισμού και συνεπώς το πλαίσιο κανόνων για τη λήψη αποφάσεων είναι δύσκολο να κατασκευαστεί.

3.7.3 Λάθη στο σχεδιασμό

Επιπλέον λάθη στη βάση γνώσης μπορεί να οδηγήσουν σε εσφαλμένη απόφαση.

3.7.4 Αδυναμία αυτόματης εξέλιξης

Τέλος τα έμπειρα συστήματα δε μπορούν να προσαρμοστούν αυτόματα στο συνεχώς εξελισσόμενο περιβάλλον παρά μόνο όταν ενημερώνεται η βάση γνώσης.

3.8 Έμπειρα Συστήματα για την επίλυση προβλημάτων

Τα έμπειρα συστήματα χρησιμοποιούνται για την επίλυση προβλημάτων με μια όμως σημαντική διαφορά από τα παραδοσιακά συστήματα επίλυσης. Στα παραδοσιακά συστήματα επίλυσης προβλημάτων η απαραίτητη γνώση και οι κανόνες επίλυσης των προβλημάτων είναι ενσωματωμένα στον κώδικα της εκάστοτε εφαρμογής λογισμικού με αποτέλεσμα τα συστήματα αυτά να είναι στατικά και να στερούνται της δυνατότητας να επεκταθούν και να προστεθεί καινούρια εμπειρία στη διαδικασία επίλυσης. Αντίθετα σε ένα έμπειρο σύστημα που χρησιμοποιείται στην επίλυση προβλημάτων η γνώση που απαιτείται για την επίλυση του προβλήματος είναι αποθηκευμένη σε δομές δεδομένων που είναι διαχωρισμένες από τις προγραμματιστικές και αλγοριθμικές δομές του προγράμματος με αποτέλεσμα να μπορεί η γνώση αυτή να τροποποιείται και να αυξάνεται καθώς νέα δεδομένα γίνονται γνωστά. Με αυτό τον τρόπο το σύστημα εξελίσσεται όχι μόνο από γνώσεις οι οποίες προστίθενται από ειδικούς αναλυτές αλλά και από το ίδιο το σύστημα με διαδικασίες αυτοβελτίωσης.

Ας θεωρήσουμε για παράδειγμα ένα πρόγραμμα λογιστικής το οποίο βασίζεται σε ένα έμπειρο σύστημα. Ένα τέτοιο σύστημα έχει τη δυνατότητα να υπολογίζει για παράδειγμα τη φορολογία με βάση κάποιους κανόνες που είναι αποθηκευμένοι σε μια γνωσιακή βάση. Οι κανόνες όμως αυτοί αλλάζουν όπως και όλες οι σχετικές παράμετροι υπολογισμού. Αν αυτή η γνώση δεν ήταν αποθηκευμένη σε ξεχωριστή δομή από το υπόλοιπο πρόγραμμα τότε οι φοροτεχνικοί σύμβουλοι δε θα ήταν σε θέση να τροποποιήσουν τα δεδομένα και θα ήταν απαραίτητη η δημιουργία ανά πάσα στιγμή μιας νέας έκδοσης.

Τα έμπειρα συστήματα χρησιμοποιούνται συχνά ως συστήματα επίλυσης προβλημάτων στον οικονομικό σχεδιασμό και στη λογιστική χρησιμοποιώντας βάσεις γνώσεων που επικεντρώνονται στο γνωστικό αντικείμενο των επιχειρήσεων. Πρόκειται για μια σημαντική εξέλιξη όσον αφορά στον τρόπο που χρησιμοποιούν οι επιχειρήσεις τα πληροφοριακά συστήματα. [Thierauf R]

4 User Interface

Ένα User Interface μιας εφαρμογής λογισμικού είναι ουσιαστικά ένα περιβάλλον Διεπαφής Ανθρώπου – Μηχανής (HMI – Human Machine Interface). Κάθε τέτοιο περιβάλλον αποσκοπεί στο να γίνει εφικτή η επικοινωνία μεταξύ της μηχανής και του χειριστή της δηλαδή να μπορεί ένας άνθρωπος να στέλνει και να λαμβάνει δεδομένα προς και από τη μηχανή αντίστοιχα με έναν κατανοητό για αυτόν τρόπο. Στις πρώτες τους μορφές τα περιβάλλοντα αυτά ήταν δύσχρηστα και εξυπηρετούσαν κυρίως το πεδίο της λειτουργικότητας (functionality) παρά της χρηστικότητας (usability) καθιστώντας έτσι το χειρισμό μιας μηχανής ή ενός υπολογιστή έργο κυρίως ενός ειδικού με τεχνικές γνώσεις παρά ενός απλού χρήστη. Καθώς όμως οι ηλεκτρονικές συσκευές έμπαιναν όλο και περισσότερο στην καθημερινή μας ζωή γινόταν όλο και πιο επιτακτική η σχεδίαση συστημάτων επικοινωνίας με τις συσκευές που θα είναι εύχρηστα και φιλικά προς τον χρήστη (user-friendly). Έτσι η έννοια της Χρηστικότητας έγινε κυρίαρχη στο χώρο σχεδίασης συστημάτων Διεπαφής και βασικός τρόπος σχεδίασης αποτέλεσε η σχεδίαση Προσανατολισμένη προς το Χρήστη (User – Centered Design).

Ένα σύστημα διεπαφής με τον χρήστη (user interface) αποτελεί ένα από τα σημαντικότερα τμήματα μιας εφαρμογής λογισμικού, κι αυτό γιατί οι χρήστες Η/Υ έχουν την τάση να αξιολογούν τα πληροφοριακά συστήματα πρωτίστως με βάση το περιβάλλον επικοινωνίας και την ευχρηστία του συστήματος διεπαφής. Πέραν όμως της ευχρηστίας, ένα ελλιπώς σχεδιασμένο περιβάλλον διεπαφής μπορεί να είναι η αιτία για σοβαρούς λάθος χειρισμούς από τον χρήστη με καταστροφικά αποτελέσματα για τη λειτουργία του συνολικού συστήματος. Μπορούμε για παράδειγμα να αναλογιστούμε τι θα συνέβαινε σε χιλιάδες χρήστες Η/Υ αν δεν είχε προβλεφθεί από του κατασκευαστές των λειτουργικών συστημάτων η εμφάνιση ενός προειδοποιητικού μηνύματος πριν την πλήρη και οριστική διαμόρφωση του σκληρού δίσκου. Επίσης ένα φτωχά σχεδιασμένο σύστημα διεπαφής μπορεί να αποτρέψει τους χρήστες από τον να χρησιμοποιήσουν μια κατά τα άλλα πάρα πολύ καλή εφαρμογή λογισμικού. Για πολλά χρόνια λόγω της έλλειψης δυνατοτήτων σχεδίασης γραφικών από την πλευρά του Υλικού των υπολογιστών, τα συστήματα επικοινωνίας με τον χρήστη βασίζονταν κυρίως στο κείμενο και στη χρήση ειδικών χαρακτήρων για τη σχεδίαση υποτυπωδών γραφικών. Ξεκινώντας από τους οικιακούς υπολογιστές και τα εξελιγμένα γραφικά περιβάλλοντά τους (GEM, MacOS) και περνώντας πολύ μεταγενέστερα στους προσωπικούς υπολογιστές και στα Windows, τα συστήματα διεπαφής εστίασαν πλέον στο γραφικό σχεδιασμό. Συνεπώς σήμερα όταν αναφερόμαστε σε User Interfaces ουσιαστικά αναφερόμαστε σε Γραφικά Περιβάλλοντα Διεπαφής με τον Χρήστη (Graphical User Interfaces – GUIs) τα οποία βασίζονται σε παράθυρα διαλόγου, παρουσίαση με εικόνες, pull-down ή pop-up menus.

4.1 Πλεονεκτήματα GUI

Τα πλεονεκτήματα των GUI είναι πολλά σε σχέση με το παρελθόν.

4.1.1 Ευκολία μάθησης

Πρώτα απ' όλα είναι πολύ εύκολη η εκμάθησή τους και χρήση τους. Χρήστες χωρίς ιδιαίτερη εμπειρία μπορούν και μαθαίνουν εύκολα την χρήση τους.

4.1.2 Multitasking

Η έννοια του multitasking δίνει τη δυνατότητα να είναι παραπάνω από μια εφαρμογές λογισμικού ταυτόχρονα ενεργές με αποτέλεσμα ο χρήστης να μπορεί να αλλάξει από διεργασία σε διεργασία και να αλληλεπιδράσει με τις διαφορετικές εφαρμογές.

4.1.3 Προσανατολισμένο στο χρήστη

Η διεπαφή πρέπει να είναι προσανατολισμένη στον χρήστη παρά στον υπολογιστή λαμβάνοντας υπόψη τα ιδιαίτερα χαρακτηριστικά των χρηστών της εφαρμογής. Για παράδειγμα ένα σύστημα γραφείου πρέπει να χρησιμοποιεί όρους όπως γράμματα, αρχεία, φάκελοι από καταλόγους αρχείων, δείκτες αρχείων, κ.α.

4.1.4 Συμβατότητα

Επίσης το σύστημα διεπαφής θα πρέπει να διατηρεί ένα κατάλληλο επίπεδο συμβατότητας. Για παράδειγμα οι εντολές και τα μενού πρέπει να έχουν την ίδια μορφή που έχει συνηθίσει ο χρήστης και σε παρόμοιες εργασίες έτσι ώστε να νιώθει οικεία προς το περιβάλλον και να εκμεταλλεύεται την όποια προηγούμενη γνώση.

4.1.5 Ανοχή σε λάθη

Επίσης το σύστημα θα πρέπει να παρέχει ανοχή σε λάθη χρήστη και να του επιτρέπει να ανακάμπτει από αυτά. Έτσι είναι επιβεβλημένη η ύπαρξη δυνατότητας undo ή η χρήση παραθύρων επιβεβαίωσης λειτουργιών καταστροφής, 'soft' deletes, κ.α.

4.1.6 Βοήθεια στο χρήστη

Τέλος άλλο ένα σημαντικό στοιχείο είναι η ενσωμάτωση συστήματος βοήθειας προς τον χρήστη με δυνατότητα χρήσης Βοήθειας (Help) ή την ύπαρξη on-line εγχειρίδιων.

4.2 Συστήματα Διεπαφής

4.2.1 Απευθείας χειρισμός

Όσον αφορά στη λειτουργικότητα Χρήστη – Συστήματος δύο προβλήματα είναι αυτά που εμφανίζονται στον σχεδιασμό διαλογικών συστημάτων και αφορούν στο πως παρέχεται η πληροφορία από τον χρήστη στο σύστημα και πως παρουσιάζεται η πληροφορία από το σύστημα στον χρήστη. Η διεπαφή και αλληλεπίδραση με τον χρήστη και η παρουσίαση της πληροφορίας πρέπει να ολοκληρώνεται μέσω ενός πλαισίου διεπαφής. Αυτό μπορεί να στηρίζεται στον απευθείας χειρισμό. Ο απευθείας χειρισμός είναι όταν η διεπαφή παρουσιάζεται στον χρήστη με ένα μοντέλο πληροφορίας που μπορεί να αλλάξει με απευθείας δράση, π.χ. μέσα από μια φόρμα ένα όνομα αλλάζει με πληκτρολόγηση νέου ονόματος πάνω από το παλιό. Τα γραφικά περιβάλλοντα (GUI's) παρέχουν απευθείας χειρισμό, π.χ. κάποια αρχεία μπορούν να διαγραφούν με μετακίνηση εικόνας σε ένα καλάθι αχρήστων. Ένα από τα πλεονεκτήματα του Απευθείας Χειρισμού είναι ότι οι χρήστες ελέγχουν πιο εύκολα τον υπολογιστή έχοντας άμεση αντίδραση από τις πράξεις τους και τα λάθη μπορούν εύκολα να βρεθούν και να διορθωθούν τους και όχι να ελέγχονται. Επίσης ο χρόνος εκμάθησης των χρηστών είναι πολύ μικρός. Πέρα όμως από τα πλεονεκτήματα, υπάρχουν και κάποια μειονεκτήματα στο μοντέλο Απευθείας Χειρισμού, όπως είναι η διαμόρφωση ενός μοντέλου πληροφορίας που μπορεί να είναι δύσκολη ενώ ο προγραμματισμός συστημάτων τέτοιων διεπαφών είναι πολύπλοκος και έχει μεγάλες απαιτήσεις από το πληροφοριακό σύστημα. Υπάρχουν διάφορα μοντέλα διεπαφής όπως:

- Desktop metaphor. Πρόκειται για ένα μοντέλο που προσομοιώνει ένα είδος γραφείου με εικόνες που αναπαριστούν αρχεία, συρτάρια, κ.α.
- Control panel metaphor. Το μοντέλο είναι κέντρο ελέγχου με οντότητες που περιλαμβάνουν κουμπιά, διακόπτες, μενού, οθόνες, χάρακες κ.α.

4.2.2 Μενού

Όπως αναφέραμε τα συστήματα διεπαφής τύπου Κέντρου Ελέγχου βασίζονται σε μεγάλο βαθμό σε Μενού. Οι χρήστες επιλέγουν από ένα κατάλογο δυνατοτήτων που παρουσιάζονται σε αυτούς από το σύστημα και η επιλογή μπορεί να γίνει είτε με το ποντίκι ή με τα βέλη. Το βασικό πλεονέκτημα των Συστημάτων με Μενού είναι ότι οι χρήστες δεν χρειάζεται να θυμούνται ονόματα εντολών και πάντα τους δίνεται η δυνατότητα να έχουν λίστα των εντολών. Έτσι απαιτείται ελάχιστη εργασία εκ μέρους τους ενώ εξαλείφεται η πιθανότητα λάθους αφού ο χρήστης ουσιαστικά δεν πληκτρολογεί μια εντολή αλλά την επιλέγει από μια λίστα. Το πρόβλημα με τη χρήση συστήματος με Μενού είναι ότι συνδυαστικές εντολές είναι δύσκολο να αναπαρασταθούν. Επίσης δίνουν τη δυνατότητα αναπαράστασης σε περιορισμένο αριθμό επιλογών. Εάν υπάρχουν πολλές επιλογές τότε πρέπει να υπάρχει δυνατότητα δόμησης μενού. Τέλος οι έμπειροι χρήστες βρίσκουν τα μενού πιο αργά από τις γλώσσες εντολών

Ένα δομημένο Μενού μπορεί να βασίζεται σε διάφορα μοντέλα:

- Ιεραρχικά Μενού. Πρόκειται για μενού που οργανώνονται σε μια ιεραρχική δομή. Η επιλογή ενός μενού οδηγεί στην εμφάνιση ενός υπό - μενού
- Μενού που ακολουθούν. Πρόκειται για το μοντέλο όπου η επιλογή ενός μενού προκαλεί την εμφάνιση ενός άλλου μενού
- Συσχετιζόμενες μονάδες ελέγχου. Στην περίπτωση αυτή όταν επιλέγεται ένα αντικείμενο του μενού εμφανίζεται μονάδα ελέγχου που προσφέρει περισσότερες επιλογές

4.2.3 Συστήματα Διεπαφής Εντολών

Εκτός από τα γραφικά περιβάλλοντα Διεπαφής υπάρχουν και τα συστήματα Διεπαφής Εντολών. Στην περίπτωση αυτή ο χρήστης πληκτρολογεί εντολές ένα λειτουργικό σύστημα όπως το UNIX. Τα συστήματα αυτά υλοποιούνται με την χρήση απλών τερματικών και οι εντολές είναι εύκολες στην επεξεργασία με χρήση μεταγλωττιστών.

Το πρόβλημα με τις Διεπαφές Εντολών είναι ότι οι χρήστες πρέπει να μάθουν και να θυμούνται μία γλώσσα εντολών. Είναι πιο κατάλληλες για έμπειρους χρήστες και ειδικούς. Η πιθανότητα λάθους είναι πολύ μεγάλη αφού οι χρήστες είναι εύκολο να κάνουν λάθη στην πληκτρολόγηση γι αυτό και πλέον προσφέρονται ως εναλλακτικές προς τις εντολές μενού και συνήθως συνυπάρχουν μαζί τους. Όπως είναι φανερό τα σύγχρονα λειτουργικά συστήματα προσφέρουν πολλαπλές Διεπαφές με τον Χρήστη. Η παρουσίαση της πληροφορίας γίνεται και με γραφικά και κείμενο.

4.2.4 Οδηγίες Χρήσης

Ένα από τα σημαντικότερα τμήματα ενός user interface είναι και το τμήμα των οδηγιών χρήσης προς τους χρήστες. Οι οδηγίες για τους χρήστες αποτελούν την ολοκλήρωση ενός συστήματος διεπαφής για να βοηθήσουν τους χρήστες για τον τρόπο παρουσίασης της πληροφορίας, για τον τρόπο εκτέλεσης μιας λειτουργίας ή για την ανάνηψη από κάποιο λάθος. Οι οδηγίες συνήθως περιλαμβάνουν μηνύματα συστήματος όπως μηνύματα λαθών, εγχειρίδια, On-line βοήθεια. Ειδικά ο σχεδιασμός μηνυμάτων λάθους είναι πολύ σημαντικός διότι ελλιπής σχεδιασμός οδηγεί σε απόρριψη του συστήματος από τον χρήστη. Επίσης όσον αφορά στο σύστημα βοηθείας, οι σχεδιαστές θα πρέπει να λαμβάνουν υπόψη ότι οι χρήστες δεν είναι τόσο καλοί στο να διαβάζουν οθόνες με κείμενο. Θα πρέπει να προτιμώνται γραφικές αναπαραστάσεις και συστήματα αλληλεπίδρασης όπως στην περίπτωση του Microsoft Troubleshooting όπου έχει γίνει υλοποίηση έμπειρου συστήματος.

Σε ένα σύστημα βοήθειας θα πρέπει να υπάρχουν πολλαπλά σημεία εισόδου ώστε να μπορεί ο χρήστης να παίρνει βοήθεια από διαφορετικά σημεία. Είναι επίσης σημαντικό να ξέρει ο χρήστης σε ποιο σημείο της βοήθειας βρίσκεται και θα πρέπει να υπάρχει δυνατότητα πλοήγησης του χρήστη στο σύστημα βοήθειας. Πέρα από το Σύστημα Βοήθειας καλό είναι να υπάρχει βοήθεια και σε μορφή ψηφιακού εγχειριδίου. Η τεκμηρίωση ενός συστήματος θα πρέπει να σχεδιάζεται για ποικιλία χρηστών από έμπειρους έως τελείως άπειρους

4.3 Χρηστικότητα (Usability) και Σχεδιασμός Προσανατολισμένος στον Χρήστη (User Centered Design)

Στην περίπτωση λοιπόν ανάπτυξης εφαρμογών λογισμικού, ένα πολύ μεγάλο μέρος της προσπάθειας αποσκοπεί στη σχεδίαση και δημιουργία ενός συστήματος Διεπαφής και Αλληλεπίδρασης Ανθρώπου – Υπολογιστή. Η επιτυχία ενός τέτοιου συστήματος εξαρτάται σε μεγάλο βαθμό από την ευκολία στη χρήση και τη φιλικότητα του περιβάλλοντος.

4.3.1 Χρηστικότητα (Usability)

Η χρηστικότητα (Usability) είναι ένας καθοριστικός παράγοντας για ένα περιβάλλον αλληλεπίδρασης ανθρώπου και μηχανής. Με τον όρο Χρηστικότητα εννοούμε το κατά πόσο εύκολο είναι ένα σύστημα αλληλεπίδρασης στην εκμάθηση χειρισμού και στο κατά πόσο επιτρέπει στους χρήστες να εργάζονται αποδοτικά (efficiently), αποτελεσματικά (effectively) και με ευκολία (comfortably) [DeLaere 1998]. Σύμφωνα με τον Shackel, για να είναι ένα σύστημα χρηστικό , πρέπει να είναι εύκολο στη μάθηση, αποτελεσματικό, προσαρμόσιμο και ευχάριστο [Preece 1990].

4.3.1.1 Βασικά χαρακτηριστικά χρηστικότητας

Η χρηστικότητα επίσης είναι ένα μέγεθος το οποίο μετράει το κατά πόσο εύχρηστο είναι ένα user interface [Nielsen 1993]. Σύμφωνα με τον Nielsen τα βασικά χαρακτηριστικά της χρηστικότητας δίνονται από την παρακάτω λίστα:

1. Δυνατότητα εκμάθησης (Learnability)

Το χαρακτηριστικό αυτό μετράει το κατά πόσο εύκολο είναι για τους χρήστες να ολοκληρώσουν βασικές λειτουργίες με την πρώτη φορά που έρχονται σε επαφή με το σύστημα. Τα συστήματα τα οποία έχουν μεγάλη ευκολία εκμάθησης δίνουν τη δυνατότητα στους χρήστες τους να φτάσουν σε υψηλά επίπεδα γνώσης των λειτουργιών του προγράμματος σε πολύ σύντομο χρονικό διάστημα [Nielsen 1993]. Ο έλεγχος κατά πόσο είναι εύκολη η εκμάθηση ενός προγράμματος με τη χρήση ενός συγκεκριμένου user interface γίνεται με τη συλλογή δεδομένων από τη διαδικασία χρήσης του προγράμματος από αρχάριους χρήστες την πρώτη φορά που χρησιμοποιούν το πρόγραμμα. [Dumas 1993]. Η βασική μέτρηση είναι ο χρόνος που απαιτείται για να φτάσουν οι χρήστες σε ένα ικανοποιητικό επίπεδο χρήσης του προγράμματος.

2. Αποδοτικότητα (Efficiency)

Αφού οι χρήστες φτάσουν σε ένα ικανοποιητικό επίπεδο θα πρέπει να εκτιμηθεί το κατά πόσο το user interface τους παρέχει τη δυνατότητα να εκτελούν τις διαθέσιμες λειτουργίες γρήγορα και αποδοτικά με οικονομία κινήσεων. [Preece 1995].

3. Προσαρμοστικότητα (Flexibility)

Πρόκειται για το βαθμό στον οποίο το σύστημα μπορεί να προσαρμόσει αλλαγές στον τρόπο χρήσης και στην λειτουργικότητά του πέρα από αυτές τις λειτουργίες που έχουν οριστεί κατά τον αρχικό σχεδιασμό και υλοποίησή του [Noyes 1999]. Τα

συστήματα θα πρέπει να σχεδιάζονται έτσι ώστε να επιδέχονται αλλαγές και διαφοροποιήσεις στον αρχικό τους σχεδιασμό.

4. Δυνατότητα Απομνημόνευσης (Memorability)

Η δυνατότητα αυτή αναφέρεται στην ικανότητα επαναφοράς στη μνήμη του τρόπου χρήσης του προγράμματος αφού παρέλθει κάποιο διάστημα κατά το οποίο ο χρήστης δεν έχει χρησιμοποιήσει το πρόγραμμα. Όσο πιο γρήγορα ανακαλέσει τον τρόπο με τον οποίο χρησιμοποιείται το πρόγραμμα τόσο πιο πετυχημένο είναι το user interface στον τομέα αυτόν [Nielsen 2000].

5. Λάθη (Errors)

Ένας αρνητικός δείκτης ο οποίος δηλώνει κακό σχεδιασμό του user interface είναι η συχνότητα εμφάνισης λαθών στον τρόπο χρήσης του προγράμματος από τους χρήστες. Είναι σημαντικό να δημιουργηθεί ένα τέτοιο περιβάλλον που θα ελέγχει και θα αποτρέπει τους χρήστες από το να κάνουν λάθος [Nielsen 2000]. Είναι επίσης σημαντικό να έχουν ενσωματωθεί μηχανισμοί οι οποίοι διευκολύνουν το χρήστη στο να ανανήψει από ένα σημαντικό λάθος. Δεν θα πρέπει οι μη αναστρέψιμες καταστάσεις να συμβαίνουν συχνά. Επίσης η επανάληψη του ίδιου λάθους επανηλλειμένα από τον χρήστη υποδηλώνει λάθος σχεδίαση του interface.

6. Ικανοποίηση (Satisfaction)

Το χαρακτηριστικό αυτό μετράει το βαθμό ικανοποίησης του χρήστη όταν χρησιμοποιεί ή ακόμα και απλώς παρακολουθεί το user interface ενός προγράμματος.

4.3.1.2 Λίστα του Nielsen- Στοιχεία σχεδίασης

Ο Nielsen έχει προτείνει και μια σειρά από ελέγχους για το βαθμό χρηστικότητας έτσι ώστε οι σχεδιαστές να είναι σε θέση να εκτιμήσουν το περιβάλλον Διεπαφής. Παρακάτω δίνεται μια λίστα με τα πιο σημαντικά σημεία που θα πρέπει να προσεχθούν κατά τη σχεδίαση ενός user interface, η οποία είναι γνωστή και ως λίστα του Nielsen [Nielsen 2000]:

Διαφάνεια της κατάστασης του συστήματος.	Το σύστημα θα πρέπει πάντα να ενημερώνει τους χρήστες για το τι συμβαίνει ανά πάσα στιγμή στην εφαρμογή.
Το σύστημα θα πρέπει να ταιριάζει με τον πραγματικό κόσμο.	Το σύστημα θα πρέπει να χρησιμοποιεί τη γλώσσα των χρηστών και όρους από την πραγματική ζωή και όχι τεχνική ορολογία. Θα πρέπει η πληροφορία να παρουσιάζεται με όσο το δυνατό πιο φυσιολογικό τρόπο.
Ελευθερία και έλεγχος χρήστη.	Οι χρήστες θα πρέπει ακόμα κι αν επιλέξουν κατά λάθος μια επιλογή να μπορούν να επιστρέψουν σε κάποια αρχική κατάσταση και όχι να εγκλωβιστούν.
Συνέχεια και σταθερότητα.	Οι χρήστες δεν πρέπει να αναρωτιούνται διαφορετικές περιπτώσεις, ενέργειες ή λέξεις μπορεί να σημαίνουν το ίδιο πράγμα.
Αποφυγή λαθών	Ο σχεδιασμός της εφαρμογής θα πρέπει να γίνει με προσοχή έτσι ώστε να προλαμβάνεται η περίπτωση λάθους. Καλό είναι οι λάθος ενέργειες να αποτρέπονται εξαρχής παρά η διαδικασία να καταλήγει σε μηνύματα λάθους.
Αναγνώριση παρά απομνημόνευση.	Ο σχεδιασμός του UI θα πρέπει να αποσκοπεί στη δημιουργία τέτοιων αντικειμένων που ο χρήστης δε θα χρειάζεται να θυμάται πληροφορίες περνώντας από ένα τμήμα σε ένα άλλο καθώς επίσης και οδηγίες για τη χρήση της εφαρμογής θα πρέπει να είναι εύκολα προσβάσιμες.
Προσαρμοστικότητα και αποτελεσματικότητα	Θα πρέπει το UI να συνδυάζει και την ευκολία αλλά και τη δυνατότητα για γρήγορη χρήση των λειτουργιών. Θα πρέπει δηλαδή να απευθύνεται και σε αρχάριους αλλά και σε έμπειρους χρήστες.
Καλαίσθητος και λιτός σχεδιασμός.	Τα παράθυρα διαλόγου μιας εφαρμογής θα πρέπει να περιέχουν τις στοιχειώδεις πληροφορίες που είναι αρκετές για την ολοκλήρωση της εργασίας και όχι περιττές πληροφορίες που μπορεί να είναι και άσχετες με το θέμα.
Αναγνώριση λαθών και ανάνηψη	Θα πρέπει να εμφανίζονται άμεσα μηνύματα σφάλματος όταν γίνει κάποιο λάθος, τα οποία θα πρέπει να είναι εκφρασμένα σε απλή και κατανοητή γλώσσα.
Βοήθεια και Τεκμηρίωση	Καλό είναι να μπορεί να λειτουργήσει ένας χρήστης μια εφαρμογή χωρίς τη χρήση βοήθειας ή

	εγχειριδίου οδηγιών. Παρ' όλα αυτά θα πρέπει να υπάρχει εύκολα πρασβάσιμη βοήθεια και τεκμηρίωση της εφαρμογής.

4.3.2 Σχεδίαση Προσανατολισμένη προς τον Χρήστης (User-Centered Design - UCD)

Η χρηστικότητα ενός user interface μπορεί να βελτιωθεί σημαντικά αν ακολουθήσει τους κανόνες που υπαγορεύει η Σχεδίαση Προσανατολισμένη προς τον Χρήστης (User-Centered Design - UCD). Σύμφωνα με τον Raossa, [Raossa Katz-Haas 1998] η σχεδίαση προτείνει μια διαδικασία αλλά και μια φιλοσοφία η οποία τοποθετεί στο κέντρο της σχεδίασης τον χρήστη. Πρόκειται για σχεδίαση η οποία εστιάζει σε παράγοντες όπως είναι η προοπτική, η μνήμη, η δυνατότητα εκμάθησης και η ευκολία επίλυσης προβλημάτων κατά τη χρήση ενός προγράμματος. Πρόκειται επίσης για μια μεθοδολογία η οποία προτείνει έναν τρόπο κατασκευής ενός user interface που επικεντρώνεται στη σχεδίαση από τη σκοπιά του χρήστη, της λειτουργίας και του περιβάλλοντος χρήσης [Gullisken 1999]. Η σχεδίαση UCD αναζητά απαντήσεις σε ερωτήσεις σχετικές με τους χρήστες του προγράμματος, τις λειτουργίες που αυτό θα πρέπει να προσφέρει και τους στόχους τους και αφού απαντηθούν οι ερωτήσεις τότε να χρησιμοποιηθούν τα ευρήματα για τη σχεδίαση και ανάπτυξη του interface [Preece 1995]. Τέτοιες ερωτήσεις μπορεί να είναι οι ακόλουθες:

- Ποιοι είναι οι χρήστες του interface.
- Ποιες είναι οι βασικές λειτουργίες και ποιος ο στόχος τους.
- Ποια είναι η εμπειρία των χρηστών στο συγκεκριμένο σύστημα ή σε κάποιο παρόμοιο.
- Ποιες λειτουργίες χρειάζονται οι χρήστες από το πρόγραμμα.
- Ποιες πληροφορίες μπορεί να χρειάζονται οι χρήστες από το πρόγραμμα και σε ποια μορφή τις χρειάζονται.
- Πως πιστεύουν οι χρήστες πως θα έπρεπε να λειτουργεί το σύστημα
- Πως θα μπορούσε το σύστημα να διευκολύνει τις εργασίες των χρηστών

Η σχεδίαση UCD εμπεριέχει τις έννοιες της Χρησιμότητας (Usefulness) και της Χρηστικότητας. Η χρησιμότητα αναφέρεται στη λειτουργικότητα της εφαρμογής και στο κατά πόσο μπορεί το πρόγραμμα να εκτελέσει τις απαραίτητες λειτουργίες που επιθυμεί ο χρήστης ενώ η χρηστικότητα, όπως είπαμε αναφέρεται στην ευκολία χρήσης.

4.3.2.1 Μεθοδολογία σχεδίασης UCD

Μια μεθοδολογία για την ανάπτυξη ενός interface βασισμένου στις αρχές της σχεδίασης UCD έχει προταθεί από την IBM και περιλαμβάνει τα εξής στάδια:

1. Προσδιορισμός της ομάδας χρηστών στους οποίους απευθύνεται (Target group). Αρχικά θα πρέπει να προσδιοριστούν τα χαρακτηριστικά των χρηστών που πρόκειται να χρησιμοποιήσουν την εφαρμογή. Στη συνέχεια θα πρέπει να προσδιοριστούν οι ανάγκες τους τις οποίες θα πρέπει να εξυπηρετεί το user interface. Ο προσδιορισμός αυτός συνήθως επιτυγχάνεται με τη χρήση ερωτηματολογίων προς ομάδες πιθανόν χρηστών όσον αφορά το αντικείμενο στο οποίο ενδιαφέρονται και την

προτεραιότητα των αναγκών που θα πρέπει να ικανοποιούνται μέσα από την εφαρμογή.

2. Ανάλυση Λειτουργιών. Κατά τη διαδικασία της σχεδίασης θα πρέπει να προσδιοριστούν οι στόχοι και οι εργασίες των χρηστών καθώς επίσης και να εξεταστούν τα εργαλεία που χρησιμοποιούν ήδη για την εκτέλεση των ίδιων εργασιών. Αυτό μπορεί να γίνει με τη χρήση ερωτηματολογίων έτσι ώστε οι χρήστες να παραθέσουν και να βάλουν σε σειρά προτεραιότητας τις εργασίες που θέλουν να κάνουν μέσα από τη εφαρμογή.

3. Εκτίμηση του ανταγωνισμού. Κατά τη σχεδίαση της εφαρμογής, θα πρέπει να εκτιμηθούν τα πλεονεκτήματα και τα μειονεκτήματα του συστήματος σε σχέση με ανταγωνιστικά συστήματα. Αυτό μπορεί να γίνει καλώντας τους χρήστες να εκτελέσουν τις ίδιες εργασίες μέσα από διάφορα ανταγωνιστικά μεταξύ τους συστήματα και να επισημάνουν τα πλεονεκτήματα α και τα μειονεκτήματα καθώς επίσης και την ικανοποίηση από τη χρήση του κάθε συστήματος.

4. Σχεδιασμός. Χρησιμοποιώντας τα αποτελέσματα από τη φάση της ανάλυσης των λειτουργιών και του ανταγωνισμού, δημιουργούνται πρωτότυπα και προτείνονται διάφορες εναλλακτικές λύσεις τις οποίες αξιολογούν οι χρήστες και ενημερώνουν για τις αλλαγές που πρέπει να γίνουν κατά την ανασχεδίαση.

5. Αποτίμηση και Επικύρωση. Οι χρήστες θα πρέπει να ερωτώνται συνεχώς για τυχόν διορθώσεις και οι απαντήσεις να χρησιμοποιούνται στον επανασχεδιασμό έτσι ώστε να βελτιώνεται το αρχικό σύστημα. Η διαδικασία αυτή συνεχίζεται μέχρι ο βαθμός ικανοποίησης των χρηστών να είναι υψηλός έτσι ώστε να επικυρωθεί η τελική μορφή του συστήματος. Η διαδικασία αυτή γίνεται με αξιολόγηση από τη μεριά του χρήστη κάποιων πρωτοτύπων όσον αφορά τη χρηστικότητά τους.

6. Τελική Αξιολόγηση. Αφού ολοκληρωθεί το σύστημα στην τελική του μορφή τότε καλούνται οι χρήστες να το αξιολογήσουν σε σχέση με τον ανταγωνισμό εκτελώντας κάθε μια λειτουργία από διαφορετικά συστήματα και βαθμολογώντας την ευκολία χρήσης καθώς και την αποδοτικότητα.

4.4 User Interface και Έμπειρα Συστήματα

Οι έρευνες στη Τεχνητή Νοημοσύνη και στο χώρο των Έμπειρων Συστημάτων για την κατανόηση από τον Η/Υ της φυσικής γλώσσας του χρήστη σταδιακά οδηγούν σε υπολογιστές που θα επικοινωνούν με τον άνθρωπο με βάση τη φυσική του γλώσσα σε γραπτή ή προφορική μορφή ενώ η επικοινωνία αυτή μπορεί να εμπλουτίζεται με την χρήση εικόνων από τον υπολογιστή. Ένα τέτοιο παράδειγμα της χρήσης γραφικών σε συνδυασμό με εργαλεία και τεχνικές των Έμπειρων Συστημάτων είναι το STE-AMER. Πρόκειται για ένα έμπειρο σύστημα που αναπτύχθηκε για την διδασκαλία της λειτουργίας ενός ατμοηλεκτρικού εργοστασίου. Το σύστημα αυτό χρησιμοποιεί τους λεγόμενους «δαίμονες» (demons). Οι «δαίμονες» παρακολουθούν συνέχεια μεγέθη για να καταγράφουν τις διακυμάνσεις τους σαν ενδείξεις. Έτσι, δίνεται η δυνατότητα να υπάρχει κάθε χρονική στιγμή μια ζωντανή και δυναμικά μεταβαλλόμενη εικόνα της λειτουργίας του εργοστασίου. Στο σημείο αυτό να αναφέρουμε την έντονη ερευνητική δραστηριότητα που αναπτύσσεται τα τελευταία χρόνια στο πεδίο σχεδιασμού και ανάπτυξης «Περιβαλλόντων Επίλυσης Προβλημάτων» (Problem Solving Environments), τα οποία προσφέρουν ένα ολοκληρωμένο περιβάλλον που καθοδηγεί (με παραστατικό τρόπο) τον χρήστη μέχρι την τελική επίλυση του προβλήματος του.

Μια τέτοια περιοχή προβλημάτων είναι αυτή της Αριθμητικής Ανάλυσης, καθώς και της χρήσης του αντίστοιχου περίπλοκου μαθηματικού λογισμικού που είναι διαθέσιμο σ' αυτήν την περιοχή. Το Έμπειρο Σύστημα που αναφέρθηκε προηγουμένως υπάγεται σε αυτό το ερευνητικό πεδίο. Οι δυνατότητες των «Περιβαλλόντων Επίλυσης Προβλημάτων» προκύπτουν από το εξαιρετικό «φιλικό» περιβάλλον επικοινωνίας με τον χρήστη των συστημάτων αυτών και από την διαφάνεια τους, δεδομένου ότι συχνά ενσωματώνουν ένα Ε.Σ. που καλύπτει την αντίστοιχη γνωστική περιοχή.

5 Περιβάλλοντα Επίλυσης Προβλημάτων

5.1 Τι είναι ένα ΠΕΠ

Ο όρος Περιβάλλοντα Επίλυσης Προβλημάτων (ΠΕΠ) είναι συγκεχυμένος. Όπως και πολλοί όροι στην Πληροφορική Επιστήμη, χρησιμοποιείται συχνά με διαφορετικούς τρόπους. Επιπλέον, τα ΠΕΠ είναι μία σχετικά νέα και διφορούμενη περιοχή έρευνας. Όμως μπορούμε να προσδιορίσουμε σε ένα βαθμό τι είναι ή τι θα έπρεπε να είναι ένα ΠΕΠ και να απαντήσουμε σε κάποια βασικά ερωτήματα όπως ποιες θα έπρεπε να είναι οι δυνατότητές τους, ποιες είναι οι ανάγκες των χρηστών, ποιοι είναι οι χρήστες, ποια είναι κάποιες βασικές υλοποιήσεις κλπ. Έχοντας στο μυαλό μας ότι ο ορισμός ΠΕΠ είναι δύσκολο να συγκεκριμενοποιηθεί, εδώ είναι μερικές απόπειρες ενός ορισμού [Γαλλόπουλος, Χούστης και Rice, 1994]:

1. Ένα ΠΕΠ είναι ένα υπολογιστικό σύστημα που παρέχει όλες τις υπολογιστικές παροχές που είναι απαραίτητες για την επίλυση ενός ορισμένου προβλήματος. Αυτές οι μορφές περιλαμβάνουν προηγμένες μεθόδους επίλυσης, αυτόματη ή ημιαυτόματη επιλογή της μεθόδου επίλυσης, και τρόπους να ενσωματώνουν εύκολα νέες μεθόδους επίλυσης. Επιπλέον, τα ΠΕΠ χρησιμοποιούν την αντίστοιχη γλώσσα των στοχοποιημένων προβλημάτων, έτσι ώστε οι χρήστες να μπορούν να τα τρέξουν χωρίς να απαιτείτε εξειδικευμένη γνώση του υπόβαθρου του ηλεκτρονικού υπολογιστή είτε σε επίπεδο υλικού ή λογισμικού. Εκμεταλλευόμενοι τις μοντέρνες τεχνολογίες όπως διαδραστικά έγχρωμα γραφικά, δυναμικούς επεξεργαστές, και δίκτυα εξειδικευμένων υπηρεσιών, τα ΠΕΠ μπορούν να εντοπίσουν εκτεταμένες διαδικασίες επίλυσης προβλημάτων και να επιτρέψει στους χρήστες να τα ανασκοπήσουν εύκολα. Γενικά, δημιουργούν μια δομή που είναι κατανοητή για όλους τους ανθρώπους έτσι ώστε να μπορούν να επιλύουν απλά ή και πολύπλοκα προβλήματα, να αναπτύσσουν λεπτομερείς αναλύσεις, και να μπορούν να τα χρησιμοποιήσουν στην εισαγωγική εκπαίδευση ή στα αρχικά στάδια της επιστήμης'.' Ένας παρόμοιος ορισμός είναι ο εξής:

2. «Ένα περιβάλλον επίλυσης προβλήματος είναι ένα υπολογιστικό σύστημα που παρέχει ένα ολοκληρωμένο και εύχρηστο σύνολο εργαλείων υψηλού επιπέδου για την επίλυση προβλημάτων σε ένα συγκεκριμένο πεδίο. Το ΠΕΠ επιτρέπει στους χρήστες να καθορίζουν και να διαμορφώνουν προβλήματα, να επιλέγουν στρατηγικές επίλυσης, να αλληλεπιδρούν και να διαχειρίζονται τους απαραίτητους πόρους υλικού και λογισμικού, να οπτικοποιούν και να αναλύουν αποτελέσματα, και να καταγράφουν και να συντονίζουν εκτεταμένες εργασίες επίλυσης προβλημάτων. Ο χρήστης επικοινωνεί με το ΠΕΠ στην γλώσσα του προβλήματος και όχι στην γλώσσα ενός συγκεκριμένου λειτουργικού συστήματος, σε προγραμματιστική γλώσσα, ή πρωτόκολλο δικτύου».

Εδώ είναι μερικές επιπλέον σημειώσεις που βοηθούν να περιγράψουμε το πώς βλέπουμε τα ΠΕΠ και πως αναπτύσσονται:

1. Ένα ΠΕΠ εφαρμόζει τις τελευταίες εξελίξεις της τεχνολογίας που βρίσκεται ακόμα και σε επίπεδο έρευνας εμπλέκοντας πολλούς επιμέρους κλάδους της Επιστήμης των Υπολογιστών έτσι ώστε να μπορέσει δημιουργήσει ένα

δυναμικό και εύχρηστο υπολογιστικό περιβάλλον για συγκεκριμένες περιοχές επίλυσης προβλημάτων κυρίως στα πεδία της επιστήμης, μηχανολογίας και κατασκευαστικής. Οι τεχνολογίες αυτές μπορούν να συμπεριλαμβάνουν τεχνικές τεχνητής νοημοσύνης, κατανεμημένα υπολογιστικά συστήματα, γραφικά και οπτικοποίηση, αλληλεπίδραση ανθρώπου – υπολογιστή, δίκτυα και Διαδίκτυο, αριθμητική ανάλυση, αντικειμενοστραφή δίκτυα, παράλληλα και κατανεμημένα δίκτυα.

2. Εναλλακτικά θα μπορούσαμε να πούμε ότι ένα ΠΕΠ είναι μια υποδομή λογισμικού η οποία θα μπορούσε να βοηθήσει μια ομάδα εκατοντάδων μηχανικών εγκατεστημένων σε διάφορα μέρη οι οποίοι συνεργάζονται στο σχεδιασμό ενός νέου αεροπλάνου και χρησιμοποιώντας μια ετερογενή συλλογή προτύπων, κωδικών και μηχανών. Πιο συγκεκριμένα η μέθοδος σχεδίασης που είναι γνωστή και ως Βελτιστοποίηση Σχεδίασης με Πολλαπλές Προδιαγραφές (multidisciplinary design optimization- MDO) που χρησιμοποιείται στην αεροναυπηγική είναι ένα καλό παράδειγμα πεδίου επίλυσης προβλημάτων όπου ένα ΠΕΠ θα μπορούσε να φανεί χρήσιμο. Στην MDO, τη σχεδίαση ενός αεροσκάφους διέπουν όλες οι αρχές σχεδίασης σχετικά με δυναμική υγρών, συστήματα ελέγχου, και δομές. Σχεδιαστικά προβλήματα μηχανικής μεγάλης κλίμακας είναι ετερογενή σε πολύ μεγάλο βαθμό εμπλέκοντας πολλά μοντέλα, κωδικούς, μηχανικά συστήματα και ανθρώπους σε διαφορετικές τοποθεσίες. Χωρίς ένα εξελιγμένο και εύχρηστο ΠΕΠ, η συγκεκριμένη σχεδίαση είναι πρακτικά αδύνατη.

Η ανάγκη για ΠΕΠ αυξάνεται διαρκώς καθώς η πολυπλοκότητα και ετερογένεια της διαδικασίας επίλυσης ενός προβλήματος αυξάνει. Παλαιότερα ένας ερευνητής που ασχολιόταν με υπολογιστικά προβλήματα χρησιμοποιούσε ένα σύνολο από σχετικά απλούστερα εργαλεία όπως τη γλώσσα προγραμματισμού FORTRAN, ένα υπολογιστή mainframe και κάποια αποτελέσματα σε μορφή πινάκων με αριθμούς και απλών σχεδίων. Σήμερα ένας ερευνητής στο συγκεκριμένο πεδίο έχει στη διάθεσή του μια σειρά από πολλές γλώσσες προγραμματισμού, από τη Fortran90 και τη HPF μέχρι τη C++ και τη Java και πολλά περιβάλλοντα οπτικού προγραμματισμού και αντιμετώπισης προβλημάτων. Οι υπολογιστές που είναι στη διάθεσή τους χρησιμοποιούν τις ταχύτατα αναπτυσσόμενες αρχιτεκτονικές παράλληλου προγραμματισμού ενώ τα αποτελέσματα μπορούν να οπτικοποιηθούν σε τρισδιάστατη μορφή, με animation. Επίσης το περιβάλλον στο οποίο δουλεύουν είναι συνεργατικό με πολλούς ερευνητές από διάφορα μέρη του κόσμου. Είναι προφανής η ανάγκη χρήσης ενός εύχρηστου περιβάλλοντος που θα βοηθάει τους ερευνητές να συνεργαστούν και να διαχειριστούν όλα τα διαθέσιμα μέσα επίλυσης.

5.2 Χαρακτηριστικά ΠΕΠ

Οι ακόλουθοι ορισμοί αναπαριστούν μια λίστα χαρακτηριστικών τα οποία είναι επιθυμητά, άλλα περισσότερο και άλλα λιγότερο, σε ένα ΠΕΠ:

5.2.1 Προσανατολισμένο στο πρόβλημα (Problem oriented).

Το ΠΕΠ θα πρέπει να αφήνει τον επιστήμονα να συγκεντρωθεί στις γνώσεις του, χωρίς να απαιτήτε να γίνει ένας αυτοδίδακτος γνώστης μέσω διαδικτύου.

5.2.2 Ενοποίηση (Integrated).

Πολλά προβλήματα και στρατηγικές επίλυσης είναι ετερογενείς: σε μοντέλα, κωδικούς, εφαρμογές, μηχανισμούς ακόμη και ορολογία. Μια από τις πιο προκλητικές απαιτήσείς ενός καλώς σχεδιασμένου ΠΕΠ είναι το να μπορέσει να ενοποιήσει όλους αυτούς τους ετερογενείς παράγοντες έτσι ώστε να μπορεί ο χρήστης να έχει έστω την ψευδαίσθηση ότι εργάζεται σε ένα αναμενόμενο και επίμονο περιβάλλον.

5.2.3 Διανεμημένο (Distributed).

Ένα ΠΕΠ μπορεί να χρειαστεί να διανεμηθεί έτσι ώστε να διευκολύνει την απομακρυσμένη συνεργασία, ή απλά να εξυπηρετήσει κάποιον χρήστη σε απομακρυσμένο μέρος.

5.2.4 Συνεχιζόμενο (Persistent).

Οι διαδικασίες επίλυσης προβλημάτων δεν είναι μεμονομένα γεγονότα. Η διαδικασία διανέμεται τόσο σε χρόνο όσο και σε διαστήματα χρόνου. Ένα ΠΕΠ θα πρέπει να το έχει προβλέψει αυτό ίσως με κάπια έννοια συνεχιζόμενων εργασιών ή με μια αποθήκη προβλημάτων και επιλύσεων κτλ. Κάθέ διαδικασία επίλυσης προβλήματος έχει κάποιο στοιχείο από προηγούμενους υπολογισμούς που μπορεί να χρειαστεί να αναχαιτήσει είναι πιθανόν αν εξάγει συμπεράσματα και λύσεις που θα χρησιμοποιηθούν σε επόμενες διαδικασίες.

5.2.5 Δυναμικά (Powerful).

Δεν έχει σημασία πόσο καλό φαίνεται ένα ΠΕΠ αν δεν έχει διαθέσιμο δυναμικό υλικό και λογισμικούς πόρους ώστε να μπορεί να επιλύσει προβλήματα μέγιστου ενδιαφέροντος. Σημειώστε ότι παρόλα αυτά, σε εκπαιδευτικά συστήματα μπορεί η καλή χρήση να πραγματοποιηθεί και από λιγότερο δυναμικά ΠΕΠ.

5.2.6 Ανοιχτά, ελαστικά, προσαρμόσημα (Open, flexible, adaptive).

Σε πολλά περιβάλλοντα είανι σημαντικό οι σχεδιαστές των ΠΕΠ και έμπειροι χρήστες να μπορούν να ενισχύουν την λειτουργικότητα ενός ΠΕΠ.

5.2.7 Γραφιστικά (Graphical, visual).

οι εφαρμογές μεγαλύτερου σκέλους απαιτούν οπτικοποίηση των αποτελεσμάτων, πολλά τα οποία βασίζονται σε γραφικά δεδομένα.

5.2.8 Έξυπνα (Intelligent).

Σε συγκεκριμένες ρυθμίσεις είναι πιθανό να μπορεί ένα ΠΕΠ να παρέχει μερικές έμπειρες «συμβουλές», να επιλέξεις ανάμεσα σε πολλαπλές μαθηματικές μεθόδους ΄΄η ακόμη και να συμβουλέψει έναν εργοδηγό σε ένα εργοστάσιο σχετικά με τις διαδικασίες ελέγχου.

5.3 Στάδια Επίλυσης Προβλημάτων

Ένας σημαντικός παράγοντας για να κατανοήσουμε τα Περιβάλλοντα Επίλυσης Προβλημάτων είναι να καθορίσουμε ποιες επιμέρους φάσεις περιλαμβάνει μια διαδικασία επίλυσης προβλήματος. Τα επιμέρους αυτά στάδια είναι αυτά τα οποία ένας αναλυτής θα επιθυμούσε να περιέχονται σε ένα ΠΕΠ. Έτσι κάποια από τα πιο σημαντικά στάδια στη διαδικασία επίλυσης προβλημάτων είναι τα εξής:

5.3.1 Έρευνα.

Ο επιστήμονας που καλείται να επιλύσει ένα πρόβλημα επιθυμεί να έχει στη διάθεσή του κάποια μοντέλα ή αλγορίθμους τους οποίους μπορεί και να συγκρίνει μέσα από σενάρια. Έτσι χρησιμοποιώντας πειραματικά κάποια δεδομένα μπορεί να συγκρίνει διάφορους αλγορίθμους έτσι ώστε να επιλέξει τον πιο κατάλληλο ή να σχεδιάσει ένα καλύτερο μοντέλο το οποίο θα χρησιμοποιηθεί στη διαδικασία επίλυσης.

5.3.2 Σχεδιασμός και ανάπτυξη.

Ένας μηχανικός προσπαθεί διαρκώς να σχεδιάσει ένα καλύτερο σύστημα. Έτσι στη διαδικασία επίλυσης είναι σημαντικό να υπάρχουν εκείνα τα εργαλεία που θα του δώσουν τη δυνατότητα να σχεδιάσει ένα καλύτερο μοντέλο. Συνήθως τα μαθηματικά μοντέλα είναι σταθερά και συγκεκριμένα παρόλα αυτά επιτρέπουν κάποιο είδος παραμετροποίησης. Η διαδικασία σχεδιασμού ενός καλύτερου μοντέλου μπορεί να είναι αυτοματοποιημένη σε ένα βαθμό αλλά συνήθως χρειάζεται σε κάθε βήμα τη παρέμβαση του μηχανικού..

5.3.3 Διαδικασία Παραγωγής.

Ένας διευθυντής εργοστασίου ή ένας υπεύθυνος τεχνικού τομέα θα πρέπει να παρακολουθεί διαρκώς τη διαδικασία παραγωγής και και να προσαρμόζει τη διαδικασία ανάλογα με τις εκάστοτε συνθήκες π.χ. ιδιότητες των πρώτων υλών. Θα πρέπει να υπάρχει η δυνατότητα προσαρμογής του μοντέλου στις συνθήκες που θα προκύψουν κατά τη διάρκεια της παραγωγής. Την επίπτωση του μοντέλου στη διαδικασία παραγωγής όσον αφορά την ποιότητα και το χρόνο παράδοσης ένα περιβάλλον επίλυσης προβλημάτων θα πρέπει να μπορεί να τις προβλέπει.

5.3.4 Εκπαίδευση.

Στην προσπάθεια να κατανοήσουμε μια περιοχή του προβλήματος είναι πολύ σημαντικό ο αναλυτής να έχει στη διάθεσή του μια σειρά από μοντέλα, παραδείγματα παρόμοιων περιπτώσεων και μεθόδων έτσι ώστε να μπορεί να τις μελετήσει, να τις κατανοήσει και να είναι σε θέση να αντιμετωπίσει τα νέα προβλήματα.

5.4 Τεχνολογίες ΠΕΠ

Όπως προκύπτει από τα παραπάνω, ένα Περιβάλλον Επίλυσης Προβλημάτων δεν είναι απλώς μια αρχιτεκτονική ή μια συγκεκριμένη εφαρμογή λογισμικού αλλά είναι ένα σύνολο τεχνολογιών από το σύστημα διεπαφής με το χρήστη (user Inteface) μέχρι το μηχανισμό επεξεργασίας των δεδομένων, την παρουσίαση των αποτελεσμάτων και το μηχανισμό ανάδρασης και επανατροφοδότησης των αποτελεσμάτων για την αναπροσαρμογή της διαδικασίας επίλυσης [J.R. Rice & R.F. Boisvert 1996]. Συνεπώς περιγράφοντας ένα ΠΕΠ, προσπαθούμε να περιγράψουμε την υποδομή και τις τεχνολογίες που απαιτούνται για ένα πλήρες σύστημα. Βέβαια μεγαλύτερη σημασία από όλα τα επιμέρους τμήματα έχει ο μηχανισμός επεξεργασίας των δεδομένων για την παραγωγή αποτελεσμάτων και σε αυτόν τον τομέα χρησιμοποιείται η αιχμή της τεχνολογίας όπως είναι τα κατανεμημένα συστήματα και τα έμπειρα συστήματα καθώς και τα περιβάλλοντα προσομοίωσης.

Πιο συγκεκριμένα λοιπόν οι βασικές τεχνολογίες που χρησιμοποιούνται από τα ΠΕΠ είναι οι εξής:

5.4.1 Αλληλεπίδραση ανθρώπου μηχανής (Human-computer interaction – HCI).

Είναι σημαντικό να υπάρχει ένα περιβάλλον διαχείρισης του συστήματος που θα προσφέρει στον αναλυτή και ερευνητή του προβλήματος όλα τα απαραίτητα εργαλεία αλλά μέσα από ένα φιλικό και εύχρηστο user interface. Το σύστημα διεπαφής θα πρέπει να είναι καλά δομημένο και να εξυπηρετεί τις ανάγκες τόσο των ερευνητών καθώς και των απλών χρηστών.

5.4.2 Τεχνητή νοημοσύνη.

Σε προβλήματα συγκεκριμένου γνωστικού αντικειμένου χρησιμοποιούνται συστήματα τα οποία λειτουργούν με τη χρήση κάποιων κανόνων λήψης αποφάσεων σε συνδυασμό με προηγούμενη γνώση που είναι και αυτή ενσωματωμένη στο σύστημα.

5.4.3 Παράλληλος προγραμματισμός.

Πολύπλοκα προβλήματα απαιτούν μεγάλους υπολογιστικούς πόρους όσον αφορά την επεξεργαστική ισχύ. Έτσι μια από τις τεχνολογίες που χρησιμοποιείται σε περίπτωση επίλυσης προβλημάτων πολύ μεγάλης κλίμακας είναι η πολυεπεξεργασία ή επεξεργασία δηλαδή των δεδομένων από πολλούς επεξεργαστές ταυτόχρονα. Ο προγραμματισμός των επεξεργαστών έτσι ώστε να δουλεύουν παράλληλα ονομάζεται παράλληλος και μπορεί να προσφέρει τεράστιες δυνατότητες επεξεργαστικής ισχύος.

5.4.4 Μαθηματικά μοντέλα.

Θα πρέπει να υπάρχει μια υποδομή από μαθηματικά μοντέλα και τεχνικές μαθηματικής ανάλυσης. Τα μοντέλα αυτά θα πρέπει να είναι στη διάθεση των αναλυτών μέσα από ειδικές βιβλιοθήκες που θα περιλαμβάνονται στο ΠΕΠ.

5.4.5 Οπτικοποίηση αποτελεσμάτων.

Τα αποτελέσματα των ερευνών θα πρέπει να παρουσιάζονται από έναν εύκολα κατανοητό τρόπο αναπαράστασης που μπορεί να οδηγήσει του ερευνητές στην πιο εύκολη εξαγωγή συμπερασμάτων. Συνεπώς η οπτικοποίηση των αποτελεσμάτων είναι και αυτή ένα σημαντικό τμήμα και απαίτηση του user interface του ΠΕΠ.

6 Συστήματα Προσομοίωσης

Προσομοίωση (simulation) είναι η αναπαράσταση μιας διεργασίας με τη βοήθεια ενός μοντέλου. Σκοπός της προσομοίωσης είναι η μελέτη της συμπεριφοράς ενός συστήματος (με «δειγματοληπτική» τεχνική), όταν η εφαρμογή αναλυτικών ή άλλων αριθμητικών μεθόδων είναι ανέφικτη ή ιδιαίτερα δυσχερής. Πολλές φορές για τη λήψη κάποιων αποφάσεων θα ήταν προτιμότερο να γίνονταν δοκιμές με υποθετικά σενάρια πάνω στο μοντέλο ενός συστήματος και όχι στο πρωτότυπο σύστημα έτσι ώστε να γίνονταν αντιληπτές οι επιπτώσεις των ενεργειών αυτών στην κατάσταση του συστήματος. Η αναπαράσταση αυτή λοιπόν δίνει τη δυνατότητα να δοκιμαστεί ένα υποθετικό σενάριο σε ένα εικονικό σύστημα κάνοντας έτσι την παρέμβαση στο σύστημα οικονομικότερη, ταχύτερη και λιγότερο επικίνδυνη.

6.1.1 Είδη προσομείωσης

Η προσομοίωση διακρίνεται σε:

1. Προσομοίωση συνεχών διεργασιών (continuous process simulation) (διαφορικές εξισώσεις)
2. Προσομοίωση διακριτών γεγονότων (discrete event simulation)

6.2 Εφαρμογές προσομείωσης

Μερικές από τις εφαρμογές της προσομοίωσης είναι:

- Ο πειραματισμός πάνω σε ένα σύστημα (π.χ. μελέτη εναλλακτικών σεναρίων)
- Η στατιστική επαγωγή όπως προσδιορισμός πιθανοτήτων και οι στατιστικοί έλεγχοι Monte Carlo)
- Στοχαστική βελτιστοποίηση για αντικειμενικές συναρτήσεις με πολλά ακρότατα ή που δεν προσδιορίζονται αναλυτικά.

Η βασική διαδικασία για να μπορεί να επιτευχθεί η προσομοίωση είναι η μοντελοποίηση. Μοντέλο (model, από το λατινικό modus = τρόπος, μέτρο) είναι κάτι που αντιπροσωπεύει κάτι άλλο (το πρωτότυπο), δίνοντας έμφαση στα πιο σημαντικά χαρακτηριστικά του χάριν ευκολίας στην ανάλυσης και αποτελεσματικότητας [Chartrand, 1985]. Παραδείγματα μοντέλων είναι ένας χάρτης με μια περιοχή υπό κλίμακα, τα αρχιτεκτονικά σχέδια ή η μακέτα κτηρίου, ένα πρότυπο ή πιλοτικό σύστημα, μια σμίκρυνση ενός μοντέλου αεροπλάνου κ.λ.π.

Με τον όρο Μαθηματικό μοντέλο (ή απλώς μοντέλο στην επιστημονική γλώσσα) αναφερόμαστε σε ένα μαθηματικό σύστημα που αντιπροσωπεύει μια πραγματική οντότητα ή κατάσταση όπως για παράδειγμα ο Ευκλείδειος χώρος, ή ένα μαθηματικό σύνολο, ή ένα σύνολο εξισώσεων.

Η Προσομοίωση (simulation) είναι μια τεχνική μίμησης ενός πραγματικού συστήματος, όπως αυτό εξελίσσεται στο χρόνο [Winston, 1994]. Μοντέλο προσομοίωσης (simulation model) είναι ένα σύνολο υποθέσεων για τη λειτουργία του συστήματος, εκφρασμένων υπό μορφή μαθηματικών ή λογικών σχέσεων μεταξύ των αντικειμένων του συστήματος (και συνήθως κωδικοποιημένων σε πρόγραμμα υπολογιστή).

Υπάρχουν διάφορα μοντέλα προσομοίωσης. Υπάρχουν τα στοχαστικά ή αλλιώς: Monte Carlo·που περιλαμβάνουν τυχαίους αριθμούς ή τα ντετερμινιστικά με συγκεκριμένες τιμές. Υπάρχουν επίσης τα διακριτά όπου οι μεταβλητές κατάστασης αλλάζουν τιμή σε διακριτές χρονικές στιγμές και από την άλλη τα συνεχή όπου η κατάσταση του συστήματος αλλάζει διαρκώς.

6.3 Σύνδεση ενός Έμπειρου Συστήματος με ένα σύστημα Προσομοίωσης

Η σύνδεση ενός Έμπειρου Συστήματος με ένα σύστημα Προσομοίωσης αρκετές τεχνικές δυσκολίες σε βαθμό που καθιστούσε την παρέμβαση του ανθρώπου απαραίτητη για την επικοινωνία των δύο συστημάτων. [Nissen, 1998]. Ο χειριστής σε αυτή την περίπτωση αναλαμβάνει να χρησιμοποιήσει ένα έμπειρο σύστημα για την λήψη μιας απόφασης και τα αποτελέσματα να τα εισάγει σε ένα σύστημα προσομοίωσης για να ελέγξει τα αποτελέσματα μιας τέτοιας απόφασης σε ένα εικονικό περιβάλλον. Σε περίπτωση που θέλει να υπάρχει επανατροφοδότηση των δεδομένων στο έμπειρο σύστημα μετά την δοκιμή θα πρέπει και αυτή να γίνει ομοίως μέσω του χειριστή. Οι Flitman and Hurrion το 1987 [Flitman & Hurrion,1987]ήταν από τους πρώτους που επιχείρησαν να συνδέσουν ένα σύστημα προσομοίωση υλοποιημένο σε γλώσσα Προγραμματισμού Fortran με ένα Έμπειρο Σύστημα υλοποιημένο σε Prolog για να παράγουν ένα ενοποιημένο σύστημα λήψης αποφάσεων και προσομοίωσης. Ομοίως ο O'Keefe το 1989 [O'Keefe , 1989] υλοποίησε ένα εργαλείο που ενοποιούσε ένα σύστημα GPSS με ένα έμπειρο σύστημα σε Prolog.

Μια άλλη προσπάθεια που έχει γίνει στο παρελθόν είναι η ανάπτυξη αρχιτεκτονικών που θα επιτρέπουν τη σύνδεση οποιουδήποτε Έμπειρου Συστήματος με κάποιο σύστημα προσομοίωσης και όχι συγκεκριμένου software. Οι Artiba και Aghezzaf το 1997 [Artiba & Aghezzaf ,1997]ανέπτυξαν μια παρόμοια αρχιτεκτονική για τη σύνδεση Έμπειρων συστημάτων με αλγορίθμους βελτιστοποίησης και ευριστικούς αλγορίθμους για την παρακολούθηση πολύπλοκων προβλημάτων σχεδιασμού της Παραγωγής και χρονοπρογραμματισμού της. Ο Zeigler το 1996 [Zeigler , 1996] ενσωμάτωσε ένα Έμπειρο Σύστημα σε περιβάλλον DEVS.

Από την άλλη πλευρά πολλοί έχουν καταφέρει να συνδέσουν συγκεκριμένες υλοποιήσεις Έμπειρων Συστημάτων και Συστημάτων Προσομοίωσης όπως οι Lyu και Gunasekaran το 1997 [Lyu & Gunasekaran , 1997], που ανέπτυξαν ένα συγκεκριμένο μοντέλο προσομοίωσης μαζί με ένα έμπειρο σύστημα στο περιβάλλον SIMSCRIPTII.5 για να μοντελοποιήσουν την φόρτωση χάλυβα σε ένα λιμάνι. Η επισήμανσή τους βέβαια ήταν ότι αυτό ήταν εφικτό μόνο αν οι κανόνες του Έμπειρου Συστήματος ήταν αρκετά απλοί για να μπορέσουν να περιγραφούν στο σύστημα προσομοίωσης. Επίσης οι Standridge και Steward το 2000 [Standridge & Steward ,2000] συνέδεσαν το σύστημα προσομοίωσης SLAMSYSTEM με ένα έμπειρο σύστημα αναπτυγμένο σε C. Το 2003 οι Robinson, Edwards και Wu Yongfa [Robinson, Edwards , Wu Yongfa,2003] περιέγραψαν πως ένα εμπορικό πακέτο μπορεί να αναπτυχθεί και να χρησιμοποιηθεί για να συνδέσει δύο συγκεκριμένα συστήματα Προσομοίωσης και Λήψης Αποφάσεων, του Witness της Lanner Group και του πακέτου XpertRule της Attar Software,αντίστοιχα.

6.4 Τα Πλεονεκτήματα και Μειονεκτήματα της προσομοίωσης

Είναι προφανής η ευκολία εφαρμογής αλλαγών σε ένα εικονικό σύστημα με την χρήση ενός μοντέλου προσομοίωσης. Τα αποτελέσματα είναι άμεσα χωρίς να είναι ανάγκη να εφαρμοστούν οι αλλαγές στο πραγματικό σύστημα και χωρίς να υπάρχει ο κίνδυνος αποτυχίας. Έτσι οι αναλυτές μπορούν εκ του ασφαλούς να εφαρμόσουν υποθετικά σενάρια ελαχιστοποιώντας το κόστος και την επικινδυνότητα των ενεργειών. Από την άλλη πολλές φορές όταν το μοντέλο είναι πολύπλοκο απαιτούνται τεράστιοι υπολογιστικοί πόροι για να μην καταλήξει η προσομοίωση να είναι μια αργή και χρονοβόρα υπολογιστική διαδικασία. Επίσης η ακρίβεια των αποτελεσμάτων εξαρτάται από το μέγεθος της δειγματοληψίας των δεδομένων που έχουν εισαχθεί και είναι πιθανόν τα αποτελέσματα να απέχουν σε ένα βαθμό από την πραγματική εικόνα αν η δομή του μοντέλου δεν αντιπροσωπεύει το πραγματικό σύστημα.

6.5 Προσομείωση Χρηματαγορών

Ένας προσομειωτής χρηματαγοράς είναι ένα πρόγραμμα ή εφαρμογή που επιχειρεί να αναπαραγάγει ή να αντιγράψει μερικά ή όλα τα χαρακτηριστικά μιας πραγματικής χρηματαγοράς σε έναν υπολογιστή, έτσι ώστε ο χρήστης να μπορεί εξασκηθεί στην διαπραγμάτευση μετοχών χωρίς κανέναν οικονομικό κίνδυνο.

6.5.1 Είδη Προσομείωσης Χρηματαγορών

Οι προσομειωτές χρηματαγορών μπορούν να χωριστούν σε δύο μεγάλες κατηγορίες:

1. Χρηματοοικονομικοί Προσομειωτές
2. Προσομειωτές Φαντασίας

6.5.1.1 Χρηματοοικονομικοί Προσομειωτές

Οι χρηματοοικονομικοί προσομειωτές επιτρέπουν στους χρήστες να δημιουργήσουν ένα χαρτοφυλάκιο με βάση τα πραγματικά δεδομένα μετοχών που υπάρχουν στην αγορά, αλλά με φανταστικά χρήματα. Οι περισσότεροι από τα υπάρχοντες προσομειωτές αυτού του είδους χρησμοποιούν μια χρονοκαθυστερημένη ενημέρωση 15 με 20 λεπτών για να διασφαλίσουν ότι οι χρήστες δε θα χρησιμοποιήσουν τα δεδομένα για εμπορικούς λόγους σε κάποιο ανταγωνιστικό σύστημα. Ο σκοπός αυτών των συστημάτων είναι για να βοηθήσουν τους χρήστες να εξασκηθούν σε πραγματικά δεδομένα χρηματαγορών αλλά με εικονικά χρήματα έτσι ώστε να εξακριβώσουν το αν θα μπορούσαν να ανταπεξέλθουν στην πραγματικότητα και να αξιοποιήσουν σωστά τα κέρδη τους.

6.5.1.2 Προσομειωτές Φαντασίας

Οι προσομειωτές φαντασίας αφορούν μετοχές ή εμπορεύματα αντικειμένων του πραγματικού κόσμου τα οποία δεν περιλαμβάνονται στις λίστες ανταλλασσόμενων προϊόντων όπως οι ταινίες και οι τηλεοπτικές εκπομπές. Κάποιοι από αυτούς τους προσομεοωτές εστιάζουν στον αθλητισμό και έχουν συνδεθεί με το live betting “ζωντανό στοίχημα”.

6.5.2 Τεχνολογία και Εφαρμογές

Οι περισσότερες εφαρμογές των online προσομειωτών χρηματαγοράς τρέχουν σε Java, JavaScript, ASP ή php με μια mysql βάση δεδομένων. Μερικά από αυτά είναι «ανοικτού κώδικα», και άλλα ιδιόκτητα με κώδικα που πωλείται ως ένα πολύτιμο λογισμικό πρόβλεψης της αγοράς. (παράρτημα1). Ένα τέτοιο παράδειγμα είναι το HSX (**Hollywood Stock Exchange**)Virtual Specialist. Αυτή η τεχνολογία έχει πωληθεί σε μεγάλα κινηματογραφικά στούντιο όπως MGM και Lion’s Gate Films, όπως και στην Popular Science team (Popsci) για χρήση στο PPX σύστημά τους.
Οι προσομειωτές χρηματαγοράς μπορούν επίσης να χρησιμοποιηθούν και για άλλες λειτουργίες . Η μηχανή HSX έχει τροποποιηθεί για να ανιχνεύει τις τάσεις της επιστήμης όπως και βίντεο στο YouTube.

6.5.3 Παιχνίδια Χρηματαγορών

Τα παιχνίδια χρηματαγορών είναι θεωρητικά παιχνίδια που επιτρέπουν στους παίκτες να ανταλλάξουν μετοχές σε ένα μια εικονική ή προσομειωμένη χρηματαγορά.

Τα παιχνίδια χρηματαγορών υπάρχουν σε διάφορες μορφές αλλά η βασική ελλοχεύουσα έννοια είναι ότι αυτά τα παιχνίδια επιτρέπουν στους παίκτες να αποκτήσουν την εμπειρία ή απλά να ψυχαγωγηθούν ανταλάσσοντας μετοχές σε έναν εικονικό κόσμο που δεν υπάρχει πραγματικό ρίσκο. Μερικά από αυτά τα παιχνίδια δεν σχετίζονται με κανέναν τρόπο με πραγματικά λεφτά. Οι παίκτες ανταγωνίζονται μεταξύ τους στο ποιος θα προβλέψει την επικείμενη πορεία των μετοχών. Πολλά παιχνίδια χρηματαγόρών βασίζονται σε πραγματικά δεδομένα μετοχών από τη Nasdaq, NYSE και άλλες μεγάλες χρηματοοικονομικές αγορές.

Ένας άλλος σκοπός χρήσης τους είναι η εκπαίδευση των μελλοντικών χρηματιστών στην χρηματιστηριακή αγορά.

7 Επίλογος

Τα Περιβάλλοντα Επίλυσης Προβλημάτων είναι αναμφίβολλα, πολυσύνθετα συστήματα λογισμικού και αναδεικνύουν τη σημασία μιας επεκτάσιμης σχεδίασης, η αξία της οποίας φαίνεται άμεσα, πρακτικά, στην ευκολία (ή τη δυσκολία) εμπλουτισμού τους με καινούργια τεχνικά χαρακτηριστικά. Τα προβλήματα που εντοπίζονται αν και αρκετά δεν είναι μη επιλύσιμα. Τα κυριότερα προβλήματα εντοπίζονται στους τομείς έρευνας και σχεδιασμού των περιβάλλοντων επίλυσης προβλημάτων. Συνοπτικά διακρίνονται σε:

1. Εκμαίευση γνώσης
2. Γλώσσα προγραμματισμού και σχεδίασης
3. Σωστό σχεδιασμό από πλευράς υλικών και αλγορίθμων
4. Ανάπτυξη αυτοματοποιημένων εργαλείων για ευκολότερη σχεδίαση και χρησιμοποίηση
5. Μείωση κόστους υλοποίησης
6. Επεκτασιμότητα βιβλιοθηκών και βάσεων δεδομένων
7. Ευπροσάρμοστα στις εκάστοτε συνθήκες που επικρατούν
8. Καλά δομημένη user interface
9. Σωστά συστήματα προσομείωση χωρίς επιπλέον κόστος χρησιμοποίησης

Όπως προαναφέραμε τα Περιβάλλοντα επίλυσης προβλημάτων είναι διεπιστημονικά εργαλεία, με μεγάλο αντίκτυπο στην καθημερινότητα του επιστήμονα των εφαρμογών, άρα αφενώς οι κατασκευαστές τους οφείλουν να έχουν βαθιά γνώση των ζητημάτων που ενέχονται στη σύνθεσή τους αλλά και ο επιστήμονας των υπολογισμών να διδαχτεί να δομεί τους κώδικές του ώστε να είναι ευκολότερο να ενσωματωθούν σε ΠΕΠ (PSE Friendly), και άρα προσβάσιμοι από περισσότερους χρήστες ειδικότερα στους τομείς εκπαίδευσης και υγείας που όπως βλέπουμε χρησιμοποιείται ευρύτατα τα τελευταία χρόνια. Ήδη έχουν αρχίσει να αναπτύσσονται πλατφόρμες ΠΕΠ οι οποίες να υλοποιούν αριθμητικούς, γραφικούς και συμβολικούς υπολογισμούς όπως το μοντέλο JYLAB (Κόλλιας Γ., 2009).

Συγκεκριμένα τα στοιχεία που θα πρέπει να ερευνηθούν και να αναπτυχθούν στο μέλλον, μερικά από τα οποία έχουν ήδη αρχίσει να αναπτύσσονται είναι:

1. Οικονομικότερες πλατφόρμες για σχεδιασμό περιορισμένων ή μεγαλύτερης έκτασης υπολογισμών
2. Απομακρυσμένες υπηρεσίες υπολογιστών και servers τα οποία να υποστηρίζουν τα συγκεκριμένα συστήματα
3. Να αναπτυχθούν κατάλληλα και αποτελεσματικότερα οι πολυτομεακές εφαρμογές
4. Τα συστήματα διεπαφής προσανατολισμένα στο χρήστη να έχουν περισσότερες χρηστικότητες. Ήδη έχουν αρχίσει να αναπτύσσονται συστήματα προηγμένης τεχνολογίας όπως το Project Natal, το Aurora, το Ringo και άλλα πολλά συστήματα που θα βοηθούν και τους χρήστες αλλά και τους ερευνητές στην καλύτερη και αποδοτικότερη χρησιμοποίηση των συστημάτων
5. Όπως αναφέραμε και παραπάνω να εφαρμοστεί η ενσωμάτωση της φυσικής γλώσσας σε κάθε επίπεδο βάσεων δεδομένων και βιβλιοθηκών που περιλαμβάνει ένα ΠΕΠ.
6. Να αυτοματοποιηθούν διαδικασίες έτσι ώστε η αναβάθμιση των συστημάτων και η προσαρμογή τους ανάλογα με τις εκάστοτε συνθήκες που επικρατούν να γίνεται πιο άμεσα και χωρίς τη συνεχή επέμβαση των σχεδιαστών και μηχανικών βάσεων

7. Να συνδυαστούν καλύτερα τα υλικά και οι αλγόριθμοι ώστε να υπάρξει αύξηση της υπολογιστικής ισχύος

Αν όλα αυτά υλοποιηθούν, και η υπολογιστική δύναμη αυξηθεί σε τέτοιο βαθμό, έχουμε να αναμένουμε εξελίξεις σε πολλούς τομείς όπως:

1. Τα αριθμητικά μοντέλα θα γίνουν πιο ακριβή και ολοκληρωμένα. (περισσότερα φυσικά, χημικά, φαινόμενα θα μπορούν πλέον να ενσωματωθούν σε μοντέλα ενώ οι οι προσομειώσεις σε 3D και 4D θα μπορούν να χρησιμοποιηθούν σε ευρύτερο κοινό και για περισσότερους σκοπούς)
2. Η οπτικοποίηση των αποτελεσμάτων θα μπορεί να γίνει μέσω συστημάτων προσομείωσης
3. Οι συσκευές θα ελέγχονται από υπολογιστές (όπως υπάρχει σήμερα ο αυτόματος πιλότος ή τα συστήματα αυτόματου παρκαρίσματος σε αυτοκίνητα αναμένεται οι διαδικασίες αυτές να χρησιμοποιηθούν και στην ιατρική όπως ρομποτικές χειρουργικές επεμβάσεις και σε άλλους τομείς)
4. Το υπολογιστικό περιβάλλον θα μπορεί αν κατανεμηθεί και να διανεμηθεί στο έπακρον έτσι ώστε οι πλατφόρμες και τα περιβάλλοντα επίλυσης προβλημάτων να είναι προσπελάσιμα από όλον τον κόσμο.

Το μέλλον της υπολογιστικής επιστήμης γενικότερα έχει πολλά πειθώρια ανάπτυξης και εξέλιξης σε βαθμό τέτοιο που ακόμη δεν μπορούμε να προβλέψουμε το τι μπορεί να υπάρξει στις επόμενες δεκαετίες. Όπως παλαιότερα ο κόσμος δεν μπορούσε να φανταστεί ότι θα μετακινόταν «πετώντας» έτσι και εμείς σήμερα δεν μπορούμε να φανταστούμε τι μπορεί να μας επιφυλάσσει το μέλλον και πόσο καλό ή κακό μπορεί αυτό να ενεργήσει για την ανθρώπινη φύση.

8 Βιβλιογραφία

1. Γαλλόπουλος, Χούστης και Rice, “Problem-solving environments for computational science", *IEEE Computational Science & Engineering*, 1, 1994
2. J.R. Rice & R.F. Boisvert, ``From scientific software libraries to problem-solving environments,'' *IEEE Computational Science & Engineering*, Fall, 1996
3. Σημειώσεις Μαθήματος «Τεχνητή Νοημοσύνη και Έμπειρα Συστήματα» ΕΑΠ, Πάτρα 2000.
4. K. MORIK, 1987,”Acquiring domain models”, Int. Journal of Man-Machine Studies, 93-104.
5. D. WATERMAN. 1986. “A Guide to Expert Systems”, Addison-Wesley
6. C.S. Pierce (1931-58) Collected Papers, Harvard University Press, Mass, USA
7. RAOSSA KATZ-HAAS (1998), “Usability Interface, Vol 5, No. 1, July 1998”
8. GULLIKSEN, J ET AL (1999), “User – Centered Design in Practice Problems and Possibilities”, Centre for User Oriented IT Design, Sweden
9. E. H. SHORTLIFFE, "MYCIN, A Rule Based Computer Program...", STAN-CS-74-465, Computer Science Department, Stanford University
10. BROWN, J.S., BURTON, D., AND J.S. BROWN (1982) “Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II and III. In: Intelligent Tutoring Systems,” Sleeman, D. and Brown, J.S. (eds), Academic Press.
11. DUDA, R. AND HART, P. (1973),"Pattern Clasification and Scene Analysis" Wiley-Interscience.
12. M. STEFIK, (1981), “Planning and Meta-Planning (MOLGEN: Part 2). Artificial Intelligence”, 16 (2)
13. THIERAUF R “Expert Systems in Finance and Accounting”, Quorum Books
14. CHARTRAND, G.,(1985), "*Introduction to Graph Theory"*, Dover, New York.
15. WINSTON, W, L., 1994, "*Operations Research, Applications and Algorithms"*, 3rd ed., Duxbury, Belmont.
16. NISSEN, M.E. (1998). “Redesigning Reengineering through Measurement–Driven Inference.” *MIS, Quarterly*, 22 No. 4
17. FLITMAN, A.M. AND HURRION, R.D. (1987). “Linking Discrete–Event Simulation Models with Expert Systems.” *Journal of the Operational Research Society*, 38 No. 8
18. O’KEEFE, R.M. (1989) “The Role of Artificial Intelligence in Discrete–Event Simulation.” In: Widman, L.E., Loparo, K.A. and Neilsen, N.R. eds_.
19. ARTIBA, A. AND AGHEZZAF, E.H. (1997). “An Architecture of a Multi–Model System for Planning and Scheduling.” *International Journal of ComputerIntegrated Manufacturing*, 10 No. 5
20. ZEIGLER, B.P., CHO, T.H. AND ROZENBLIT, J.W.(1996). “A Knowledge–Based Simulation Environment for Hierarchical Flexible Manufacturing.” *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, 26 No. 1

21. LYU, J. AND GUNASEKARAN, A. (1997). “An intelligent simulation model to evaluate scheduling strategies in a steel company.” *International Journal of Systems Science*, 28 No. 6

22. STANDRIDGE, C.R. AND STEWARD, D. (2000). “Using Expert Systems for Simulation Modeling of Patient Scheduling.” *Simulation*, 75 No. 3

23. ROBINSON, S., EDWARDS, J.S. AND YONGFA, W. (1998). “An Expert Systems Approach to Simulating the Human Decision Maker.” *Winter Simulation Conference*

24. LANNER GROUP (2001). *Witness 2001*. Lanner Group, Redditch, UK.

25. ATTAR SOFTWARE (2000). *XpertRule KBS Reference Manual*. Attar Software, Leigh, UK.

26. DE LAERE,K, ET AL (1998), “Human – Computer Interaction and change in Self Appraisal”, The electronic Mirror 11 June 1998

27. PREECE, J. & KELLER, L (1990), “Human – Computer Interaction” , Prentice Hall International, UK

28. NIELSEN (1993) “Usuability Engineering”, Morgan Kaufmann Academic Press.

29. DUMAS, J. & REDISH, J. (1993), “A Practical Guide to Usability Testing”, Prentice Hall

30. PREECE (1995), “A guide to usablitiy human factors in Computing”, Open University Press.

31. NOYES, J & BABER, C (1999), “User-Centered Design of Systems”, Springer-Verlag Berlin Heidelberg, New York

32. NIELSEN J. (2000), “Designing Web Usability: The Practice of Simplicity”, New Riders Publishing, Indianapolis.

9 Παράρτημα 1

US005950176A

United States Patent [19]

Keiser et al.

[11] **Patent Number: 5,950,176**

[45] **Date of Patent: Sep. 7, 1999**

[54] **COMPUTER-IMPLEMENTED SECURITIES TRADING SYSTEM WITH A VIRTUAL SPECIALIST FUNCTION**

[75] Inventors: **Timothy Maxwell Keiser**; **Michael R. Burns**, both of Los Angeles, Calif.

[73] Assignee: **HSX, Inc.**, Los Angeles, Calif.

[21] Appl. No.: **08/620,906**

[22] Filed: **Mar. 25, 1996**

[51] **Int. Cl.**[6] **G06F 17/60**

[52] **U.S. Cl.** **705/37**; 705/36; 705/35

[58] **Field of Search** 705/37, 36

[56] **References Cited**

U.S. PATENT DOCUMENTS

3,499,646	3/1970	Burgess, Jr. et al.	273/278 X
3,573,747	4/1971	Adam et al.	705/37
3,581,072	5/1971	Nymeyer	705/37
4,412,287	10/1983	Braddock, III	705/37
4,597,046	6/1986	Musmanno et al.	705/36
4,674,044	6/1987	Kalmus et al.	705/37
4,903,201	2/1990	Wagner	705/37
4,980,826	12/1990	Wagner	705/37
5,077,665	12/1991	Silverman et al.	705/37
5,101,353	3/1992	Lupien et al.	705/37
5,136,501	8/1992	Silverman et al.	705/37
5,270,922	12/1993	Higgins	705/37
5,297,032	3/1994	Trojan et al.	705/37
5,375,055	12/1994	Togher et al.	705/37
5,508,913	4/1996	Yamamoto et al.	705/37
5,557,517	9/1996	Daughterty, III	705/37
5,689,652	11/1997	Lupien et al.	705/37
5,692,233	11/1997	Garman	705/37
5,727,165	3/1998	Ordish et al.	1/1
5,845,266	12/1998	Lupien et al.	705/37
5,905,974	5/1999	Fraser et al.	705/37

OTHER PUBLICATIONS

Chan, K.C. et al. "Market structure and the intraday pattern of bid–ask spreads for NASDAQ securities," The Journal of Business, v. 68, n. 1, p. 35–, Jan. 1995.

Howard, Barbara. "The trade: technology aims to take the final step," Institutional Investor, v. 25, n. 1, p. S15–, Jan. 1991.

Hakansson, Nils H. et al. "On the feasibility of automated market making by a programmed specialist," Journal of Finance, vol. XL, No. 1, pp. 1–20, Mar. 1985.

Lindsey, Richard R. and Ulrike Schaede. "*Specialist vs. Saitori*: market–making in New York and Tokyo," Financial Analysts Journal, v. 48, n. 4, pp. 48–57, Jul. 1992.

Freund, William C. "Trading stock around the clock: the future growth of global electronic markets," California Management Review, v. 34, n. 1, pp. 87–, 1991.

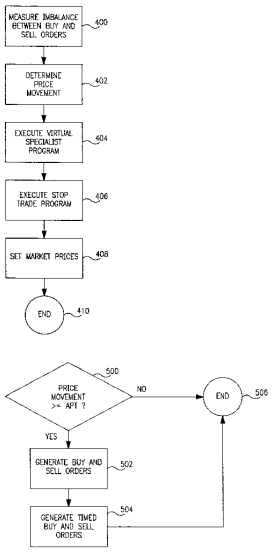
Bloomfield, Robert. "The interdependence of reporting discretion and informational efficiency in laboratory markets," The Accounting Review, v. 71, pp. 493–511, Oct. 1996.

Primary Examiner—Allen R. MacDonald
Assistant Examiner—Michele Stuckey Crecca
Attorney, Agent, or Firm—Brown Raysman Millstein Felder & Steiner LLP

[57] **ABSTRACT**

The present invention discloses a method, apparatus, and article of manufacture for a computer-implemented financial management system that permits the trading of securities via a network. A server computer receives buy and sell orders for derivative financial instruments from a plurality of client computers. The server computer matches the buy orders to the sell orders and then generates a market price through the use of a virtual specialist program executed by the server computer. The virtual specialist program responds to an imbalance in the matching of the buy and sell orders.

12 Claims, 4 Drawing Sheets



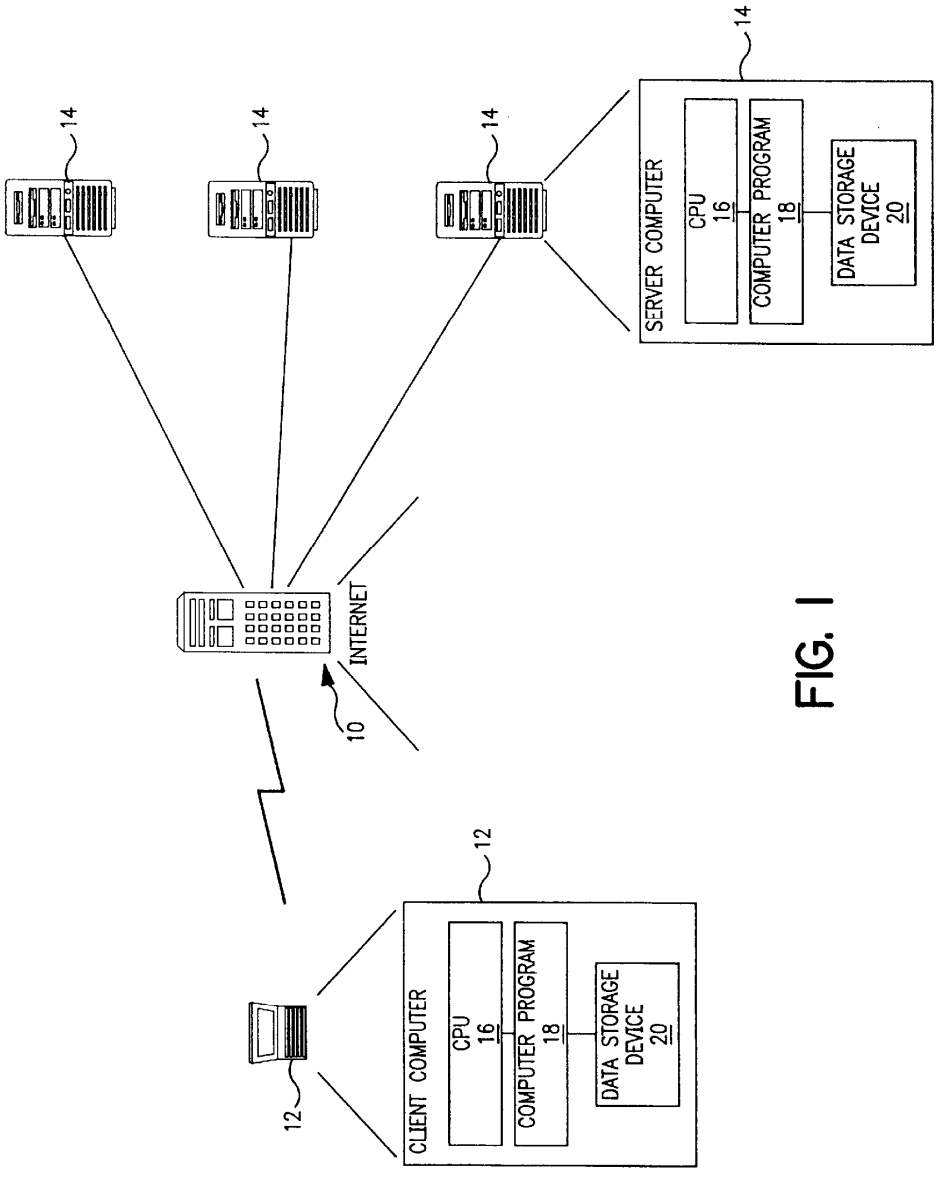

14
14
14
14
SERVER COMPUTER
CPU
16
COMPUTER PROGRAM
18
DATA STORAGE DEVICE
20
INTERNET
10
12
12
CLIENT COMPUTER
CPU
16
COMPUTER PROGRAM
18
DATA STORAGE DEVICE
20

FIG. 1

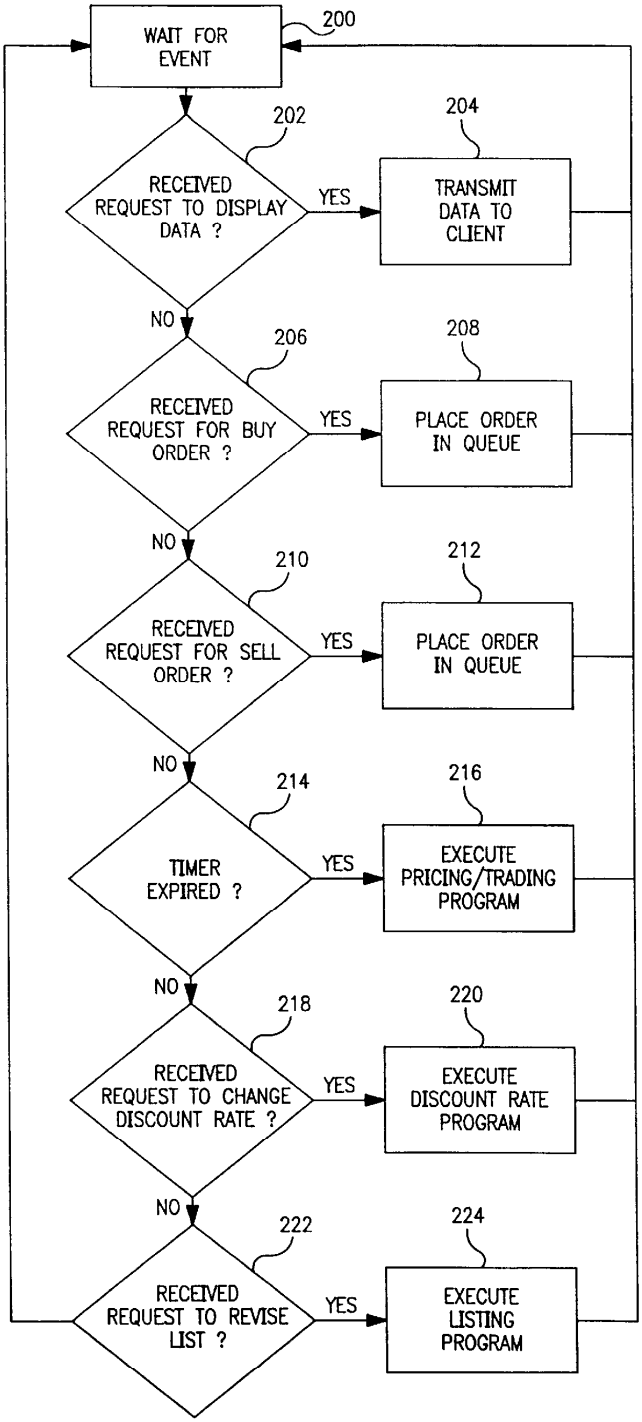

WAIT FOR EVENT
200
202
RECEIVED REQUEST TO DISPLAY DATA ?
YES
204
TRANSMIT DATA TO CLIENT
NO
206
RECEIVED REQUEST FOR BUY ORDER ?
YES
208
PLACE ORDER IN QUEUE
NO
210
RECEIVED REQUEST FOR SELL ORDER ?
YES
212
PLACE ORDER IN QUEUE
NO
214
TIMER EXPIRED ?
YES
216
EXECUTE PRICING/TRADING PROGRAM
NO
218
RECEIVED REQUEST TO CHANGE DISCOUNT RATE ?
YES
220
EXECUTE DISCOUNT RATE PROGRAM
NO
222
RECEIVED REQUEST TO REVISE LIST ?
YES
224
EXECUTE LISTING PROGRAM

FIG. 2

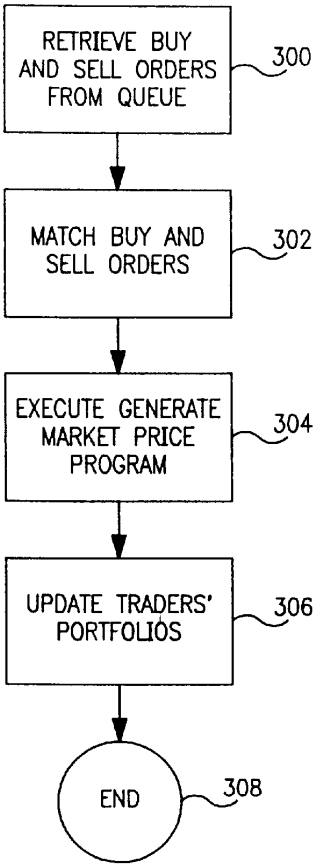

RETRIEVE BUY AND SELL ORDERS FROM QUEUE
300
MATCH BUY AND SELL ORDERS
302
EXECUTE GENERATE MARKET PRICE PROGRAM
304
UPDATE TRADERS' PORTFOLIOS
306
END
308

FIG. 3

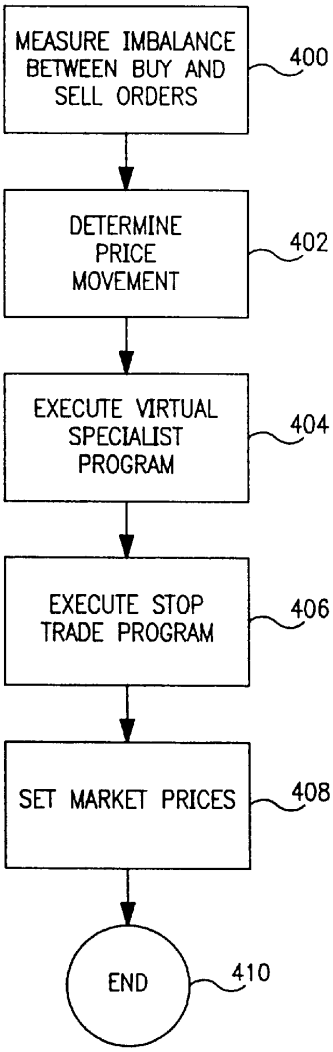

MEASURE IMBALANCE BETWEEN BUY AND SELL ORDERS
400
DETERMINE PRICE MOVEMENT
402
EXECUTE VIRTUAL SPECIALIST PROGRAM
404
EXECUTE STOP TRADE PROGRAM
406
SET MARKET PRICES
408
END
410

FIG. 4

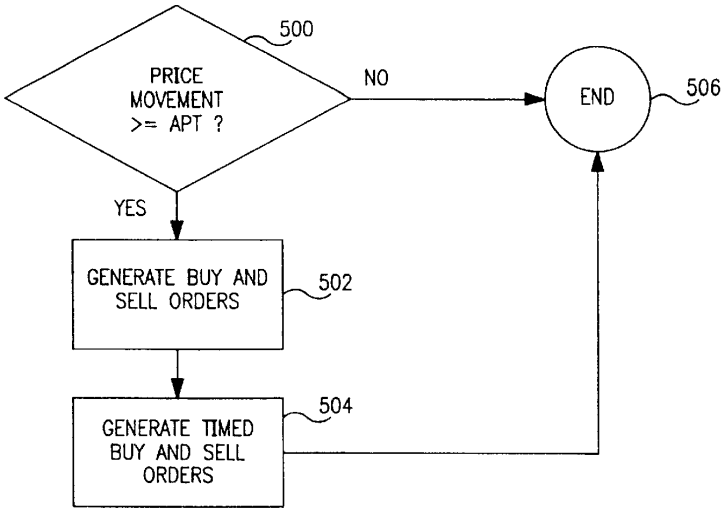

500
PRICE MOVEMENT >= APT ?
NO
END
506
YES
GENERATE BUY AND SELL ORDERS
502
GENERATE TIMED BUY AND SELL ORDERS
504

FIG. 5

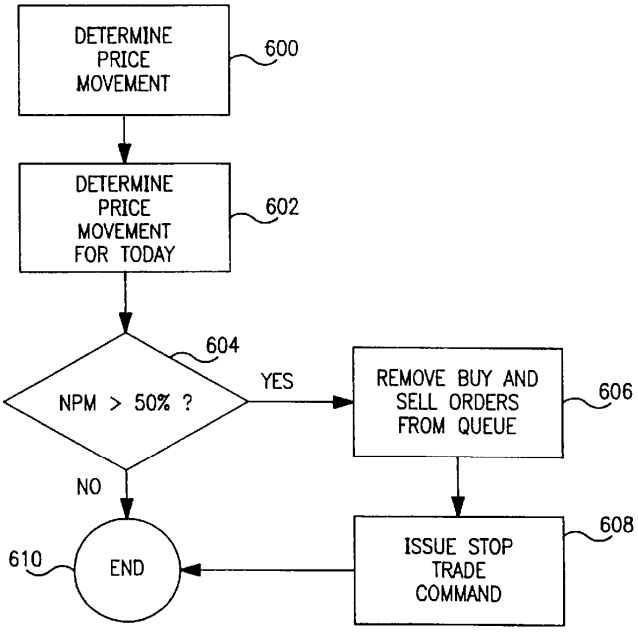

DETERMINE PRICE MOVEMENT
600
DETERMINE PRICE MOVEMENT FOR TODAY
602
604
NPM > 50% ?
YES
REMOVE BUY AND SELL ORDERS FROM QUEUE
606
NO
610
END
ISSUE STOP TRADE COMMAND
608

FIG. 6

COMPUTER-IMPLEMENTED SECURITIES TRADING SYSTEM WITH A VIRTUAL SPECIALIST FUNCTION

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates in general to computer-implemented financial systems, and in particular to an improved automated securities trading system.

2. Description of Related Art

Computer-implemented securities trading systems are well known in the art. One such system is that disclosed U.S. Pat. No. 4,674,044, issued to Kalmus et al., entitled "Automated Securities Trading System", and incorporated by reference herein. These computer-implemented securities trading systems obtain bid and asked prices for securities from a database and then execute trades based on the bid and asked prices. However, there is generally still a human component to such systems.

For example, most financial markets also employ one or more market makers called "specialists." These specialists fill customer orders from the specialist's inventory position if there are no matches for the customer orders in the open market. In the prior art, the specialist function is not automated, but is performed by a firm or individual. Thus, there is a need in the art for an improved computer-implemented trading system that includes an automated specialist function to create a market for the securities traded and to lessen the volatility of smaller securities markets.

SUMMARY OF THE INVENTION

To overcome the limitations in the prior art described above, and to overcome other limitations that will become apparent upon reading and understanding the present specification, the present invention discloses method, apparatus, and article of manufacture for a computer-implemented financial management system that permits the trading of securities via a network. In accordance with the present invention, a server computer receives buy and sell orders for derivative financial instruments from a plurality of client computers. The server computer matches the buy orders to the sell orders and then generates a market price through the use of a virtual specialist program executed by the server computer. The virtual specialist program responds to an imbalance in the matching of the buy and sell orders.

An object of the present invention is to lessen the price volatility of derivative financial instruments traded in narrower markets.

A feature of the present invention is a virtual specialist program that engages in trading in the market to offset the price volatility and to provide liquidity to the market.

BRIEF DESCRIPTION OF THE DRAWINGS

Referring now to the drawings in which like reference numbers represent corresponding parts throughout:

FIG. **1** is a block diagram of an exemplary hardware environment of the preferred embodiment of the present invention;

FIG. **2** is a flowchart illustrating the general logic of the present invention;

FIG. **3** is a flowchart illustrating the logic of the pricing/trading program of the present invention;

FIG. **4** is a flowchart illustrating the logic of the generate market price program of the present invention;

FIG. **5** is a flow diagram illustrating the logic of the virtual specialist program of the present invention; and

FIG. **6** is a flow diagram illustrating the logic of the stop trading program of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

In the following description of the preferred embodiment, reference is made to the accompanying drawings which form a part hereof, and in which is shown by way of illustration a specific embodiment in which the invention may be practiced. It is to be understood that other embodiments may be utilized and structural changes may be made without departing from the scope of the present invention.

OVERVIEW

The present invention comprises a computer-implemented trading system for derivative financial instruments. The present invention accepts buy and sell orders from traders for the derivative financial instruments, sets a market price based on the supply and demand, and participates in the market as a trader in order to minimize price volatility. One preferred embodiment of the present invention is a computer-implemented "Hollywood Stock Exchange", which may be implemented as a simulation (i.e., game) or as an actual trading system for derivative financial instruments representing movies, talent, CDs, and television programs. These derivatives could be purchased with dollars or with a virtual currency known as "Hollywood dollars" which are controlled by a virtual reserve bank.

The derivative financial instruments are identified by a Current Trading List displayed for the traders that comprises a list of movies in various stages of production, talent, and other entertainment-oriented assets. The list contains:

- name of the derivative financial instrument;
- genre of the movie (action-adventure, mystery, western, comedy, etc.);
- production status (scripting, pre-production, filming, editing, release, home-video, etc.);
- number of shares in circulation;
- last trading price (printed every 15 minutes);
- price movement (i.e. ±Hollywood Dollars) since the previous midnight (PST);
- price movement since the previous mid-day;
- price movement year to date;

Traders are able to view the list sorted by:

- name, alphabetically;
- genre, alphabetically;
- production status, alphabetically;
- most active (number of shares traded yesterday);
- biggest gainers;
- biggest losers; and
- fastest movers today (e.g., fastest 20 movers up and fastest 20 movers down).

Similar information would be provided for other derivative financial instruments offered on the Hollywood Stock Exchange.

Each trader's portfolio is identified by a Portfolio data structure that comprises the trader's account status. This information includes:

- the amount of cash in the trader's account (paid interest at the system discount rate plus some increment, compounded daily);

current percentage rate paid to cash;

the total value of held stocks at the last selling price;

the total value of held bonds at the last selling price;

total portfolio value (TPV) (cash+bonds+stocks);

percentage of TPV in cash;

percentage of TPV in bonds; and

percentage of TPV in stocks.

Traders can generate any number of different reports for display, including:

lists of stocks and bonds being traded (see above);

index of total Hollywood stocks (HSXI) expressed as a number, with 1000 defined as the aggregate total stock price value on opening day, wherein HSXI=(today's gross stock-value)/(opening day gross stock-value);

index of total Hollywood bonds (HBXI) expressed as a number, with 1000 defined as the aggregate total bond price value on opening day, wherein HBXI=((today's gross bond-value)/(opening day gross bond-value));

index of total Hollywood Stock Exchange (HMXI) comprised of all stocks and bonds, and expressed as a number, with 1000 defined as the aggregate total stock price value on opening clay, wherein HMXI=((today's gross market-value)/(opening day gross market-value));

lists of the top market performers, e.g., the top 10 traders in percentage portfolio growth calculated as net portfolio value−change=(% change of cash)+(% change of stocks)+(% change of bonds), and for each of the categories: yesterday (midnight to midnight), last week (7 days, ending midnight, each thursday), last month (closes at midnight last calendar day of month), last quarter (closes at midnight on last day of last month/quarter), year-to-date (running daily total of percentage value changes)/(days year-to-date), and annually (closes at midnight on December 31 each year);

overall market condition report, including a list of stopped issues with:

name;

last trading price;

time that stop-trade condition occurred;

percentage the issue actually moved on-the-day before the stop-trade;

number of total shares and/or bonds traded today;

dollar value of total trades today;

number of buy and sell trades today; and

number of buy and sell trades this month.

Use of the above information will guide traders in making future buy and sell orders.

HARDWARE ENVIRONMENT

FIG. **1** is a block diagram that illustrates an exemplary hardware environment for the preferred embodiment of the present invention, and more particularly, illustrates a typical distributed computer system using the Internet **10** to connect client computers **12** executing for example, Web browsers, to server computers **14** executing a computer program embodying the present invention. A typical combination of resources may include client computers **12** that are personal computers or work stations connected via the Internet **10** to server computers **14** that are personal computers, work stations, minicomputers, or mainframes.

Generally, both the client computers **12** and the server computers **14** are comprised of one or more CPUs **16**, various amounts of RAM **20** storing computer programs and other data, and other components typically found in computers. In addition, both the client computers **12** and the server computers **14** may include one or more monitors, and fixed or removable data storage devices **20** such as hard disk drives, floppy disk drives, and/or CD-ROM drives. Also included may be input devices such as mouse pointing devices and keyboards.

Both the client computers **12** and the server computers **14** operate under the control of an operating system, such as Windows, Macintosh, UNIX, etc. Further, both the client computers **12** and the server computers **14** each execute one or more computer programs **18** under the control of their respective operating systems. The present invention is preferably implemented as one or more computer programs **18** executed by the server computer **14**, although in alternative embodiments these computer programs **18** may also be executed on the client computer **12**.

Generally, the computer programs **18** implementing the present invention are tangibly embodied in a computer-readable medium, e.g. one or more of the fixed and/or removable data storage devices **20** attached to the computer. Under control of the operating system, the computer programs **18** may be loaded from the data storage devices **20** into the RAM of the computer for subsequent execution by the CPU **16**. The computer programs **18** comprise instructions which, when read and executed by the computer, causes the computer to perform the steps necessary to execute the steps or elements of the present invention.

Those skilled in the art will recognize that the exemplary environment illustrated in FIG. **1** is not intended to limit the present invention. Indeed, those skilled in the art will recognize that other alternative hardware environments may be used without departing from the scope of the present invention.

GENERAL LOGIC OF THE TRADING SYSTEM

FIG. **2** is a flowchart illustrating the general logic of the present invention.

Block **200** represents the server computer **14** waiting for the next event to occur. Once the event occurs, control is transferred to blocks **202–224** to identify the event and respond accordingly.

Block **202** is a decision block that represents the server computer **14** determining whether it received a request to display data from the client computer **12**. If so, block **204** represents the server computer **14** transmitting data to the client computer **12** for subsequent display. The data transmitted for display preferably includes at least three types of data: the current list of trading derivative financial instruments, the trader's portfolio, and other reports generated by the server computer **14**.

Block **206** is a decision block that represents the server computer **14** determining whether it received a request to submit a buy order from the client computer **12** for a particular derivative financial instrument, e.g., stock or bond. If so, block **208** represents the server computer **14** processing the buy order by placing it in a queue in the memory of the server computer **14**. The buy order is a data structure comprising:

trader's account number;

trader's name;

the time and date of the order;

the stock or bond to buy;

the cash balance in the trader's account;

a text-field where the trader may enter the total number to buy (generally in multiples of 100);

The buy order waits in the queue for the expiration of a predetermined "sweep pricing cycle". In the preferred embodiment, the sweep pricing cycle occurs every 15 minutes although other intervals could be used. The market price the trader actually pays for the derivative financial instrument is determined by the aggregate supply/demand for the derivative financial instrument at the end of the sweep pricing cycle during which the order was placed. The market price is set by the pricing/trading program executed by the server computer, which is described below in FIG. **3**. The trader's account is then charged the market price for the derivative financial instrument. If the purchase uses up all available cash in the trader's account, the trader is "loaned" enough money to pay for the purchase, and their account is charged interest at a predetermined rate, e.g., 18% a year compounded daily, on the negative account balance. The interest is charged against the trader's account until they accumulate more cash to zero out the balance, either by selling stocks or buying dollars.

Block **210** is a decision block that represents the server computer **14** determining whether it received a request to submit a sell order from the client computer **12**. If so, block **212** represents the server computer **14** processing the sell order by placing it in a queue in the memory of the server computer **14**. The sell order is a data structure comprising:

- trader's account number;
- trader's name;
- the time and date of the order;
- the stock or bond to sell;
- the amount of the stock or bond in the trader's account;
- a text-field where the trader may enter the total number to sell (generally in multiples of 100);

Like the buy order, the sell order waits in the queue for the expiration of the predetermined sweep pricing cycle. The market price at which the trader actually sells the derivative financial instrument is determined by the aggregate supply/demand for the derivative financial instrument at the end of the sweep pricing cycle during which the order was placed. The market price is set by the pricing/trading program executed by the server computer, which is described below in FIG. **3**. The trader's account is then credited with the market price for the derivative financial instrument.

The sell order can be either produced by a trader or generated by the server computer **14**, as will be explained in more detail below. For a sell order produced by a trader, the trader views a list of stocks or bonds owned by the trader on a monitor attached to the client computer and chooses to sell a quantity at the market price.

When the trader requests to view the list of stocks, the server computer **14** transmits certain information to the client computer **12** for display, including, for each stock owned, the last trading price (LTP), the quantity of stocks, the purchase price, and the date purchased. Similarly, when viewing the list of bonds, the server computer **14** transmits certain information to the client computer **12** for display, including, for each bond owned, the last trading price (LTP), the interest rate being earned for each kind of bond, the quantity of bonds, the purchase price, and the date purchased.

Block **214** is a decision block that represents the server computer **14** determining whether an internal timer for the sweep pricing cycle has expired. If so, block **216** represents the server computer **14** processing the timer by executing a pricing/trading program as described in FIG. **3**.

Block **218** is a decision block that represents the server computer **14** determining whether it received a request to change the discount rate. If so, block **220** represents the server computer **14** executing a discount rate program. In order to add or subtract liquidity, the server computer **14** occasionally steps in to act as a virtual reserve bank and adjust the discount rate. The discount rate is adjusted based on the performance of the specific industry of the market. For the Hollywood Stock Exchange, the discount rate is adjusted to add or subtract liquidity to affect the growth of the entertainment industry. When the server computer **14** lowers the discount, all the bonds seem to be a better deal, because the bonds are paying a fixed rate interest that never changes. This encourages traders to buy more bonds, and this surge in buying demand causes a correlated increase in bond prices as described above. The same thing happens to stocks, because traders are making less money on the interest being paid on the cash balance in their trading account. When the server computer **14** raises the discount, the bonds seem to be a worse deal, since their advantage over the discount is smaller. Thus, the server computer **14** relaxes the buying pressures or demands for bonds, which should result in additional sell orders, or at least slow the buying of bonds, thus decreasing their prices as they trade in the market. Likewise, stocks seem less attractive, since traders could make more money by keeping cash in their accounts and getting interest on it.

Block **222** is a decision block that represents the server computer **14** determining whether it received a request to revise the derivative list. If so, block **224** represents the server computer **14** executing a listing program. The server computer **14** determines whether the list of derivatives trading in the system should be revised. The list could be revised to reflect new derivative offerings, expired derivatives, and delisted derivatives.

When a new derivative is offered, the price is based on the derivative's potential value. For example, for a new stock offering, which represents a movie on the Hollywood Stock Exchange, the initial price of the stock could be based on the movie's potential box office revenue. For a bond offering, which represents talent on the Hollywood Bond Exchange, the price of the bond could be based on the Hollywood Reporter's Star Power Index. A bond representing a talent with a low Star Power Index of **15** would be issued with a higher yield than a bond representing a talent with a high Star Power Index rating.

A warrant with a strike price is attached to the new derivative when it is offered. When the derivative and warrant are first issued, the warrant is of no value until the strike price is reached. For a stock, the strike price could be reached after the movie has grossed a certain level of revenue. When a derivative is delisted from the exchange, a stock due to the movie ending its production run or a talent due to retirement or death, for example, the warrants are called and the traders are paid the value of the warrants, thus providing off-balance sheet financing for studios.

PRICING/TRADING PROGRAM

FIG. **3** is a flowchart illustrating the logic of the pricing/trading program of the present invention. Block **300** represents the server computer **14** retrieving the buy and sell orders that have accumulated in the queue during the period since the prior sweep pricing cycle. Block **302** represents the server computer **14** matching the buy orders with the sell orders, although it is likely that an identical number of buy and sell orders would not have accumulated in the queue during the period. Block **304** represents the server computer **14** executing the generate market price program described in FIG. **4** to determine the market price for the derivative

financial instruments. After the market price is determined, block **306** represents the server computer **14** updating the traders' portfolios to reflect the buy and sell orders in the queue being processed at the market price. Block **308** represents the end of the pricing/trading program.

GENERATE MARKET PRICE PROGRAM

FIG. **4** is a flowchart illustrating the logic of the generate market price program of the present invention. One purpose of the generate market price logic is to generate a market price for a derivative financial instrument that reflects the demand or lack of demand for the derivative financial instrument in the market. Block **400** represents the server computer **14** measuring the imbalance between the buy and sell orders during the period since the prior sweep pricing cycle. Block **402** represents the server computer **14** determining the price movement of a derivative financial instrument caused by the imbalance in buy and sell orders. Block **404** represents the server computer **14** executing a virtual specialist program as described in FIG. **5** to provide stability and liquidity to the market. Block **406** represents the server computer **14** executing the stop trade program, as described in FIG. **6**, to stop trading in a derivative financial instrument if the projected price movement is excessive during the trading day and threatens the integrity of the market for that instrument. Block **408** represents the server computer **14** setting the market price, which becomes the price the pricing/trading program uses to update the traders' portfolios. Block **410** represents the end of the generate market price program.

In measuring the imbalance between buy and sell orders, as represented by block **400**, the absolute difference between the number of sells and the number of buys is defined as the net movement in sweep (NMS). A sweep increment variable (SIV) is defined as the increase or decrease in price caused by an incremental imbalance in the number of buy orders and sell orders. A lot movement variable (LMV) represents the incremental lot size that will result in a price increase or decrease of one SIV. The projected price movement (PM) can be expressed as:

$$PM=(NMS/LMV)*SIV.$$

For example, with 42,000 buy orders and 30,000 sell orders for a particular stock, the NMS=(42,000−30,000)= 12,000. With SIV=$0.25 and LMV=5000, the price movement of the particular stock will be (12,000/5,000)*0.25= $0.50. Thus, the market price of the particular stock will be $0.50 greater than the last trading price.

One can easily see that, with such a pricing scheme, there is the potential for great volatility in the price of a derivative financial instrument and the eventual loss of investor confidence in the market mechanism. In exchanges such as the Hollywood Stock Exchange, it would be possible for one or more individuals to pursue trading strategies that would purposely cause drastic price fluctuations.

In order to encourage growth and stability in the capital market regulated by the trading system of the present invention, a virtual specialist program is executed by the server computer, as represented by block **404** in FIG. **4**. In executing the virtual specialist program, the server computer **14** regulates the trading by actively trading in the market out of a virtual specialist portfolio (VSP). The virtual specialist portfolio initially contains half of all the issued shares of each derivative financial instrument.

VIRTUAL SPECIALIST PROGRAM

FIG. **5** is a flow diagram illustrating the logic of the virtual specialist program of the present invention. Block **500** is a decision block that represents the server computer **14** determining whether or not the price movement during the sweep pricing cycle is greater or equal to an adjusted price movement threshold (APT). The APT is a constant in the memory of the server computer **14**. If the APT is greater than the price movement, then the server computer **14** does not trade in the market. If the price movement is greater than or equal to the APT, then the server computer **14** trades out of a virtual specialist portfolio. The level of trading by the server computer **14** is determined by the amount that the price movement exceeded the APT. The greater the price movement, the more shares the server computer **14** trades to offset the price movement.

In an exemplary embodiment of the present invention, the ATP=1.25 and the server computer **14** performs the following steps: if PM=APT then the server computer **14** matches 20% of unmatched shares; if PM=APT+0.25 then the server computer **14** matches 20%of unmatched shares; if PM=APT+0.50 then the server computer **14** matches 30% of unmatched shares; if PM=APT+0.75 then the server computer **14** matches 40% of unmatched shares; if PM=APT+ 1.0 then the server computer **14** matches 50% of unmatched shares; if PM=APT+1.25 then the server computer **14** matches 60% of unmatched shares; if PM=APT+1.50 then the server computer **14** matches 70% of unmatched shares; or if PM=APT+1.75 then the server computer **14** matches 80% of unmatched shares.

Block **502** represents the server computer **14** generating a buy or a sell order to offset the price movement. The buy or sell order generated by the server computer **14** is placed in the queue with the trader buy and sell orders to be processed during the next sweep cycle.

Since the virtual specialist portfolio initially includes half of all the securities traded, the server computer **14** could eventually deplete the virtual specialist portfolio or cause the virtual specialist portfolio to own all the shares of a stock. In order to maintain a balanced virtual specialist portfolio, and provide some liquidity to the market, the server computer **14** generates additional buy and sell orders to offset orders generated in response to the price movement exceeding the APT. Block **504** represents the server computer **14** generating timed buy and sell orders. In one embodiment of the invention, the server computer **14** assess each stock and each bond in the virtual specialist portfolio. The server computer **14** determines the deficit or surplus in the item, and then place 1/288th of the deficit as a "timed recovery order" into each successive 15 minute segment for the next 3 days. When the pricing/trading program 255 matches buy and sell orders as represented by block **320**, the pricing/trading program **255** includes any "timed recovery orders" outstanding for the last **3** days in the sweep. These orders are matched with the traders' buy and sell orders. Block **506** represents the end of the virtual specialist program.

STOP TRADING PROGRAM

FIG. **6** is a flow diagram illustrating the logic of the stop trading program of the present invention. Block **600** represents the server computer **14** determining the price movement of a stock caused by the imbalance in buy and sell orders. Block **602** represents the server computer **14** measuring the price movement on the day, not just during the sweep cycle period. Block **604** is a decision block that represents the server computer **14** determining whether the net price movement (NPM) within one "trading day" (i.e., midnight-midnight) is greater than 50% up or down. As represented by block **606**, the buy and sell orders are

removed from the queue if the net price movement is greater than 50% for a stock trading above $20. At that point, the trading in that issue is stopped within the 15 minute period until further notice. All orders (buy and sell) for that stock during this sweep are unfilled. The trading has stopped due to "excessive order imbalance".

For example, assume that the Last Trading Price (LTP) for "Rambo-17" is $67 (+7.5 on-the-day). During one 15-minute sweep pricing cycle, the server computer **24** receives buy orders for 655,000 shares of "Rambo-17". Also, the server computer **14** receives sell orders for 35,000 shares of "Rambo-17". The server computer **14** evaluates the price movement for the sweep pricing cycle, and tests it to see if the net projected price movement "on-the-day" is greater than 50%. If it would be greater than 50%, it stops trading in that instrument only. In this example, there is a net order-imbalance of 620,000 shares, which would create an up movement in price of (+620,000/5000)*$0.25=+$31.00. Since the total movement on the day would be the $7.50 so far plus the additional $31.00, the net projected price movement on the day would be $31.00+$7.50=$38.50. If the opening price that day was $59.50, the percentage projected price movement for the day is $38.50/$59.50=64%. Since the projected net price movement would be greater than 50%, the trading is stopped for that instrument. If the projected price movement was less than 50%, the price of the instrument would be adjusted accordingly and trade in that stock continued. Block **608** represents the STOP TRADE order that issues regarding the particular stock. Traders who issued a buy or sell order for the stock are notified that the order has not been filled due to excessive order imbalance during the trading day. Finally, block **610** represents the end of the stop trading program.

CONCLUSION

The foregoing description of the preferred embodiment of the invention has been presented for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise form disclosed. Many modifications and variations are possible in light of the above teaching. It is intended that the scope of the invention be limited not by this detailed description, but rather by the claims appended hereto.

What is claimed is:

1. A computerized method for regulating market price in a computerized trading system, the system receiving buy orders and sell orders for an instrument, the method comprising:

measuring an imbalance between buy orders and sell orders for the instrument received over a given period;

computing a projected price movement based on the measured imbalance between the number of buy and sell orders;

setting a market price for the instrument based upon the received buy and sell orders and the measured imbalance; and

automatically generating additional buy orders or sell orders for the instrument at the market price to guarantee execution of some or all of the received buy or sell orders if the projected price movement is greater than or equals a predetermined price movement threshold.

2. The method of claim **1**, wherein the step of measuring the imbalance comprises computing an absolute difference between the number of buy orders and the number of sell orders.

3. The method of claim **2**, wherein the projected price movement is computed from the difference and stored variables.

4. The method of claim **3**, wherein the stored variables include a sweep increment variable and a lot movement variable.

5. The method of claim **3**, wherein the step of computing the projected price movement comprises computing the projected price movement using the following equation:

$$PM=(NMS/LMV)*SIV,$$

where PM represents the projected price movement, NMS represents the absolute difference between the number of buy and sell orders, LMV represents the lot movement variable, and SIV represents the sweep increment variable.

6. The method of claim **1**, further comprising generating a number of additional orders in direct proportion to an amount by which the projected price movement exceeds the predetermined price movement threshold.

7. A computerized method for regulating market price in a computerized trading system that receives buy orders and sell orders for an instrument, the method comprising:

measuring an imbalance between buy and sell orders received for the instrument over a plurality of periods;

computing a projected price movement for each of the periods based on the measured imbalance between the number of buy and sell orders;

computing a total price movement in the instrument for the plurality of periods based upon the projected price movement during the periods; and

stopping trading activity in the instrument if the computed total price movement exceeds an excessive order threshold.

8. The method of claim **7**, comprising computing the excessive order threshold from an opening market price for the instrument determined prior to the plurality of periods.

9. The method of claim **8**, wherein the step of computing the excessive order threshold comprises computing the excessive order threshold as 50% of the opening market price for the instrument.

10. The method of claim **1**, wherein the step of automatically generating additional buy and sell orders comprises automatically generating orders to guarantee execution of all of the received buy and sell orders.

11. A computer-readable storage medium for storing program code means for, when executed, causing a computer to perform a computerized method for regulating market price in a computerized trading system that receives buy orders and sell orders for an instrument, the method comprising:

measuring an imbalance between buy and sell orders received for the instrument over a plurality of periods;

computing a projected price movement for each of the periods based on the measured imbalance between the number of buy and sell orders;

computing a total price movement in the instrument for the plurality of periods based upon the projected price movement during the periods; and

stopping trading activity in the instrument if the computed total price movement exceeds an excessive order threshold.

12. A computer-readable storage medium for storing program code means for, when executed, causing a computer to perform a computerized method for regulating

market price in a computerized trading system that receives buy orders and sell orders for an instrument, the method comprising:

measuring an imbalance between buy and sell orders received for the instrument over a plurality of periods;

computing a projected price movement for each of the periods based on the measured imbalance between the number of buy and sell orders;

computing a total price movement in the instrument for the plurality of periods based upon the projected price movement during the periods; and

stopping trading activity in the instrument if the computed total price movement exceeds an excessive order threshold.

* * * * *