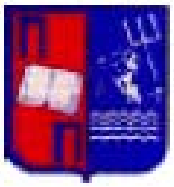
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΑ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

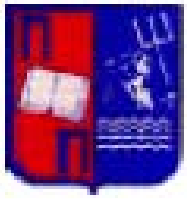


ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

ΠΑΥΛΟΥ ΠΕΤΡΑΝΤΩΝΑΚΗ ΤΟΥ ΜΑΤΘΑΙΟΥ

ΔΙΟΙΚΗΣΗ ΚΙΝΔΥΝΟΥ ΣΕ ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ
ΕΦΑΡΜΟΓΕΣ ΣΕ ΣΥΣΤΗΜΑΤΑ ΕΘΝΙΚΗΣ ΑΜΥΝΑΣ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΑ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ



ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

ΠΑΥΛΟΥ ΠΕΤΡΑΝΤΩΝΑΚΗ ΤΟΥ ΜΑΤΘΑΙΟΥ

ΔΙΟΙΚΗΣΗ ΚΙΝΔΥΝΟΥ ΣΕ ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ ΕΦΑΡΜΟΓΕΣ ΣΕ ΣΥΣΤΗΜΑΤΑ ΕΘΝΙΚΗΣ ΑΜΥΝΑΣ

Τριμελής Συμβουλευτική Επιτροπή

Επιβλέπων
Καθηγητής Παναγιωτόπουλος Ιωάννης- Χρήστος

Μέλη
Καθηγητής Σίσκος Ιωάννης
Αναπληρωτής Καθηγητής Θεοδωρίδης Ιωάννης

Στους γονείς μου.

ΕΥΧΑΡΙΣΤΙΕΣ

Μια σελίδα δεν φτάνει για να εκφράσω τις ευχαριστίες μου σε όσους μου συμπαραστάθηκαν αυτά τα έξι χρόνια έρευνας και συνέβαλαν ο καθένας από το δικό του μετερίζι στην ολοκλήρωση αυτής της διατριβής.

Θέλω να ευχαριστήσω ιδιαίτερα τον Επιβλέποντα καθηγητή κ. Ιωάννη – Χρήστο Παναγιωτόπουλο για την μεγάλη τιμή που μου έκανε να με δεχθεί σαν επιστημονικό του συνεργάτη. Η απόφαση του αυτή αποτελεί για εμένα ορόσημο στην ζωή μου. Τον ευχαριστώ θερμά για την καθοδήγηση του, τη συμπαράσταση του και τη βοήθεια του σε όλη την ερευνητική μου πορεία. Υπήρξε πηγή γνώσης και σωστός Ακαδημαϊκός.

Θερμές ευχαριστίες στα μελη της τριμελής Επιτροπή τον Καθηγητή κ. Ιωάννη Σίσκο και τον Αναπληρωτή Καθηγητή κ. Ιωάννη Θεοδωρίδη για την άψογη συνεργασία καθώς και τις εποικοδομητικές συμβουλές, τις παρατηρήσεις και τα εύστοχα σχόλια.

Εκφράζω τις ευχαριστίες μου επίσης στο φίλο και συνεργάτη Γεωργιάδη Πέτρο για την πολύτιμη βοήθεια που μου πρόσφερε με τις παρατηρήσεις του και συμβουλές του κατά τη διάρκεια υλοποίησης του λογισμικού της διδακτορικής διατριβής. Ευχαριστώ την Αγγελική Θεοδόση για τη στήριξη της, τις οδηγίες της, και την ανοχή της στην γκρίνια μου όλο αυτό τον καιρό. Παράλληλα εκφράζω τις θερμές μου ευχαριστίες στο φίλο και συνεργάτη Ρίγγα Αναστάσιο για τις συμβουλές του σε θέματα προσομοίωσης και μοντελοποίησης της προτεινόμενης μεθοδολογίας.

Τέλος το πιο σημαντικό ευχαριστώ το απευθύνω στους γονείς μου που στερήθηκαν πολλά για να μπορέσουν να με σπουδάσουν και να γίνω ένας ολοκληρωμένος ακαδημαϊκός πολίτης. Ευχαριστώ για την αγάπη τους, τη στοργή τους, τα ξενύχτια τους και τις αγωνίες τους για μένα. Τους ευχαριστώ που με πίεζαν να διαβάσω για να μπορέσω να γίνω αυτό που είμαι σήμερα. Η διδακτορική μου διατριβή είναι ένα μικρό δώρο ευγνωμοσύνης σε αυτούς και ανταμοιβή των κόπων τους.

ΠΡΟΛΟΓΟΣ

Η παρούσα διδακτορική διατριβή εστιάζεται στη δημιουργία και ανάπτυξη της Σύγχρονης Θεωρίας Διοίκησης Κινδύνου σε πολύπλοκα Πληροφοριακά Συστήματα. Σκοπός της διδακτορικής διατριβής είναι η ανάπτυξη μιας θεωρίας αντιμετώπισης κρίσεων σε Ψηφιακά Πληροφοριακά Συστήματα. Μελετά και αναλύει την υπάρχουσα δομή των μεθοδολογιών Ανάλυσης ρίσκου με σκοπό την αξιολόγηση τους. Ειδικότερα παρουσιάζονται και αναλύονται υπάρχοντα μοντέλα Διαχείρισης Κρίσεων προσαρμοσμένα σε ψηφιακά πληροφοριακά συστήματα.

Παράλληλα ορίζονται θεμελιώδης αρχές της Σύγχρονης Θεωρίας Διοίκησης Κινδύνου με σκοπό την παρουσίαση μιας ολοκληρωμένης θεωρίας διαχείρισης κρίσεων και αντιμετώπισης Ασταθών παραγόντων σε δυναμικά μεταβαλλόμενα περιβάλλοντα λειτουργίας. Αναλύονται οι δομές και μέθοδοι αντιμετώπισης κρίσεων με τη χρήση μαθηματικών εργαλείων. Επιπλέον οριοθετούνται οι κανόνες λειτουργίας του μοντέλου αντιμετώπισης κρίσεων, προσδιορίζοντας τους κίνδυνους εκείνους που κάνουν το χώρο ασταθή και ιδιαίτερα εχθρικό μέσα σε ένα συγκεκριμένο ορίζοντα χρονοπρογραμματισμού.

Το δεύτερο σκέλος της Θεωρίας περιλαμβάνει τις καταστάσεις όπου μια κρίση έχει υλοποιηθεί. Η πρόληψη απέτυχε να καταστείλει την κρίση με αποτέλεσμα καταστροφικές δυνάμεις να ενεργοποιηθούν στο σύστημα μας. Το μοντέλο προσδιορίζει τις δυνάμεις αυτές και ενεργοποιεί το αντίστοιχο σύστημα ανάκτησης με το ελάχιστο δυνατό κόστος ενεργοποίησης.

Τέλος παρουσιάζεται ένα λογισμικό βοήθημα αντιμετώπισης κρίσεων και ελαχιστοποίησης κόστους ανάκτησης στηριζόμενο στο παρουσιαζόμενο μοντέλο. Η αξιολόγηση του μοντέλου και ο έλεγχος αξιοπιστίας έγινε με την χρήση με την υλοποίηση αντιστοίχων πιθανών σεναρίων κρίσεως. Όλες οι ακραίες και ομαλές καταστάσεις του μοντέλου αντιμετώπισης κρίσεων προσομοιώνονται δίνοντας στον διαχειριστή του συστήματος την δυνατότητα βελτιστοποίησης και επισήμανσης των αδυναμιών του συστήματος.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 1

ΕΙΣΑΓΩΓΗ

1.ΕΙΣΑΓΩΓΗ

1.1 Η ΝΕΑ ΨΗΦΙΑΚΗ ΕΠΟΧΗ

"Σοφός δεν είναι όποιος ξέρει τα πολλά αλλά όποιος ξέρει τα χρήσιμα."

Αισχύλος

Ο εικοστός αιώνας ήταν ο χρυσός αιώνας της πληροφορικής, όποτε μεγάλες και σημαντικές ανακαλύψεις έλαβαν χώρα, δίνοντας μια νέα ώθηση στην ανάπτυξη καινοτόμων εφαρμογών. Μια πληθώρα νέων εφαρμογών έκανε την εμφάνιση της που τάραξε τα λιμνάζοντα ύδατα στον επιστημονικό τομέα, με αποτέλεσμα να δημιουργηθεί μια νέα επιστήμη, η οποία ξεπέρασε σε δυναμική σημαντικούς επιστημονικούς κλάδους, όπως τα μαθηματικά και τη φυσική. Η πληροφορική γεννήθηκε, λοιπόν, μέσα από τα σπλάχνα των δυο παραπάνω επιστημών, αναπτύχθηκε και τις υπερκέρασε, αποκτώντας την πρωτοκαθεδρία στον επιστημονικό κλάδο. Οι Θεμελιώδεις δομές της αναπτύχθηκαν, με σκοπό να συμβάλουν, με τη σειρά τους, στη θεμελίωση του επιστημονικού αυτού κλάδου, να επηρεάσουν άλλους σημαντικούς επιστημονικούς κλάδους, αλλά παράλληλα και τομείς της καθημερινότητας του Ανθρώπου.

Οι σύγχρονες ψηφιακές εφαρμογές τροποποίησαν τη δομή των σύγχρονων δυτικοευρωπαϊκών συστημάτων, με άμεση απόρροια θεμελιώδεις δομές των συστημάτων να αλλάξουν δραματικά, πολλές από αυτές να βελτιωθούν, να αναβαθμιστούν ή χειρότερα να εγκαταλειφθούν, αφού δεν ήταν ικανές να ικανοποιήσουν το νέο πλαίσιο αναγκών που δημιουργήθηκε. Έτσι, συντελέστηκαν σημαντικές εξελίξεις, καθώς πάγιες ιδέες και αντιλήψεις αντικαταστάθηκαν από νέες, πιο προοδευτικές, ικανές να ακολουθήσουν τις απαιτήσεις της νέας Ψηφιακής εποχής.

Η ανάπτυξη της πληροφορικής, όπως προαναφέρθηκε, επηρέασε σημαντικούς επιστημονικούς κλάδους, ενώ παράλληλα άλλαξε τη δομή και τη λειτουργία πολλών πτυχών της καθημερινότητας του Ανθρώπου. Τομείς όπως η Οικονομία, η Υγειά, η Παιδεία, το Τραπεζικό σύστημα, οι Μεταφορές και ιδιαίτερα η Άμυνα άλλαξαν ριζικά. Οι σύγχρονες ψηφιακές εφαρμογές της Πληροφορικής, επιπρόσθετα βελτίωσαν πολλούς τομείς των σύγχρονων δυτικοευρωπαϊκών κοινωνικό-πολιτικών συστημάτων, έδωσαν ώθηση στις τοπικές κοινωνίες, συνέβαλαν στην

βελτίωση του βιοτικού επιπέδου του Ανθρώπου, άλλαξαν τον τρόπο ζωής των σύγχρονων κοινωνιών. Ταυτόχρονα, επέφεραν ρηξικέλευθες αλλαγές σε ζωτικούς τομείς, αλλάζοντας ριζικά εδραιωμένες αντιλήψεις όσον αφορά τον τρόπο αντιμετώπισης μιας μεγάλης μερίδας προβλημάτων.

Η νέα Ψηφιακή εποχή είναι η δεύτερη μεγάλη τεχνολογική επανάσταση στη σύγχρονη ιστορία του Ανθρώπου. Μια επανάσταση που άλλαξε το ρου της ιστορίας. Η βιομηχανική επανάσταση απετέλεσε την αρχή της εξέλιξης με την αρωγή της οποίας επήλθαν τεράστιες αλλαγές. Νέα οικονομικά, πολιτικά και κοινωνικά συστήματα εμφανίστηκαν, παραγκωνίζοντας τα ήδη υπάρχοντα. Η ζωή του Ανθρώπου άλλαξε. Τα φέουδα, οι γαιοκτήμονες, οι άρχοντες αντικαταστάθηκαν από τις βιομηχανίες, τους βιομηχάνους, την αστική τάξη. Τα άλογα αντικαταστάθηκαν από τις μηχανές. Σε σύντομο χρονικό διάστημα επήλθε και η Ψηφιακή επανάσταση. Με την ανακάλυψη του υπολογιστή και την κατασκευή νέων υπολογιστικών συστημάτων οι μηχανές αντικαταστάθηκαν από τους υπολογιστές, οι εργάτες από τα ρομπότ και η καύσιμη ύλη, που τροφοδοτεί την μηχανή, από τα bits και τα bytes του υπολογιστή.

Άμεσο επακόλουθο αυτής της αλλαγής ήταν να επηρεαστούν σημαντικά όλα τα Μοντέρνα συστήματα. Η πλειοψηφία αυτών ψηφιοποιήθηκε προκειμένου να είναι ικανά να αντεπεξέλθουν στις απαιτήσεις των καιρών, ενώ όσα δεν μπόρεσαν να ακολουθήσουν τις εξελίξεις παραγκωνίστηκαν και αντικαταστάθηκαν από αλλά ικανά να ικανοποιήσουν τις απαιτήσεις της ψηφιακής εποχής. Η μετάβαση αυτή οδήγησε στη δημιουργία τεράστιων Ψηφιακών συστημάτων, που είχαν πολύπλοκη δομή και λειτουργία. Η πλειοψηφία αυτών είναι κατανεμημένα σε διάφορα μήκη και πλάτη της υφηλίου [Stephenson 1997].

Αφού, λοιπόν, η κατανομή των σύγχρονων ψηφιακών συστημάτων αύξησε την πολυπλοκότητα τους, η αύξηση αυτή οδήγησε με τη σειρά της στη δημιουργία Συστημάτων ευάλωτων σε Κινδύνους. Η πορεία και των δυο δεδομένων - Συστήματα και Κίνδυνοι – ήταν αλληλένδετη και με ταυτόχρονη ανάπτυξη. Σημεία ορόσημα για την ανάπτυξη βελτίωση και επαναπροσδιορισμό της Σύγχρονης Θεωρίας Διοίκησης Κινδύνου. Τα Σύγχρονα Πληροφοριακά Συστήματα καλύπτουν πληθώρα αναγκών σε διάφορους τομείς της καθημερινότητας είναι, όμως, απροστάτευτα από τους κίνδυνους που τα απειλούν. Η πιθανότητα μη ομαλής λειτουργίας ενός πληροφοριακού συστήματος είναι πάντοτε αυξημένη, καθώς είναι συχνά τα φαινόμενα όπου ένα Ψηφιακό Σύστημα δεν κατάφερε να

αντεπεξέλθει σε μια Κρίση με αποτελέσματα ευλόγως οδυνηρά, που επηρεάζουν τόσο τους χρήστες τους συστήματος, όσο και την εταιρεία, οργανισμό που έχει επενδύσει ένα σεβαστό πόσο χρημάτων. Στο νέο περιβάλλον λειτουργίας τέτοιες κρίσεις είναι καθημερινά φαινόμενα και χρήζουν άμεσης αντιμετώπισης.

1.2 ΤΟ ΑΝΤΙΚΕΙΜΕΝΟ ΤΗΣ ΔΙΑΤΡΙΒΗΣ

Η κατάτμηση, η διασπορά και η διαδικτυακή επικοινωνία βελτίωσαν τη λειτουργία των Ψηφιακών Συστημάτων. Παράλληλα, ελαχιστοποίησαν το κόστος στις επικοινωνίες και έδωσαν μια νέα ώθηση στη δομή και υλοποίηση των Σύγχρονων Πληροφοριακών Συστημάτων. Μηδενίστηκε, με αυτό τον τρόπο, η απόσταση μεταξύ κομβικών σημείων κάθε υποσυστήματος, αυξάνοντας την λειτουργική τους ικανότητα. Έτσι, η τηλεδιαχείριση ήταν πια εφικτή, ελαχιστοποιώντας - κατά πολύ - το κόστος λειτουργίας και συντήρησης του συνολικού Συστήματος.

Όλα, όμως, τα παραπάνω καθώς και άλλα, που θα αναλυθούν σε επόμενα κεφάλαια, αποτέλεσαν σημαντικό παράγοντα αστάθειας και τρωτότητας. Άνοιξαν την Κερκόπορτα της Θεωρίας των Πληροφοριακών Συστημάτων και φανέρωσαν με την σειρά τους τις αδυναμίες που έχουν τα πληροφοριακά συστήματα. Αδυναμίες σημαντικές που καταδεικνύουν την ανάγκη συνεχούς προστασίας από τους Κινδύνους. Το πιο σημαντικό, όμως, είναι ότι συνέβαλαν στην πάγια αντίληψη ότι τίποτα δεν είναι εκατό τοις εκατό ασφαλές, με συνέπεια να ανοίξει ο ασκός του Αιόλου για τη δημιουργία ασταθών καταστάσεων, μέσα στις οποίες δρα ένα Πληροφοριακό σύστημα. Οι καταστάσεις είναι συχνά χαοτικές, δαιδαλώδεις και με αβέβαια αποτελέσματα. Επομένως, οδηγούν στη δημιουργία νέων δομών του συστήματος άλλοτε ενισχύοντας το και άλλοτε, όπως συμβαίνει στην πλειοψηφία των περιπτώσεων, αποδυναμώνοντας το ή οδηγώντας το σε ανεξέλεγκτη μεταβολή, δηλαδή στην καταστροφή.

1.3 ΣΤΟΧΟΣ ΤΗΣ ΔΙΑΤΡΙΒΗΣ

Η σύγχρονη Τεχνολογία και ιδιαίτερα η εξέλιξη της πληροφορικής συνέβαλε στην υπερπήδηση εμποδίων που αποτελούσαν τροχοπέδη για την εξέλιξη της ανθρωπότητας. Τα Συστήματα (οικονομικά, κοινωνικά, γεωπολιτικά, άμυνας) βρέθηκαν αντιμέτωπα με καταστάσεις συγκεχυμένες, μη ελεγχόμενες, που η αντιμετώπιση τους ήταν κατά

κανόνα αδύνατη. Ειδικά, οι ασύμμετρες απειλές και οι αντίστοιχες κρίσεις απειλούν την κατάρρευση των σύγχρονων Συστημάτων, όπως αυτά έχουν οικοδομηθεί σήμερα. Σκοπός της εργασίας αυτής είναι η προσπάθεια πρόληψης και αποφυγής Κρίσεων, που κατά κανόνα εμφανίζονται σε Ασταθείς Χώρους Δεδομένων. Γίνεται προσπάθεια ελαχιστοποίησης του κινδύνου που αντιμετωπίζει ένα Σύστημα με τη βοήθεια της *Ψηφιακής Τεχνολογίας*, και ειδικότερα με την βοήθεια των ψηφιακών εργαλείων ικανών όχι μόνο να προλάβουν μια Κρίση, αλλά και να την αντιμετωπίσουν, αποκλιμακώνοντας την με το ελάχιστο δυνατό κόστος.

Η ανάγκη αποφυγής Κρίσεων οδήγησε στην Ανάπτυξη της Θεωρίας του Κινδύνου. Μιας θεωρίας επαρκούς να βοηθήσει ένα σύστημα για να αντιμετωπίσει επιτυχώς μια επικίνδυνη κατάσταση, αφού του παρέχει τα απαραίτητα εφόδια, ώστε να υπολογίσει την κρίση και τις συνέπειες που θα έχει τόσο σε αυτό όσο και στο ευρύτερο περιβάλλον εργασίας. Όπως προκύπτει, σε πολλές περιπτώσεις, οι Κρίσεις είναι δυνατόν να αντιμετωπιστούν.

Στόχος της διατριβής είναι η ανάπτυξη μιας αποτελεσματικής Θεωρίας Αντιμετώπισης Κρίσεων ειδικευμένης σε Ψηφιακά Πληροφοριακά Συστήματα που αφορούν κυρίως την Εθνική Άμυνα. Η θεωρία προσπαθεί, από την δική της οπτική σκοπιά, να προσφέρει ένα βοηθητικό εργαλείο στον αναλυτή και να καλύψει κενά πάνω στον τομέα της Διαχείρισης του Κινδύνου σε Πληροφοριακά Συστήματα, παρουσιάζοντας ταυτόχρονα γενικές θεωρητικές αρχές. Εισάγει νέες έννοιες πάνω στα πληροφοριακά συστήματα, όπως για παράδειγμα η έννοια της μετάλλαξης, ενώ προσπαθεί τόσο ποιοτικά όσο και ποσοτικά να προσδιορίσει τον Κίνδυνο που απειλεί το υπό μελέτη Σύστημα.

Μέσα από το τρίπτυχο Πρόληψη – Αποφυγή – Αντιμετώπιση, η διατριβή προσπαθεί να αποδείξει ότι οι κίνδυνοι είναι δυνατό να προβλεφθούν με ικανοποιητική ακρίβεια. Δυστυχώς, είναι αρκετές οι φορές που μια Απειλή δεν μπόρεσε να αντιμετωπιστεί, επειδή ολιγωρία και παραλείψεις οδήγησαν στην τέλεση μιας καταστροφής. Είναι επιτακτική επομένως, η ανάγκη εύρεσης του κατάλληλου συστήματος Ανάκτησης των καταστραμμένων δεδομένων, με σκοπό την πλήρη ή μερική αποκατάσταση των ζημιών.

Μέσα από την διατριβή, αποδεικνύεται ότι μπορεί το μοντέλο να δώσει τη δυνατότητα στον Αναλυτή να χρησιμοποιήσει το αποτελεσματικότερο αντίμετρο με το ελάχιστο κόστος Ανάκτησης. Το

όφελος είναι διπλό και ιδιαίτερα σημαντικό, αφού επιτυγχάνεται η μερική ή η ολική ανάκτηση, ενώ παράλληλα, ελαχιστοποιείται το κόστος ανάκτησης που είναι ένας βασικός παράγοντας με μεγάλο ειδικό βάρος. Ένας παράγοντας που προσμετράται και λαμβάνεται σοβαρά υπόψη, κατά την διαδικασία ανάκτησης, αντισταθμίζοντας το κόστος Ανάκτησης με το κόστος Απώλειας Δεδομένων.

Η μεθοδολογία Πρόληψης και Αντιμετώπισης των Κινδύνων, όπως θα αναλυθεί σε επόμενα κεφάλαια, υλοποιείται μέσα από ένα λογισμικό το οποίο υποβοηθά τον αναλυτή στη λήψη συγκεκριμένων αποφάσεων. Σε στιγμές Κρίσης, το λογισμικό αυτό συμβάλλει στην αποσαφήνιση των αδυναμιών του συστήματος καθώς και στη λήψη των κατάλληλων και γρήγορων αποφάσεων. Φυσικά, πρωταρχικό ρόλο στο όλο Σύστημα έχει, είχε και θα έχει ο Άνθρωπος. Αυτός είναι ο τελικός παράγοντας που θα αποφασίσει πως θα λειτουργήσει την ώρα της κρίσης και τι μέτρα θα λάβει, προκειμένου να αντεπεξέλθει στη δύσκολη στιγμή, ωστόσο, τις πρώτες στιγμές της κρίσης, το «σοκ» επικρατεί και ο ανθρώπινος παράγοντας συχνά παραλύει με άμεση συνέπεια την μεγιστοποίηση της καταστροφής (κόστους).

1.4 Η ΔΟΜΗ ΤΗΣ ΔΙΑΤΡΙΒΗΣ

Σε αυτό το τμήμα παρουσιάζεται αναλυτικά η δομή της διδακτορικής διατριβής, η οποία αποτελείται από οχτώ κεφάλαια και τη Βιβλιογραφία. Το πρώτο κεφάλαιο αποτελεί την εισαγωγή της διατριβής. Σε αυτήν περιγράφεται η νέα ψηφιακή εποχή μέσα από την άνθιση της επιστήμης των υπολογιστών. Παράλληλα, παρουσιάζεται το αντικείμενο της διατριβής, ενώ επισημαίνονται και οι στόχοι της παρούσης επιστημονικής μελέτης. Στο τέλος, περιγράφεται η δομή των κεφαλαίων.

Το δεύτερο κεφάλαιο της διατριβής έχει τίτλο «*Ανάλυση ρίσκου και Θεωρία καταστροφών*». Αναλυτικότερα, περιγράφονται οι βασικές έννοιες πάνω στις οποίες έχει θεμελιωθεί ως σήμερα η Θεωρία Διαχείρισης Ρίσκου. Επιπλέον, παρουσιάζονται εννοιολογικές δομές αντιμετώπισης ρίσκου, ιδιαίτερα σε πληροφοριακά συστήματα ή συστήματα διαχείρισης έργου. Μέσα από μελέτες ετών παρουσιάζεται το πλαίσιο μέσα στο οποίο κινείται, δρα και αναπτύσσεται η διαχείριση ρίσκου έως και σήμερα, ενώ, την ίδια στιγμή, αναλύονται - μέσα από εκτενή μελέτη - οι υπάρχουσες μεθοδολογίες διαχείρισης ρίσκου και παρουσιάζεται η λειτουργία τους και τα οφέλη που παρέχουν σε ένα Κατανεμημένο Πληροφοριακό Σύστημα. Η ανάλυση ρίσκου είναι σε

πλήρη σχέση με την Ευρύτερη Θεωρία Καταστροφών. Έτσι, μέσα από τη θεωρία και τα μοντέλα καταστροφών προσαρμόζεται, αναπτύσσεται και υλοποιείται η Θεωρία Κινδύνου στα Πληροφοριακά Συστήματα.

Το τρίτο κεφάλαιο έχει τίτλο *«Μεθοδολογία Πρόληψης-Αντιμετώπισης και Αποφυγής Κινδύνων»*. Στο κεφάλαιο αυτό, αναλύεται διεξοδικά και θα παρουσιαστεί το μοντέλο Πρόληψης – Αποφυγής – Αντιμετώπισης κρίσεων. Το πρώτο κομμάτι του κεφαλαίου θα είναι αφιερωμένο στην πρόληψη. Παρουσιάζεται ένα μοντέλο πρόληψης καταστροφών με όλα τα βοηθήματα εκείνα που θα ενισχύσουν την αμυντική θέση του Αναλυτή Κινδύνου. Μέσα από το σημείο αυτό, γίνεται η μετάβαση στο δεύτερο κομμάτι του κεφαλαίου, που περιλαμβάνει την Αποφυγή με την υλοποίηση, χρησιμοποίηση των αντίμετρων εκείνων που είναι ικανά να προλάβουν τον κίνδυνο. Τέλος, μέσα από το τρίτο κομμάτι της εφαρμογής παρουσιάζεται το καινοτόμο κομμάτι της εργασίας. Το κομμάτι της Αντιμετώπισης (αποκλιμάκωσης), το οποίο είναι το πιο σημαντικό, αφού δυστυχώς δεν μπορούν όλες οι κρίσεις να αποφευχθούν. Ένα κομμάτι βαρύνουσας σημασίας για την ομαλή λειτουργία του συστήματος, με το οποίο ο αναλυτής κινδύνου παροτρύνεται να χρησιμοποιήσει το κατάλληλο πακέτο Ανάκτησης καταστραμμένων δεδομένων που θα του επιφέρει τη βέλτιστη λύση. Σαν βέλτιστη λύση θεωρείται, ασφαλώς, η ενεργοποίηση κατάλληλων αντίμετρων με το ελάχιστο δυνατό κόστος.

Το τέταρτο κεφάλαιο της διατριβής έχει τίτλο «*Ψηφιακά Συστήματα Εθνικής Άμυνας»*. Περιγράφονται τα Ψηφιακά Συστήματα εθνικής άμυνας πάνω στα οποία εφαρμόζεται η διατριβή. Τα συστήματα αυτά παρουσιάζουν μια ιδιαιτερότητα στον τρόπο και στο περιβάλλον λειτουργίας καθώς και στις εξειδικευμένες ικανότητες που οφείλει να έχει το στρατιωτικό προσωπικό, που τα χειρίζεται. Οποιαδήποτε κρίση σε ένα τέτοιο σύστημα είναι καταστροφική έως θανατηφόρα για τους χρήστες του. Επίσης, αναλύεται η δομή ενός σημαντικού ψηφιακού συστήματος το οποίο έχει πρόσφατα τεθεί σε εφαρμογή και αναπτύσσεται, ο «Ψηφιακός Στρατιώτης».

Το πέμπτο κεφάλαιο της διατριβής έχει τίτλο *« Υλοποίηση Σεναρίων Κρίσεων σε Ψηφιακά Συστήματα Εθνικής Άμυνας»,* όπου παρουσιάζονται προσομοιωμένα σενάρια κρίσεων και ελέγχεται η αξιοπιστία του προτεινόμενου μοντέλου.

Το έκτο κεφάλαιο έχει τίτλο « *Ψηφιακά Πρωτόκολλα Διαχείρισης Κρίσεων»*. Στο κεφάλαιο αυτό παρουσιάζονται η δομή και τρόπος

δημιουργίας ψηφιακών πρωτοκόλλων διαχείρισης κρίσεων τα οποία δρουν σε ένα ασταθές, δυναμικά μεταβαλλόμενο περιβάλλον λειτουργίας. Παράλληλα, παρουσιάζεται αλγόριθμος λειτουργίας των πρωτοκόλλων μέσα σε ασταθές περιβάλλον.

Στο έβδομο κεφάλαιο, δίνονται τα συμπεράσματα της διδακτορικής διατριβής. Αρχικά, δίνονται κάποια συμπερασματικά σχόλια και παρατηρήσεις και στην πορεία παρουσιάζεται η συμβολή της διατριβής στον επιστημονικό χώρο. Επιπλέον, παρατίθενται οι δημοσιεύσεις, οι οποίες υποστηρίζουν την έρευνα της διδακτορικής διατριβής.

Στο όγδοο κεφάλαιο παρατίθεται ο κώδικας της εφαρμογής, η οποία αναπτύχθηκε κατά την διάρκεια της έρευνας της διδακτορικής διατριβής. Παράλληλα, παρουσιάζεται η εφαρμογή που δημιουργήθηκε στα πλαίσια της διδακτορικής έρευνας. Πρόκειται για ένα εργαλείο με το οποίο ο αναλυτής του συστήματος αποκτά τη δυνατότητα οροθέτησης των κινδύνων του συστήματος. Επιπλέον, η εφαρμογή ενημερώνει το χρήστη για τυχόν αδυναμίες του συστήματος, ενώ, σε περίπτωση καταστροφικής κρίσης ο αναλυτής του συστήματος μπορεί να χρησιμοποιήσει το κατάλληλο αντίμετρο με το ελάχιστο δυνατό κόστος ενεργοποίησης.

Στο κεφάλαιο «*Βιβλιογραφία»,* γίνεται μια βιβλιογραφική αναφορά σε όλες τις εργασίες που μελετήθηκαν και χρησιμοποιήθηκαν για τις ανάγκες της συγγραφής της διδακτορικής διατριβής.

ΚΕΦΑΛΑΙΟ 2

ΑΝΑΛΥΣΗ ΡΙΣΚΟΥ ΚΑΙ ΘΕΩΡΙΑ ΚΑΤΑΣΤΡΟΦΩΝ

2. ΘΕΩΡΙΑ ΡΙΣΚΟΥ

2.1 ΕΙΣΑΓΩΓΗ

"Η τέχνη του πολέμου μας διδάσκει να μη στηριζόμαστε στην πιθανότητα ότι ο εχθρός δεν θα έρθει αλλά στην ετοιμότητα μας να τον υποδεχθούμε."

Sun Zu, Η τέχνη του πολέμου. Κεφ 8. §11

Τις τελευταίες δεκαετίες υπήρξε μια ραγδαία εξέλιξη της Τεχνολογίας. Ένας μεγάλος αριθμός εφαρμογών συνέβαλε στην αλλαγή του τρόπου ζωής. Προβλήματα που στο παρελθόν φαίνονταν δυσεπίλυτα αντιμετωπίστηκαν, ανοίγοντας νέους δρόμους προόδου, εξέλιξης και προοπτικής. Πολλά από αυτά τα τεχνολογικά επιτεύγματα πέτυχαν τη μείωση του κόστους παραγωγής, άλλα πάλι μείωσαν σημαντικά το χρόνο και την απόσταση ενώ, ευρύτερα δεν έμεινε κανένας τομέας της σύγχρονης ζωής ανεπηρέαστος, αλλάζοντας τον τρόπο ζωής του Ανθρώπου. Ιδιαίτερα ευαίσθητοι τομείς, όπως **η Άμυνα, η Οικονομία,** αλλά και η **Γεωπολιτική** επηρεάσθηκαν σημαντικά σε επίπεδο οριστικής μετάλλαξης αυτών σε πολλά από τα υποσυστήματα τους.

Όπως ήταν αναμενόμενο, η πρόοδος αυτή έφερε νέα δεδομένα στους χώρους εφαρμογής της άγνωστα μέχρι τότε. Τα Σύγχρονα Αμυντικά Συστήματα είναι σήμερα πλήρως εξαρτημένα από τη Σύγχρονη Τεχνολογία και Πληροφορική. Επιπρόσθετα, δημιουργήθηκαν νέοι χώροι δεδομένων, που χαρακτηρίζονται από Αστάθεια μέσα στον ορίζοντα προγραμματισμού**.** Τα **Δυναμικά μεταβαλλόμενα πεδία** των χώρων αυτών οδηγούν αναγκαστικά στη **μη Ομαλή, μη προβλέψιμη λειτουργία** του Συστήματος που υπάρχει μέσα τους.

Άμεση συνέπεια αυτής της συνεχούς μεταβολής των πεδίων είναι η **μετάλλαξη** πολλών Συστημάτων και η εμφάνιση νέων απροσδιόριστων μορφών στα συστήματα με συχνά εκφυλισμένες πια δομές. Παράλληλα, προκαλείται η διάσπαση των συνεκτικών δεσμών, μεταξύ των υποσυστημάτων που απαρτίζουν το σύστημα, και τακτικά παρατηρείται η ολική κατάρρευση του. Συνεπώς, η κατάρρευση αυτή συντελεί καθοριστικά στη δημιουργία Κρίσεων και κινδυνικών καταστάσεων. Οι Κρίσεις αυτές είναι συχνά απροσδιόριστης μορφής, με ανεπάντεχα αποτελέσματα, που καταλήγουν κατά κανόνα σε μεταλλάξεις των Συστημάτων, όπως ήδη αναφέρθηκε. Πολλές φορές οι κρίσεις αυτές οφείλονται σε **Ασύμμετρες Απειλές** που δύσκολα μπορούν να

προβλεφθούν και να αποκλιμακωθούν. Αυτή είναι η αιτία που οι ασύμμετρες απειλές καταλήγουν να έχουν συχνά καταστροφικές συνέπειες.

2.2 ΘΕΩΡΙΑ ΚΑΤΑΣΤΡΟΦΩΝ

2.2.1 ΓΕΝΙΚΕΣ ΕΝΝΟΙΕΣ

Η εμφάνιση ασταθών χώρων δεδομένων αύξησε σημαντικά την πιθανότητα εμφάνισης καταστροφικών δυνάμεων πάνω σε ένα σύστημα. Το περιβάλλον λειτουργίας έπαψε πια να είναι ομαλό. Οι μη ομαλές καταστάσεις άλλαξαν τη δομή λειτουργίας των πληροφοριακών συστημάτων, εισάγοντας νέες έννοιες και λειτουργίες. Οι αλλαγές αυτές γίνονται με τρόπο που τα συμβατικά μαθηματικά μοντέλα δεν μπόρεσαν να υπολογίσουν, αφού είναι αναγκαία η χρήση χαοτικών μαθηματικών μοντέλων. Το 1960 η Θεωρία των Καταστροφών που αναπτύχθηκε από το Γάλλο Μαθηματικό Rene Thom και επεκτάθηκε αργότερα από τον Christopher Zeeman κατάφερε να εξηγήσει, με ποιοτικό τρόπο, τις καταστροφικές δυνάμεις που προκαλούν ασυνέχειες σε ομαλούς χώρους δεδομένων.

Είναι γεγονός ότι υπάρχουν φαινόμενα που διαταράσσουν τη συνέχεια μεταξύ των μεταβλητών, οι οποίες λαμβάνουν μέρος σε αυτά. Χαρακτηριστικό παράδειγμα από τη φυσική ο κλάδος της Θερμοδυναμικής με τη διαφορική εξίσωση Van der Waals για την εντροπία. Η Θεωρία Καταστροφών προσπαθεί να ερμηνεύσει τέτοια φαινόμενα, αρκεί να υπόκεινται σε ορισμένους περιορισμούς, έχει θεμελιωθεί στην τοπολογία, ενώ τα αποτελέσματα της είναι ποιοτικά και όχι ποσοτικά. Είναι, λοιπόν, κατάλληλη για να περιγράφει και να προβλέπει το σχήμα των διεργασιών. Η θεωρία των καταστροφών είναι ένας τρόπος να δούμε τι υπάρχει πιο πέρα. Βασίζεται στην παραδοχή της δομικής σταθερότητας, τονίζοντας την ποιοτική κανονικότητα. Εισαγωγικές έννοιες είναι τι είναι **Σύστημα** και η έννοια της **Καταστροφής**.

Ως **Σύστημα** ορίζεται, σύμφωνα με την γενική θεωρία συστημάτων, ένα **σύνολο από αλληλοεπιδρώμενα δομικά στοιχεία** τα οποία αποτελούν μέρος μιας οντότητας. Τα δομικά αυτά στοιχεία είναι συνδεδεμένα και πολλές φορές αλληλοεξαρτώμενα, έτσι ώστε να εκτελούν συγκεκριμένες λειτουργίες που απαιτούνται για την ομαλή ύπαρξη του συστήματος. Ένα σύστημα χωρίζεται συνήθως σε τρία

βασικά υποσυστήματα που το καθένα επεξεργάζεται και παράγει διαφορετικά δεδομένα ανάλογα με την επίδραση που δέχεται από το εξωτερικό περιβάλλον τη συγκεκριμένη χρονική στιγμή. Αυτά συνήθως είναι [Κιουντούζης 1995]:

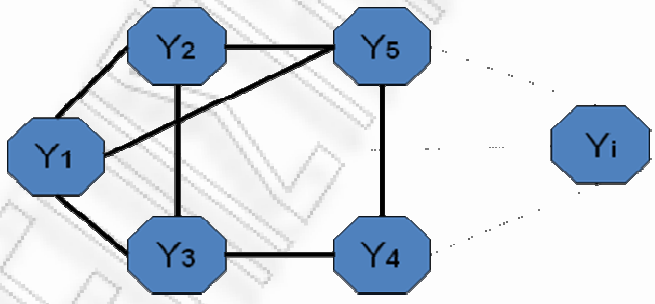
- **Το Φυσικό Σύστημα Παραγωγής.**
- **Το Σύστημα Λήψης Αποφάσεων.**
- **Το Ψηφιακό Σύστημα**.

Επιπλέον κάθε Πραγματικός Χώρος μπορεί να ορισθεί ως ένα Σύστημα (Σ) με δομή (Δ) και λειτουργικότητα (Λ) της μορφής:

$$\Sigma = (\Delta , \Lambda)$$

όπου Δ: το σύνολο των Υποχώρων του Συστήματος,
$\Delta=\{Y_1,Y_2,\ldots,Y_i ,\ldots Y_n\}$ και
Λ: οι σχέσεις των Υποχώρων του Συστήματος,
$\Lambda=\{(Y_1 , Y_2),\ldots,(Y_i , Y_j), \ldots\}$

Το παρακάτω σχήμα παρουσιάζει τη δομή ενός τέτοιου χώρου.



Σχήμα 2.1
Δομή ενός Πραγματικού Χώρου δεδομένων

Κάθε σύγχρονο Ολοκληρωμένο Πληροφοριακό Σύστημα προϋποθέτει την ορθή συλλογή πληροφορίας προκειμένου να επιτευχθεί η ομαλή και συνεχής λειτουργία του. Η περισσότερη πληροφορία δεν είναι στις κορυφές Yi του Συστήματος, αλλά στις σχέσεις (Yi,Yj) μεταξύ των κορυφών.

Καταστροφή ονομάζουμε την **απότομη μεταβολή της ομαλής πορείας** ενός συστήματος μετά από την επίδραση σε αυτό μιας εξωτερικής αιτίας ή, όπως αναφέρει ο Ι. Θ. Χαΐνης, «καταστροφή είναι

το απότομο άλμα που επιτρέπει στο σύστημα να υφίσταται, όταν κανονικά θα όφειλε να πάψει να υπάρχει»[Χαΐνης 2005].

Βασική προϋπόθεση ώστε να είναι δυνατή η μελέτη ενός καταστροφικού φαινομένου, είναι, όπως είπαμε και παραπάνω, τα φαινόμενα που εξετάζουμε να υπόκεινται σε ορισμένους περιορισμούς καθώς επίσης και να υπάρχουν a priori ορισμένα δεδομένα. Μια ακόμη προϋπόθεση είναι να μην υπάρχει εξωτερική επίδραση τέτοια που να μεταβάλλει την ευστάθεια του συστήματος, εφόσον η Θεωρία Καταστροφών εφαρμόζεται μόνο σε ευσταθή συστήματα. Για αυτό το λόγο, δεν μπορεί να εφαρμοστεί στον υπό μελέτη Ασταθή χώρο δεδομένων της παρούσης Διατριβής.

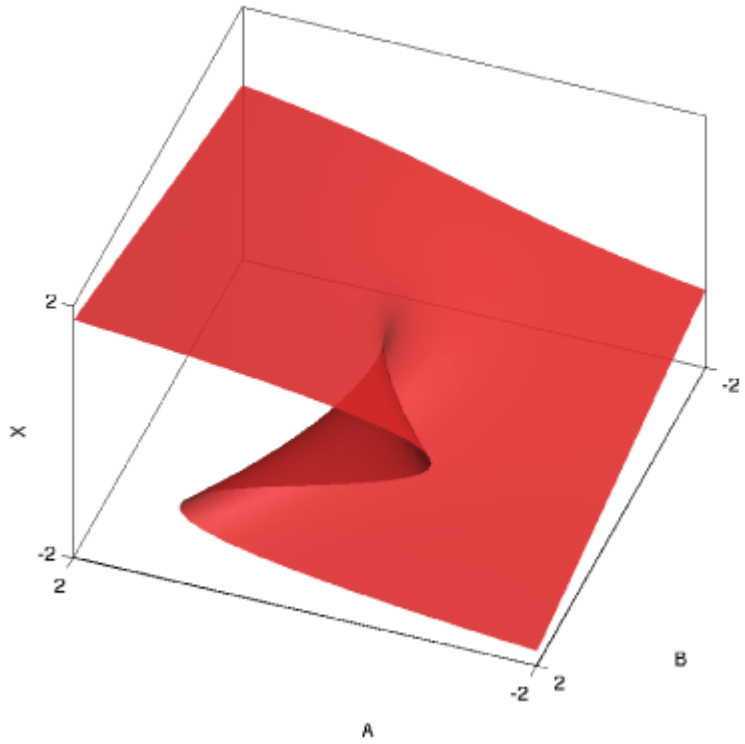
Παρόλα αυτά, αν πληρούνται οι παραπάνω βασικές προϋποθέσεις σύμφωνα με το **θεώρημα κατατάξεως** του Rene Thom, καθένα από τα καταστροφικά φαινόμενα που παρατηρούνται στη φύση κατατάσσονται χωριστά σε μια από τις **επτά στοιχειώδεις καταστροφές**. Καταστροφή, σύμφωνα με τον Thom, είναι κάθε **ασυνεχής μετάβαση** που συμβαίνει όταν ένα σύστημα μπορεί να έχει περισσότερες από μία σταθερές καταστάσεις. Σημαντικό είναι βέβαια ότι η αναγωγή του μοντέλου μιας καταστροφής σε μια από τις επτά στοιχειώδεις δεν είναι μονοσήμαντη. Μπορεί, επομένως, ένα μοντέλο καταστροφής να αντιστοιχισθεί σε δυο ή περισσότερες από τις επτά και όχι πάντα σε μια.

2.2.2 ΜΟΝΤΕΛΑ ΕΙΔΙΚΩΝ ΚΑΤΑΣΤΡΟΦΙΚΩΝ ΣΥΝΑΡΤΗΣΕΩΝ

Οι στοιχειώδεις καταστροφές της Ειδικής Θεωρίας Καταστροφών, όπως προαναφέραμε, χωρίζονται σε επτά κατηγορίες. Οι καταστροφές αυτές αποτελούν την αρχική αιτία για τη μετάβαση ενός πληροφοριακού συστήματος από μια κατάσταση σε μια άλλη. Μπορούν να παρασταθούν με γραφήματα που δείχνουν σταθερές καταστάσεις σαν σειρές σημείων σε ένα συγκεκριμένο περιβάλλον λειτουργίας. Σύμφωνα με τους Thom και Zeeman, κάθε γράφημα έχει έναν άξονα για κάθε παράγοντα ελέγχου, που καθορίζει την συμπεριφορά του συστήματος και έναν άξονα συμπεριφοράς.

Έτσι, η απλούστερη στοιχειώδης καταστροφή είναι **η πτυχή** που έχει έναν άξονα έλεγχου και συμπεριφοράς και είναι δυσδιάστατη. Ένα τέτοιο σύστημα σε αυτήν την κατάσταση μπορεί να οδηγηθεί σε συγκεκριμένη δομή ελάχιστου δυναμικού [Woodcock 1988].

Η καταστροφή **τύπου αιχμής** συμβαίνει σε συστήματα με δυο παράγοντες έλεγχου. Το γράφημα είναι τρισδιάστατο. Κάθε σημείο παριστάνει κατάσταση ισορροπίας, εκτός των σημείων που βρίσκονται στην πιέτα και είναι ιδιαίτερα ασταθή. Ένα τέτοιο γράφημα είναι της μορφής $V(x,u,v) = x^4+ux^2+vx$. Τα σημεία στα όρια της πιέτας είναι ημιασταθή. (Εικόνα 2.1)

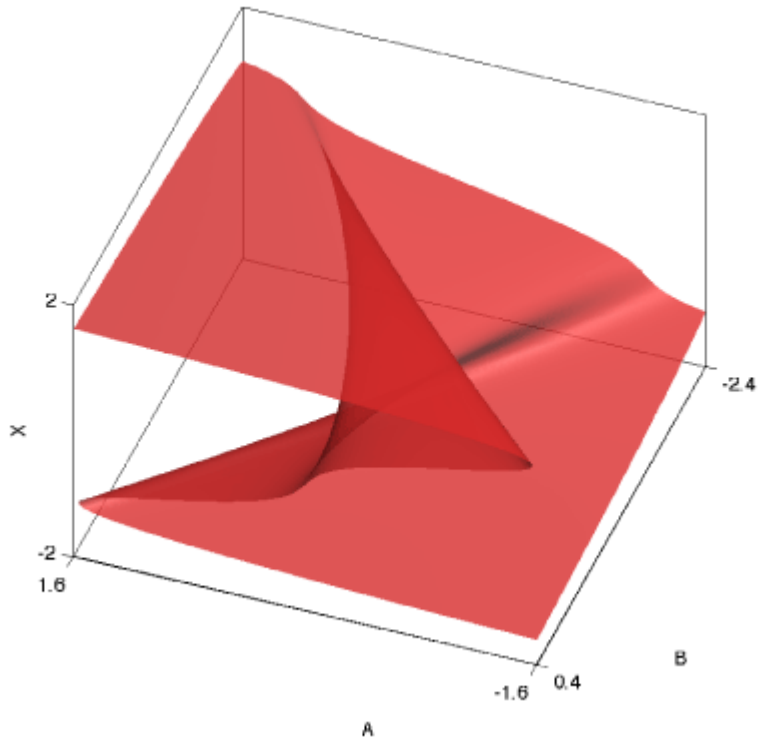


Εικόνα 2.1
Γράφημα καταστροφής τύπου αιχμής

Η καταστροφή **τύπου χελιδονοουράς** χρησιμοποιείται για την μελέτη καταστροφικών διεργασιών, όπου η συμπεριφορά εξαρτάται από τρεις παράγοντες ελέγχου. Το γράφημα είναι τετρασδιάστατο. Η καταστροφή χελιδονοουράς υλοποιείται όταν ένα σύστημα αφήνει την επιφάνεια προς άλλο στρώμα της επιφάνειας ή σε σημείο κάτω από αυτήν.

Η καταστροφή **τύπου πεταλούδας** εξαρτάται από τέσσερις παράγοντες έλεγχου. Το γράφημα είναι πενταδιάστατο. Το πρότυπο της πεταλούδας παρουσιάζει ποικιλομορφία συμπεριφορών λόγω του μεγάλου αριθμού παραγόντων έλεγχου. Όπως φαίνεται και από το σχήμα, (Εικόνα 2.2) το γράφημα σχηματίζει τρία ξεχωριστά στρώματα με το

μεσαίο στρώμα να είναι σε μια κατάσταση συμβιβασμού μεταξύ των δυο άλλων ακραίων καταστάσεων. [Woodcock 1988]



Εικόνα 2.2
Γράφημα καταστροφής τύπου πεταλούδας

Το γράφημα **ομφαλικής καταστροφής (υπερβολικό)** έχει δυο άξονες συμπεριφοράς, ενώ είναι ένα πενταδιάστατο γράφημα. Η γεωμετρία του καλύπτει ένα πλήθος χαρακτηριστικών συμπεριφοράς και η μεγάλη συνθετικότητά του δυσκολεύει αρκετά στη μελέτη του. Παρόλα αυτά, όμως η χρησιμότητα του είναι μεγάλη αφού καλύπτει ένα ευρύ φάσμα φυσικών φαινομένων, για τα οποία απαιτείται η μελέτη μεγάλου αριθμού παραγόντων.

Το γράφημα **ομφαλικής καταστροφής (ελλειπτικό)** έχει και αυτό δυο άξονες συμπεριφοράς. Είναι ένα πενταδιάστατο γράφημα. Ο μεγάλος αριθμός παραγόντων συμπεριφοράς το καθιστά ένα χρήσιμο εργαλείο προτυποποίησης καταστροφικών φαινομένων που χαρακτηρίζουν τον κλάδο των κοινωνικών επιστήμων.

Το γράφημα **ομφαλικής καταστροφής (παραβολικό)** έχει δυο άξονες συμπεριφοράς. Είναι, σε αντίθεση με τα προηγούμενα

γραφήματα, ένα εξαδιάστατο γράφημα. Σε αυτή την περίπτωση, η δομή των γραφημάτων είναι πλούσια. Αυτή η γεωμετρία δείχνει τη μεγάλη χρησιμότητα τους σαν ποιοτικά πρότυπα για σύνθετα φυσικά φαινόμενα, όπως το μηχανολογικό σχέδιο και η υδροδυναμική, προσφέροντας λύσεις και πιο πολύπλοκα προβλήματα στον τομέα της ψυχολογίας [Woodcock 1988]. Βέβαια, όλες αυτές οι κατηγορίες καταστροφών έχουν νόημα στην περίπτωση των ομαλών δυναμικών χωρών δεδομένων, όπου μέσα στον ορίζοντα προγραμματισμού τα δεδομένα μπορούν να αλλάξουν, αλλά σύμφωνα με κάποιο γνωστό μαθηματικοποιημένο νόμο.

2.3 ΟΡΟΛΟΓΙΑ ΔΙΑΧΕΙΡΙΣΗΣ ΚΙΝΔΥΝΟΥ

Η ασφάλεια των υπολογιστών υποφέρει από το ανακριβές λεξιλόγιο που χρησιμοποιεί. Σήμερα, έχουν γίνει αρκετές προσπάθειες για την διευθέτηση αυτής της εννοιολογικής κρίσης. Παραθέτουμε μια λίστα με τους πιο συχνά χρησιμοποιούμενους όρους της διαχείρισης κινδύνων, καθώς και τον ορισμό τους. Μέσα από αυτήν τη λίστα, μπορούμε εύκολα να διακρίνουμε και τα υπάρχοντα είδη κινδύνων που μπορεί να βρεθεί εκτεθειμένο ένα σύστημα.

Εκμετάλλευση πρόσβασης	Ακατάλληλη χρήση των πόρων του συστήματος
Παρακολούθηση τηλεπικοινωνιακών γραμμών	Μη εξουσιοδοτημένη παρακολούθηση των τηλεπικοινωνιακών μέσων
Εξουσιοδοτημένη εκμετάλλευση πρόσβασης από μη εργαζόμενο	Εκμετάλλευση πρόσβασης από έναν εξωτερικό συνεργάτη
Ιός υπολογιστή	Ένα λογισμικό το οποίο-άμεσα ή έμμεσα-μπορεί να δημιουργήσει ένα επικίνδυνο περιστατικό
Απώλεια αξιοπιστίας συστήματος δεδομένων	Απώλεια δεδομένων πληροφοριακού συστήματος
Denial of service	Μη εξουσιοδοτημένη διακοπή της εξυπηρέτησης του δικτύου ή των εφαρμογών του
Καταστροφή υπολογιστικών πόρων	Ίδιο με το σαμποτάζ
Καταστροφή δεδομένων	Διαγραφή ή μόνιμη αλλοίωση δεδομένων

Εκμετάλλευση πρόσβασης από εργαζόμενο	Ακατάλληλη χρήση του συστήματος από εργαζόμενο
Οικονομική απάτη	Η χρήση ψεύτικων τεχνικών για οικονομικό κέρδος
Hacking σε τηλεφωνική γραμμή	Παρόμοιο με την τηλεφωνική απάτη
Απώλεια, διαγραφή ή μετατροπή δεδομένων	Αφαίρεση κατοχής πληροφοριών λόγω κλοπής
Εσωτερική εκμετάλλευση δικτυακής πρόσβασης	Ακατάλληλη χρήση της δικτυακής πρόσβασης από κάποιον που έχει εξουσιοδοτημένη πρόσβαση
Κλοπή φορητού υπολογιστή	Μη εξουσιοδοτημένη αφαίρεση του φορητού υπολογιστή
Διαρροή σημαντικών πληροφοριών	Ίδιο με την κλοπή σημαντικών πληροφοριών
Παραποίηση λογισμικού εφαρμογών	Μη εξουσιοδοτημένη αλλαγή των εφαρμογών
Παραποίηση λογισμικού συστήματος	Μη εξουσιοδοτημένη αλλαγή του λογισμικού συστήματος
Σαμποτάζ δεδομένων ή δικτύου	Καταστροφή δεδομένων ή επηρεασμός της λειτουργίας του δικτύου
Εισχώρηση ξένου προσώπου στο σύστημα	Μη εξουσιοδοτημένη πρόσβαση στο σύστημα από κάποιον ξένο που δεν συνεργάζεται με τον ιδιοκτήτη του συστήματος
Παρακολούθηση τηλεπικοινωνιών	Μη εξουσιοδοτημένη παρακολούθηση μιας τηλεφωνικής συνομιλίας
Τηλεπικοινωνιακή απάτη	Κλοπή τηλεπικοινωνιακών υπηρεσιών
Κλοπή υπολογιστικών πόρων	Χρήση του υπολογιστή ή του δικτύου για μη ορθό σκοπό
Κλοπή δεδομένων, ανταλλαγή μυστικών	Ίδιο με την κλοπή σημαντικών πληροφοριών
Trojan horse	Ένα κρυφό λογισμικό που ενεργοποιείται για να μεταβάλλει κάτι στον υπολογιστή, μια μορφή ιού του υπολογιστή
Μη εξουσιοδοτημένη εισαγωγή στο δίκτυο	Πρόσβαση στο δίκτυο χωρίς προηγούμενη συγκατάθεση του

	διαχειριστή δικτύου
Μόλυνση από ιούς	Η εισχώρηση ιών σε ένα υπολογιστή

2.4 ΔΙΑΧΕΙΡΙΣΗ ΚΙΝΔΥΝΟΥ

2.4.1 ΓΕΝΙΚΕΣ ΕΝΝΟΙΕΣ

Ο **κίνδυνος (risk)**, η δυνατότητα ζημιάς ή απώλειας, περιγράφεται κυρίως με την εξάρτηση μεταξύ της απειλής και της ευπάθειας ή της επίπτωσης και της δυνατότητας. Έχουν αναπτυχθεί πολλοί ορισμοί με τους οποίους περιγράφεται ο κίνδυνος, ανάλογα με το είδος της εφαρμογής που μελετάται. Ο κίνδυνος ορίζεται σαν μια ενέργεια ή σύνολο ενεργειών πάνω σε ένα πληροφοριακό σύστημα με **αρνητικό συνήθως αποτέλεσμα**.

Στη **μηχανική** ορίζεται ως κίνδυνος η πιθανότητα να συμβεί ένα ατύχημα επί τις συνέπειες του ατυχήματος συνήθως μετρημένες σε θανάτους ή απώλεια χρημάτων.

Στα **πληροφοριακά συστήματα** κίνδυνος ορίζεται ως η πιθανότητα ένα αρνητικό συμβάν να λάβει χώρα, επί την πιθανότητα να εμφανιστούν κάποιες αδυναμίες και επί το πιθανό αποτέλεσμα. Συνεπώς, η μέτρηση του κινδύνου μπορεί να προσδιοριστεί ως παράγωγο των τιμών της απειλής, της ευπάθειας και των πόρων:

Κίνδυνος = Πόρος x Απειλή x Ευπάθεια

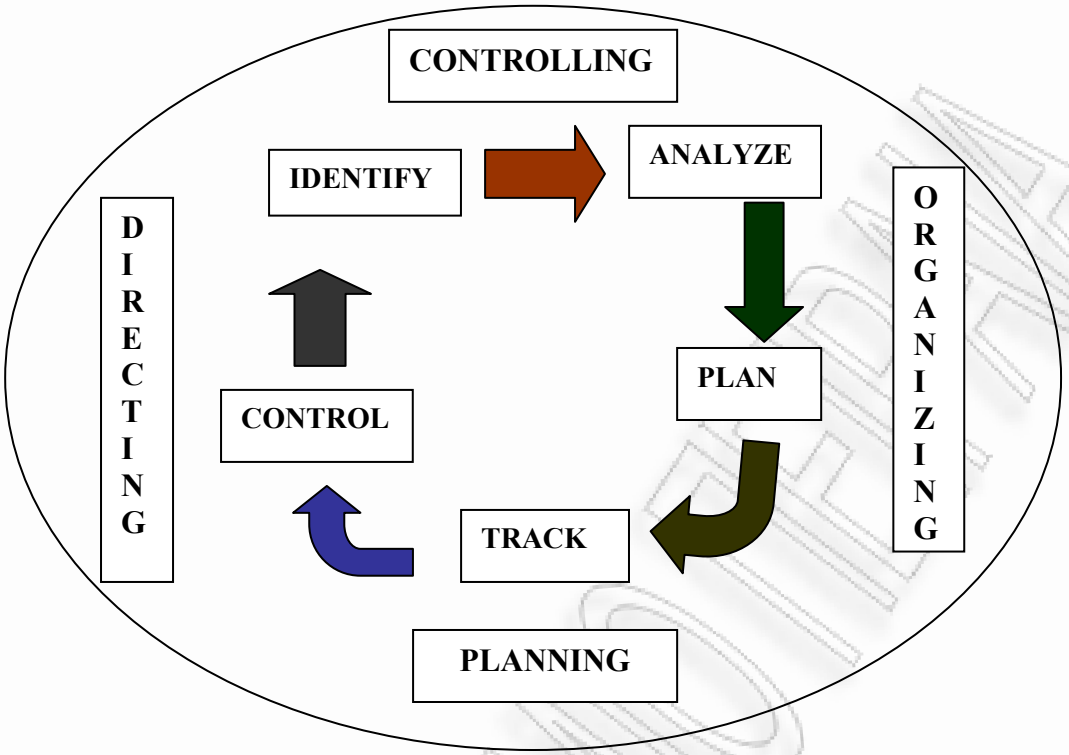
Το γενικό πλαίσιο καθορίστηκε στις αρχές της δεκαετίας του 1990, όπου παρουσιάσθηκε μια δομημένη διαδικασία διαχείρισης του κίνδυνου σαν **υποσύνολο της θεωρίας διαχείρισης έργων (project management)**. Ως έργο (Project) είναι μιά πολύπλοκη Εργασία, που πρέπει να γίνει μέσα σε έναν δεδομένο **ορίζοντα προγραμματισμού Δt = t2 – t1** με δεδομένα του παρόντος **t1**, έτσι ώστε να μεγιστοποιείται το κέρδος με ικανοποίηση των περιορισμών μέχρι και την χρονική στιγμή **t2**. Ένα έργο P αναλύεται σε Ενέργειες με ολική χρονική διάταξη:

$$E=\{e_1,e_2,\ldots,e_i,\ldots,e_n\}$$

Κάθε e_i είναι ένα **σύνολο** από **Δράσεις .** Το σύνολο **E** είναι μιά διαμέριση του Έργου : **Union(e_i) = Σύνολο Δράσεων, Τομή(e_k,e_d)=Ø**

Τα περισσότερα έργα εκείνη την περίοδο άρχισαν να έχουν αυξημένη πολυπλοκότητα, πράγμα που τα έκανε ιδιαίτερα ευάλωτα σε πληθώρα κινδύνων. Δυστυχώς, μέχρι και σήμερα, η πολυπλοκότητα των σύγχρονων ψηφιακών συστημάτων και ιδιαίτερα, η ευρεία διασπορά των υποσυστημάτων τους δεν έπαψε να υπάρχει, πράγμα που τα κάνει ιδιαίτερα ευάλωτα και ευπαθή σε πολύπλοκους, πολυπληθείς και πολύ αποτελεσματικούς κινδύνους. Η διαχείριση ρίσκου χειρίζεται τις λεπτές ισορροπίες στη διάρκεια δημιουργίας ενός project, κοιτώντας πάντα μπροστά σε **μεσομακροπρόθεσμο ορίζοντα προγραμματισμού**. Η διαχείριση ρίσκου προσφέρει στον ευρύτερο κλάδο του **Project Management μια δομημένη προσέγγιση της αναγνώρισης κινδύνων και της ανάλυσής των συνεισφέροντας θετικά στην ομαλή ανάπτυξη και επιτυχημένη ολοκλήρωση του υπό δημιουργία πληροφοριακού έργου** [Higuera et al. 1994]. Η αλληλεπίδραση των δυο θεωριών φαίνεται στο Σχήμα 2.2. Συγκεκριμένα, μέσα στο πλαίσιο υπάρχουν οι τέσσερις βασικές δομές - που σύμφωνα με το συγγραφέα - αντιπροσωπεύουν τον ευρύτερο τομέα της θεωρίας διαχείρισης έργου. **Η οργάνωση, ο σχεδιασμός, η διεύθυνση και ο έλεγχος** είναι οι θεμελιώδεις αυτές δομές. Μέσα σε αυτό το περιβάλλον βρίσκεται εμφωλευμένο το ανατροφοδοτούμενο πλαίσιο πάνω στο οποίο στηρίζεται η θεωρία ρίσκου στα σύγχρονα πληροφοριακά συστήματα και περιλαμβάνει **την αναγνώριση, την ανάλυση, το σχεδιασμό και τον έλεγχο** [Higuera et al. 1994]. Η **Project Risk Analysis and Management** είναι το σύνολο των διαδικασιών που απαιτούνται να σχεδιαστούν, προκείμενου να ελαχιστοποιηθεί ο κίνδυνος που απειλεί το υπό μελέτη σύστημα, ακόμα και με ολική κατάρρευση. Στόχος της μεθοδολογίας είναι η αναγνώριση των κινδύνων, που απειλούν το υπό μελέτη σύστημα [Norris 1992]. Αυτό το γεγονός έκανε του διαχειριστές των έργων να στραφούν σε μεθοδολογίες ελαχιστοποίησης του κινδύνου. Το γενικό πλαίσιο αποτελείται από πέντε τομείς πάνω στους οποίους ο project manager αναλύει τους κινδύνους, που απειλούν το πληροφοριακό σύστημα:

- **πόροι,**
- **απειλές,**
- **επιπτώσεις,**
- **πιθανότητα,**
- **συστήματα ασφαλείας.**



Σχήμα 2.2
Αλληλεπίδραση Ανάλυσης Ρίσκου στο ευρύτερο πλαίσιο της θεωρίας έργου.
[Higuera et al. 1994]

Υπάρχει ομοφωνία μεταξύ των ειδικών στην ασφάλεια των πληροφοριακών συστημάτων ότι **δεν μπορεί να υπάρξει 100% ασφάλεια**, με άλλα λόγια μηδενικός κίνδυνος. Ακόμα και αν υποθέσουμε ότι η ολική εξάλειψη του κινδύνου είναι δυνατή, αυτό θα παρεμποδιζόταν από τους περιορισμούς του προϋπολογισμού που τυχόν θα υπάρχουν και οφείλονται στις περισσότερες περιπτώσεις στο κόστος των μέτρων που πρέπει να ληφθούν καθώς στοιχίζουν περισσότερο από το κόστος των πόρων, οι οποίοι θα προστατεύονταν. Συνεπώς, η έμφαση σε ότι αφορά τους κινδύνους επεκτείνεται από την **αποφυγή του κινδύνου** στη **διαχείριση του κινδύνου (Risk Management).**

Διαχείριση κινδύνου είναι το σύνολο των διαδικασιών που απαιτούνται να αναπτυχθούν σε ένα έργο προκείμενου να μειωθεί η πιθανότητα εμφάνισης καταστροφικών δυνάμεων σε αυτό. Παράλληλα, στόχος της διαχείρισης κινδύνου είναι η αποφυγή των αρνητικών συνεπειών που θα έχει η υλοποίηση ενός επικίνδυνου φαινομένου. Πολλές φορές, βέβαια, η διαχείριση κινδύνου αντιμετωπίζει σοβαρά διλήμματα, αφού καλείται να υλοποιήσει διαδικασίες αντιμετώπισης κινδύνων με χαμηλή πιθανότητα εμφάνισης, αλλά υψηλή καταστροφική

δύναμη και κινδύνων με υψηλή πιθανότητα εμφάνισης, αλλά χαμηλή καταστροφική ικανότητα.

2.4.2 ΜΕΘΟΔΟΛΟΓΙΕΣ ΔΙΑΧΕΙΡΙΣΗΣ ΚΙΝΔΥΝΟΥ

Η διαχείριση ρίσκου παρόλα αυτά είναι προαπαιτούμενο όχι μόνο για τη δημιουργία μιας πολιτικής ασφαλείας, αλλά και για τη βήμα προς βήμα κατασκευή ενός ολοκληρωμένου, σταθερού, με γερά θεμέλια, σύγχρονου πληροφοριακού συστήματος. Το κυριότερο στοιχείο είναι η αναγνώριση των αντιμέτρων προστασίας και η χρήση των καταλληλότερων από αυτά ώστε να αντιμετωπισθεί με επιτυχία μια τυχόν μη ομαλή λειτουργία του συστήματος.

Η διαχείριση κινδύνου λοιπόν, είναι ένας κλάδος της γενικής θεωρίας διαχείρισης έργου που έχει παρουσίασει ιδιαίτερη ανάπτυξη την τελευταία δεκαετία. Οι απόψεις των μελετητών είναι σε γενικές γραμμές ταυτόσημες, με έντονη επιρροή μεταξύ των συγγραφέων.

Σύμφωνα με τον Wideman [Wideman 1992] υπάρχουν **τέσσερα βασικά στάδια** στη διαχείριση κρίσεων:

- **Η Αναγνώριση Κινδύνων (Risk Identification),**
- **Η Ποσοτικοποίηση Κινδύνων (Risk Quantification),**
- **Η Ανταπόκριση Κινδύνων (Risk Response Development),**
- **Η Έλεγχος Κινδύνων (Risk Response Control).**

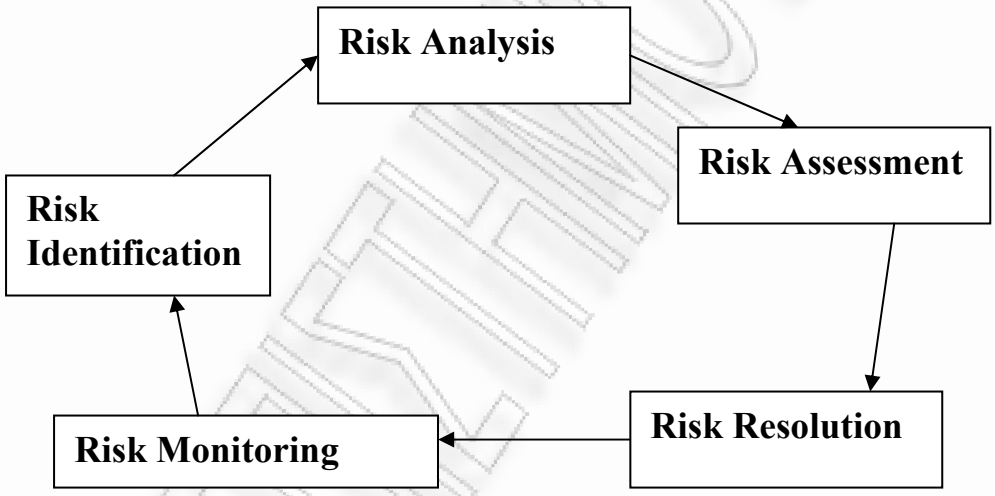
Από την άλλη μεριά, οι Chapman και Ward [Chapman Ward 1997] αυξάνουν τα στάδια μελέτης του κινδύνου. Η νέα αυτή άποψη προτείνει **εννέα επίπεδα** διαχείρισης κινδύνου. Ειδικότερα:

- **Προσδιορισμός (Define),**
- **Εστίαση (Focus),**
- **Αναγνώριση (Identify),**
- **Δομή (Structure),**
- **Κυριότητα (Ownership),**
- **Εκτίμηση (Estimate),**
- **Αξιολόγηση (Evaluate),**
- **Σχεδιασμός (Plan),**
- **Διαχείριση (Manage).**

Το 1993, ο Eloff ορίζει **πέντε τομείς διαχείρισης κρίσεων,** όπως φαίνονται παρακάτω: [Eloff 1993].

- **Η Αναγνώριση Κινδύνων (Risk Identification),**
- **Η Ανάλυση Κινδύνων (Risk Analysis),**
- **Η Αποτίμηση Κινδύνου (Risk Assessment),**
- **Η Επίλυση Κινδύνου (Risk Resolution),**
- **Παρακολούθηση Κινδύνων (Risk Monitoring).**

Οι πέντε αυτοί τομείς του Eloff έχουν κυκλική σύνδεση, όπως φαίνεται και από το παρακάτω σχήμα, (Σχήμα 2.3) και έχουν σαν σκοπό τη συνεχή βελτίωση του συστήματος και την αποτελεσματική αντιμετώπιση των κρίσεων.



Σχήμα 2.3
Μοντέλο διαχείρισης κρίσεων σύμφωνα με τον Eloff

Η πιο πρόσφατη προσέγγιση έγινε το 2000 από το Ινστιτούτο Διαχείρισης Έργου [PMI 2000] των Η.Π.Α. Σύμφωνα με την ανάλυση του ινστιτούτου, η διαχείριση κρίσεων είναι μια **πολυεπίπεδη μεθοδολογία,** η οποία χωρίζεται σε **έξι επίπεδα**:

- **Ανάπτυξη Σχεδίου Διαχείρισης Κινδύνου(Risk Management Planning),**
- **Αναγνώριση Κινδύνων (Risk Planning),**
- **Ποιοτική Ανάλυση Κινδύνων (Qualitative Risk Analysis),**
- **Ποσοτική Ανάλυση Κινδύνων (Quantitative Risk Analysis),**

- **Ανταπόκριση στον Κίνδυνο (Risk Response Planning),**
- **Παρακολούθηση και Έλεγχος Κινδύνων (Risk Monitoring and Control).**

Η βασική ανάλυση κινδύνου και η διαχείριση του ορίζονται - στην πλειοψηφία των περιπτώσεων - με παραπλήσιες δομές. Αυτό φαίνεται άλλωστε από την ονομασία και τη λειτουργία των διαφόρων επιπέδων αντιμετώπισης κινδύνου των μεθοδολογιών, οι οποίες παρουσιάσθηκαν παραπάνω.

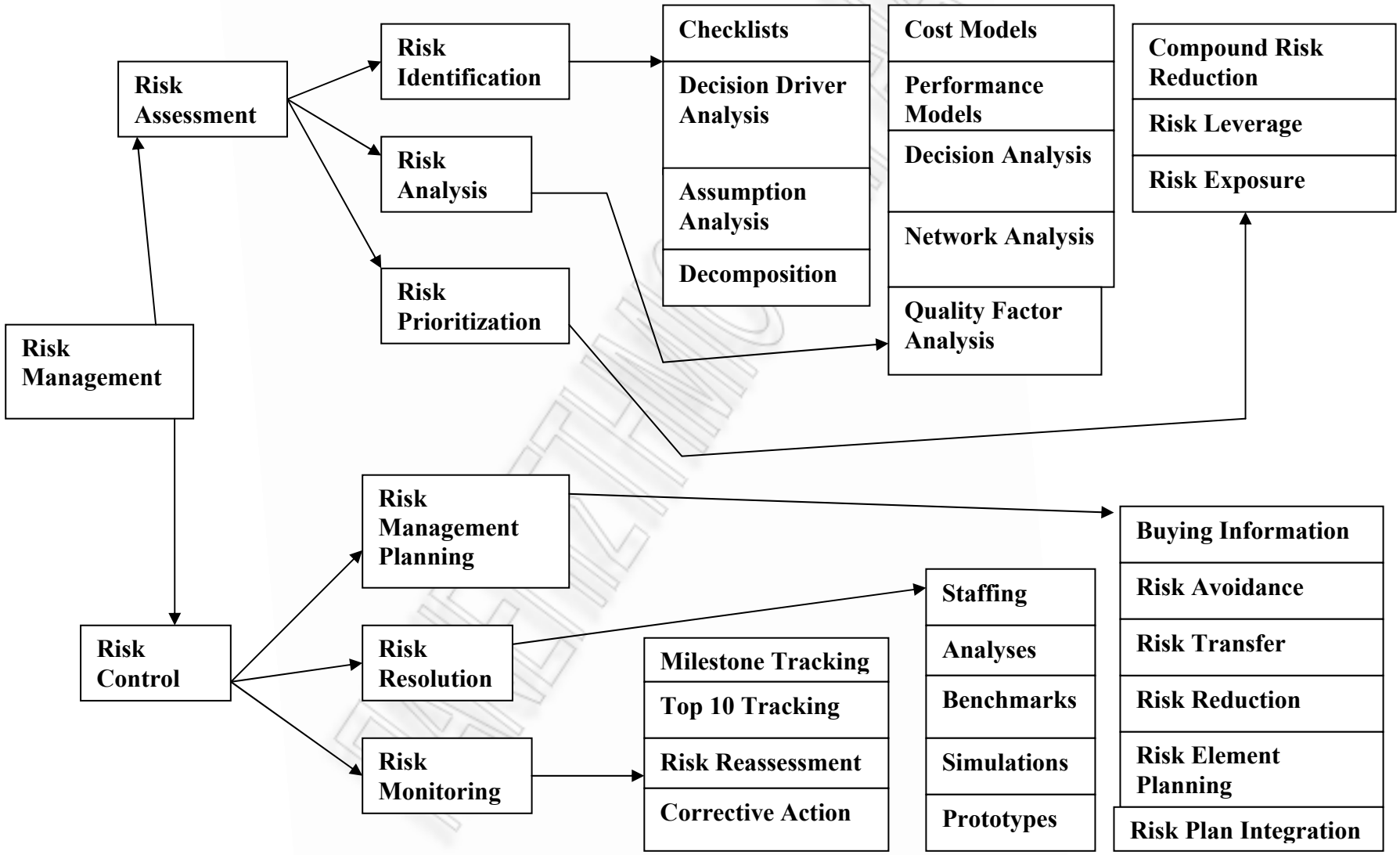
Έτσι, σε γενικές γραμμές, οι περισσότερες μεθοδολογίες στην προσπάθεια τους να γίνει η **ανάλυση του κινδύνου (Risk Analysis),** με εγκυρότητα και επιτυχία, κινούνται σε ένα στερεότυπο μοτίβο. Το μοτίβο αυτό περιλαμβάνει την **αναγνώριση (Risk Identification)** των κινδύνων, οι οποίοι απειλούν το πληροφοριακό σύστημα. Επιπλέον, περιλαμβάνει την **αποτίμηση των επιπέδων των κινδύνων (Risk Assessment),** που υπολογίζεται από τις γνωστές τιμές των δομών του συστήματος και των επιπέδων των απειλών του. Παράλληλα, σημαντικό ρόλο παίζουν και οι αδυναμίες αυτών των δομών. Η **διαχείριση του κινδύνου** περιλαμβάνει, εκτός από την αναγνώριση, την επιλογή και την υιοθεσία των αντιμέτρων που δικαιολογούνται από τους αναγνωρισμένους κινδύνους που απειλούν την ομαλή λειτουργία του πληροφοριακού συστήματος. Σκοπός της διαχείρισης είναι η μείωσή των κινδύνων σε αποδεκτά επίπεδα.

Η διαχείριση του κινδύνου, συγκεφαλαιώνοντας, είναι η διαδικασία της **επιλογής (Risk Resolution)** και της **υλοποίησης μέτρων (Risk Monitoring)** για την ελαχιστοποίηση των επιπτώσεων - απειλητικών συμβάντων σε μια επιχείρηση. Η ελαχιστοποίηση της επίπτωσης δε σημαίνει και αποφυγή του κινδύνου. Η **τέλεια ασφάλεια**, ή η αποφυγή όλων των απειλών που συμβαίνουν είναι **απείρως δαπανηρή, αν όχι ακατόρθωτη** με συνέπεια να μην αποτελεί ένα λογικό στόχο. Ο σκοπός είναι η επιλογή μέτρων διαχείρισης του κινδύνου, που η αξία της διατήρησης τους να **είναι σημαντικά κατώτερη** από το αναμενόμενο κόστος των ζημιών. Ο λόγος της μείωσης του αναμενόμενου κόστους προς το κόστος υλοποίησης των αντίμετρων λέγεται **Επιστροφή στην Επένδυση** (Return On Investment (ROI)).

Σημαντική είναι η άποψη του Boehm [Boehm 1991] σχετικά με τον τρόπο ανάλυσης του κινδύνου. Η πρόταση αυτή αποτελεί την θεμελιώδη βάση για τις μετέπειτα δομές διαχείρισης κρίσεων, οι οποίες παρουσιάσθηκαν σε προηγούμενες παραγράφους. Το παρακάτω σχήμα

ορίζει το πλαίσιο μέσα στο οποίο κινείται η μεθοδολογία που προτείνεται από τον Boehm.

2


Risk Management
Risk Assessment
Risk Identification
Risk Analysis
Risk Prioritization
Checklists
Decision Driver Analysis
Assumption Analysis
Decomposition
Cost Models
Performance Models
Decision Analysis
Network Analysis
Quality Factor Analysis
Compound Risk Reduction
Risk Leverage
Risk Exposure
Risk Control
Risk Management Planning
Risk Resolution
Risk Monitoring
Milestone Tracking
Top 10 Tracking
Risk Reassessment
Corrective Action
Staffing
Analyses
Benchmarks
Simulations
Prototypes
Buying Information
Risk Avoidance
Risk Transfer
Risk Reduction
Risk Element Planning
Risk Plan Integration

Η ανάλυση ρίσκου, σύμφωνα με τον Boehm, χωρίζεται σε δυο αρχικά στάδια. Το πρώτο προκαταρτικό βασικό επίπεδο περιλαμβάνει, όπως φαίνεται και στο παραπάνω σχήμα, το στάδιο της **αποτίμησης ρίσκου (risk assessment).** Το δεύτερο αντίστοιχο στάδιο περιλαμβάνει το κομμάτι που αφορά τον **έλεγχο του κίνδυνου (risk control).** Τα δυο αυτά επίπεδα είναι στην ουσία προπομποί για την εξειδικευμένη ανάλυση που κάνει παρακάτω ο συγγραφέας.

Καθένα από τα επίπεδα αυτά χωρίζεται σε τρία υποεπίπεδα, τα οποία αποτελούν τον κορμό της μεθοδολογίας. Τα υποεπίπεδα αυτά είναι το κομμάτι εκείνο στο οποίο εφαρμόζεται η θεωρία **αντιμετώπισης** κρίσεων, την οποία προτείνει ο συγγραφέας. Κάθε υποεπίπεδο ανάλογα με το είδος της λειτουργίας που επιτελεί χρησιμοποιεί συγκεκριμένα μοντέλα και χρήσιμες δομές για την επιτυχημένη λειτουργία του συστήματος ελέγχου και πρόληψης κρίσεων και για την αποφυγή δυσάρεστων καταστάσεων.

Τα θεμελιώδη στοιχεία της μεθοδολογίας Boehm [Boehm (1991)] είναι έξι, όπως φαίνεται και στο προηγούμενο σχήμα, και περιλαμβάνουν συγκεκριμένα:

- **Η Αναγνώριση Κινδύνου (Risk Identification),**
- **Η Ανάλυση Κινδύνου (Risk Analysis),**
- **Η Κατηγοριοποίηση Κινδύνου (Risk Prioritization),**
- **Η Σχεδίαση Διαχείρισης Κινδύνου (Risk Management Planning),**
- **Η Επίλυση Κινδύνου (Risk Resolution),**
- **Η Παρακολούθηση Κινδύνου (Risk Monitoring).**

Αυτά τα έξι στοιχεία αποτέλεσαν την ραχοκοκαλιά πάνω στην οποία θεμελιώθηκε η πλειοψηφία των μεθοδολογιών διαχείρισης κρίσεων σε ψηφιακά συστήματα. Ένα δείγμα από αυτές παρουσιάσθηκε παραπάνω. Μέσα σε αυτό το μονοπάτι, λοιπόν, θα κινηθεί και το μοντέλο διαχείρισης κρίσεων, το οποίο αναπτύσσει η παρούσα διατριβή προσθέτοντας σημαντικές τροποποιήσεις.

Παράλληλα, ένα σημείο που πρέπει να επισημανθεί είναι το γεγονός ότι η συντριπτική πλειοψηφία των μεθοδολογιών διαχείρισης ρίσκου ασχολείται με την ελαχιστοποίηση της πιθανότητας εμφάνισης ενός αρνητικού συμβάντος.

Ένα καίριο ερώτημα που γεννάται είναι τι συμβαίνει στην περίπτωση που το αρνητικό γεγονός λάβει χώρα. Είναι ικανές οι μεθοδολογίες αυτές να καλύψουν την περίπτωση υλοποίησης της κρίσης;

Η άποψη του Boehm αποτελεί, όπως είδαμε, μια πρωτοποριακή ανάλυση του κίνδυνου, καθώς καλύπτει όλους τους τομείς της πρόληψης του κινδύνου, προτείνοντας αναλυτικά βήματα για τη σωστή αντιμετώπιση μιας κρίσης σε ένα σύγχρονο πληροφοριακό σύστημα. Η μέθοδος αυτή καλύπτει δυστυχώς μόνο την πρόληψη, ενώ αδυνατεί να προσφέρει λύσεις στην πιθανή περίπτωση υλοποίησης ενός κινδύνου. Με λίγα λόγια, αποφεύγεται να αναλυθεί η περίπτωση ανάκτησης δεδομένων κατά τη διάρκεια και μετά το τέλος ενός συμβάντος. Συμπερασματικά, η μέθοδος του Boehm είναι χρήσιμη για αντίμετρα όχι όμως για διαχείριση Κρίσεως.

Ιδιαίτερο ενδιαφέρον παρουσιάζει η μεθοδολογία διαχείρισης ρίσκου του Fairley [Fairley 1994]. Η μεθοδολογία κάνει ένα μεγάλο και σημαντικό βήμα, προσθέτοντας νέα στοιχεία στη μέχρι τότε υπάρχουσα αντίληψη για τη διαχείριση ρίσκου. Καλύπτει το κενό που είχε αφήσει ο Boehm, συμπεριλαμβάνοντας την περίπτωση υλοποίησης ενός καταστρεπτικού φαινομένου. Η μέθοδος του Fairley αποτελείται από **επτά στάδια ανάλυσης και διαχείρισης** του ρίσκου σε ένα πληροφοριακό σύστημα, όπως παρουσιάζονται κάτωθι:

- **Η Αναγνώριση Κινδυνικών Παραγόντων (Identify Risk Factors),**
- **Ο Υπολογισμός Πιθανότητας Ρίσκου και συνεπειών στο σύστημα (Assess risk probabilities and effects on the project),**
- **Η Ανάπτυξη στρατηγικών για περιορισμό αναγνωρισμένων Κινδύνων (Develop strategies to mitigate identified risks),**
- **Η Παρακολούθηση Κινδυνικών Παραγόντων (Monitor risk factors),**
- **Η Ανακοίνωση Πλάνου Αντιμετώπισης (Invoke a contingency plan),**
- **Η Διαχείριση Κρίσεως (Manage Crisis),**
- **Η Ανάκτηση από την Κρίση (Recover from Crisis).**

Το πρώτο στάδιο της μεθοδολογίας περιλαμβάνει την αναγνώριση των κινδύνων, οι οποίοι απειλούν το πληροφοριακό σύστημα, ενώ

παράλληλα, επισημαίνει τις αλλαγές που μπορεί να εμφανίσει ο κίνδυνος, προσπαθώντας να προσδιορίσει όλες τις μορφές του κινδύνου.

Το δεύτερο στάδιο υπολογίζει τον κίνδυνο σαν γινόμενο της πιθανότητας εμφάνισης επί τη συνέπεια, που θα έχει το επικίνδυνο φαινόμενο στην ομαλή λειτουργία του έργου και επισημαίνει τα καταστροφικά αποτελέσματα κάθε πιθανού επικίνδυνου φαινομένου πάνω στη δομή του project, τονίζοντας ότι οι συνέπειες είναι θετικές ή αρνητικές, ανάλογα με την οπτική γωνία που αντιμετωπίζονται τα πράγματα σε συγκεκριμένες χρονικές στιγμές.

Το τρίτο στάδιο - σύμφωνα πάντα με την μέθοδο του Fairley - αναπτύσσει στρατηγικές για την αντιμετώπιση των αναγνωρισμένων παραγόντων υλοποίησης κρίσεων. Οι δυο στρατηγικές που προτείνονται στο επίπεδο αυτό είναι ο **σχεδιασμός ενεργειών (Action Planning)** και ο **σχεδιασμός μηχανισμού διατήρησης (Contingency Planning).** Ο σχεδιασμός ενεργειών αναφέρεται στα βήματα που πρέπει ο διαχειριστής του έργου να κάνει, προκειμένου να αντιμετωπίσει άμεσα μια κρίση. Είναι, δηλαδή, αντίμετρα τα οποία πρέπει να υλοποιηθούν άμεσα, ώστε να αποφευχθεί μια καταστροφή. Ένα αντίμετρο αποτελεί η έγκυρη αντικατάσταση του πεπαλαιωμένου hardware και ένα δεύτερο είναι η νέα αρχιτεκτονική σχεδιασμού έργου. Και τα δυο μαζί μπορούν να τεθούν σε εφαρμογή με μικρό κόστος απωλειών για τη συνολική δομή του έργου. Η δεύτερη κατηγορία είναι στο σύνολο των αντιμέτρων εκείνων τα οποία υλοποιούνται μακροπρόθεσμα, προκειμένου να αντιμετωπίσει μια μελλοντική κρίση.

Το τέταρτο επίπεδο ασχολείται με την παρακολούθηση των παραγόντων κινδύνου, η οποία πραγματοποιείται με τη βοήθεια αναφορών προόδου από τα επιμέρους τμήματα.

Ενδιαφέρον παρουσιάζουν τα επόμενα τρία επίπεδα. Τα τρία τελευταία επίπεδα στην ουσία είναι ένα γενικευμένο επίπεδο πάνω στο οποίο ορίζεται η μετά την κρίση κατάσταση του έργου. Δημιουργείται, συγκεκριμένα, ένα σχέδιο αντιμετώπισης καταστροφών, προκειμένου να αποφευχθεί μια κρίση. Δυστυχώς, όμως, αυτό δεν είναι πάντα εφικτό, με αποτέλεσμα, μέσα από τα δυο τελευταία επίπεδα η μεθοδολογία να προσπαθεί να χειριστεί το καταστροφικό φαινόμενο σε πρωταρχικό στάδιο. Παρουσιάζεται, λοιπόν, για πρώτη φορά μια **επιφανειακή ερμηνεία της έννοιας της ανάκτησης (recovery)**. Στην ουσία, δεν πρόκειται για ανάκτηση συστήματος, αλλά για αξιολόγηση της

υπάρχουσας κατάστασης διαχείρισης της κρίσης και επαναπροσδιορισμού του project μέσα από συγκεκριμένα βήματα, τα οποία οφείλουν να κάνουν οι κατασκευαστές του έργου, προκειμένου να μπορέσουν να το θέσουν σε επαναλειτουργία.

Ευρύτερα, στις περισσότερες μεθοδολογίες διαχείρισης έργου - project ο κίνδυνος τείνει να μελετάται από μια πολλή αρνητική όψη. Όπως προαναφέρθηκε, ο κίνδυνος είναι κάτι που μπορεί να συμβεί και επηρεάζει αρνητικά την προσπάθεια επίτευξης του στόχου. Αντίθετα, μια άλλη άποψη θέλει τον κίνδυνο να μην είναι πάντοτε η αρνητική συνέπεια. Πράγματι, μια κρίση δεν περιγράφει πάντοτε κάτι το οποίο βαίνει λανθασμένα, αλλά είναι σε γενικές γραμμές μια νέα κατάσταση που τελικά συμβαίνει διαφορετικά από μια άλλη κατάσταση, η οποία αναμενόταν ή ήταν σχεδιασμένο να συμβεί. Αυτή η άποψη επιτρέπει την πιθανότητα ο κίνδυνος να μετατραπεί σε ευκαιρία, αν διαχειρισθεί καταλλήλως. Ένα τέτοιο ιστορικό παράδειγμα στην επιστήμη της Πληροφορικής είναι ο B. Gates το 1981 με την εμφάνιση της ΜικροΠληροφορικής, η οποία βέβαια κατέστρεψε κάθε τι που είχε σχέση με την Πληροφορική των Μεγάλων Mainframes!

Έτσι, λοιπόν, ένας άλλος ορισμός του κινδύνου είναι: **Ένα μελλοντικό συμβάν ή ακολουθία συμβάντων, που υπάρχει το ενδεχόμενο να εμφανιστεί απώλεια ή επίπτωση σε αντικείμενα είτε θετική είτε αρνητική.** Είναι, συνεπώς, πιο ορθό να μιλάμε όχι απλώς για διαχείριση του κινδύνου, αλλά για Διοίκηση Κινδύνου και των επιπτώσεων του.

2.5 ΑΝΑΓΝΩΡΙΣΗ ΚΙΝΔΥΝΟΥ

Η αναγνώριση του κινδύνου είναι οπωσδήποτε το πρώτο βήμα για την αντιμετώπισή του, αλλά είναι και ο τομέας που φέρνει στην επιφάνεια όλα τα ανθρώπινα μειονεκτήματα, τα οποία εμπλέκονται μέσα στη λήψη αποφάσεων. Είναι προφανές ότι αν ο αναλυτής δεν έχει σκεφτεί την πιθανότητα εμφάνισης ενός κινδύνου, δεν μπορεί να δράσει και να σχεδιάσει αντίμετρα, προκειμένου να αντιμετωπίσει μια κρίσιμη κατάσταση. Σε μια ανώριμη επιχείρηση, ο προϊστάμενος του project μπορεί να θεωρηθεί υπεύθυνος για την εμφάνιση μιας πρωτόγνωρης απειλής μόνο και μόνο επειδή επεσήμανε την ύπαρξη της.

Η ανάπτυξη σχεδίου του project είναι ένα σημαντικότατο βήμα για την αναγνώριση του κινδύνου. Πρέπει να κατανοηθεί καλά η σημαντικότητα του **Κρίσιμου Μονοπατιού (Critical Path)** μέσα στο σχέδιο, που δεν είναι άλλο από το συντομότερο χρόνο που χρειάζεται κάποιος για την ολοκλήρωση της εργασίας και τη φύση της διαλειτουργικότητας των επιμέρους εργασιών.

Το επόμενο βήμα είναι η διαπίστωση του κύριου αιτίου, το οποίο προκαλεί τους κινδύνους που ανιχνεύονται. Αν η λίστα των κινδύνων περιέχει κινδύνους όπως «Το project ενδέχεται να ξεμείνει από χρήματα» ή «Το project μπορεί να υπερβεί χρονικά τα διαθέσιμα όρια», τότε το σχέδιο πρέπει να επαναπροσδιοριστεί, καθώς κανένα από αυτά δεν είναι κίνδυνος. Στην πραγματικότητα είναι η συνδυαστική των συνεπειών, κάποιων άλλων παραγόντων που δεν πήγαν κατ' ευχήν. Παρόμοια, αν ένας κίνδυνος είναι της μορφής «Το προσωπικό δεν είναι έμπειρο», στην πραγματικότητα περιγράφεται μια κατάσταση, η οποία προϋπάρχει και ίσως να δημιουργήσει ακολούθως ποικιλόμορφες παρενέργειες, εκτός αν ληφθούν μέτρα (αντίμετρα), όπως ειδική εκπαίδευση και ανασχεδιασμό θέσεων και ατόμων.

Το πρώτο βήμα για την αντιμετώπιση των κινδύνων γίνεται με την συμπλήρωση ενός ερωτηματολόγιου που αποτελεί την πλέον συνηθισμένη μέθοδο εύρεσης λύσεων, αυτή τη στιγμή, στην αγορά. Για την συμπλήρωσή του, όμως, χρειάζεται προσπάθεια και ανάλυση, καθώς πρέπει να συνταχθεί με τέτοιο τρόπο, ώστε να εκμαιεύει τις σημαντικές πληροφορίες καθώς και να συμπληρωθεί σωστά και εμπεριστατωμένα, ώστε να αρχίσει σταδιακά να μετατρέπεται το ακατέργαστο πρωταρχικό πλάνο σε ολοκληρωμένο σύστημα αντιμετώπισης κρίσεων. Γενικά τα ερωτηματολόγια για να αποδώσουν, θα πρέπει να είναι κλειστά και διχοτομικά και είναι δυνατή η εφαρμογή της Ιεραρχικής Ανάλυσης.

Το σύστημα έχει ένα αρχικό επίπεδο κινδύνου, πριν εφαρμοστούν οποιαδήποτε αντίμετρα. Τα αντίμετρα αυτά, υποθέτοντας ότι οι τιμές τους καθορίζονται από τις ίδιες παραμέτρους που χρησιμοποιούν οι απειλές, οι ευπάθειες και η εκτίμηση των πόρων, μπορούν να μειώσουν τον κίνδυνο (κλειδωμένες πόρτες, firewalls) περιορίζοντας τις ευπάθειες (επαγρύπνηση, patches, hotfixes) ή ελαχιστοποιώντας την αξία των πόρων (κρυπτογράφηση). Μετά την καταμέτρηση των αποτελεσμάτων από κάθε συνδυασμό απειλής, ευπάθειας, πόρου και αντίμετρου, ο εναπομένων κίνδυνος έχει καθοριστεί.

Σήμερα, οι πιο πρόσφατες μεθοδολογίες ανάλυσης κινδύνου ξεκινούν με την αναγνώριση και την κοστολόγηση των πόρων, ακολουθούμενη από την αναγνώριση των απειλών, οι οποίες είναι πιθανές να συμβούν με τις σχετιζόμενες ευπάθειες. Ο τελικός κίνδυνος καθορίζεται από το συνδυασμό των αναγνωρισμένων πόρων, απειλών και ευπαθειών. Ο συνδυασμός αυτός αποτελεί το δείκτη εκείνο που απαιτείται για να προταθούν τα καλύτερα αντίμετρα. Κατά τη διάρκεια αυτής της διαδικασίας, δυο διαφορετικά σχέδια μέτρησης μπορούν να εφαρμοστούν για τα στοιχεία του κινδύνου. Το πρώτο περιλαμβάνει την ποσοτική προσέγγιση του κινδύνου, ενώ το δεύτερο στηρίζεται στην ποιοτική αποτίμηση του, που αντιμετωπίζει το σύστημα στο εχθρικό περιβάλλον λειτουργίας. Η **ποσοτική προσέγγιση** αντιλαμβάνεται τον κίνδυνο με αριθμητικούς όρους, όπως η **αναμενόμενη απώλεια χρημάτων και η πιθανότητα να συμβεί αυτό (Ετήσια Προσδοκώμενη Απώλεια, ALE)**. **Η ποιοτική προσέγγιση** δεν έχει αριθμητικές τιμές και είναι συχνά βασισμένη σε γνώμες. Τα αποτελέσματα συνοψίζονται σε έννοιες όπως **χαμηλή, μέτρια και υψηλή**. Τα πλεονεκτήματα και τα μειονεκτήματα και των δύο προσεγγίσεων είναι πολλά ανάλογα με το είδος του συστήματος που μελετάται και το περιβάλλον λειτουργίας μέσα στο οποίο δρά, ωστόσο, σε κάθε περίπτωση υπάρχει απόσταση μεταξύ της αρχικής προσέγγισης και της τελικής κατάστασης μετά την κρίση.

2.5.1 ΠΟΙΟΤΙΚΗ ΑΝΑΛΥΣΗ ΚΙΝΔΥΝΟΥ

Έχοντας αναγνωρίσει ένα εύρος από κινδύνους είναι αναγκαία η μελέτη των πιο σοβαρών από αυτούς, με σκοπό να εξακριβωθεί που πρέπει να επικεντρωθεί η προσοχή σχετικά με τους πόρους - που απαιτούνται - για την δημιουργία του έργου. Οι άνθρωποι, σε γενικές γραμμές δεν έχουν πάντα τη δυνατότητα να αναλύσουν τον κίνδυνο με απόλυτη επιτυχία. Συχνά, τείνουν να παίρνουν αποφάσεις που επηρεάζονται από τη συναισθηματική τους διάθεση σε μια κατάσταση κρίσεως, παρά από την αντικειμενική θέση του σχετικού κίνδυνου. Πολλές φορές, μάλιστα, επικεντρώνουν την προσοχή τους σε έναν κίνδυνο που συνέβη πρόσφατα, αδιαφορώντας για επικίνδυνα γεγονότα που έχουν συμβεί στο παρελθόν με την ίδια συχνότητα επανεμφάνισης, όπως ο προαναφερθείς υπό μελέτη κίνδυνος. Αντίστοιχα και οι περισσότεροι αναλυτές πιστεύουν ό,τι θέλουν να πιστέψουν – επιλεκτικά, ενώ παράλληλα φιλτράρουν τις πληροφορίες, οι οποίες δεν υποστηρίζουν τη δική τους άποψη. Επίσης, η πλειοψηφία τους εμφανίζει σημαντικό έλλειμμα κριτικής γνώσης που έχει σαν αποτέλεσμα να υπάρχει

ανεπάρκεια ικανότητας στο να μελετούν την πιθανότητα με μια ολιστική οπτική.

Μια τεχνική, που κρατά όσο μπορεί πιο περιορισμένη την υποκειμενικότητα αυτή, είναι η τεχνική Delphi. Χρησιμοποιώντας αυτή την τεχνική, οι απόψεις συλλέγονται συμπληρωματικά και ανώνυμα και μετά διασταυρώνονται από ειδικούς. Οι ειδικοί απλά μελετούν τα καταχωρημένα δεδομένα, που παρουσιάζονται, αδιαφορώντας για τις προσωπικότητες που εμπλέκονται σε διάφορα επίπεδα λειτουργίας. Για να αποφασιστεί πόσο σημαντικός είναι ένας κίνδυνος, μελετώνται δύο κύριες παράμετροι:

- **Η πιθανότητα,**
- **Το αποτέλεσμα.**

Ως πιθανότητα ορίζεται ένας κίνδυνος που συμβαίνει σε συγκεκριμένο ορίζοντα χρονοπρογραμματισμού μέσα στο υπό μελέτη σύστημα, ενώ, ως αποτέλεσμα οι συνέπειες, που εμφανίζονται στο σύστημα, στην περίπτωση που ο κίνδυνος υλοποιηθεί.

Από την άλλη μεριά, η επίδραση των συνεπειών μπορεί να αναλυθεί ως προς τις παρακάτω παραμέτρους:

- **Του Χρόνου,**
- **Του Κόστους,**
- **Της Ποιότητας,**
- **Της Εγγύτητας του κινδύνου.**

Η κλίμακα για τη μέτρηση της πιθανότητας και της επίδρασης μπορεί να είναι αριθμητική ή ποιοτική, αλλά και οι δύο περιπτώσεις πρέπει να έχουν κατανοηθεί επακριβώς. Πολύ συχνά χρησιμοποιείται η κλίμακα **Υψηλή, Μεσαία και Χαμηλή**. Αυτή η κλίμακα είναι ασαφής για τα περισσότερα projects. Αντίθετα, μια κλίμακα της μορφής 1-100 είναι υπερβολικά λεπτομερής. Μια προτεινόμενη κλίμακα - όπως φαίνεται στη διεθνή βιβλιογραφία - έχει ως εξής:

Κλίμακα	**Πιθανότητα**
Πολύ χαμηλή	Δύσκολο να συμβεί
Χαμηλή	Μπορεί να συμβεί περιστασιακά
Μέτρια	Είναι πιθανό να μη συμβεί
Υψηλή	Είναι πιθανό να συμβεί

Πολύ υψηλή	Είναι σχεδόν βέβαιη να συμβεί

Κλίμακα	**Επίδραση**
Πολύ χαμηλή	Μηδαμινή επίδραση
Χαμηλή	Αμελητέα επίδραση στο χρόνο, το κόστος ή την ποιότητα
Μέτρια	Σημειωτέα επίδραση στο χρόνο, το κόστος ή την ποιότητα
Υψηλή	Σημαντική επίδραση στο χρόνο, το κόστος ή την ποιότητα
Πολύ υψηλή	Απειλή για την επιτυχία του project

Άμεσο συμπέρασμα των παραπάνω - και όπως φαίνεται από τις περισσότερες μεθοδολογίες, οι οποίες εφαρμόζονται μέχρι σήμερα - είναι η προσπάθεια ποσοτικοποίησης της ποιοτικής αυτής ανάλυσης. Η ποσοτικοποίηση αυτή δίδεται παρακάτω:

- **Κόκκινοι κίνδυνοι,**
- **Πράσινοι κίνδυνοι.**

Οι κόκκινοι κίνδυνοι, σημασιολογικά, είναι τα **φαινόμενα εκείνα που προσευχόμαστε να μη συμβούν**. Είναι στην ουσία μαύρες οπές του συστήματος, οι οποίες οδηγούν σε δυσμενείς καταστάσεις. Είναι κίνδυνοι που πρέπει να αντιμετωπιστούν άμεσα, προκειμένου να αποφευχθούν οδυνηρά καταστροφικά φαινόμενα. Αυτό απαιτεί διαρκή και επίπονη προσπάθεια, καθώς και επένδυση σε χρόνο και χρήμα.

Εντούτοις, με τον όρο πράσινοι κίνδυνοι εννοούμε καταστάσεις κρίσεως, **οι οποίες δεν είναι οδυνηρές για την ομαλή λειτουργία του συστήματος.** Πρόκειται απλώς για σπάνιες καταστάσεις κρίσεως που δεν μπορούν όμως να αγνοηθούν. Οφείλει, λοιπόν, ο διαχειριστής του συστήματος να καλύψει τα ευάλωτα κενά με την αναγκαία δαπάνη χρημάτων που θα δοθούν για δημιουργία αντίμετρων.

2.5.2 ΠΟΣΟΤΙΚΗ ΑΝΑΛΥΣΗ ΚΙΝΔΥΝΟΥ

Η προηγούμενη ποιοτική ανάλυση του κινδύνου είναι η πιο κατανοητή και απλή για όλους τους υπεύθυνους του project. Αν θέλουμε να ποσοτικοποιήσουμε τους όρους υψηλό, μεσαίο, χαμηλό χρειάζεται να γίνουν σαφείς προσδιορισμοί και να κατανοηθεί τι είναι το υψηλό κόστος. Παράλληλα, πρέπει να αναλυθεί επαρκώς ποια είναι η επίπτωση του χρόνου για το συγκεκριμένο σύστημα, πριν να οριστούν οι κίνδυνοι

με σαφήνεια. Ακολουθεί ένας πίνακας της μορφής αυτής, ο οποίος αποσαφηνίζει τις ασάφειες πάνω στους όρους:

Επίδραση	Κόστος	Χρόνος	Ποιότητα
Πολύ χαμηλή	Μικρές διακυμάνσεις στον προϋπολογισμό που απορροφούνται από άλλα επιμέρους έξοδα	Ελάχιστες αστοχίες σε σχέση με τον προγραμματισμό	Ελάχιστη μείωση χωρίς ολική επίδραση στη χρησιμότητα
Χαμηλή	Απαιτεί μικρή επιπλέον χρηματοδότηση από την επιχείρηση	Ελάχιστη αστοχία στα σημεία «κλειδιά» ή στους δημοσιευμένους χρόνους	Αδυναμία να συμπεριληφθούν κάποια στοιχεία
Μέτρια	Απαιτεί σημαντική επιπλέον χρηματοδότηση	Επιδρά αρνητικά στην εμπιστοσύνη του συστήματος	Σημαντικά στοιχεία λειτουργικότητας δεν θα είναι διαθέσιμα
Υψηλή	Απαιτεί σημαντικό επαναπροσδιορισμό του οικονομικού μοντέλου για την επίτευξη των στόχων	Αδυναμία να επιτευχθούν τα deadlines που ορίζει το σχέδιο δράσης	Αδυναμία να ικανοποιηθούν σημαντικές λειτουργίες
Πολύ υψηλή	Αυξάνει την απειλή της βιωσιμότητας του συστήματος	Η καθυστέρηση διακινδυνεύει την βιωσιμότητα του συστήματος	Το αποτέλεσμα του συστήματος είναι μη αξιοποιήσιμο

2.5.3 ΛΗΨΗ ΑΠΟΦΑΣΕΩΝ (Decision Making)

Λαμβάνοντας υπόψη το περιβάλλον της επιχείρησης και των διαθέσιμων πόρων, οι ειδικοί που ασχολούνται με τη λήψη αποφάσεων σε μια επιχείρηση μπορούν να υλοποιήσουν μια ή περισσότερες στρατηγικές διαχείρισης κινδύνου.

Υπάρχει ένα μεγάλο εύρος απειλών και ευπαθειών καθώς και πολλά διαφορετικά επιχειρηματικά περιβάλλοντα και λύσεις που επιτάσσουν την ανάγκη για τη σταθεροποίηση των ασταθών

χαρακτηριστικών των σύγχρονων περιβαλλόντων λειτουργίας. Δυστυχώς, στην πλειοψηφία των περιπτώσεων αυτό δεν είναι εφικτό, αφού υπάρχουν σημαντικές αντικειμενικές δυσκολίες.

Η διαχείριση του κινδύνου και η αντιμετώπιση κρίσεων έρχεται να καλύψει αυτή την αδυναμία των σύγχρονων υπολογιστικών συστημάτων. Ο διαχειριστής του συστήματος καλείται, εκτός από τον σχεδιασμό και τη λειτουργία του, να αναπτύξει και ένα σύνολο άλλων ειδικών ικανοτήτων, προκείμενου να μπορέσει να αντεπεξέλθει στις απαιτήσεις των σύγχρονων δυναμικά μεταβαλλόμενων πλαισίων λειτουργίας. Απαραίτητη προϋπόθεση για την ομαλή λειτουργία και συντήρηση των ψηφιακών συστημάτων είναι η ικανότητα του διαχειριστή να προβλέπει, να αντιμετωπίζει, να αποφεύγει καταστροφικά φαινόμενα, ενώ παράλληλα οφείλει να μπορεί να ανακτήσει κατεστραμμένα - πολύτιμα τις περισσότερες φορές - υποσυστήματα.

Αυτό κάνει την υλοποίηση της ανάλυσης κινδύνου και τις μεθοδολογίες μέτρησης πολύ δύσκολο να εφαρμοστούν άρτια και το αποτέλεσμά τους να εξαρτάται από την εμπειρία των ατόμων που εμπλέκονται στη διαδικασία αυτή, κάτι το οποίο δημιουργεί ασυνέπεια και μερικές φορές μη ικανοποιητικά αποτελέσματα. Επιπροσθέτως, απαιτείται να συλλεχθεί μεγάλος όγκος πληροφοριών και να γίνει ένα πλήθος – που στην περίπτωση της ποσοτικής μεθοδολογίας είναι αρκετά περίπλοκο – υπολογισμών. Για να προσδιοριστούν αυτά τα προβλήματα έχουν αναπτυχθεί διάφορα αυτόματα εργαλεία που υπόσχονται αυξημένη παραγωγικότητα μέσω της ελαχιστοποίησης της εργασίας και του χρόνου της ανάλυσης, καθώς και μέσω της κανονικοποίησης των διαφορών μεταξύ των εμπειριών του Προσωπικού. Υπάρχει ένα μεγάλο πλήθος πακέτων διαχείρισης κινδύνου, παρόλο που αυτό το πλήθος στην πλειοψηφία του είναι πεπαλαιωμένο και μη μοντελοποιημένο.

2.6 ΛΟΓΙΣΜΙΚΟ ΑΝΑΛΥΣΗΣ, ΔΙΑΧΕΙΡΙΣΗΣ ΚΙΝΔΥΝΩΝ

2.6.1 Η ΜΕΘΟΔΟΛΟΓΙΑ @RISK

Μεθοδολογία: Ποσοτική

Το @RISK είναι ένα πρόγραμμα που προστίθεται στα 123/Symphony/Excel, τα οποία είναι προορισμένα για ανάλυση του κινδύνου, το οποίο χρησιμοποιεί την εξομοίωση Monte Carlo. Οι κατανομές της πιθανότητας προσθέτονται στα κελιά, χρησιμοποιώντας

30 νέες κατανομές πιθανότητας οι οποίες είναι ενσωματωμένες στις συναρτήσεις. Κάνοντας χρήση μενού της γνωστής μορφής του Lotus ή του Excel, επιτρέπουν στους χρήστες να επιλέξουν δειγματοληψία της μορφής Monte Carlo ή Latin Hypercube, να διαλέξουν τα εξαγόμενα διαστήματα τιμών καθώς και να εκτελέσουν προσομοίωση. Τα αποτελέσματα παρουσιάζονται γραφικά και στατιστικά και ακολούθως υπολογίζονται και εμφανίζονται σε μορφή αναφοράς.

2.6.2 Η ΜΕΘΟΔΟΛΟΓΙΑ ALRAM (Automated Livermore Risk Analysis Methodology)

Μεθοδολογία: Ποσοτική

Αποτελώντας ένα σύστημα που αναπτύχθηκε αποκλειστικά για κυβερνητικές εφαρμογές, η μεθοδολογία αυτή είναι σχεδιασμένη να επιτρέπει το διαχωρισμό των συνδυασμών πόρων/απειλών, έτσι ώστε μόνο **οι υψηλής επιρροής γνωστοί κίνδυνοι να εξετάζονται.** Η μεθοδολογία αυτή επικεντρώνει την προσοχή της στην αποτελεσματικότητα των συνιστώμενων ελέγχων ασφάλειας καθώς και των ήδη υπαρχόντων. Το ALRAM χωρίζεται σε τρεις βασικές φάσεις που περιλαμβάνουν το σχεδιασμό του έργου, την ανάλυση του κινδύνου αυτού και βέβαια τη λήψη των αποφάσεων για αυτούς τους διαπιστωμένους κινδύνους. Η αρχική φάση καθορίζει τον σκοπό της ανάλυσης, τους απαραίτητους πόρους και το Προσωπικό που απαιτείται. Η δεύτερη φάση συλλέγει και αναλύει τα δεδομένα που συλλέχθηκαν στην πρώτη φάση. Σε αυτή τη δεύτερη φάση, τα στοιχεία του κινδύνου αναγνωρίζονται μέσω της διαπίστωσης των ανάλογων απειλών και τα αποτελέσματα παρέχονται σαν είσοδος για την τελική φάση της λήψης των αποφάσεων. Η τελική φάση αυτή παρουσιάζει τις εκτιμήσεις του κόστους για κάθε προτεινόμενη λύση μαζί με το πλάνο για την επιλογή και την ιεράρχηση των απειλών.

2.6.3 Η ΜΕΘΟΔΟΛΟΓΙΑ ARES (Automated Risk Evaluation System) Version 1.1

Μεθοδολογία: Ποσοτική

Το ARES χρησιμοποιεί μια μηχανή αποτελεσμάτων η οποία είναι βασισμένη σε κανόνες και καθοδηγούμενη από μενού και λίστα επιλογών για την περάτωση της ανάλυσης του κινδύνου. Δεν πρόκειται για ένα σύστημα βασισμένο σε αριθμούς, καθώς το ARES συλλέγει δεδομένα

από το χρήστη (τοποθεσία, τηλεφωνικό αριθμό, διεύθυνση κτλ) με οδηγίες συμπλήρωσης άδειων φορμών. Τα στοιχεία αυτά παρουσιάζουν το σύστημα ασφαλείας του υπολογιστή και του λειτουργικού του συστήματος, μαζί με ένα μεγάλο εύρος άλλων πληροφοριών, που συλλέγονται από τη λίστα των επιλογών. Τα στοιχεία που έχουν συλλεχθεί περιλαμβάνουν τα **υψηλότερα βαθμολογημένα δεδομένα** στο σύστημα, το επίπεδο πιστοποίησης του χρήστη, το επίπεδο εμπιστοσύνης του συστήματος και διάφορα άλλα πρακτικά θέματα, όπως τη διαχείριση των κωδικών και άλλα. Όταν παράγονται οι τελικές ή οι ενδιάμεσες αναφορές, το ARES συγκρίνει τις λίστες που έχουν συλλεχθεί με τους βασικούς κανόνες. Η συμπερασματική αναφορά αποτελείται από εξώφυλλο, επιστολές έγκρισης και απαρίθμηση των πιθανών κινδύνων στον υπεύθυνο ασφαλείας, δίνοντας του τη δυνατότητα επιλογής ή διόρθωσης καθενός από τους πιθανούς κινδύνους ως μέρος της διαδικασίας επικύρωσης. Η αναφορά γράφεται σε ένα ASCII αρχείο, επιτρέποντας στον υπεύθυνο να ελέγξει επιμελώς την αποτελεσματικότητα του λογισμικού στο συγκεκριμένο εργασιακό περιβάλλον.

2.6.4 Η ΜΕΘΟΔΟΛΟΓΙΑ BDSS (Bayesian Decision Support System)

Μεθοδολογία: Ποσοτική/Ποιοτική

Το BDSS είναι προγραμματισμένο να συλλέγει στοιχεία αξιολόγησης και να συντάσσει ερωτήσεις, οι οποίες βοηθούν στην επίλυση των πιθανών κινδύνων, χρησιμοποιώντας ποσοτικές βάσεις δεδομένων που παρέχονται από τον προμηθευτή. Πιο συγκεκριμένα, ο χρήστης μπορεί να συμπεριλάβει απειλές που είναι ειδικές ως προς τις ανάγκες του, και να συλλέγει τις εμπειρίες του με την βοήθεια ειδικών αλγόριθμων που θα επεξεργαστούν τα δεδομένα μαζί με την ποσοτική τους βάση γνώσης. Με αυτόν τον τρόπο, συλλέγονται αυτές οι απειλές, τα ευάλωτα σημεία και το μοντέλο διαλέγει ασφαλείς τρόπους. Επιπλέον, το σύστημα αυτό αξιολογεί και βαθμολογεί τις απειλές, πριν και μετά την υλοποίηση της προτεινόμενης λύσης, έτσι ώστε να είναι δυνατή η παρουσίαση του προβλήματος και να γίνεται δυνατή η αποφυγή της επανάληψης τους. Τα αποτελέσματα της ανάλυσης παρουσιάζονται τυπικά σε γραφικό περιβάλλον με καμπύλες κινδύνου βασισμένες στο κόστος της απώλειας και στην πιθανότητα της επανεμφάνισης. Οι κεντρικοί αλγόριθμοι του BDSS είναι βασισμένοι στο θεώρημα του Bayes που διευθύνουν την αβεβαιότητα και τις στατιστικές μεθόδους. Το λογισμικό του BDSS παράγει μια ποικιλία τυπωμένων αναφορών καθώς

και αρχεία ASCII που μπορούν να εξάγονται από το χρήστη στο διαθέσιμο χώρο του επεξεργαστή κειμένου. Υπάρχει ευελιξία στο πως το BDSS θα χρησιμοποιείται. Για παράδειγμα, η ανάλυση της ευπάθειας που παρέχει η εφαρμογή του BDSS, διαθέτει μια ποσοτική παρουσίαση των αδυναμιών του συστήματος προστασίας.

Ας επιχειρήσουμε να συγκρίνουμε την κλασική ανάλυση κινδύνου με τα αποτελέσματα του BDSS με ένα σενάριο. Έστω ότι η εταιρία Α έχει ένα κέντρο δεδομένων με ιδιαίτερο ενδιαφέρον για δύο συγκεκριμένα θέματα – τη φωτιά και τη λανθασμένη χρήση των πόρων. Βασισμένο σε αυτές τις δύο ανησυχίες, ο αναλυτής του κινδύνου συνέταξε την αναφορά διαχείρισης κινδύνου και κατά την αναζήτηση των δεδομένων του κατέληξε στα εξής:

- Οι πόροι της εταιρίας Α αξίζουν περίπου 1 εκατομμύριο δολάρια.
- Το περιβάλλον επεξεργασίας πληροφοριών είναι ένα κτίριο με ξύλινο περίβλημα χωρίς αντιπυρική προστασία.
- Η φύλαξη του χώρου είναι χαλαρή. Τα καύσιμα κρατούνται για λίγο χρονικό διάστημα, πριν απομακρυνθούν και το κάπνισμα επιτρέπεται.
- Η στατιστική φωτιάς που υπάρχει διαθέσιμη δείχνει ότι με τέτοιες συνθήκες συμβαίνει μια φωτιά κάθε 10 χρόνια.
- Μια σημαντική φωτιά θα προκαλέσει απώλειες των 500.000 δολαρίων σε ανασφάλιστους πόρους.
- Εκατό χρήστες υπολογιστών και προσωπικό συστημάτων χρησιμοποιούν τον υπολογιστή για προσωπικούς λόγους με μέσο όρο μια φορά το μήνα και μέσο κόστος 50 δολάρια.

Με αυτά τα δεδομένα συντάσσεται ο κλασσικός αλγόριθμος ανάλυσης κινδύνου όπου:

EF = Παράγων έκθεσης
SLE = Απλή Έκθεση Απώλειας
ARO = Ετήσιος Ρυθμός Συμβάντων
ALE = Ετήσια Εκτιμώμενη Απώλεια

ΚΛΑΣΙΚΟΣ ΑΛΓΟΡΙΘΜΟΣ

Απειλή = Φωτιά
ΚόστοςΠόρων X EF = SLE x ARO = ALE
$1.000.000 x 0.5 = $500.000 x 0.1 = $50.000

Απειλή = Λανθασμένη χρήση πόρων
ΚόστοςΠόρων X EF = SLE x ARO = ALE
$1.000.000 x 0.00005 = $50 x 1.200 = $60.000

Παρατηρούμε ότι η ALE που παράγεται από τον κλασσικό αλγόριθμο είναι $50.000 και $60.000, αντίστοιχα, με συνέπεια να παρουσιάζει την **απειλή της φωτιάς λιγότερο σημαντική**! Η κλασική ανάλυση είναι ανίκανη να διαχωρίσει την ALE για ένα καταστροφικό γεγονός.

Ακολούθως εξετάζεται η συμπεριφορά του BDSS στο ίδιο παράδειγμα.

ΑΛΓΟΡΙΘΜΟΣ BDSS

Απειλή = Φωτιά		**Απειλή = Λανθασμένη χρήση πόρων**	
	Εμπιστοσύνη		Εμπιστοσύνη
Εύρος Αξίας Πόρων	**Παράγοντας**	**Εύρος Αξίας Πόρων**	**Παράγοντας**
$950.000 - $1.050.000	90%	$950.000 - $1.050.000	90%
Παράγων Έκθεσης		**Παράγων Έκθεσης**	
Κατανομή		*Κατανομή*	
0,4 – 0,6	80%	0,00004 – 0,00006	80%
ARO Κατανομή		*ARO Κατανομή*	
0,05 – 0,15	80%	1,100 – 1,300	80%
ALE		*ALE*	
$46.033		**$42.746**	

Συμπεραίνουμε ότι οι ποσοτικοί παράγοντες που χρησιμοποιούνται για τον εντοπισμό του κινδύνου είναι πιο έξυπνοι. Εδώ βλέπουμε ότι η καταστροφή που θα **επιφέρει η φωτιά είναι λίγο σημαντικότερη** από αυτήν της κακής χρήσης των πόρων.

2.6.5 Η ΜΕΘΟΔΟΛΟΓΙΑ BUDDY SYSTEM

Μεθοδολογία: Ποιοτική

Το Buddy System είναι μια αυτόματη μεθοδολογία ανάλυσης κινδύνου για περιβάλλοντα μικροϋπολογιστών και καλύπτει δύο στάδια:

1) **Την επισκόπηση στη λήψη αντίμετρων.**
2) **Την ανάλυση της ασφάλειας και διαχείριση της (SAM).**

Αυτό το πακέτο λογισμικού μελετά το επίπεδο της ευπάθειας και είναι βασισμένο στο ήδη εγκατεστημένο σύστημα προστασίας. Ανάλογα με το επίπεδο των πληροφοριών που επεξεργάζονται στο σύστημα, διαπιστώνεται αν το επιθυμητό επίπεδο ευπάθειας είναι αποδεκτό. Συνιστώμενες ενέργειες για τη διόρθωση της κάθε ευπάθειας που ξεφεύγει από το επιθυμητό εύρος, παρέχονται μέσω της χρήσης των δυναμικών "what if" σεναρίων. Μια βάση δεδομένων, που περιέχει πάνω από 100 συστήματα προστασίας, περιέχεται στο πακέτο λογισμικού. Περαιτέρω, η Διαχείριση του Κινδύνου από το σύστημα επιτρέπει στον αναλυτή να παρακολουθεί τις προτεινόμενες διορθωτικές ενέργειες μέσω αναφορών.

2.6.6 Η ΜΕΘΟΔΟΛΟΓΙΑ Control Matrix (CONTMAT)

Μεθοδολογία: Προσεγγιστική πίνακα

Αυτή η μεθοδολογία επιτρέπει την αξιολόγηση των ελέγχων της εφαρμογής, τους αντικειμενικούς σκοπούς της καθώς και τους κινδύνους, χρησιμοποιώντας προσέγγιση πίνακα. Ο πίνακας παρέχει μια περίληψη της ασφάλειας της εφαρμογής και του περιβάλλοντος ελέγχου. Αυτό επιτρέπει στο χρήστη και στην ομάδα ασφαλείας να αντιλαμβάνονται γρήγορα που χρειάζονται περαιτέρω συστήματα προστασίας. Μια βάση δεδομένων από ελεγκτικές τεχνικές, που μπορούν να υλοποιηθούν για την περιφρούρηση υψηλού κινδύνου περιοχών, συμπεριλαμβάνεται στο πακέτο.

2.6.7 Η ΜΕΘΟΔΟΛΟΓΙΑ CONTROL-IT

Μεθοδολογία: Ποιοτική

Πρόκειται για έλεγχο με προσέγγιση φύλλου εργασίας. Αυτό το λογισμικό παρέχει μια προσέγγιση ελέγχου με χρήση φύλλων εργασίας για το σχεδιασμό των ελέγχων μέσα σε μικροεπεξεργαστικά περιβάλλοντα. Αναγνωρίζει ποιοι έλεγχοι είναι απαραίτητοι για τη διασφάλιση των επαρκών ασφαλειών της επιχείρησης ή των επιστημονικών συστημάτων. Το πακέτο λογισμικού περιέχει τέσσερα ξεχωριστά συστήματα:

- Το 1ο πακέτο (Σχεδιασμός Ελέγχων σε Υπολογιστικά Συστήματα) είναι ένα εκπαιδευτικό εργαλείο που διδάσκει τον χρήστη πως να σχεδιάζει και να αναπτύσσει έναν πίνακα ελέγχου.
- Το 2ο πακέτο (Αξιολόγηση των Κινδύνων του Πίνακα) διδάσκει τη χρήση των Delphi και τη Σύγκριση Κινδύνων για την αξιολόγηση των απειλών και των ελέγχων τους.
- Το 3ο πακέτο (Αυτοματοποιημένος Σχεδιασμός Πίνακα Ελέγχου από Υπολογιστή) είναι ένα πακέτο υλοποίησης πινάκων ελέγχου που περιέχει μια βάση δεδομένων από ελέγχους καθώς και μια ξεχωριστή βάση από απειλές και εξαρτήματα υπολογιστή. Αυτό το πακέτο επιτρέπει σε κάποιον να σχεδιάσει έναν πρόχειρο πίνακα ελέγχου, να αναζητήσει τη βάση δεδομένων από ελέγχους και να τους μεταφέρει σε μια λίστα από πίνακες ελέγχου.
- Το 4ο πακέτο (Προβολή Κειμένου και Παρουσίαση Γραφικών) χρησιμοποιείται για το σχεδιασμό του τελικού πίνακα από τις ακολουθίες απειλών, εξαρτημάτων και ελέγχων.

2.6.8 Η ΜΕΘΟΔΟΛΟΓΙΑ CORA

Μεθοδολογία: Ποσοτική

Η βοήθεια που προσφέρει το CORA διατίθεται σε δύο διαδικασίες. Πρώτα, το CORA παρέχει ένα περιβάλλον για να οργανώσει, να συλλέξει, να αποθηκεύσει και να πιστοποιήσει δεδομένα που περιγράφουν τον κίνδυνο και τις εκθέσεις σε απώλειες μιας επιχείρησης. Το CORA χρησιμοποιεί σκεπτόμενους αλγόριθμους για να κατασκευάσει ένα Ποσοτικό Μοντέλο Κινδύνου. Πιο συγκεκριμένα υπολογίζει το **Κόστος Απλού Συμβάντος (SOL)** και τα αποτελέσματα τα προβάλλει γραφικά. Ακολούθως, οι χρήστες του μπορούν να πειραματιστούν με το ποσοτικό μοντέλο της επιχείρησης για να εκτελέσουν **Εκτιμήσεις Διαχείρισης Κινδύνου** και να διαπιστωθεί το πακέτο των βέλτιστων λύσεων. Αυτό το πακέτο λογισμικού αντικαθιστά επάξια τα IST/RAMP και το CRITI-CALC. Το CORA έχει μοναδικά χαρακτηριστικά:

- Μπορεί να εισάγει βασικά στοιχεία από το βασικό πληροφοριακό σύστημα της επιχείρησης, οπότε μπορεί να συλλέγει δεδομένα που είναι έγκυρα.
- Οι ειδικοί μπορούν να καθορίσουν παράγοντες κινδύνου και να τους αποθηκεύσουν ως αρχεία “risk rules”. Έτσι, έχουν τη δυνατότητα να προσαρμόσουν πλήρως τις ιδιότητες της διαχείρισης.
- Ο Financial Simulator ενισχύει τα οικονομικά αποτελέσματα.

- Έχει βάση δεδομένων που επιτρέπει τη βέλτιστη ανάκτηση δεδομένων των IT Systems.

2.6.9 Η ΜΕΘΟΔΟΛΟΓΙΑ CRAMM (CCTA Risk Analysis and Management Methodology)

Μεθοδολογία: Ποιοτική

Η CRAMM είναι ένα τυπικό σύστημα ανάλυσης και διαχείρισης κινδύνου το οποίο είναι ανεπτυγμένο από την Βρετανική Κυβέρνηση και την BIS Applied Systems Limited. Η CRAMM αποτελείται από τρία στάδια το καθένα από τα οποία υποστηρίζεται από ερωτηματολόγια και οδηγίες καθοδήγησης.

Το 1^{ο} στάδιο εκτελεί μια αξιολόγηση των βοηθημάτων του συστήματος ή του δικτύου. Από αυτή τη διαδικασία καθορίζονται οι ποιοτικές τιμές για τα βοηθητικά δεδομένα σε κλίμακα από το 1 έως το 10, ταξινομημένα σύμφωνα με τις πιθανές παρενέργειες των αλλαγών, της έλλειψης διαθεσιμότητας και της καταστροφής. Τα φυσικά βοηθήματα κοστολογούνται και με κριτήριο την αντικατάσταση ή το κόστος της επιδιόρθωσης και μετατρέπονται σε κλίματα από 1 έως 10. Οπουδήποτε η τιμή των βοηθημάτων είναι χαμηλή (3 και κάτω) το σύστημα τίθεται υπό αξιολόγηση. Αυτό είναι το πρώτο στάδιο βασικής προστασίας και τα αποτελέσματα αυτά προωθούνται στο επόμενο στάδιο.

Το 2^{ο} στάδιο εξακριβώνει τις απειλές και τα ελαττώματα της κάθε ομάδας βοηθημάτων και βαθμολογεί το ζευγάρι αυτό σε κλίμακα 1 έως 5, όπου το 5 αντανακλά το χειρότερο σενάριο.

Το 3^{ο} στάδιο αφορά την επιλογή των συστημάτων ασφαλείας, που αναφέρεται σε μια βιβλιοθήκη με περισσότερες από 900 εφαρμογές. Ακολούθως, η διαχείριση στοχεύει στη λήψη απόφασης για το πιο κατάλληλο σύστημα ασφαλείας και η CRAMM παρέχει λειτουργίες που προορίζονται για τη διερεύνηση αυτών των επιλογών. Ένα εύρος από αναφορές διαχείρισης είναι διαθέσιμες.

Το λογισμικό CRAMM παρέχει, επίσης, ένα σύστημα κωδικού ασφαλείας για τη μείωση των κινδύνων από μη εξιουσιοδοτημένη πρόσβαση στα δεδομένα που αναλύονται. Ενδείξεις που υποδηλώνουν την ευαισθησία των πληροφοριών παρέχονται σε όλες τις οθόνες καθώς και στο τυπωμένο αντίτυπο αυτών.

2.6.10 Η ΜΕΘΟΔΟΛΟΓΙΑ CRITI-CALC

Μεθοδολογία: Ποσοτική/Ποιοτική

Αυτό το προϊόν χρησιμοποιεί τη μέθοδο της Ετήσιας Προσδοκώμενης Απώλειας (ALE) για τον προσδιορισμό της ποσότητας και της κρισιμότητας του κινδύνου, κάτω από τις οποίες είναι εκτεθειμένες οι εφαρμογές. Το λογισμικό αυτό συλλέγει πληροφορίες για κάθε πιθανού είδους αστοχία της εφαρμογής, για το κόστος της λήψης αντιγράφων ασφαλείας καθώς και το κόστος της ανάκτησης αυτών. Ακολούθως, χρησιμοποιεί αυτές τις πληροφορίες για τον υπολογισμό της Ετήσιας Προσδοκώμενης Απώλειας για κάθε εφαρμογή. Η κρισιμότητα κάθε εφαρμογής καθορίζεται από την πιθανότητα της απώλειας δεδομένων, η οποία μπορεί να προέρχεται είτε από διακοπή της επεξεργασίας είτε από μια συλλογή 14 πιθανών παραγόντων καθυστέρησης. Ο χρήστης αλληλεπιδρά με το σύστημα, χρησιμοποιώντας οθόνες οι οποίες προβάλλουν πληροφορίες για την τρέχουσα έκθεση, που βρίσκεται σε κίνδυνο. Όταν ο χρήστης έχει λάβει γνώση για τα αρχικά δεδομένα, μια ανάλυση του τύπου "what if" εκτελείται, μεταβάλλοντας τα υπάρχοντα δεδομένα ως μια μέθοδο πιστοποίησης της αποτελεσματικότητας των συστημάτων ασφαλείας. Οι πληροφορίες που περιέχονται στις αναφορές που παράγονται μπορούν να χρησιμοποιηθούν για τη βελτιστοποίηση των σχεδίων αντιμετώπισης απρόοπτων συμβάντων. Η ALE, σαν συνάρτηση με παράγοντα τη μέγιστη διάρκεια διακοπής λειτουργίας, είναι συγκρίσιμο μέγεθος με το κόστος της λήψης αντιγράφων ασφαλείας και με την αυτόματη βέλτιστη ανάκτηση των δεδομένων.

2.6.11 Η ΜΕΘΟΔΟΛΟΓΙΑ GRA/SYS

Μεθοδολογία: Ποιοτική

Το GRA/SYS είναι ένα εργαλείο σχεδιασμένο για την προσφορά βοήθειας στους εσωτερικούς ελεγκτές και στο προσωπικό ασφαλείας, ώστε να αναπτύσσουν ένα σχέδιο καθορισμού προτεραιοτήτων για την εποπτεία των κινδύνων της επιχείρησης. Πιο συγκεκριμένα, το λογισμικό προετοιμάζει την εφαρμογή και τη δραστηριότητα των υπολογιστών, καθώς επίσης καθορίζει τον αριθμό των κινδύνων για διάφορες περιοχές ελέγχου ύψιστης σημασίας. Θέτει βαθμολογίες κινδύνου, οι οποίες αντικατοπτρίζουν τον βαθμό επικινδυνότητας στην επιχείρηση, όπου υπολογίζεται και δίνεται προτεραιότητα κατά φθίνουσα σειρά σε κλίμακα

από 1 έως 9, με το 9 να αναπαριστά τη χειρότερη περίπτωση. Μια, επιπλέον, αναφορά που αντικατοπτρίζει το πλήθος των φορών που κάθε κίνδυνος συμβαίνει ή πρόκειται να συμβεί προετοιμάζεται. Χρησιμοποιώντας τις παραγόμενες αναφορές από αυτό το πακέτο λογισμικού, ο χρήστης είναι ικανός να αναγνωρίζει τους κινδύνους στους οποίους απαιτούνται πιο αποτελεσματικά συστήματα ασφαλείας.

2.6.12 Η ΜΕΘΟΔΟΛΟΓΙΑ IST/RAMP (International Security Technology/Risk Analysis Management Program)

Μεθοδολογία: Ποσοτική

Το IST/RAMP είναι ένα λογισμικό που παραμένει στον κεντρικό υπολογιστή και αναλαμβάνει την ανάλυση κινδύνου με μια αυτοτελή μονάδα εισόδου η οποία εγκαθίσταται στο PC. Το λογισμικό υπολογίζει την Ετήσια Προσδοκώμενη Απώλεια καθώς και την απώλεια ενός και μοναδικού συμβάντος. Το σύστημα μπορεί να παρέχει, επίσης, και ποιοτική ανάλυση. Το IST/RAMP παράγει φόρμες συλλογής δεδομένων για να βοηθήσει την ανάλυση του κινδύνου, τη βέλτιστη οργάνωση και τον έλεγχο της συλλογής των δεδομένων. Πέντε κατηγορίες απωλειών καθορίζονται:

- **διακοπή υπηρεσιών,**
- **φυσική απώλεια και ζημιά,**
- **απάτη,**
- **μη εξουσιοδοτημένη αποκάλυψη απορρήτων**
- **φυσική κλοπή.**

Μια βιβλιοθήκη από βάσεις δεδομένων επιτρέπουν στον αναλυτή να συντηρεί τα ίχνη των αλλαγών στα δεδομένα εισόδου. Η δυνατότητα για χρήση του "what if" επιτρέπει στον αναλυτή να διαλέγει τα πιο συμφέροντα, από άποψη κόστους, αντίμετρα ασφαλείας.

Το RAMP↔LINK είναι ένα καθοδηγούμενο από μενού σύστημα εισαγωγής δεδομένων το οποίο χρησιμοποιεί τις πληροφορίες κινδύνου, που εισάγονται από τον αναλυτή, για τη δημιουργία ενός αρχείου, το οποίο μπορεί να μεταφορτωθεί στο IST/RAMP για επεξεργασία.

2.6.13 Η ΜΕΘΟΔΟΛΟΓΙΑ JANBER

Μεθοδολογία: Ποιοτική

Το Janber ενεργοποιεί ένα ερωτηματολόγιο ναι/όχι και μια λίστα επιλογών για τη συλλογή πληροφοριών του ήδη υπάρχοντος συστήματος ασφαλείας. Το λογισμικό ζυγίζει τα τοποθετημένα συστήματα ασφαλείας και τα μετρά σε σχέση με την ταξινόμηση προτεραιότητας των δεδομένων που επεξεργάζονται στο σύστημα. Αυτά τα ταξινομημένα επίπεδα δεδομένων ξεκινούν από υψίστης ευαισθησίας αταξινόμητα μέχρι ιδιαίτερα ταξινομημένα δεδομένα. Η ανάλυση παρέχει ένα χαρακτηρισμό του επιπέδου της ευπάθειας από 2-28, με το 28 να παριστάνει το χειρότερο σενάριο. Οι ευπάθειες, τα συστήματα ασφαλείας και τα βάρη τους μπορούν να προεγκατασταθούν από τον προμηθευτή, σχεδιασμένα, έτσι ώστε να ικανοποιούν τις απαιτήσεις της επιχείρησης. Τα συστήματα ασφαλείας, που απαιτούνται, αλλά δεν υλοποιούνται, σημαδεύονται στην αναφορά και στη συνέχεια παρέχονται προτεινόμενες οδηγίες και συμβουλές σύμφωνα με το καταστατικό της επιχείρησης. Έτσι λοιπόν, οι χρήστες έχουν τη δυνατότητα να εκτελούν σενάρια "what if" για την εκτίμηση της αποτελεσματικότητας συγκεκριμένων συστημάτων ασφαλείας.

Επιπλέον, η εφαρμογή του Janber επιτρέπει στους χρήστες να καθορίζουν πρότυπες τιμές για συγκεκριμένα πεδία δεδομένων. Τα αποτελέσματα από τη συλλογή των δεδομένων και της ανάλυσής τους συντηρούνται σε ξεχωριστή βάση δεδομένων. Ο προμηθευτής συστήνει η ανάλυση και η συλλογή των δεδομένων να γίνεται από διαφορετικό προσωπικό και αυτό για να διασφαλιστεί ότι η ανάλυση, που εκτελείται από τον υπεύθυνο ασφάλειας των υπολογιστών, επιτυγχάνει τα βέλτιστα αποτελέσματα, καθώς το λογισμικό παρέχει τη δυνατότητα να ανιχνεύει τις ενέργειες που απορρέουν από την προηγούμενη εκτίμηση. Το Janber, λοιπόν, δημιουργεί μια βάση δεδομένων από πληροφορίες σε όλα τα συστήματα που επιθεωρήθηκαν και ταυτόχρονα, παρέχει μια βάση δεδομένων από ερωτήσεις για το σχεδιασμό των απρόοπτων συμβάντων και τις λειτουργίες ανάκτησης αυτών.

2.6.14 Η ΜΕΘΟΔΟΛΟΓΙΑ LAVA (Los Alamos Vulnerability and Risk Assessment)

Μεθοδολογία: Ποσοτική/Ποιοτική

Το LAVA επιμελείται ερωτηματολόγια τα οποία έχουν ως αποτέλεσμα την αναγνώριση των ελλειπών συστημάτων ασφαλείας, καλύπτοντας τομείς από τη διαχείριση των κωδικών ασφαλείας μέχρι την ασφάλεια του προσωπικού και τις πρακτικές του εσωτερικού ελέγχου. Το λογισμικό αξιολογεί τις πιθανές παρενέργειες και τις συνέπειες στην επιχείρηση και παράγει την τελική έκθεση απωλειών. Το LAVA λαμβάνει υπόψη του τρία είδη απειλών:

- **τους φυσικούς και περιβαλλοντικούς κινδύνους,**
- **τις τυχαίες και εσκεμμένες ανθρώπινες απειλές,**
- **τις απομακρυσμένες ανθρώπινες απειλές.**

Το LAVA παρέχει λεπτομερείς αναφορές ποιοτικών και ποσοτικών αποτελεσμάτων των κινδύνων που αναγνωρίζονται.

2.6.15 Η ΜΕΘΟΔΟΛΟΓΙΑ MARION

Μεθοδολογία: Ποσοτική/Ποιοτική

Το MARION εκτιμά τους κινδύνους της επιχείρησης που σχετίζονται με τα πληροφοριακά συστήματα, απορροφώντας γνώση από μια μεγάλη βάση δεδομένων πραγματικών περιστατικών. Το λογισμικό ενσωματώνει ένα ερωτηματολόγιο για την εκτίμηση του επιπέδου ασφαλείας, που εφαρμόζεται ήδη μέσα στην επιχείρηση. Κάθε ερώτηση έχει προσδιοριστεί με ένα δείκτη βάρους ο οποίος αντανακλά τη σχετική σημαντικότητα, σύμφωνα με την ανάλυση της υποβοηθητικής βάσης δεδομένων με τα συμβάντα. Ένας βαθμός προσδιορίζεται για κάθε ερώτηση, με το αποτέλεσμα και το βαθμό να αποθηκεύεται. Το λογισμικό υπολογίζει τον τελικό βαθμό για κάθε μια από τις 27 κατηγορίες ασφαλείας και παρουσιάζει τα αποτελέσματα γραφικά και σε τυπωμένη μορφή. Όταν το προφίλ ασφαλείας καθοριστεί, το λογισμικό συγκρίνει κάθε κατηγορία με τις κανονικές τιμές οι οποίες προέρχονται από τη βάση δεδομένων. Το λογισμικό χρησιμοποιεί τις πληροφορίες για το κόστος, το οποίο επίσης κρατείται στη βάση δεδομένων για τον υπολογισμό της εκτιμώμενης δαπάνης σε σχέση με το γενικό προϋπολογισμό ασφαλείας. Τα υπολογισμένα έξοδα αναλύονται,

σύμφωνα με τις προδιαγραφές ασφάλειας της επιχείρησης και παρουσιάζονται γραφικά σε λεπτομερείς πίνακες. Η δυνατότητα του για σενάρια "what if" επιτρέπει σε κάποιον να χρησιμοποιήσει διαφορετικούς προϋπολογισμούς για να διαπιστώσει τις επιπτώσεις στο προφίλ ασφαλείας. Οι επιπτώσεις των προτεινόμενων μέτρων μπορούν, επίσης, να αναπαρασταθούν.

2.6.16 Η ΜΕΘΟΔΟΛΟΓΙΑ MicroSecure Self Assessment

Μεθοδολογία: Ποιοτική

Πρόκειται για ένα αυτοματοποιημένο εργαλείο το οποίο επιτρέπει στους χρήστες του υπολογιστή να πραγματοποιούν μια αυτοαξιολόγηση ασφαλείας. Το λογισμικό αναλύει το περιβάλλον του υπολογιστή, καθορίζει τις ευπάθειες και συστήνει ελέγχους ασφαλείας. Τα προτεινόμενα συστήματα ασφαλείας σχεδιάζονται για να αυξήσουν την ασφάλεια και να μειώσουν την έκθεση σε κίνδυνο σε έξι τομείς που περιλαμβάνουν:

- **την ακεραιότητα του συστήματος,**
- **την ασφάλεια των δεδομένων,**
- **την αξιοπιστία, την ακεραιότητα των δεδομένων,**
- **τη λήψη αντιγράφων ασφαλείας,**
- **την ανάκτηση λόγω καταστροφής,**
- **την εμπιστευτικότητα και την ιδιωτικότητα**.

Το λογισμικό μπορεί να παραμετροποιηθεί για να πληροί τις ιδιαίτερες ανάγκες της επιχείρησης.

2.6.17 Η ΜΕΘΟΔΟΛΟΓΙΑ MINIRISK

Μεθοδολογία: Ποιοτική

Το MINIRISK είναι ένα εργαλείο σχεδιασμένο για να βοηθά την αντιμετώπιση των ευπαθειών της ασφάλειας υπολογιστών σε ένα μικροεπεξεργαστικό περιβάλλον. Ένα ερωτηματολόγιο επιτρέπει στην επιχείρηση να εκτιμά την επάρκεια και την πληρότητα των ανεξάρτητων συστημάτων ασφαλείας και να επανεκτιμά τους τομείς εκείνους στους οποίους νέα συστήματα ασφαλείας έχουν υλοποιηθεί. Κατά τη διάρκεια της επεξεργασίας των απαντήσεων του ερωτηματολογίου, ο χρήστης

αναγνωρίζει τα ελλιπή συστήματα ασφαλείας ανάμεσα σε 10 με 50 κατηγορίες ευπαθειών που κυμαίνονται από τη διαχείριση των κωδικών ασφαλείας μέχρι το σχεδιασμό αντιμετώπισης απρόοπτων και τον εσωτερικό έλεγχο. Συστήματα ασφαλείας και έλεγχοι που θεωρούνται απαραίτητοι από την επιχείρηση καθορίζονται σε κάθε κατηγορία για ανασκόπηση. Η απουσία των ζωτικών συστημάτων ασφαλείας μεταβάλλει το επίπεδο της ευπάθειας σε μια κλίμακα από το 0 μέχρι το 9, με το 0 να αντιστοιχεί στην καλύτερη περίπτωση και το 9 στη χειρότερη. Το MINIRISK εφαρμόζει κατώτατο όριο με το οποίο αξιολογεί τις ευπάθειες που το υπερβαίνουν και συνεπώς ξεπερνούν τα αποδεκτά επίπεδα κινδύνου.

2.6.18 Η ΜΕΘΟΔΟΛΟΓΙΑ PRISM Risk Analysis and Simulation for the PC

Μεθοδολογία: Ποσοτική

Το PRISM υποστηρίζει την ανάπτυξη της μοντελοποιημένης ανάλυσης κινδύνου, κάνοντας χρήση της εξομοίωσης, της ανάλυση της ευαισθησίας καθώς και της γραφικής αναπαράστασης των αποτελεσμάτων. Περιέχει, επίσης, σύστημα λειτουργιών για την αποθήκευση, την ανάκτηση, την προβολή και την μετατροπή των υπαρχόντων μοντέλων. Εκτός από τις απλές αλγεβρικές συναρτήσεις, το PRISM επιτρέπει τη χρήση γραμματικής παρόμοιας με την BASIC για τη μοντελοποίηση πιο σύνθετων εφαρμογών.

2.6.19 Η ΜΕΘΟΔΟΛΟΓΙΑ RA/SYS (Risk Analysis System)

Μεθοδολογία: Ποσοτική

Το RA/SYS είναι ένα αυτοματοποιημένο σύστημα ανάλυσης κινδύνου που λειτουργεί με μια σειρά από διασυνδεδεμένα αρχεία, τα οποία μπορούν να βοηθήσουν σε πάνω από 50 ευπάθειες και 65 απειλές. Οι υπολογισμοί πραγματοποιούνται σε ζευγάρια απειλών-ευπαθειών για να παράγουν βαθμολογίες και συχνότητες απειλών. Μια αναφορά συνοψίζει τις εκτιμήσεις απωλειών, την ανάλυση του οικονομικού κέρδους και της επιστροφής του κόστους στον επενδυτή.

2.6.20 Η ΜΕΘΟΔΟΛΟΓΙΑ RANK-IT

Μεθοδολογία: Ποσοτική

Το RANK-IT είναι ένα πακέτο λογισμικού που προορίζεται για την εκτίμηση κινδύνου που χρησιμοποιεί τη μέθοδο της Delphi. Η Delphi είναι ένα σύστημα εξειδικευμένης προσέγγισης για τη βαθμολόγηση των κινδύνων. Το λογισμικό αυτό αυτοματοποιεί τη **μέθοδο Delphi,** προσθέτοντας τη Συγκριτική Βαθμολόγηση Κινδύνων για την εξασφάλιση μιας ενοποιημένης λίστας βαθμολογημένων κινδύνων ή για τον υπολογισμό του ποσοστού των τιμών κινδύνου. Κάθε βαθμολογημένο αντικείμενο έχει μια αριθμητική τιμή που μπορεί να χρησιμοποιηθεί σαν παράγοντας βάρους ή σαν απόλυτα αριθμητική τιμή.

Το RANK-IT χρησιμοποιείται για τη βαθμολόγηση των απειλών του συστήματος, των ελέγχων, των ευπαθειών, των εξαρτημάτων ή σε οποιοδήποτε άλλον τομέα. Επίσης, μπορεί να χρησιμοποιηθεί για τη βαθμολόγηση εναλλακτικών επιχειρηματικών αποφάσεων άλλων τύπων είτε ποσοτικοποιημένων είτε όχι.

Ο προμηθευτής αναφέρει ότι ο χρόνος που απαιτείται για την παραγωγή μιας βαθμολόγησης κινδύνου, χρησιμοποιώντας αυτό τον συνδυασμό της Delphi και της Συγκριτικής Βαθμολόγησης Κινδύνου, κυμαίνεται μεταξύ 30 λεπτών και 3 ωρών. Ωστόσο το πρόβλημα δεν είναι βέβαια ο χρόνος που απαιτείται, αλλά η ορθότητα της βαθμολόγησης του κινδύνου.

2.6.21 Η ΜΕΘΟΔΟΛΟΓΙΑ RiskCALC

Μεθοδολογία: Ποσοτική

Σκοπός του RiskCALC είναι να υπολογίζει την Ετήσια Προσδοκώμενη Απώλεια ή παράλληλα με την χρήση ερωτηματολόγιων - ανάλογα με το είδος των απαντήσεων – χρησιμοποιεί άλλη μέθοδο μέτρησης. Ο χρήστης μπορεί προαιρετικά να αλλάξει τις τιμές των μεταβλητών του RiskCALC για να προσδιορίσει τα πιο εποικοδομητικά - από άποψη κόστους - συστήματα ασφαλείας και να προβάλλει τα αποτελέσματα στην οθόνη του χρήστη. Το RiskCALC είναι ένα μέρος από την οικογένεια των λογισμικών εργαλείων που παρουσιάζονται παρακάτω. Καθένα από αυτά παρέχει ένα τυπικό ASCII αρχείο για την εισαγωγή και την εξαγωγή των μεταβλητών του RiskCALC.

Πρωταρχικά, το RiskCALC επιτρέπει στο χρήστη να απαντάει σε ερωτήσεις και να τυπώνει αναφορές, οι τιμές των οποίων αποσπώνται από το ερωτηματολόγιο και εισάγονται αυτόματα.

Δευτερευόντως, το Risk Minimizer αναγνωρίζει τους πιο σημαντικούς κινδύνους της επιχείρησης μέσα από την ολοκληρωμένη ανάλυση. Το Risk Minimizer μπορεί να χρησιμοποιηθεί με άλλα εργαλεία διαχείρισης κινδύνου, τα οποία χρησιμοποιούν το είδος αρχείων του RiskCALC.

Τρίτον, το System Manager βοηθά στο σχεδιασμό ή την προσαρμογή των ήδη υπαρχόντων μοντέλων ανάλυσης κινδύνου, αλλά με ισχυρές παραδοχές (assumptions).

Τέταρτον, το Demonstration Models επιτρέπει στο χρήστη να αναπτύξει ένα προσαρμοσμένο ερωτηματολόγιο ή να επιλέγει ένα που μοντελοποιεί διάφορα σενάρια κινδύνου.

2.6.22 Η ΜΕΘΟΔΟΛΟΓΙΑ RISKPAC

Μεθοδολογία: Ποσοτική/Ποιοτική

Το RiskPAC είναι ένα σύστημα που χρησιμοποιεί ερωτηματολόγιο για την αλληλεπίδραση με το χρήστη και τη μέτρηση του κινδύνου σε κυβερνητικές εφαρμογές και άλλους τομείς. Οι απαντήσεις του χρήστη στο ερωτηματολόγιο αποθηκεύονται σε ξεχωριστά αρχεία, που καλούνται δημοσκοπήσεις. Διαφορετικές δημοσκοπήσεις συγκρίνονται για να διαπιστωθούν τα αποτελέσματα των διορθωτικών μέτρων ή για να εκτελεστούν αναλύσεις “what if”. Οι ερωτήσεις στο ερωτηματολόγιο ομαδοποιούνται σε κατηγορίες, παρόμοιες με ένα βιβλίο διαιρεμένο σε κεφάλαια. Κάθε κατηγορία βαθμολογείται, παρέχοντας μια λεπτομερή και λογική ανάλυση του υποκειμένου. Η αναφορά που παράγει το RiskPAC παρέχει το βαθμό του κινδύνου σε κάθε κατηγορία. Βασισμένο, λοιπόν, σε αυτό το βαθμό για την κάθε κατηγορία, το RiskPAC παρέχει συνιστώμενες ενέργειες για τη διόρθωση των ευπαθειών (καθώς μια βάση δεδομένων από διορθωτικές ενέργειες περιέχεται σε καθένα από τα ερωτηματολόγια).

Το RiskPAC, επίσης, περιέχει ένα module ανάλυσης, τον Υπολογιστή A.L.E., για την ανάλυση της Ετήσιας Εκτιμώμενης Απώλειας. Πολλαπλά A.L.E. spreadsheets μπορούν να δημιουργηθούν.

Μια λίστα από τις περιγραφές των απειλών αποθηκεύονται σε ξεχωριστά αρχεία, τα οποία μπορούν να φορτωθούν μέσα σε spreadsheets, μειώνοντας τη διαδικασία εισαγωγής δεδομένων και διευκολύνοντας την ανάλυση "what if". Αναδιπλούμενες λίστες στο spreadsheet περιέχουν τα προτερήματα, τις απειλές, την επιρροή του δολαρίου και τη συχνότητα των συμβάντων. Οι τιμές του A.L.E. υπολογίζονται καθόσον ο χρήστης δουλεύει.

Το RiskPAC System Manager, το οποίο διατίθεται ξεχωριστά, χρησιμοποιείται για τη δημιουργία ή τη διαφοροποίηση των ερωτηματολογίων. Το RiskPAC System Manager επιτρέπει στο χρήστη να εισάγει ένα σετ ερωτήσεων, απαντήσεων και διορθωτικών ενεργειών και να τα μετατρέψει σε ένα εξαιρετικό σύστημα εκτίμησης κινδύνου. Προφανώς, δεν λειτουργεί για απροσδόκητες καταστάσεις, τις οποίες ο χρήστης δεν είχε καν υπ' όψιν του.

2.6.23 Η ΜΕΘΟΔΟΛΟΓΙΑ RISKWATCH

Μεθοδολογία: Ποσοτική/Ποιοτική

Το RISKWATCH είναι ένα εργαλείο διαχείρισης της ασφάλειας, που αποτελείται από εφτά modules.

- Το 1ο module είναι ένα εργαλείο ανάλυσης κινδύνου, το οποίο καλείται να εκτελέσει μια τυπική ανάλυση κινδύνου στα Κέντρα Αποφάσεων, στις εφαρμογές, στα δίκτυα ή σε απομακρυσμένες περιοχές.
- Το 2ο module υποστηρίζει τον τρέχοντα σχεδιασμό διαχείρισης κινδύνου.
- Το 3ο module αναπτύσσει ένα σχέδιο ασφάλειας.
- Το 4ο module αναπτύσσει σχέδια απρόοπτων περιστατικών.
- Το 5ο module διευθύνει ένα Τεστ Ασφαλείας και Αξιολόγησης των επιλεγμένων συστημάτων ασφαλείας.
- Το 6ο module είναι ένα πρόγραμμα γραφικών.
- Το 7ο module είναι ένα εργαλείο Εξειδικευμένης Ανάπτυξης Συστημάτων.

Το RISKWATCH περιέχει ένα εργαλείο ανάπτυξης ερωτηματολογίων, το οποίο επιτρέπει να εισάγονται οι ερωτήσεις και να ομαδοποιούνται. Τα modules μπορούν να λειτουργούν και ανεξάρτητα.

Επιπλέον, το RISKWATCH έχει μια ενσωματωμένη ευφυής βάση για την εξασφάλιση της εσωτερικής ασφάλειας. Έχει τη δυνατότητα να παράγει αναφορές, συμπεριλαμβανομένης και της αναζήτησης κειμένου, καθώς και γραφικό πρόγραμμα για την ερμηνεία των αποτελεσμάτων της ανάλυσης σε διαγράμματα ράβδων ή πίττας. Κανένα άλλο λογισμικό δεν απαιτείται.

Στη συνέχεια, το RISKWATCH είναι σχεδιασμένο να παρέχει όλα τα προαπαιτούμενα των Ομοσπονδιακών πρακτορείων σε ότι αφορά την ανάλυση του κινδύνου και διαπιστώνει αυτόματα την υλοποίηση των συστημάτων ασφαλείας, με δείκτη Επιστροφής στην Επένδυση για κάθε σύστημα.

Τέλος, το RISKWATCH μπορεί να δημιουργήσει ερωτηματολόγια σε βοηθητικές μνήμες για εύκολη διανομή, ή να συλλέξει πληροφορίες μέσω δικτυακών ρυθμίσεων. Μια αποτίμηση των ευπαθειών δημιουργείται αυτόματα για κάθε απομακρυσμένη τοποθεσία. Ο έλεγχος του ιστορικού των ενεργειών συντηρείται σε όλα τα στάδια και τις διαδικασίες του προγράμματος. Νέες εκδόσεις του είναι διαθέσιμες κάθε χρόνο.

2.6.24 Η ΜΕΘΟΔΟΛΟΓΙΑ SOS (Security On-Line System)

Μεθοδολογία: Ποσοτική/Ποιοτική

Το SOS είναι ένα εργαλείο το οποίο έχει σχεδιαστεί για τη διαχείριση της ασφάλειας και του κινδύνου ενός συστήματος. Ο χρήστης ξεκινά καθορίζοντας την ταυτότητα του συστήματός του στην γνωσιακή βάση δεδομένων. Χρησιμοποιώντας τη βάση δεδομένων αυτή, γίνεται μια βασική αποτίμηση των κινδύνων γίνεται για να καθοριστεί η έκθεση απώλειας δολαρίων ή η μη εξουσιοδοτημένη αλλαγή των δεδομένων του συστήματος. Ο χρήστης μπορεί, ακολούθως, να χρησιμοποιήσει ένα προκαθορισμένο ή σχεδιασμένο ερωτηματολόγιο για την προσωπική του αποτίμηση, την επισκόπηση της ασφάλειας των δεδομένων ή τον έλεγχο του συστήματος. Με αυτές τις πληροφορίες, ο χρήστης μπορεί να αναπτύξει μια βάση δεδομένων από απειλές, ευπάθειες και συστήματα ασφαλείας, η οποία μπορεί να χρησιμοποιηθεί ώστε να γραφτεί ένα σχέδιο απρόοπτων συμβάντων.

Η προσέγγιση αυτή του SOS επιτρέπει τη χαρτογράφηση των δεδομένων η εφαρμογή διαμένει και εντοπίζεται το επίπεδο του κινδύνου

για τον υπολογιστή, τις εφαρμογές, τα τοπικά δίκτυα, τις επικοινωνίες των δεδομένων, τα συστήματα βάσεων δεδομένων, τα κέντρα διαχείρισης δεδομένων, των λειτουργικών συστημάτων, των προϊόντων ασφαλείας, των συστημάτων που βρίσκονται υπό κατασκευή και του υλικού των υπολογιστών.

2.6.25 Η ΜΕΘΟΔΟΛΟΓΙΑ FMEA (Failure Mode Effect Analysis)

Μεθοδολογία: Ποιοτική

Το Failure Mode and Effect Analysis (FMEA) είναι ένα εργαλείο βελτίωσης ποιότητας, το οποίο σε αντίθεση με άλλα αντίστοιχα εργαλεία παράγει σημαντικά αποτελέσματα, χωρίς να απαιτεί πολύπλοκες στατιστικές αναλύσεις. Το FMEA είναι μια συστηματική μέθοδος αναγνώρισης και πρόληψης προβλημάτων σε προϊόντα και διαδικασίες πριν αυτά εμφανιστούν. Τα FMEA εστιάζουν στην πρόληψη σφαλμάτων, βελτίωση της ασφάλειας και αύξηση της ικανοποίησης των πελατών.

Σκοπός των FMEA είναι η πρόληψη προβλημάτων στις διαδικασίες και τα προϊόντα πριν εμφανιστούν. Με τη χρήση τους στις διαδικασίες σχεδίασης και κατασκευής, βοηθούν σημαντικά στη μείωση του κόστους αναγνωρίζοντας δυνατές βελτιώσεις στα αρχικά στάδια των προαναφερθέντων διαδικασιών, όταν οι αλλαγές είναι ακόμα εύκολες και φθηνές. Το αποτέλεσμα είναι η μείωση ή ακόμα και η εξάλειψη της ανάγκης λήψης μέτρων εκ των υστέρων που μπορούν να οδηγήσουν ακόμα και σε κρίσεις στα τελικά στάδια της διαδικασίας.

Στόχος μίας μελέτης FMEA είναι ο εντοπισμός όλων των τρόπων που ένα προϊόν ή μία διεργασία μπορούν να παρουσιάσουν πρόβλημα. Οι τρόποι αυτοί ονομάζονται Καταστάσεις Αποτυχίας και κάθε ένας από αυτούς έχει κάποια πιθανότητα να συμβεί, η οποία ονομάζεται σχετικό ρίσκο. Κάθε Κατάσταση Αποτυχίας έχει κάποιες ενδεχόμενες συνέπειες, με μερικές συνέπειες να είναι πιο πιθανές από κάποιες άλλες. Η αξιολόγηση των παραπάνω περιλαμβάνει τους εξής τρεις παράγοντες:

- **Σοβαρότητα (Severity):** Οι επιπτώσεις στην περίπτωση που παρουσιαστεί το πρόβλημα.
- **Εμφάνιση (Occurrence):** Η πιθανότητα ή συχνότητα εμφάνισης του προβλήματος.

- **Εντοπισμός (Detection):** Η πιθανότητα να εντοπιστεί το πρόβλημα πριν γίνει αντιληπτός ο αντίκτυπος του.

Καθένας από τους παραπάνω παράγοντες αποτιμάται σε μία κλίμακα από το 1 έως το 10 και στη συνέχεια πολλαπλασιάζονται οι τρεις αριθμοί. Το αποτέλεσμα που προκύπτει ονομάζεται Αριθμός Προτεραιότητας Ρίσκου (Risk Priority Number, RPN) και παίρνει τιμές μεταξύ του 1 και του 1000. Ο RPN χρησιμοποιείται για να κατατάξουμε σε σειρά προτεραιότητας τις διορθωτικές κινήσεις που απαιτείται να γίνουν ώστε να εξαλειφθούν ή να μειωθούν οι Καταστάσεις Αποτυχίας. Οι Καταστάσεις Αποτυχίας με το μεγαλύτερο RPN πρέπει να αντιμετωπιστούν πρώτα, αλλά πρέπει να δοθεί μεγάλη προσοχή και στις περιπτώσεις που η αποτίμηση του Severity είναι υψηλή (9 ή 10) ανεξάρτητα από τον RPN.

Μετά τις διορθωτικές κινήσεις οι Severity, Occurrence και Detection αποτιμώνται ξανά και έτσι προκύπτει ένας νέος RPN. Η διαδικασία αυτή επαναλαμβάνεται με συνεχείς βελτιώσεις και διορθωτικές κινήσεις μέχρι ο RPN να φτάσει σε αποδεκτές τιμές για όλες τις Καταστάσεις Αποτυχίας. Όλα τα FMEA προϊόντος και διαδικασίας ακολουθούν τα παρακάτω βήματα:

ΒΗΜΑ 1: Μελέτη προϊόντος/διαδικασίας.
ΒΗΜΑ 2: Brainstorming για πιθανά προβλήματα.
ΒΗΜΑ 3: Απαρίθμηση ενδεχόμενων συνεπειών για κάθε πρόβλημα.
ΒΗΜΑ 4: Αποτίμηση του παράγοντα Severity για κάθε συνέπεια.
ΒΗΜΑ 5: Αποτίμηση του παράγοντα Occurrence για κάθε πρόβλημα.
ΒΗΜΑ 6: Αποτίμηση του παράγοντα Detection για κάθε πρόβλημα ή/και συνέπεια.
ΒΗΜΑ 7: Υπολογισμός του Αριθμού Προτεραιότητας Ρίσκου (RPN) για κάθε συνέπεια.
ΒΗΜΑ 8: Απόδοση προτεραιότητας αντιμετώπισης κάθε προβλήματος.
ΒΗΜΑ 9: Λήψη μέτρων για τα προβλήματα υψηλής προτεραιότητας.
ΒΗΜΑ 10: Υπολογισμός των νέων Αριθμών Προτεραιότητας Ρίσκου (RPN).

2.6.25.1 ΠΛΕΟΝΕΚΤΗΜΑΤΑ FMEA

- Απλό στην προετοιμασία και την υλοποίηση.
- Δεν απαιτεί πολύ χρόνο για την ολοκλήρωση του.
- Δεν απαιτεί πολύπλοκες στατιστικές αναλύσεις.

- Οι ομάδες FMEA αποτελούνται από λίγα άτομα.
- Πληροί τις προϋποθέσεις του συστήματος QS-9000.

2.6.25.2 ΜΕΙΟΝΕΚΤΗΜΑΤΑ FMEA

- Θα μπορούσε να χαρακτηριστεί «απλοϊκό».
- Βασίζεται αποκλειστικά στον ανθρώπινο παράγοντα, οπότε μπορεί να υπάρξουν λανθασμένες εκτιμήσεις.
- Δεν υποστηρίζεται από κάποιο σοβαρό λογισμικό.
- Ο δείκτης RPN δεν είναι αποδεκτός σε όλους τους χώρους δεδομένων και συστημάτων εφαρμογών.

2.6.26 Η ΜΕΘΟΔΟΛΟΓΙΑ PRA(Preliminary Risk Analysis)

Μεθοδολογία: Ποιοτική

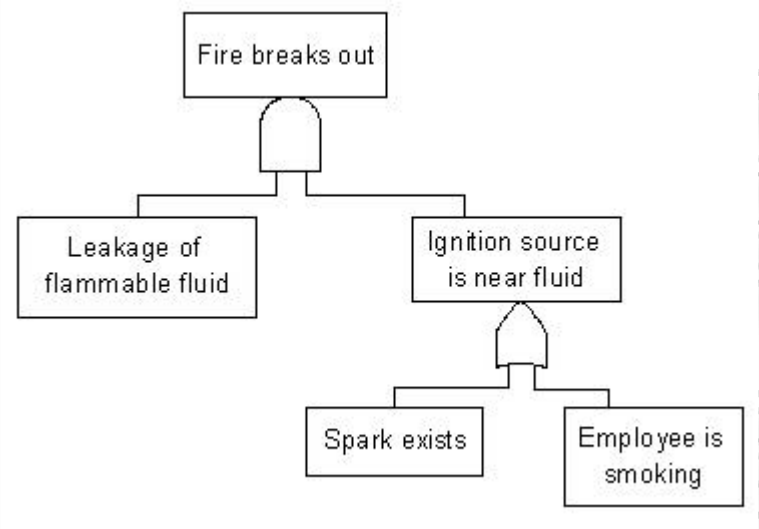
Η Preliminary Risk Analysis Preliminary risk analysis or hazard analysis είναι μια ποιοτική τεχνική η οποία περιλαμβάνει ανάλυση του γεγονότος που μπορεί να μετατραπεί σε ατύχημα.

Με αυτή την τεχνική οι πιθανές δυσμενείς καταστάσεις εντοπίζονται και στη συνέχεια αναλύονται. Για κάθε γεγονός προτείνονται βελτιώσεις και λύσεις. Το αποτέλεσμα αυτής της μεθοδολογίας είναι μια βάση για τον προσδιορισμό των κατηγόριων στους οποίους κατατάσσονται οι κίνδυνοι και τις μεθόδους αντιμετώπισης τους. Τα γεγονότα και οι κίνδυνοι κατηγοριοποιούνται έτσι ώστε να χρησιμοποιούνται τα κατάλληλα αντίμετρα ανάλογα με τη σημαντικότητα τους.

2.6.27 Η ΜΕΘΟΔΟΛΟΓΙΑ Fault tree analysis

Μεθοδολογία: Ποσοτική/Ποιοτική

Η μεθοδολογία fault tree είναι ένα λογικό διάγραμμα το οποίο δείχνει τη σχέση μεταξύ πτώσης συστήματος και πτώσης των στοιχείων του συστήματος. Είναι μια τεχνική που στηρίζεται σε αφαιρετική λογική. Ένα ανεπιθύμητο γεγονός πρώτα ορίζεται και προσδιορίζονται οι σχέσεις μεταξύ των πτώσεων.



Εικόνα 2.3
Παράδειγμα Ανάλυσης με τηνFault Tree Analysis

Η Fault tree μπορεί να χρησιμοποιηθεί τόσο ποιοτικά όσο και ποσοτικά. Η διάφορα μεταξύ τους είναι ότι η ποιοτική ανάλυση είναι πιο χαλαρή και δεν απαιτεί αυστηρή λογική όπως η ποσοτική ανάλυση.

2.6.28 Η ΜΕΘΟΔΟΛΟΓΙΑ Event tree analysis

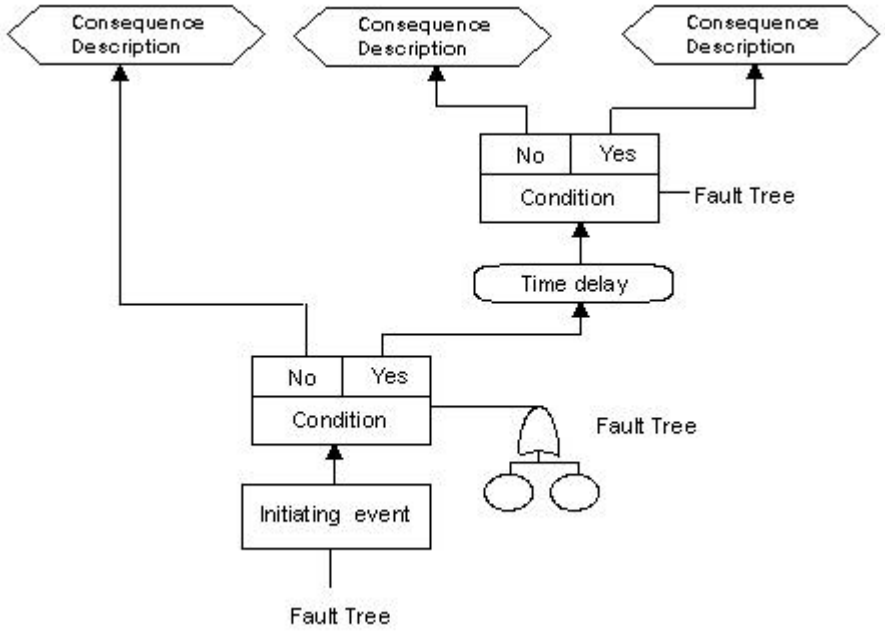
Μεθοδολογία: Ποσοτική/Ποιοτική

Η Event tree analysis είναι μια μεθοδολογία που παρουσιάζει τη συχνότητα των αποτελεσμάτων που εμφανίζονται μετά την υλοποίηση ενός επιλεγμένου αρχικού γεγονότος. Χρησιμοποιείται κυρίως για την ανάλυση επιπτώσεων προ-καταστροφικών φαινομένων και μετα-καταστροφικών καταστάσεων, ενώ χρησιμοποιείται επίσης στον προσδιορισμό κρίσεων σε πυρηνικά εργοστάσια.

2.6.29 Η ΜΕΘΟΔΟΛΟΓΙΑ Cause-Consequence Analysis

Μεθοδολογία: Ποσοτική/Ποιοτική

Η Cause-consequence analysis (CCA) είναι ένα μείγμα από fault tree και event tree analysis. Συνδυάζει ανάλυση αιτιών και ανάλυση αποτελεσμάτων με τη βοήθεια των δυο τεχνικών. Σκοπός της CCA είναι ο προσδιορισμός αλυσίδας γεγονότων που οδηγούν σε πτώση. Με τη βοήθεια πιθανοθεωρητικών μοντέλων καθορίζεται η επικινδυνότητα των γεγονότων και το ρίσκο του συστήματος όπως φαίνεται στο παρακάτω σχήμα. (Εικόνα 2.4)



Εικόνα 2.5
Παράδειγμα Cause-Consequence Analysis

2.6.30 Η ΜΕΘΟΔΟΛΟΓΙΑ Management Oversight Risk Tree

Μεθοδολογία: Ποσοτική/Ποιοτική

Η MORT είναι μια διαγραμματική μεθοδολογία που ορίζει ασφαλή προγραμματιστικά στοιχεία σε σωστή και λογική σειρά. Η ανάλυση της χρησιμοποιεί μέσα της fault tree, όπου το ανώτερο γεγονός είναι ένα από τα ακόλουθα συνήθως:

"Damage, destruction, other costs, lost production or reduced credibility of the enterprise in the eyes of society".

Η MORT έχει πάνω από 1500 πιθανά βασικά γεγονότα πάνω στα οποία στηρίζεται η ανάλυση της και αφορούν τομείς της πρόληψης ατυχημάτων, administration και management. Η MORT χρησιμοποιείται για ανάλυση ατυχημάτων και εκτίμηση προγραμμάτων ασφαλείας.

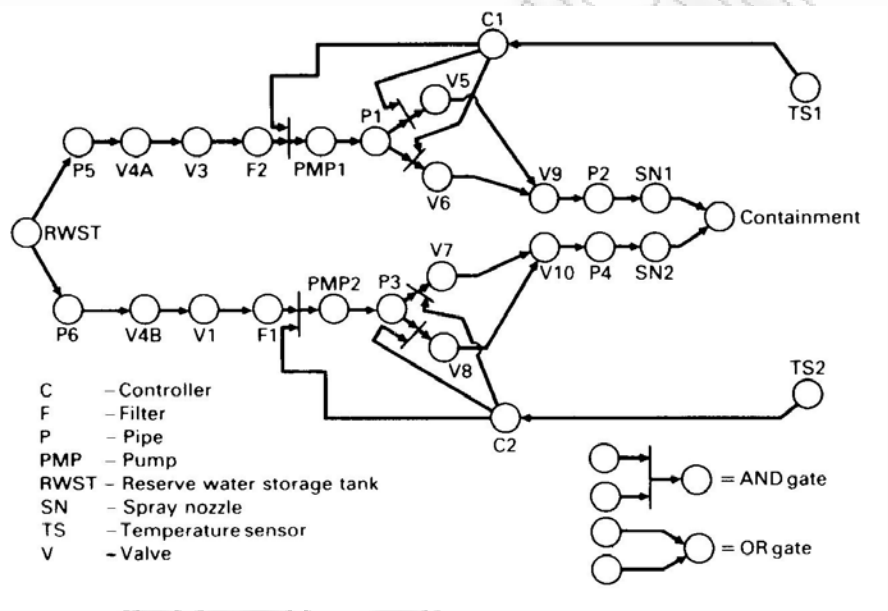
2.6.31 Η ΜΕΘΟΔΟΛΟΓΙΑ Digraph/Fault Graph

Μεθοδολογία: Ποσοτική/Ποιοτική

Η μεθοδολογία fault graph method/digraph matrix analysis χρησιμοποιεί μαθηματικά και γραφήματα στηριζόμενα στη θεωρία γραφημάτων όπως ειδικότερα το "path set" (ένα σύνολο από μοντέλα που

κινούνται σε ένα μονοπάτι) και τη "reachability" (το συνολικό σύνολο όλων των πιθανών μονοπατιών μεταξύ δυο κόμβων).

Η μέθοδος αυτή χρησιμοποιεί λογικές πύλες AND και OR. Ο πίνακας συνδεσιμότητας δείχνει αν ένας ελαττωματικός κόμβος θα οδηγήσει στην κορυφή του γραφήματος. Οι μετρήσεις δείχνουν αν απλοί κόμβοι ή ζευγάρια κόμβων οδηγούν σε πτώση του συστήματος. Η Digraph method επιτρέπει feed back loops τα οποία ενισχύουν τη δυναμικότητα του συστήματος.



Εικόνα 2.6
Σχήμα Γραφήματος Digraph/Fault Graph

2.6.32 Η ΜΕΘΟΔΟΛΟΓΙΑ Dynamic Event Tree Analysis Method

Μεθοδολογία: Ποσοτική/Ποιοτική

Η Dynamic event tree analysis method (DETAM) είναι μια μεθοδολογία η οποία χειρίζεται τη χρονική ανάπτυξη του υλικού του συστήματος, υπολογίζει μεταβλητές τιμές με τη βοήθεια ενός σεναρίου κρίσεως. Γενικά, ένα δυναμικό δέντρο είναι ένα δέντρο γεγονότων στο οποίο επιτρέπεται η διακλάδωση σε διαφορετικά χρονικά σημεία. Αυτή η προσέγγιση περιέχει 5 χαρακτηριστικά:

- Σύνολο διακλαδώσεων.
- Σύνολο μεταβλητών που καθορίζουν το σύστημα.
- Κανόνες διακλαδώσεων.
- Συχνότητα επέκτασης κανόνα.

- Ποσοτικά εργαλεία.

Το σύνολο των διακλαδώσεων αναφέρεται στο σύνολο των μεταβλητών που καθορίζουν το χώρο των πιθανών διακλαδώσεων για κάθε κόμβο του δέντρου. Κανόνες διακλάδωσης από την άλλη μεριά αναφέρονται σε κανόνες που καθορίζουν ποιο κλαδί πρέπει να λάβει μέρος. Η προσέγγιση αυτή μπορεί να χρησιμοποιηθεί για την αναπαράσταση ενός ευρύτερου πλήθους συμπεριφορών τελεστών, ενώ παράλληλα μοντελοποιεί τις συνέπειες της πράξης ενός τελεστή και λειτουργεί ως πλαίσιο λήψης αποφάσεων για το διαχειριστή του συστήματος. Τέλος επιτρέπει τη δοκιμή διαδικασιών κινδύνου για την εξαγωγή χρήσιμων συμπερασμάτων.

2.7 ΣΥΜΠΕΡΑΣΜΑΤΑ

Η αλλαγή της νοοτροπίας πρόληψης, για την καλύτερη αντιμετώπιση των κινδύνων, είναι δύσκολο να συντελεστεί μέσα σε μια νύχτα, ενώ απαιτεί κόστος. Όμως, όσο περισσότερο μελετώνται οι κίνδυνοι τόσο περισσότερο θα γίνονται αντιληπτές οι επιπτώσεις τους στον πραγματικό κόσμο όπως στα project, που αναλαμβάνονται.

Ο κίνδυνος δεν αντιμετωπίζεται απλά επειδή αγνοείται ή αποτυγχάνει το σχέδιο πρόληψης του. Αν υπάρχει η δυνατότητα δημιουργίας σχεδίων και προϋπολογισμών, που θα λαμβάνουν υπόψη τους κινδύνους, τότε οι επιχειρήσεις θα αναπτύξουν ένα καλύτερο κλίμα εμπιστοσύνης όσον αφορά το χειρισμό projects, ενώ θα ελέγχουν δυναμικότερα τα ευάλωτα σημεία τους.

Δεν πρέπει ποτέ να ξεχνά κανείς ότι οι κίνδυνοι μπορούν να μετατραπούν σε ευκαιρίες και η προληπτική προσέγγιση στη διαχείριση τους μπορεί να δημιουργήσει νέες ευκαιρίες για την επιχείρηση και τους διαχειριστές του συστήματος. Το απόσταγμα των παραπάνω συνοψίζεται στα ακόλουθα:

- Τα υπάρχοντα συστήματα διαχείρισης κινδύνων εστιάζονται στη μελέτη και βαθμολόγηση των υφισταμένων δομών ασφάλειας.
- Η πλειοψηφία των συστημάτων ασχολείται με την πρόληψη κινδύνων.
- Κατά κανόνα δεν ασχολούνται με σενάρια καταστροφής.
- Δεν ενδιαφέρονται για την ανάκτηση καταστραμμένων δεδομένων.

- Μέρος των συστημάτων ασφάλειας αντιμετωπίζει συγκεκριμένα είδη κινδύνων.
- Δεν αντιμετωπίζονται όλα τα πιθανά φαινόμενα κρίσεως που μπορούν να εμφανιστούν.
- Οι κίνδυνοι υπολογίζονται ότι υλοποιούνται με υποκειμενικά, πιθανοθεωρητικά κριτήρια.
- Η πλειοψηφία των συστημάτων δεν υπολογίζει τους κινδύνους που είναι απίθανο να συμβούν, θεωρώντας τους αμελητέους.
- Δεν υπολογίζεται το κόστος ενεργοποίησης αντίμετρων του συστήματος.
- Δεν υπολογίζεται το κόστος ανάκτησης.
- Δεν υπολογίζεται η αξία των καταστραμμένων υλικών.
- Δεν υπολογίζεται το κόστος συντήρησης ενός τέτοιου συστήματος.

Τα παραπάνω κενά έρχεται να καλύψει η Μεθοδολογία Διαχείρισης Κρίσεων, όπως θα αναλυθεί σε επόμενο κεφάλαιο της διατριβής, η οποία δίνει έμφαση σε απειλές – καταστροφές και κρίσης σε μη ομαλούς χώρους δεδομένων, όπου κυριαρχεί η Θεωρία Αβεβαιότητας, παρά η θεωρία πιθανοτήτων.

ΚΕΦΑΛΑΙΟ 3

ΜΕΘΟΔΟΛΟΓΙΑ ΠΡΟΛΗΨΗΣ, ΑΝΤΙΜΕΤΩΠΙΣΗΣ ΚΑΙ ΑΠΟΦΥΓΗΣ ΚΙΝΔΥΝΩΝ

3. ΠΡΟΛΗΨΗ - ΑΝΤΙΜΕΤΩΠΙΣΗ ΚΙΝΔΥΝΩΝ

3.1 ΕΙΣΑΓΩΓΗ

"Μια μονάχα μέρα φτάνει να ρίξει και να ανορθώσει όλα τα ανθρώπινα πράγματα."

Σοφοκλής (Αίας)

Δεν υπάρχει μηχανικός λογισμικού στη Γη που να μη νιώθει άσχημα μετά από μια πτώση Εξυπηρετητή (server failure). Ιδιαίτερα, σε εφαρμογές πραγματικού χρόνου σε μεγάλες βάσεις δεδομένων με κατανεμημένη δομή, όπως σε μεγάλους οργανισμούς εθνικής εμβέλειας ή και σε θέατρα στρατιωτικών εφαρμογών, οι δυσκολίες ανάκτησης, που προκύπτουν, μετά από την πτώση του Λογισμικού ή και του Υλικού είναι εντονότερες. Πρόσφατες μελέτες στην Αμερική (1998) δείχνουν ότι ένα 20% των δεδομένων δεν μπορεί να ανακτηθεί λόγω μόνιμης απώλειας με μεγάλο κόστος για τις εταιρείες. Στο σημείο αυτό, πρέπει να υπογραμμίσουμε ότι το κόστος απώλειας των δεδομένων συσπειρώνεται γύρω από την αξία αυτών καθαυτών των δεδομένων και λιγότερο από παράγοντες, όπως η μείωση της παραγωγικότητας των υπαλλήλων και των τεχνικών υπηρεσιών. Τα υπάρχοντα Συστήματα Επανάκτησης (recovery) δίνουν κάποιες λύσεις σε κλασικές περιπτώσεις Πτώσης Υλικού ή Λογισμικού. Ωστόσο, όταν τα δεδομένα είναι εκτεθειμένα σε απειλές διαδικτύου ή όταν ο χώρος χαρακτηρίζεται ως **ανώμαλος χώρος δεδομένων**, τότε η ύπαρξη αποτελεσματικών μεθόδων ανάκτησης σπανίζει και είναι ορατή η ανάγκη νέων τρόπων αντιμετώπισής τους.

Η 11^{η} Σεπτεμβρίου αποτελεί ημερομηνία ορόσημο στον τομέα της αντιμετώπισης καταστροφών. Μετά την μέρα αυτή τίποτα δεν θα ήταν ίδιο όπως πριν. Μια ημερομηνία που αποτέλεσε σταθμό στην εφαρμογή των ήδη υπαρχόντων μεθοδολογιών αντιμετώπισης κρίσεων, αναθεώρησης και δημιουργίας νέων, πιο ευέλικτων, γρήγορων και αποτελεσματικών μορφών ανάκτησης. Οι νέες αυτές μεθοδολογίες προσπαθούν να αντιμετωπίσουν απειλές από πτώσεις, που μέχρι σήμερα θεωρούσαμε αμελητέες, αλλά τελικά αποδείχθηκαν γεγονότα μείζονος σημασίας. Ιδιαίτερα, μετά την καταστροφική αυτή ημέρα, μετά τις απώλειες σε ανθρώπινες ζωές έγινε κατανοητό ότι δεν μπορούμε σε ένα σύστημα ανάκτησης να βασιζόμαστε στον ανθρώπινο παράγοντα καθώς σε αυτήν την περίπτωση αποδείχθηκε ο πλέον εύθραυστος και μάλιστα τις πρώτες στιγμές της κρίσης είναι σχεδόν ανύπαρκτος!

3.2 ΘΕΜΕΛΙΩΔΕΙΣ ΑΡΧΕΣ

3.2.1 ΕΙΣΑΓΩΓΗ

Η εμφάνιση τόσων πολλών μορφών απειλών οδήγησε στη λήψη αντίστοιχου αριθμού αντιμέτρων. Είναι γενικά αντιληπτό ότι οι κίνδυνοι μεταλλάσσονται συνέχεια, παρουσιάζοντας νέες μορφές, πιο περίπλοκες και συχνά πιο επικίνδυνες, ξεφεύγοντας από τα στενά όρια κατηγοριοποίησης τους.

Τα μέτρα αντιμετώπισης τέτοιων κινδύνων, για να είναι αποτελεσματικά, θα πρέπει και αυτά να συνεχίσουν να μεταλλάσσονται με τον ίδιο ρυθμό. Για πολλές κατηγορίες κινδύνων τα μέτρα είναι ακριβώς τα ίδια.

Ένα επιτυχημένο σχέδιο αναδιοργάνωσης από καταστροφή προσφέρει μια μέθοδος, που μέσα από κατάλληλα βήματα προσπαθεί να επαναφέρει το σύστημα στην αρχική του κατάσταση και μάλιστα με ενέργειες που λαμβάνουν χώρα πριν, μετά και κατά τη διάρκεια μιας καταστροφής.

3.2.2 ΠΟΛΥΠΛΟΚΟΤΕΡΕΣ ΜΟΡΦΕΣ ΑΝΑΚΤΗΣΗΣ ΑΠΟ ΠΤΩΣΗ

Η μετάβασή μας σε πολυπλοκότερα σχέδια μας παραπέμπει από το χώρο της πληροφορικής και της ασφάλειας πληροφοριακών συστημάτων στο χώρο της Ανάλυσης Ρίσκου (Risk Management). Μεταπηδάμε, λοιπόν, σε ένα χώρο, όπου ο αναλυτής πρέπει να υπολογίσει και να προβλέψει το ρίσκο, με σκοπό να προβεί σε εκείνον το σχεδιασμό ενεργειών που θα ελαχιστοποιήσει το κόστος καταστροφής. Σε γενικές γραμμές, θα πρέπει να υπάρχει η δυνατότητα αναγνώρισης της παρούσας κατάστασης και κατανόησης των δομών του συστήματος ως προς:

- Την ανάλυση αυτών των δομών σε απλούστερες μορφές και εύρεση των κεντρικών σημείων αυτοκατάρρευσης του συστήματος.
- Το σχεδιασμό εφικτών, ικανών και αναλυτικών αντιμέτρων, ώστε να αποφευχθεί ενδεχόμενη κατάρρευση κρίσιμων σημείων του συστήματος και τέλος,
- Τον έλεγχο και την πιστοποίηση αξιοπιστίας μέσα από αλλεπάλληλες δοκιμές αντοχής [Wood (1996)].

3.2.3 ΓΕΝΙΚΟ ΠΛΑΙΣΙΟ ΑΝΑΠΤΥΞΗΣ ΣΤΡΑΤΗΓΙΚΗΣ ΑΝΑΚΤΗΣΗΣ ΑΠΟ ΠΤΩΣΗ

Μια επιτυχημένη μεθοδολογία ανάκτησης δεδομένων από πτώση οφείλει να στηρίζεται σε αρχές σύγχρονες, οι οποίες να λαμβάνουν υπόψη το γενικότερο κοινωνικό-οικονομικό πλαίσιο μέσα στο οποίο δρουν και υλοποιούνται. Κύριο στοιχείο τους πρέπει να είναι η προβλεψιμότητα και η δυνατότητα αποφυγής δυσμενών καταστάσεων εν τη γενέσει τους.

Ένα θετικό στοιχείο για ένα επιτυχημένο πλαίσιο ανάπτυξης είναι η ύπαρξη ικανών δομών επικοινωνίας, ανταλλαγής απόψεων και τεχνογνωσίας μεταξύ των τμημάτων που συμμετέχουν στο έργο υλοποίησης του συστήματος ανάκτησης. Η επικοινωνία είναι σημαντικό κομμάτι και γίνεται ακόμη πιο σημαντικό, όταν αυτή στηρίζεται στην αμοιβαία εκτίμηση, στο σεβασμό και το ομαδικό πνεύμα μεταξύ των μελών. Τα στοιχεία αυτά οδηγούν σε γρήγορη και επιτυχημένη υλοποίηση εφαρμογών ανάκτησης από πτώση.

Τέλος, ένα στοιχείο που ίσως να μη φαίνεται σημαντικό, αλλά είναι πράγματι αναγκαίο και αποτελεί το κομμάτι εκείνο που συντελεί στην εύρωστη λειτουργία του συστήματος είναι η δημιουργία εύχρηστων εγχειριδίων χρήσης του (manuals) καθώς και η προσομοίωσή του κάτω από αντίξοες συνθήκες, που τείνουν να αγγίξουν πραγματικές καταστάσεις κρίσης.

Με το πρώτο κομμάτι μέσα από ένα καλογραμμένο, φιλικό στο χρήστη και αναλυτικό manual επιτυγχάνουμε την εύκολη εκμάθηση του, την παρουσίαση του σε άπειρους χρήστες καθώς και την αναλυτική εμφάνιση της δομής του, σε περίπτωση που θα χρειαστεί μετατροπή κάποιο τμήμα του.

Με τη συνεχή προσομοίωσή του λαμβάνουμε εκείνα τα δεδομένα τα οποία θα καθορίσουν την αξιοπιστία του συστήματος, την αντοχή του σε ώρα κρίσης καθώς και πλούσιο υλικό για τυχόν ατέλειες και βελτιώσεις, που θεωρούνται αναγκαίες.

3.2.4 ΜΟΡΦΟΠΟΙΗΣΗ ΠΛΑΙΣΙΟΥ ΑΝΑΠΤΥΞΗΣ

Η δομή ενός τέτοιου συστήματος θα πρέπει να είναι επαναλαμβανόμενη. Ένα σύστημα ανάκτησης από πτώση θα πρέπει να αυτοτροφοδοτείται συνεχώς από δεδομένα, προκειμένου να βελτιώνεται

και να εξελίσσεται. Παρόλα αυτά, δεν μπορούμε να παραβλέψουμε τον ανθρώπινο παράγοντα που συντελεί στην ανάπτυξη ενός τέτοιου σχεδίου.

Αναλυτικότερα, ένα επιτυχημένο σύστημα αντιμετώπισης καταστροφών ξεκινά αρχικά από **την Αναγνώριση**. Ο υπεύθυνος του συστήματος κάνει μια αρχική βολιδοσκόπηση του χώρου πάνω στον οποίο θα εφαρμοστεί το μοντέλο αντιμετώπισης καταστροφών. Γίνεται μια πρώτη καταγραφή της δομής και της ιεραρχίας του συστήματος, με σκοπό την εύκολη κατανόηση του. Έτσι, δίνεται η δυνατότητα να μεταβούμε εύκολα στο επόμενο βήμα, **την Ανάλυση**.[Higuera (1994)]

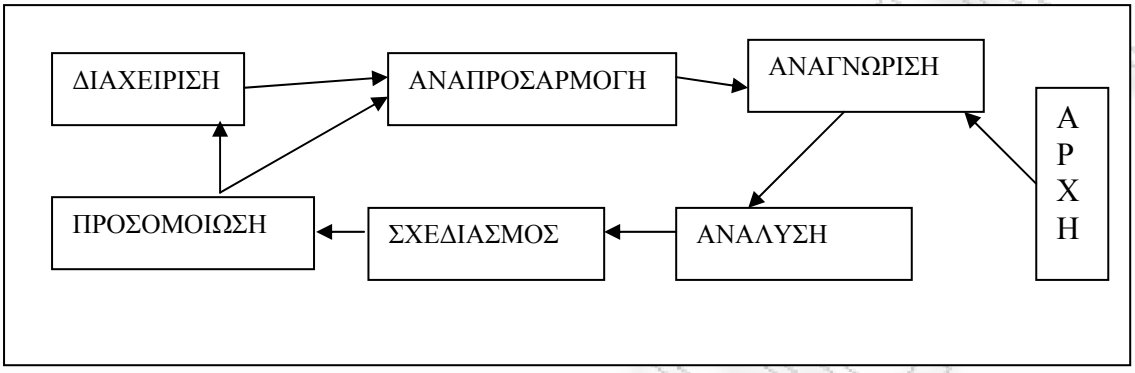
Το σημαντικότερο βήμα, αφού εκεί **δημιουργούνται τα θεμέλια ενός επιτυχημένου σχεδιασμού,** είναι η ανάλυση. Μέσα από συλλογική δουλειά και οργάνωση αναλύονται οι δομές και λαμβάνονται κρίσιμες αποφάσεις για παρεμβάσεις σε ευαίσθητους τομείς του συστήματος. Η επιλογή αυτών των τομέων γίνεται με βάση τις πληροφορίες που συλλέγονται μέσα από το ίδιο το σύστημα και τους φορείς που το χειρίζονται.

Μετά από τα δυο προηγούμενα βήματα φθάνουμε στο **κρίσιμο σημείο της υλοποίησης και του Σχεδιασμού.** Εδώ υλοποιούνται οι δομές και λαμβάνονται αντίμετρα προστασίας των κεντρικών σημείων αυτοκατάρρευσης του συστήματος δηλαδή πτώσεις που οδηγούν σε ολοκληρωτικές καταστροφές σε επίπεδο μεταλλάξεων, οπότε το σύστημα μετά την πτώση, οδηγείται σε μια διαφορετική δομή - τις περισσότερες φορές μη προβλέψιμη εκ των προτέρων .

Ένα σωστό και βιώσιμο μοντέλο πρέπει να έχει τη δυνατότητα αναπροσαρμογής. Αυτό είναι το κλειδί για την επιτυχία και τη μακροβιότητα ενός τέτοιου συστήματος, πράγμα που επιτυγχάνεται με την **Προσομοίωση του κάτω από αντίξοες συνθήκες**. Μέσα από τις συνεχείς προσομοιώσεις του λαμβάνονται πολλά χρήσιμα στοιχεία και πληροφορίες που συντελούν στη βελτίωση, ανάπτυξη και αναπροσαρμογή του συστήματος. Καταφέρνει το σύστημα, έτσι, να αυτοτροφοδοτείται με νέα στοιχεία, να βελτιώνει τυχόν ελαττώματα και να μειώνει την απόσταση που το χωρίζει από το πραγματικό υπό μελέτη σύστημα.

Τέλος, **αναγκαίο κομμάτι - όπως σε όλα τα συστήματα - είναι και η Διαχείριση**. Η επίβλεψη, δηλαδή, της φυσιολογικής ροής των διεργασιών του συστήματος και η άμεση παρέμβαση, όπου παραστεί

ανάγκη για την αποτροπή δυσμενών καταστάσεων. Σχηματικά η ροή εργασιών μπορεί να αναπαρασταθεί, όπως φαίνεται στο σχήμα 3.1.



Σχήμα 3.1
Ροή εργασίας διαχείρισης κρίσεων

Η ορθή ροή της εργασίας που ακολουθεί η μεθοδολογία αποτελεί τον παράγοντα κλειδί για ένα επιτυχημένο σύστημα ανάκτησης από πτώση. Η Ανάκτηση δεν μπορεί να είναι σαφώς αποτέλεσμα μιας και μόνης κίνησης μετά την πτώση, αλλά ενός κύκλου ενεργειών-εργασιών που εξασφαλίζουν την επιτυχία του συστήματος ανάκτησης.

Είναι, επίσης, κατανοητό ότι οι κρίσεις δεν αποφεύγονται με ευχολόγια αλλά με ολοκληρωμένη και έγκαιρη αντιμετώπιση. Οι συνέπειες από μια κρίση είναι καταστροφικές για μια εταιρεία, αφού όχι μόνο θα ζημιωθεί από την απώλεια των δεδομένων της, αλλά θα ζημιωθεί και από την προσπάθεια ανάκτησης τους. Η ζημιά στην ουσία δεν είναι μόνο οικονομική, αλλά θα αμαυρώσει και το κύρος της εταιρείας, μειώνοντας την αξιοπιστία της προς τους πελάτες της, γεγονός που δύσκολα αποκαθίσταται.

Τα σύγχρονα πληροφοριακά συστήματα - όπως προαναφέρθηκε - δρουν σε ένα εχθρικό περιβάλλον λειτουργίας. Η επικινδυνότητα του περιβάλλοντος οφείλεται στην ολοένα αυξανόμενη εξάρτηση από τα δικτυοκεντρικά πληροφοριακά συστήματα, τις εκτεταμένες εσωτερικές εγκαταστάσεις επικοινωνιών των ψηφιακών πληροφοριακών συστημάτων, την εκρηκτική ανάπτυξη του Διαδικτύου και τους ισχυρούς δεσμούς μεταξύ των επιχειρηματικών εταίρων. Η περαιτέρω άνθηση του e-business και του e-government διευκολύνθηκε από την πρόοδο στις τεχνολογίες των πληροφοριών και των τηλεπικοινωνιών, η οποία δημιούργησε με τη σειρά της πολλές νέες δυνατότητες, αλλά και ένα περιβάλλον με πληθώρα κινδύνων.

Όντας αντιμέτωποι με αυτές τις μεγάλες προκλήσεις και λαμβάνοντας υπόψη άλλες πτυχές, όπως την ευρέως διαδεδομένη εσωτερική απειλή και τις μη τεχνικές διαρροές πληροφοριών ασφάλειας, οι ειδικοί στην ασφάλεια πληροφοριακών συστημάτων πρέπει να αξιολογούν τους εξειδικευμένους κινδύνους της επιχείρησης τους, για να διασφαλίσουν το πιο κατάλληλο επίπεδο ασφάλειας, επιτρέποντας την αδιάκοπη ροή των λειτουργιών της επιχείρησης.

Η επιπόλαιη ή σε αρκετές περιπτώσεις ελλιπής ανάλυση του κίνδυνου οδηγεί πολύ συχνά σε τραγικά αποτελέσματα με οδυνηρές συνέπειες όχι μόνο για το ίδιο το σύστημα αλλά και για τους χειριστές του. Πληθώρα ιστορικών παραδειγμάτων πιστοποιεί του λόγου το ακριβές όπως αναλύεται σε επόμενες παραγράφους.

Τον Μάϊο του 1982 στον πόλεμο των Falkland ο Αγγλικός στόλος έχασε το τότε σύγχρονο αντιτορπιλικό Sheffield, διότι το Ολοκληρωμένο Πληροφοριακό Σύστημα (ΟΠΣ) προστασίας του σκάφους λειτουργούσε στο επίπεδο οργάνωσης σε θέατρα επιχειρήσεων με Σοβιέτ στη βόρεια θάλασσα, με άμεση συνέπεια να μην αντιδράσει το αντιπυραυλικό Sea Dart που διέθετε όταν αεροσκάφος της Αργεντινής εκτόξευσε έναν Γαλλικό Exocet, δηλαδή έναν "φιλικό" πύραυλο του ΝΑΤΟ ! Φαινόμενο Προγραμματιστικής Σκληρότητας.[Walker P. (1983)]

Τον Ιανουάριο του 1986 το διαστημικό λεωφορείο Challenger καταστρέφεται στην εκτόξευση παρασύροντας τους 7 αστροναύτες του σε τραγικό θάνατο. Όπως επίσημα από την ΝΑΣΑ ανακοινώθηκε το ατύχημα έγινε διότι το ΟΠΣ δεν έκανε σε πραγματικούς χρόνους ενημέρωση των ESS και MIS, με άμεση συνέπεια τον κακό συντονισμό Αναδόχων και ανώτερης διοίκησης του όλου προγράμματος![Garrity et al. (1998)]

Το 1995 μετά από λανθασμένη λειτουργία του ΟΠΣ των πρώτων βοηθειών του Λονδίνου LASCAD, φονεύονται 30 ασθενείς και χάνονται σχεδόν 65 εκατομμύρια λίρες που ήταν το κόστος σχεδιασμού και ανάπτυξης αυτού![Beynon-Davies P. (1995)]

Επομένως, οδηγούμαστε στο συμπέρασμα ότι η ανάλυση του κινδύνου είναι μια εργασία αναγκαία. Μια εργασία που αν δεν υλοποιηθεί, θα αποτελέσει την Δαμόκλειο Σπάθη για την ανάπτυξη και επέκταση επιτυχημένων, λειτουργικών και αξιόπιστων πληροφοριακών συστημάτων.

3.2.5 ΜΕΤΑΛΛΑΚΤΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΚΑΙ ΜΕΤΑΛΛΑΞΕΙΣ

3.2.5.1 ΟΡΙΣΜΟΣ ΜΕΤΑΛΛΑΚΤΙΚΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Τα σύγχρονα ψηφιακά πληροφοριακά συστήματα την τελευταία δεκαετία, ιδιαίτερα στην Ευρώπη και την Αμερική ενσωμάτωσαν νέες ψηφιακές δομές. Απώτερος στόχος τους ήταν η κάλυψη όλων των πτυχών της καθημερινότητας του Ανθρώπου. Οι Εφαρμογές Λογισμικού, σήμερα περιέχουν ένα σύγχρονο ψηφιακό πληροφοριακό συστήματος, καθώς είναι το βασικό δομικό συστατικό της ομαλής λειτουργίας της όλης εφαρμογής.

Για να μπορέσει ένα τέτοιο λογισμικό να καλύψει ένα τόσο μεγάλο φάσμα οφείλει να έχει ειδικές ικανότητες που να ξεπερνούν τα στενά πλαίσια βάσει των οποίων δραστηριοποιούνται τα σύγχρονα πληροφοριακά συστήματα. **Η Ακεραιότητα, η Αξιοπιστία μπορεί να είναι αναγκαία, αλλά δεν είναι από μόνα τους ικανά χαρακτηριστικά,** ώστε να προσφέρουν με τη σειρά τους την απαιτούμενη ώθηση.

Η **Προσαρμοστικότητα** είναι το ειδικό χαρακτηριστικό, που απαιτείται, προκειμένου να επιτευχθεί ο στόχος, μολονότι πολλές φορές η έννοια αυτή παρερμηνεύεται. Ένα τέτοιο πληροφοριακό σύστημα θα έχει τη δυνατότητα:

- **να ανταποκρίνεται στις αλλαγές που συντελούνται στο περιβάλλον λειτουργίας του, αποφεύγοντας δυσμενείς καταστάσεις.**
- **να αναδομείται, δημιουργώντας νέα δομικά εξελικτικά στοιχεία.**
- **να αναπτύσσει άμεσα νέα χαρακτηριστικά σε σύντομο χρονικό διάστημα.**
- **να έχει την ικανότητα αναδόμησης και αυτοδιόρθωσης δυσλειτουργιών.**
- **να συγκεντρώνει διαχρονικά όλη την τεχνογνωσία και εμπειρία του Συστήματος.**

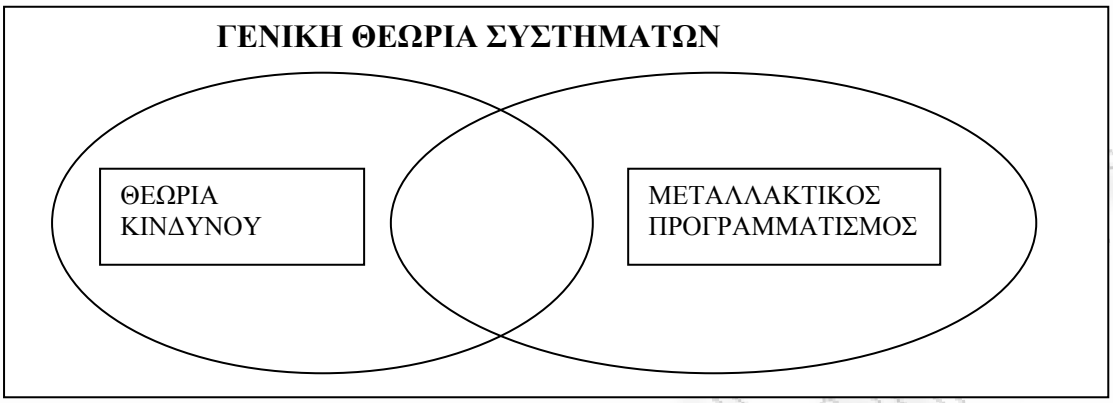
Τα παραπάνω αποτελούν και τις θεμελιώδεις αρχές του **Μεταλλακτικού Προγραμματισμού**. Αρχές που συντελούν στη δημιουργία πληροφοριακών συστημάτων ικανών να προσαρμόζονται στις απαιτήσεις του περιβάλλοντος λειτουργίας, ενώ παράλληλα να μπορούν να συνεχίσουν να υπηρετούν το σκοπό για τον οποίο

δημιουργήθηκαν [Kalligatsis (2005)]. Ο Μεταλλακτικός Προγραμματισμός στην ουσία είναι η δημιουργία πληροφοριακών ψηφιακών συστημάτων κατάλληλων να μεταμορφώνονται. Ένα μεταλλακτικό πληροφοριακό σύστημα πλεονεκτεί έναντι ενός συμβατικού πληροφοριακού Συστήματος. Η εκτεταμένη ευλυγισία του σε συνδυασμό με την ικανότητα να παρεμβαίνει τόσο στο σχεδιασμό όσο και στον προγραμματισμό των δομικών του κλάσεων το καθιστά ισχυρό παράγοντα ανάπτυξης ενός σύγχρονου οργανισμού. Αποτελεί την αιχμή του δόρατος για την ψηφιακή ανάπτυξη ενός οργανισμού στον 21^{ο} αιώνα και τη Νέα Ψηφιακή Εποχή με τη νέα τάξη πραγμάτων.

Ακολούθως, σημαντικό πλεονέκτημα ενός Μεταλλακτικού Ψηφιακού Συστήματος είναι η ικανότητα του να αναπροσαρμόζεται σε πληθώρα εταιρικών αλλαγών που συντελούνται μέσα στο περιβάλλον λειτουργίας. Η προσαρμογή αυτή γίνεται τις περισσότερες φορές χωρίς προβλήματα, άκοπα καθώς και χωρίς αυστηρούς περιορισμούς και αποκλίσεις από τον πραγματικό κόσμο.[Mavrommatis (2005)].

Ο Μεταλλακτικός προγραμματισμός αποτελεί μετεξέλιξη του αντικειμενοστραφή προγραμματισμού, προσθέτοντας το νέο χαρακτηριστικό της δομικότητας στην ευρύτερη έννοια του προγραμματισμού ψηφιακών εφαρμογών. Ο Μεταλλακτικός προγραμματισμός μπορεί να γίνει μέσο-μακροπρόθεσμα ένα σημαντικό εργαλείο δημιουργίας ευέλικτων ψηφιακών συστημάτων. Μπορεί να αποτελέσει το θεμέλιο λίθο για τη δημιουργία μιας νέας αντίληψης στη δημιουργία ψηφιακών πληροφοριακών συστημάτων επιδέξιων να προσαρμοστούν σε ιδιαίτερα περιβάλλοντα λειτουργίας. Η προσαρμοστικότητα αυτή θα συμβάλλει στην ανάπτυξη δομών τέτοιων, ώστε να αποφεύγονται οι κρίσιμες καταστάσεις με το λιγότερο δυνατό κόστος.

Ο Μεταλλακτικός προγραμματισμός (Mutative Programming) μπορεί να αποτελέσει συμπλήρωμα της Θεωρίας του Κινδύνου κάτω από κατάλληλες συνθήκες και με την προσθήκη θεσμικών θεμελιωδών αρχών. Το επόμενο σχήμα παρουσιάζει παραστατικά τα προαναφερθέντα (Σχήμα 3.2). Κρίσεις σε Εχθρικά Περιβάλλοντα εργασίας θα μπορούν να αφομοιωθούν και να αντιμετωπιστούν, όταν τα πληροφοριακά συστήματα αποκτήσουν την κατάλληλη υποδομή.



Σχήμα 3.2
Αλληλοσυσχέτιση Θεωρίας Κινδύνου-Μεταλλακτικού Προγραμματισμού

3.2.5.2 Η ΕΝΝΟΙΑ ΤΗΣ ΜΕΤΑΛΛΑΞΗΣ

Σε προηγούμενη παράγραφο αναλύσαμε την έννοια του μεταλλακτικού προγραμματισμού σαν εργαλείο δημιουργίας μιας νέας δομής σύγχρονων πληροφοριακών συστημάτων ικανών να αντεπεξέλθουν σε προβλήματα που εμφανίζονται σε Ασταθή Περιβάλλοντα λειτουργίας. Η βασική αρχή στο μεταλλακτικό προγραμματισμό - αλλά όπως θα δούμε και παρακάτω - και στην Θεωρία Κινδύνου είναι η **Μετάλλαξη** ενός πληροφοριακού συστήματος**.**

Μετάλλαξη ενός ψηφιακού πληροφοριακού συστήματος ορίζουμε μια αλλαγή που συμβαίνει σε συγκεκριμένη στιγμή μέσα στο χρονικό ορίζοντα λειτουργίας του συστήματος, ικανή να αλλάξει τη συνολική δομική και λειτουργική ικανότητα του. Στον Μεταλλακτικό Προγραμματισμό το Λογισμικό Εφαρμογών ακολουθεί τις μεταλλάξεις του φυσικού Συστήματος που ψηφιακά εκφράζει.

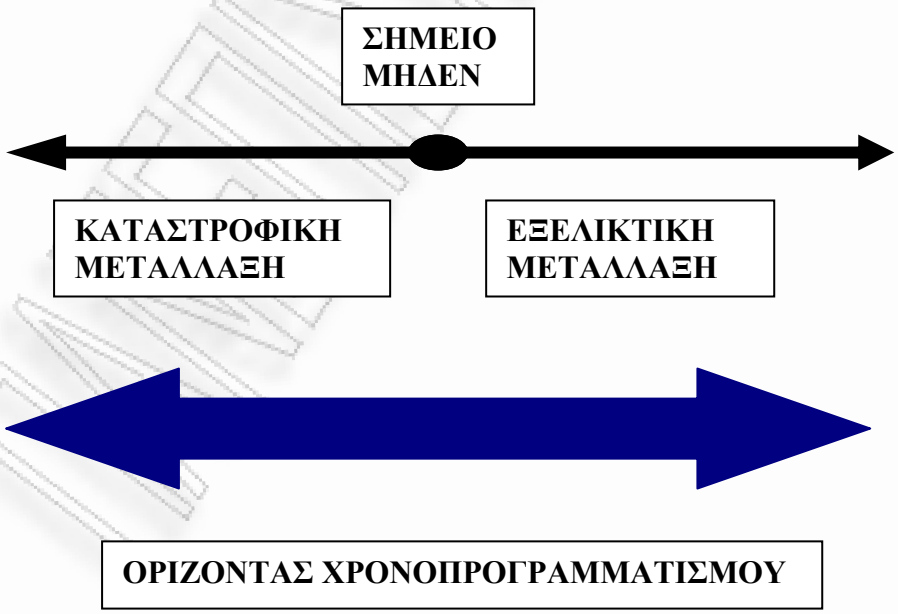
Η αλλαγή αυτή μπορεί να συμβεί στο ίδιο το σύστημα ή σε κάποιο από τα υποσυστήματα του. Η Μετάλλαξη στο σύστημα συμβαίνει όταν αλλάζει η δομή του περιβάλλοντος λειτουργίας δηλαδή, όταν η συνιστάμενη των δυνάμεων, που επιδρούν σε αυτό, είναι μεγαλύτερη από την αντίδραση του ίδιου του συστήματος.

Η μετάλλαξη δεν οφείλεται αποκλειστικά σε εξωγενείς παράγοντες που επιδρούν στο σύστημα, προερχόμενοι κυρίως από αστάθειες του περιβάλλοντος λειτουργίας. Μεταλλακτικές καταστάσεις σε

συγκεκριμένες στιγμές του χρονοπρογραμματισμού μπορούν να εμφανιστούν, όταν επιδράσουν στο σύστημα και ενδογενείς παράγοντες.

Οι αστάθειες και οι δυσλειτουργίες είναι βασικοί λόγοι κατάρρευσης συνεκτικών δομών του συστήματος ή των υποσυστημάτων αυτών. Όταν μια μεταλλακτική κρίση συμβαίνει, το αποτέλεσμα είναι συνήθως καταστροφικό, καθώς υποσυστήματα του κυρίως συστήματος αποδομούνται και χρήσιμες λειτουργίες για την ορθή λειτουργία του συστήματος τίθενται ανενεργές. **Οι αλυσιδωτές μεταλλακτικές αντιδράσεις, αν εφαρμοστούν σε συγκεκριμένο χρονικό σημείο μέσα στον ορίζοντα λειτουργίας του συστήματος, είναι ικανές να οδηγήσουν και στην οριστική καταστροφή του συστήματος.**

Η μετάλλαξη **δεν έχει πάντα αρνητική κατάληξη**. Συχνά ένα σύστημα κάνει υπέρβαση με μετάλλαξη για να σπάσει την στατικότητα του. Όπως ορίσθηκε παραπάνω, η μετάλλαξη είναι μια αλλαγή που συμβαίνει στο σύστημα. Αυτή η αλλαγή μπορεί κάτω από συγκεκριμένες συνθήκες να αναβαθμίσει τη λειτουργική ικανότητα του συστήματος, προσθέτοντας του νέα χαρακτηριστικά που βελτιώνουν και βελτιστοποιούν το μεταλλαγμένο σύστημα. Γίνεται εύκολα αντιληπτό ότι **μια τέτοια κατάσταση είναι στην ουσία μια Εξελικτική κατάσταση για το σύστημα**. Το παρακάτω σχήμα συνοψίζει τα προαναφερθέντα και παριστά αυτό το φαινόμενο. (Σχήμα 3.3)



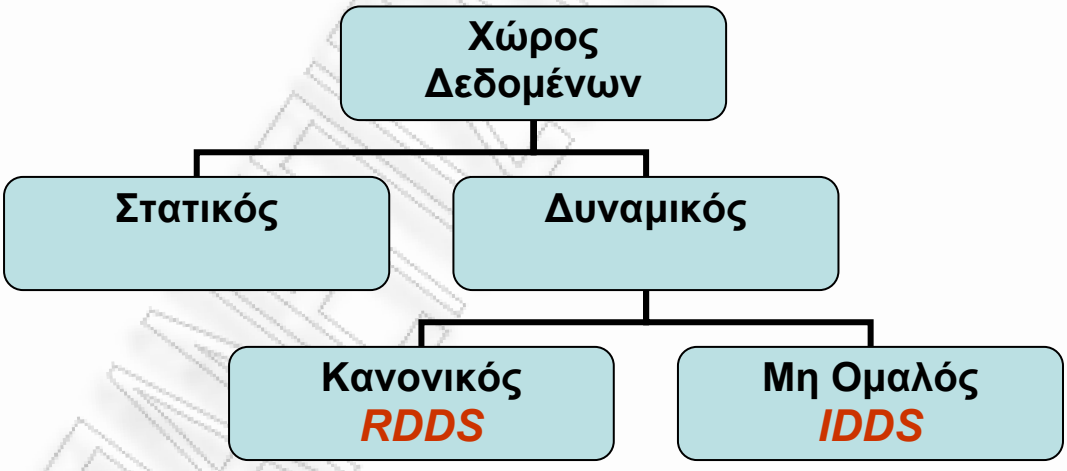
Σχήμα 3.3
Ορισμός Μεταλλακτικών Καταστάσεων

Η Μετάλλαξη η οποία λαμβάνει χώρα στο **Σημείο Μηδέν** αποτελεί ορόσημο για τη μελλοντική κατάσταση του συστήματος. Το σημείο μηδέν είναι σημείο καμπής που θα οδηγήσει το ψηφιακό πληροφοριακό σύστημα στην Κρίση ή την Ανέλιξη, σε Μαύρη Λειτουργική Οπή (Black Operational Holes) ή σε Λευκή Λειτουργική Οπή (White Operational Holes). [Panayiotopoulos 1992]

3.2.6 ΧΩΡΟΙ ΔΕΔΟΜΕΝΩΝ

Βασικό συστατικό πάνω στο οποίο θα στηριχθεί, στη συνέχεια, η Σύγχρονη Θεωρία Διοίκησης Κινδύνου είναι ο **Χώρος Δεδομένων (Data Space).** Είναι το περιβάλλον με το οποίο αλληλεπιδρά και λειτουργεί ένα σύγχρονο ψηφιακό πληροφοριακό σύστημα. Ο χώρος δεδομένων είναι παράγοντας βαρύνουσας σημασίας. Τα δεδομένα του εφοδιάζουν το πληροφοριακό σύστημα, προσφέροντας του λόγο ύπαρξης.

Ένας χώρος δεδομένων, ανάλογα με το είδος των δυνάμενων που ασκούνται, τη δομή του καθώς και τα πληροφοριακά συστήματα που δρουν μέσα σε αυτόν, χωρίζεται σε δυο βασικές κατηγόριες, όπως φαίνεται και από το παρακάτω σχήμα (Σχήμα 3.4).



Σχήμα 3.4
Δομή Χώρου Δεδομένων

Ο αρχικός Χώρος Δεδομένων, ανάλογα με το είδος των δεδομένων που περιλαμβάνει, χωρίζεται σε δυο βασικές κατηγορίες, στο **στατικό και το δυναμικό χώρο δεδομένων.** Ο στατικός χώρος δεδομένων αποτελείται από σταθερές δυνάμεις. Η δομή των δεδομένων είναι σταθερή μέσα στον ορίζοντα χρονοπρογραμματισμού του συστήματος.

Το πληροφοριακό σύστημα λειτουργεί σταθερά, ενώ το αποτέλεσμα της λειτουργίας του είναι προβλέψιμο και προκύπτει μετά την ομαλή επεξεργασία των δεδομένων του χώρου από το σύστημα. Η επεξεργασία των δεδομένων γίνεται με γνωστές μεθόδους του μαθηματικού προγραμματισμού, που προσφέρουν βελτιστοποιημένες ή καλές λύσεις. Η σταθερότητα αυτή προσφέρει ασφάλεια και σιγουριά αλλά είναι ένα σπάνιο φαινόμενο, ειδικότερα στις μέρες μας, όπου η εξέλιξη των πληροφοριακών συστημάτων είναι τέτοια, ώστε η δομή τους να αποτελείται από κατανεμημένα και διάσπαρτα υποσυστήματα. **Η πλειοψηφία των υπαρχόντων χώρων δεδομένων είναι στην πραγματικότητα σήμερα Δυναμικά μεταβαλλόμενη και ιδιαίτερα Ασταθής.** [Panayiotopoulos (1992)].

Στη Θεωρία Διαχείρισης Κινδύνου είναι αναγκαία η λήψη κρίσιμων αποφάσεων στον παρόντα χρόνο λειτουργίας του υπό μελέτη συστήματος. Οι αποφάσεις αυτές πρέπει να είναι σωστές και γρήγορες, αφού το αποτέλεσμα τους σε κλάσματα του δευτερόλεπτου μπορεί να αποτρέψει μια κρίση ή να επιτρέψει ακόμα και την καταστροφή του συστήματος. Αυτό οφείλεται στο γεγονός ότι ο χώρος των δεδομένων είναι δυναμικά μεταβαλλόμενος. Ορίζεται DS ο χώρος των δεδομένων μέσα στον οποίο λειτουργεί το σύστημα S και πάνω σε αυτό υλοποιούνται οι δυνάμεις του χώρου. Έτσι έχουμε:

$$\mathbf{DS = (D, F, RL, \Delta t)}$$

Έτσι ώστε:

$\Delta t = t_f - t_p$ Ο χρονικός ορίζοντας προγραμματισμού λειτουργίας του συστήματος, με t_p τον παρόντα χρόνο και t_f το τέλος του χρονικού ορίζοντα.

D Τα δεδομένα του συστήματος.

RL Οι κανόνες, αλγόριθμοι και γενικά οι Διαδικασίες που απαιτούνται για να υλοποιηθεί η Σύγχρονη Θεωρία Διοίκησης Κινδύνου.

$F=\{f_i\}, \quad i \in N^* = \{1,2,...,n\}$ Σύνολο καταστροφικών δυνάμεων (Catastrophe Forces) που επηρεάζουν το σύστημα S. Είναι ανενεργές στο t_p, αλλά μπορούν να ενεργοποιηθούν

μέσα στο Δt. Είναι η έννοια της απειλής που γίνεται καταστροφή.

Ο χώρος DS ονομάζεται δυναμικός χώρος δεδομένων. Είναι πράγματι ένας δυναμικός χώρος δεδομένων, αφού τα δεδομένα που δέχεται το σύστημα από την αλληλεπίδραση του με το περιβάλλον λειτουργίας είναι και αυτά δυναμικά με αποτέλεσμα η δομή τους συνεχώς να αλλάζει. Η δομή του περιβάλλοντος λειτουργίας δεν είναι σταθερή στο διάστημα Δt. Η αστάθεια αυτή οφείλεται στο γεγονός ότι εμφανίζονται δυνάμεις μέσα στο περιβάλλον λειτουργίας, οι οποίες συνεχώς μεταλλάσσονται και δύσκολα αντιμετωπίζονται. Οι δυναμικοί χώροι δεδομένων χωρίζονται σε δυο υποκατηγορίες. Η πρώτη υποκατηγορία είναι ο ομαλός δυναμικός χώρος δεδομένων, όπου τα δεδομένα του χώρου μπορούν να μεταβάλλονται δυναμικά. **Η μεταβολή αυτή μπορεί να προβλεφθεί στηριζόμενη σε μαθηματικούς κανόνες, που προσδιορίζουν τις αλλαγές που συμβαίνουν μέσα στον ορίζοντα Δt.**

Αντίθετα, σε έναν Ασταθή (μη Ομαλό) Δυναμικό Χώρο Δεδομένων τα στοιχεία του μεταβάλλονται **με μη γνωστές και πολύπλοκες διαδικασίες**. Είναι ουτοπία να πιστεύει κανείς ότι μπορεί να προβλέψει και να κατανοήσει το μηχανισμό αλλαγής των δεδομένων του περιβάλλοντος. Αυτό συμβαίνει γιατί σε μια χρονική στιγμή t_1 η πληροφορία είναι δεδομένη, αλλά σε μελλοντικό χρόνο θεωρείται ατελής και κάτω του κρίσιμου σημείου πληροφόρησης. [Panayiotopoulos (1992)]

3.3 ΓΕΝΙΚΗ ΘΕΩΡΙΑ ΔΙΑΧΕΙΡΙΣΗΣ ΚΙΝΔΥΝΟΥ

3.3.1 ΕΙΣΑΓΩΓΗ

Τα σύγχρονα ψηφιακά πληροφοριακά συστήματα λειτουργούν, όπως προαναφέρθηκε, σε ένα δυναμικό, ασταθές, μεταβαλλόμενο περιβάλλον λειτουργίας. Αυτή η αλληλεπίδραση με το περιβάλλον τους προσφέρει διευρυμένες δυνατότητες λειτουργίας και ευελιξίας, αποδεσμεύοντας τα από στατικά συμβατικά μοντέλα.

Η δυναμική μεταβολή του περιβάλλοντος κρύβει κίνδυνους που απειλούν την ομαλή λειτουργία και ανάπτυξη του πληροφοριακού συστήματος. Οι κίνδυνοι που μετουσιώνονται σε κρίσεις οδηγούν συχνά στην κατάρρευση του συστήματος, που δύσκολα μπορεί να αποφευχθεί, αλλά και εξίσου δύσκολα μπορεί να αντιμετωπισθεί. Σε περιπτώσεις κρίσεων είναι επιτακτική η ανάγκη ενός συστήματος διαχείρισης

κρίσεων, ικανού να αντιμετωπίσει ουσιαστικά την επερχόμενη κατάρρευση του συστήματος, δίνοντας τη δυνατότητα στο σύστημα να ανακτήσει τα καταστραμμένα δεδομένα του. **Η ανάκτηση καταστραμμένων δεδομένων είναι ιδιαίτερα αναγκαία για ένα σύστημα,** αν θέλει να αυτοχαρακτηρίζεται σαν σύγχρονο ψηφιακό πληροφοριακό σύστημα. Σημαντική προϋπόθεση για τη σωστή και επιτυχημένη ανάκτηση είναι - όπως θα αναλυθεί σε επόμενη παράγραφο - η γρήγορη, ολοκληρωμένη και με το ελάχιστο δυνατό κόστος διαδικασία αποκατάστασης των ζημιών.

Από την άλλη μεριά δεν είναι όμως σπάνιες οι φορές όπου το σύστημα μετά από μια κρίση τίθεται σε κατάσταση μερικής λειτουργίας. Υποσυστήματα του υπάρχοντος συστήματος καταρρέουν υποσκάπτοντας την δομική συνεκτικότητα του. Μεταλλακτικές καταστάσεις εμφανίζονται συμβάλλοντας στην περαιτέρω όξυνση της δυσλειτουργικότητας του συστήματος.

Το κενό έρχεται να καλύψει η **Γενική Θεωρία Διαχείρισης Κινδύνου,** που μέσα από **ένα ολοκληρωμένο μοντέλο Πρόληψης – Αντιμετώπισης Κρίσεων επιτυγχάνεται** η ελαχιστοποίηση της δυνατότητας εμφάνισης καταστροφικών δυνάμεων πάνω σε ένα ψηφιακό πληροφοριακό σύστημα το οποίο δρα σε ένα Εχθρικό, Δυναμικά Μεταβαλλόμενο περιβάλλον. Παράλληλα με την Πρόληψη, υπάρχει και η Ανάκτηση των καταστραμμένων δεδομένων σαν απόρροια υλοποίησης επικίνδυνων δυναμικών φαινομένων, που προέρχονται από ασταθή δεδομένα του περιβάλλοντος λειτουργίας. Παρουσιάζεται ένα μοντέλο ανάκτησης με το ελάχιστο δυνατό κόστος του καταλληλότερου recovery, προσφέροντας πληθώρα δυνατοτήτων, οι οποίες δεν μπορούσαν να γίνουν εκμεταλλεύσιμες έως τώρα.

3.3.2 ΔΟΜΗ ΓΕΝΙΚΗΣ ΜΕΘΟΔΟΛΟΓΙΑΣ ΔΙΑΧΕΙΡΙΣΗΣ ΚΙΝΔΥΝΟΥ

Η Γενική Μεθοδολογία Διαχείρισης Κινδύνου είναι ένα χρήσιμο εργαλείο για την ολοκληρωμένη ανάπτυξη ενός σύγχρονου ψηφιακού πληροφοριακού συστήματος. Η θεωρία πρέπει να λαμβάνεται σοβαρά υπόψη κατά τη συνολική διαδικασία ανάπτυξης και υλοποίησης ενός πληροφοριακού συστήματος και να αποτελεί επέκταση της Θεωρίας Πληροφοριακών Συστημάτων, όπως αυτή έχει υλοποιηθεί από διάφορους μελετητές.

Η Γενική Θεωρία Διαχείρισης Κινδύνου δομικά χωρίζεται σε δυο μεγάλες υποκατηγορίες στην πρόληψη και στην ανάκτηση. Σκοπός της πρόληψης είναι η μελέτη στο αρχικό στάδιο υλοποίησης του πληροφοριακού συστήματος. Μέσα από εκτενή και ολοκληρωμένη ανάλυση των δομών του ψηφιακού συστήματος ομαδοποιούνται και επισημαίνονται όλοι οι κίνδυνοι που το απειλούν καθ' όλη την διάρκεια ανάπτυξης και λειτουργίας του μέσα σε ένα συγκεκριμένο ορίζοντα χρονοπρογραμματισμού *Δt*. Μια σωστή και ολοκληρωμένη μελέτη του ψηφιακού πληροφοριακού συστήματος απαιτεί την πλήρη κατηγοριοποίηση των κινδύνων που το απειλούν **χωρίς αποκλεισμούς**. Όλοι οι κίνδυνοι θεωρούνται αρχικά ισοπίθανα φαινόμενα, ικανά να πλήξουν ανά πάσα στιγμή το σύστημα μέσα στον ορίζοντα λειτουργίας του. Φυσικά όπως αντιλαμβάνεται κανείς, υπάρχουν κατηγορίες κινδυνικών φαινομένων (ιοί, πτώσεις Server), οι οποίες υλοποιούνται με μεγαλύτερη συχνότητα σε σχέση με άλλες κατηγορίες κινδυνικών φαινομένων (γεωλογικά φαινόμενα, πλημμύρες), οι οποίες εμφανίζονται σπανιότερα. Παρόλα αυτά υπάρχουν στιγμές μέσα σε συγκεκριμένο διάστημα λειτουργίας κατά τη διάρκεια των οποίων η εμφάνιση ενός τέτοιου σπάνιου φαινομένου μπορεί να προκαλέσει ανυπολόγιστες καταστροφές. Αυτό το φαινόμενο οφείλεται στο γεγονός ότι ορισμένα **καταστροφικά φαινόμενα εμφανίζονται σε ανύποπτες στιγμές με έντονη σφοδρότητα** καταλαμβάνοντας εξαπίνης τους αμυντικούς μηχανισμούς. Τα φαινόμενα της 11^{ης} Σεπτεμβρίου 2001 στην Νέα Υόρκη έδειξαν ότι πρέπει να είμαστε πάντα προετοιμασμένοι για το αναπάντεχο. Η έγκαιρη προετοιμασία και η πρόληψη, λοιπόν, είναι το κλειδί της επιτυχίας.

Παράλληλα, μέσα από τη μεθοδολογία πρόληψης που υλοποιείται σαν παρακλάδι της γενικής θεωρίας παρουσιάζονται χρήσιμα εργαλεία πρόληψης και αντιμετώπισης κινδυνικών φαινομένων, τα οποία θα αναλυθούν σε επόμενες παραγράφους.

Η δεύτερη υποκατηγορία της Ανάκτησης περιλαμβάνει όλες εκείνες τις καταστροφικές δυνάμεις που υλοποιήθηκαν στο σύστημα. Η υλοποίηση τους οφείλεται είτε σε αδυναμία του συστήματος να προλάβει την κρίση και να την αντιμετωπίσει εν τη γενέσει της είτε στη δημιουργία αλυσιδωτών καταστροφικών φαινομένων που δεν είχαν προβλεφθεί από το Πληροφοριακό σύστημα. Η Γενική Θεωρία Διαχείρισης Κινδύνου αντιμετωπίζει ολοκληρωμένα κάθε κρίση η οποία λαμβάνει χώρα σε οποιαδήποτε στιγμή μέσα στον ορίζοντα χρονοπρογραμματισμού του συστήματος. Επιπλέον, **αντιμετωπίζεται οποιαδήποτε καταστροφική**

κρίση έχει λάβει χώρα με το ελάχιστο δυνατό κόστος, όπως θα αναλυθεί στη συνέχεια.

Η δεύτερη υποκατηγορία της Ανάκτησης είναι ιδιαίτερα σημαντική, αφού στην πραγματικότητα δεν αφήνει τα πράγματα στην τύχη τους. Επεμβαίνει δυναμικά σε μια καταστροφική κρίση, δίνοντας σημαντικό βοήθημα στη διαχείριση του ψηφιακού πληροφοριακού συστήματος και στις αποφάσεις που πρέπει να λάβει άμεσα, ώστε να **ελαχιστοποιήσει την καταστροφή.** Ακόμα και σε καταστάσεις κρίσεων, η γενική θεωρία διαχείρισης κινδύνων επεμβαίνει δυναμικά, προσφέροντας λύσεις οι οποίες είναι υλοποιήσιμες, πραγματικές και προσφέρουν τη μέγιστη δυνατή απόδοση.

3.3.3 ΓΕΝΙΚΕΣ ΑΡΧΕΣ

Προτού αναλυθούν τα τμήματα της γενικής θεωρίας θα ορισθεί ο χώρος μέσα στον οποίο εφαρμόζεται το μοντέλο διαχείρισης κρίσεων. Θα γίνει μια σύντομη αναφορά στα δομικά σύνολα που αποτελούν τα θεμέλια της θεωρίας. Είναι οι χώροι μέσα στους οποίους αναπτύσσεται και λειτουργεί το ψηφιακό πληροφοριακό σύστημα. Παράλληλα, είναι οι δεξαμενές από όπου ξεπετάγονται οι δυνάμεις που δρουν πάνω σε αυτό είτε για να προκαλέσουν μια κρίση είτε για να συμβάλλουν στην αποτροπή της.

Ένα ψηφιακό πληροφοριακό σύστημα όπως αναφέρθηκε δρα σε ένα ασταθή δυναμικά μεταβαλλόμενο χώρο δεδομένων. Ας υποθέσουμε ότι Ω είναι το εχθρικό περιβάλλον λειτουργίας μέσα στο οποίο λειτουργεί το πληροφοριακό σύστημα στον ορίζοντα χρονοπρογραμματισμού Δt. Το σύστημα S είναι ένα σύγχρονο, ολοκληρωμένο, ψηφιακό πληροφοριακό σύστημα το οποίο λειτουργεί στον χώρο Ω. Άμεση συνέπεια των παραπάνω είναι το γεγονός ότι το σύστημα θα είναι πολύπλοκο και πιθανότατα κατανεμημένο σε διάφορες κορυφές του χώρου δεδομένων Ω. Θα αποτελείται από ένα πολυπληθές σύνολο υποσυστημάτων που αλληλεπιδρούν μεταξύ τους. Ας υποθέσουμε ότι το σύστημα S αποτελείται από ένα σύνολο n υποσυστημάτων, όπως φαίνεται παρακάτω:

$$S=\{s_1, s_2,\ldots,s_n\} \quad , \quad S^*=\{1,2,\ldots,n\} \qquad (1)$$

Λόγω της Αστάθειας και της δυναμικότητας του χώρου σε συνδυασμό με την πολυπλοκότητα και τη διασπορά του πληροφοριακού συστήματος ένα πλήθος καταστροφικών δυνάμεων είναι ικανό να λάβει

χώρα μέσα στο χρόνο Δt. Ας υποθέσουμε ότι C είναι το σύνολο των καταστροφικών δυνάμεων τότε ισχύει:

$$C= \{c_1, c_2, \ldots, c_m\} \quad , \quad C^*=\{1,2,\ldots,m\} \quad (2)$$

Οι καταστροφικές δυνάμεις που μπορούν να ενεργήσουν μέσα σε χρόνο Δt είναι απρόβλεπτες. Πολλές από αυτές μπορούν αν όχι να προβλεφθούν τουλάχιστον, όμως, έστω και εμπειρικά να επισημανθούν. Μια σωστή και οργανωμένη ανάλυση του συστήματος και του τρόπου λειτουργίας του μπορεί να επισημάνει τις καταστροφικές δυνάμεις που βρίσκονται εν υπνώσει (απειλές), αλλά μπορούν να ενεργοποιηθούν. Οι εν υπνώσει καταστροφικές δυνάμεις για το σύστημα είναι οι Απειλές που οφείλουν να κρατούν σε εγρήγορση τον διαχειριστή του συστήματος, προκειμένου να αποφευχθούν καταστροφικά φαινόμενα. Ας υποθέσουμε ότι T είναι το σύνολο των απειλών που επηρεάζουν τα υποσυστήματα του συστήματος S.

$$T = \{t_1, t_2, \ldots, t_k\} \qquad (3)$$

Για να μπορέσει να αντιμετωπίσει τις απειλές το σύστημα χρειάζεται ένα σύνολο από ικανά αντίμετρα. Ο βαθμός επιτυχίας συναρτάται από την ποσότητα των αντίμετρων που χρησιμοποιούνται αλλά και από την ποιότητα τους. Ας υποθέσουμε ότι CM είναι το σύνολο των αντίμετρων που υπάρχουν στο σύστημα S.

$$CM = \{cm_1, cm_2, \ldots, cm_\lambda\} \qquad CM^* = \{1,2,\ldots,\lambda\} (4)$$

Δυστυχώς οι απειλές δε μένουν απειλές. Υπάρχουν πολλές περιπτώσεις που μια απειλή δεν αντιμετωπίζεται επιτυχώς. Καταστροφές εμφανίζονται με άσχημες συνέπειες για την ομαλή λειτουργία του συστήματος. Ας θεωρήσουμε C το σύνολο των καταστροφών που ενεργοποιούνται στο χρόνο Δt.

Ένα ολοκληρωμένο σύστημα διαχείρισης κρίσεων πρέπει να παρέχει όλα τα εχέγγυα, προκειμένου να αντιμετωπιστεί ένα καταστροφικό φαινόμενο. Είναι επιτακτική η ανάγκη εφαρμογής κατάλληλων διαδικασιών ανάκτησης των κατεστραμμένων δεδομένων, ώστε να εξασφαλισθεί η ομαλή λειτουργία του συστήματος. Ας υποθέσουμε ότι R είναι το σύνολο των αντίμετρων που απαιτούνται για την ανάκτηση των δεδομένων του συστήματος S, που καταστράφηκαν από κάποια δύναμη D.

$$R= \{r_1 , r_2 ,...,r_j ,...r_\theta\} , R^*=\{1,2,...,\theta\} \qquad (6)$$

Από τους παραπάνω ορισμούς εύκολα εξάγεται το συμπέρασμα ότι τα σύνολα T και CM είναι εξ ορισμού ασυμβίβαστα, όπως το ίδιο ισχύει και για τα σύνολα R και D. Παράλληλα, τα σύνολα CM και R δεν είναι ασυμβίβαστα, αφού υπάρχει σχέση μεταξύ τους καθώς ένα αντίμετρο μπορεί να γίνει και μέτρο αντιμετώπισης ενός καταστροφικού φαινομένου. Όμοια, και τα σύνολα T και D έχουν κοινά σημεία μεταξύ τους, αφού μια απειλή μπορεί να μετεξελιχθεί σε καταστροφικό φαινόμενο.

Η γενική θεωρία διαχείρισης κινδύνου στηρίζεται σε ένα σύνολο παραδοχών, που απαιτούνται, προκείμενου να προσδιοριστούν τα σύνολα τα οποία αποτελούν τα θεμέλια της θεωρίας. Είναι οι δεξαμενές από όπου αντλούνται δεδομένα και υλοποιούνται οι εφαρμογές της θεωρίας. Ειδικότερα:

- Μια απειλή μπορεί να αντιμετωπιστεί από ένα ή περισσότερα είδη αντίμετρων ή ένα σύνολο από αντίμετρα.
- Ένα αντίμετρο μπορεί να είναι λύση για μια ή περισσότερες απειλές του συστήματος.
- Μια απειλή μπορεί να έχει σαν αποτέλεσμα μια καταστροφή.
- Μια καταστροφή μπορεί να αντιμετωπίζεται με περισσότερες από μια τεχνικές ανάκτησης
- Μια τεχνική ανάκτησης μπορεί να απαντά σε περισσότερες από μια καταστροφές.

3.3.4 ΠΙΘΑΝΑ ΚΕΝΤΡΙΚΑ ΣΗΜΕΙΑ ΑΥΤΟΚΑΤΑΡΡΕΥΣΗΣ

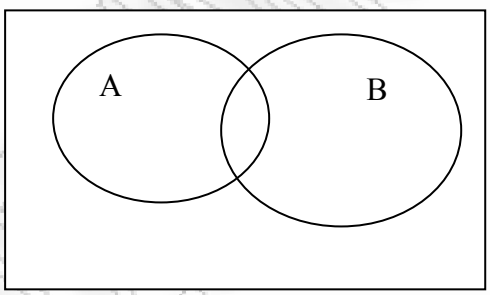
Σε αυτήν την παράγραφο, θα προσπαθήσουμε να εντοπίσουμε τα πιθανά σημεία αυτοκατάρρευσης ενός ολοκληρωμένου ψηφιακού πληροφοριακού συστήματος. Στόχος της γενικής θεωρίας συστημάτων είναι η πρόληψη και η αποφυγή εμφάνισης τέτοιων φαινομένων που ίσως οδηγήσουν και στην ολοκληρωτική καταστροφή του συστήματος.

Ένα ολοκληρωμένο Πληροφοριακό Σύστημα για να μπορέσει να αντεπεξέλθει σε κρίσιμες καταστάσεις οφείλει να μπορεί να εξελίσσεται και να μεταβάλλεται διαχρονικά, ενώ παράλληλα θα πρέπει να έχει τη δυνατότητα να χειρίζεται ασταθή δυναμικά δεδομένα. Είναι, λοιπόν, λογικό, εξαιτίας της συνεχούς μεταβλητότητας των δεδομένων, το σύστημα να πρέπει σε τακτά χρονικά διαστήματα να έχει τη δυνατότητα

του συνεχούς ελέγχου και της συντήρησης, προκειμένου να μπορεί να χειριστεί μεταβαλλόμενες καταστάσεις. Κατά συνέπεια, θα μπορέσει είτε από μόνο του, είτε με εξωτερική καθοδήγηση (διαχειριστής συστήματος) να αποφύγει πιθανές **Πτώσεις** και να καλύψει ατέλειές του.[Violanti 2000]

Στη συνέχεια, θα προσπαθήσουμε να εντοπίσουμε αυτές τις ατέλειες του συστήματος, παρουσιάζοντας μια πρώτη εκτίμηση για τον εντοπισμό κεντρικών σημείων αυτοκατάρρευσης, ενώ παράλληλα θα μελετήσουμε και άλλες περιπτώσεις κατά τις οποίες το σύστημα δεν καταρρέει ολοκληρωτικά, αλλά απλώς μεταλλάσσεται, συνεχίζοντας τη λειτουργία του με ορισμένες διαφορετικές δομές από πριν. [Wood 1996]

Δε βλάπτει τη γενικότητα να θεωρήσουμε, ένα σύστημα το οποίο αρχικά αποτελείται από δύο υποσυστήματα Α και Β που στη γενική τους μορφή παρουσιάζουν **δεσμούς εξάρτησης**. Γραφικά, θα μπορούσε κάτι τέτοιο να παρασταθεί με το παρακάτω διάγραμμα:

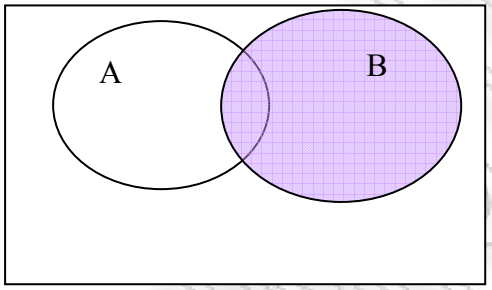


Σχήμα 3.6
Δυο αλληλοεξαρτώμενα υποσυστήματα του συστήματος S.

Θεωρούμε σαν σύνολο Α το σύνολο των στοιχείων που απαιτείται να λειτουργούν ορθά, για να λειτουργήσει πλήρως και σωστά το υποσύστημα Α. Ομοίως, Β το σύνολο των στοιχείων που απαιτείται να λειτουργούν ορθά, για να λειτουργήσει πλήρως και σωστά το υποσύστημα Β. Η τομή στο Σχήμα 3.6 αντιπροσωπεύει τους δεσμούς εξάρτησης των δύο υποσυστημάτων. Τα σημεία της τομής των δύο αυτών υποσυστημάτων είναι τα κοινά τους στοιχεία. Παράλληλα, η τομή παρουσιάζει τα στοιχεία εκείνα που είναι **αναγκαία** για να μπορέσουν ταυτόχρονα να λειτουργήσουν ομαλά τα δύο αλληλεξαρτώμενα υποσυστήματα. Ένα άλλο χαρακτηριστικό των στοιχείων της τομής είναι το γεγονός ότι τα στοιχεία αυτά πρέπει να καλύπτονται, για να είναι σε θέση να λειτουργεί το γενικότερο σύστημα σε μια υποτυπώδη κατάσταση.

3.3.4.1. ΚΑΤΑΡΡΕΥΣΗ ΤΩΝ ΣΤΟΙΧΕΙΩΝ ΤΟΥ ΣΥΝΟΛΟΥ Α –Β

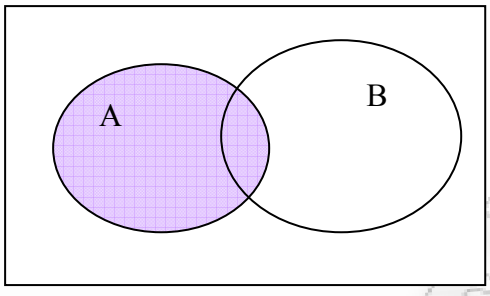
Στην περίπτωση Α-Β, όπως γίνεται αντιληπτό, το αρχικό μας σύστημα μετά την κατάρρευση αποτελείται από τα στοιχεία που βρίσκονται στο σύνολο Β και τα στοιχεία της τομής. Αποτέλεσμα εδώ **είναι η μερική Κατάρρευση του συστήματος**, και η μετάλλαξή του σε ένα νέο. Το μεταλλαγμένο νέο σύστημα αποτελείται από το υποσύστημα Β το οποίο συνεχίζει την ομαλή λειτουργία του, ενώ το υποσύστημα Α έχει εκφυλιστεί και λειτουργεί μόνο στα στοιχεία της τομής. Τέλος, παρατηρούμε ότι οι δεσμοί εξάρτησης παραμένουν σταθεροί, όπως φαίνεται στο παρακάτω σχήμα (Σχήμα 3.7):



Σχήμα 3.7
Κατάρρευση των στοιχείων του συνόλου Α –Β

3.3.4.2 ΚΑΤΑΡΡΕΥΣΗ ΤΩΝ ΣΤΟΙΧΕΙΩΝ ΤΟΥ ΣΥΝΟΛΟΥ Β-Α

Αντίθετα με την προηγούμενη περίπτωση (Α1), στην παρούσα περίπτωση Β-Α (Α2), το αρχικό μας σύστημα αποτελείται από τα στοιχεία που βρίσκονται στο σύνολο Α και τα στοιχεία της τομής. Αποτέλεσμα και εδώ **δεν είναι η ολοκληρωτική Κατάρρευση του συστήματος**, αλλά η μετάλλαξή του σε ένα νέο σύστημα. Το μεταλλαγμένο νέο σύστημα αποτελείται από το υποσύστημα Α το οποίο συνεχίζει την ομαλή λειτουργία του, ενώ το υποσύστημα Β, όπως και στην παραπάνω περίπτωση έχει εκφυλιστεί και λειτουργεί μόνο στοιχειωδώς. Τέλος, παρατηρούμε ότι οι δεσμοί εξάρτησης παραμένουν σταθεροί και σε αυτήν την κατάσταση. Η περίπτωση αυτή θεωρείται **δυϊκή** περίπτωση της Α1. Το Σχήμα 3.8 παρουσιάζει αυτή την περίπτωση.



Σχήμα 3.8
Κατάρρευση των στοιχείων του συνόλου B –A

3.3.4.3 ΚΑΤΑΡΡΕΥΣΗ ΤΩΝ ΣΤΟΙΧΕΙΩΝ ΤΟΥ ΣΥΝΟΛΟΥ (A-B)∪(B-A)

Στην περίπτωση αυτή, τα στοιχεία που αποτελούν τους δεσμούς εξάρτησης λειτουργούν ομαλά. Τόσο τα στοιχεία του υποσυστήματος Α όσο και αυτά του αντίστοιχου υποσυστήματος Β έχουν καταρρεύσει. Αποτέλεσμα αυτής της κατάστασης είναι η λειτουργία του συστήματος σε μια υποτυπώδη μορφή. Τα συνδετικά στοιχεία των δυο συνόλων είναι αυτά που συνεχίζουν να λειτουργούν μετά από την εμφάνιση της κρίσης. Συγκεκριμένα, το σύστημα **εκφυλίστηκε και συρρικνώθηκε, καλύπτοντας μόνο τις βασικές ζωτικές λειτουργίες για την συντήρησή του.** Αυτό συμβαίνει διότι τα στοιχεία της τομής των δύο συνόλων είναι **ικανά** να θέσουν σε λειτουργία το σύστημα στην **πρωτογενή** αυτή μορφή. Σε μια τέτοια περίπτωση είναι δυνατή η Ανάκτηση.[Πετραντωνάκης (2003)]

3.3.4.4 ΚΑΤΑΡΡΕΥΣΗ ΔΕΣΜΩΝ ΕΞΑΡΤΗΣΗΣ

Στην περίπτωση κατάρρευσης, έστω και ενός στοιχείου της τομής, που αποτελεί δεσμό εξάρτησης, **καταρρέει αυτόματα** όλο το σύστημα, ανεξάρτητα από την επιμέρους λειτουργία των στοιχείων των υποσυστημάτων Α, Β. Τα στοιχεία των δεσμών εξάρτησης είναι **αναγκαία,** προκειμένου να λειτουργήσει ομαλά το σύστημα ή στη χειρότερη περίπτωση (όπου το σύνολο των εκτός τομής στοιχείων των δύο υποσυστημάτων δε λειτουργεί) να λειτουργήσει σε υποτυπώδη κατάσταση. Συνεπώς, δημιουργούνται δυο αυτόνομα συστήματα τα οποία δεν έχουν κοινό δίαυλο επικοινωνίας. Το κάθε υποσύστημα στην ουσία δρα σαν νέα αυτόνομη οντότητα χωρίς να έχει τη δυνατότητα επεξεργασίας δεδομένων το ένα από το άλλο. Κατόπιν τούτου και χωρίς βλάβη της γενικότητας, συμπεραίνουμε ότι τα **στοιχεία των δεσμών εξάρτησης** αποτελούν **Κεντρικά Σημεία Αυτοκατάρρευσης** για

οποιοδήποτε τύπο Πληροφοριακού Συστήματος, ανεξάρτητα της πολυπλοκότητας του.

3.3.5 ΕΝΤΟΠΙΣΜΟΣ ΠΙΘΑΝΩΝ ΣΗΜΕΙΩΝ ΑΥΤΟΚΑΤΑΡΡΕΥΣΗΣ

Γενικεύοντας αυτά που αναλύθηκαν, είναι φανερό ότι ισχύουν και στην περίπτωση των πραγματικών ψηφιακών πληροφοριακών συστημάτων, τα οποία βέβαια αποτελούνται από περισσότερα από δύο υποσυστήματα. Έστω S το σύνολο των υποσυστημάτων του χώρου μας.(1)

Όπως ήδη αναλύθηκε, σημαντικό βήμα για την αποφυγή κατάρρευσης του συστήματος είναι ο εντοπισμός των αδυναμιών του, όπου εξωτερικοί συνήθως παράγοντες επιδρούν σε αυτές και οδηγούν είτε σε ολοκληρωτική καταστροφή του Πληροφοριακού Συστήματος είτε στην εκφυλιστική μετάλλαξη του. Βέβαια, η όλη μελέτη δεν αποσκοπεί στην εύρεση άριστων ή καλών λύσεων, που να ισχύουν για πάντα, αλλά το πρόβλημα μας επικεντρώνεται μέσα σε ένα συγκεκριμένο **ορίζοντα προγραμματισμού**. Για παράδειγμα, ο σχηματισμός των ψηφιακών στρατιωτών και του πληροφοριακού τους συστήματος μας ενδιαφέρει προφανώς για το χρονικό διάστημα της συγκεκριμένης αποστολής τους. Βέβαια, δε γνωρίζουμε πως το πληροφοριακό μας σύστημα θα γίνει, μεταβαλλόμενο σε ένα ασταθές δυναμικό περιβάλλον μέσα στον ορίζοντα προγραμματισμού, αλλά είναι δυνατό να γνωρίζουμε ποια είναι εκείνα τα οποία μπορούν να επιδράσουν ενεργοποιούμενα μέσα στον ορίζοντα προγραμματισμού πάνω στο σύστημα S, αλλάζοντας τη δομή και τη λειτουργικότητα τους. Έστω C το σύνολο των καταστροφικών δυνάμεων (παραγόντων) που είναι πιθανό να εμφανιστούν (ενεργοποιηθούν) μέσα στον ορίζοντα προγραμματισμού μας.(2)

Μια μέθοδος που οδηγεί στον εντοπισμού των πιθανών σημείων αυτοκατάρρευσης, ολοκληρωτικών ή και μεταλλακτικών, είναι η χρήση του **Χάρτη Κινδύνου.** Ο χάρτης αυτός καταγράφει την κατάσταση κινδύνου του πληροφοριακού μας συστήματος και μπορεί να εκφραστεί με τον ακόλουθο πίνακα:

X= (x_{ij}) , $\forall i \in S^*$ **,** $j \in C^*$ **,** όπου:
x_{ij}=1, αν η ενεργοποίηση του c_j αλλάζει τα δεδομένα του s_i.
x_{ij}=0, διαφορετικά.

Ο χάρτης Κινδύνου είναι ένα πρωτογενές απαραίτητο εργαλείο, γιατί αντικατοπτρίζει την κατάσταση που αντιμετωπίζει το

πληροφοριακό σύστημα μέσα στο ασταθές, δυναμικό περιβάλλον λειτουργίας. Πάνω στο χάρτη κινδύνου αποτυπώνονται όλοι οι κίνδυνοι που αντιμετωπίζει το πληροφοριακό σύστημα, χωρίς αποκλεισμούς και εξαιρέσεις. Επίσης, παρουσιάζει πιθανές αιτίες κρίσης, που απειλούν το σύστημα θεωρώντας όλους τους κινδύνους σαν εξίσου δυνατά καταστροφικά φαινόμενα. Επομένως, δεν παραθέτει κάποια κρίση ή ένα σύνολο καταστροφών, ακόμα και αν ένα τέτοιο καταστροφικό φαινόμενο είναι πολύ πιθανό να συμβεί. Συνήθως, ενεργοποιούνται ορισμένες καταστάσεις μέσα στον ορίζοντα προγραμματισμού. Έστω ένα υποσύνολο C^0 των υλοποιήσιμων καταστροφικών δυνάμεων που πραγματοποιούνται στο διάστημα Δt. Η Γενική Θεωρία Διαχείρισης Κινδύνου στηρίζεται σε τρία θεμελιώδη θεωρήματα που αποτελούν τη βάση της λειτουργίας και ύπαρξης της. Τα θεωρήματα αυτά εξηγούν με απλό και αποτελεσματικό τρόπο τις καταστάσεις στις οποίες μπορεί να βρεθεί ένα σύστημα κατά τη διάρκεια της λειτουργίας του μέσα σε ένα ασταθές, δυναμικό περιβάλλον. Οι καταστάσεις στις οποίες μπορεί ένα σύστημα S να βρεθεί μέσα στο χρονικό διάστημα Δt κατά την αλληλεπίδραση του με το εξωτερικό, εχθρικό περιβάλλον είναι τρεις και περιγράφουν όλες τις πιθανές περιπτώσεις που μπορούν να εμφανιστούν. Οι τρεις αυτές περιπτώσεις είναι ανεξάρτητες από το εύρος των καταστροφικών δυνάμεων, τη σφοδρότητα και την πιθανότητα εμφάνισης τους. Αξιοσημείωτο είναι ότι εμφανίζονται οποτεδήποτε, οπουδήποτε και με οποιοδήποτε σύνολο δυνάμεων, που υλοποιείται μέσα στο χρονικό διάστημα Δt. Έτσι, έχουμε:

Θεώρημα Ολοκληρωτικής Καταστροφής (Black Operational Hole Theorem)

Αν ισχύει:

$$(\forall i \in S^*)\ (\sum_j \mathbf{x_{ij}} >= 1\ ,\ \forall j \in C^* : \mathbf{c_j} \in C^0) \qquad (7)$$

τότε το σύστημα S καταρρέει ολοκληρωτικά.

Η απόδειξη της παραπάνω πρότασης είναι απλή και αυτονόητη. Αρκεί να φανταστεί κανείς τη δομή που θα έχει ένας τέτοιος Χάρτης Κινδύνου. Μέσα στον ορίζοντα χρονοπρογραμματισμού του συστήματος S υπάρχει η δυνατότητα μια καταστροφική δύναμη να επηρεάσει ταυτόχρονα όλα τα υποσυστήματα του υπό μελέτη συστήματος. Αυτό, όπως γίνεται αντιληπτό, θα έχει ως άμεσο αποτέλεσμα την ολοκληρωτική κατάρρευση του συστήματος S, με συνέπεια να τεθεί σε κατάσταση **Μαύρης Λειτουργικής Οπής (Black Operational Hole).** Έτσι, καταρρέουν όλοι οι δεσμοί συνεκτικότητας μεταξύ των υποσυστημάτων

καθώς και η λειτουργία αυτών, όπως παρουσιάσθηκε παραπάνω. Επιπλέον, μια κατάσταση μαύρης λειτουργικής οπής μπορεί να εμφανιστεί ακόμα και στην περίπτωση που ένα σύνολο δυνάμεων επιδράσει πάνω στα υποσυστήματα του συστήματος S, επηρεάζοντας το σύνολο αυτών. Το θεώρημα προσδιορίζει τα κεντρικά σημεία αυτοκατάρρευσης του συστήματος που εμφανίζονται μέσα στο διάστημα Δt, όπως αυτά αναλύθηκαν σε προηγούμενη παράγραφο.

Θεώρημα Λειτουργικής Ηρεμίας (White Operational Hole Theorem)

Αν ισχύει:

$$(\forall i \in S^{*})(\sum_{j} \mathbf{x_{ij}=0}\ ,\ \forall j \in C^{*} : \mathbf{c_j} \in C^{0}) \qquad (8)$$

τότε το σύστημα S βρίσκεται σε λειτουργική ηρεμία.

Όπως στο προηγούμενο θεώρημα έτσι και εδώ η απόδειξη γίνεται εύκολα αντιληπτή, καθώς όταν μια καταστροφική δύναμη C^0 δεν επηρεάζει κανένα από τα υποσυστήματα του συστήματος S, τότε το σύστημα λειτουργεί ομαλά και βρίσκεται σε **Λευκή Λειτουργική Οπή (White Operational Hole).** Με αυτό τον τρόπο και όλα τα υποσυστήματα λειτουργούν ομαλά, μένοντας ανεπηρέαστα από την αστάθεια του χώρου δεδομένων.

Μεταλλακτικό Θεώρημα (Gray Operational Hole Theorem)

Αν ισχύει:

$$(\exists k \in S^{*})(\sum_{j} \mathbf{x_{kj}=0})\ \forall j \in C^{*} : \mathbf{c_j} \in C^{0}) \ (9)$$

τότε το σύστημα S μεταλλάσσεται χωρίς αναγκαστικά να καταστραφεί.

Η κατάσταση γίνεται πιο πολύπλοκη στο τρίτο θεμελιώδες θεώρημα, που περιγράφεται με την παραπάνω πρόταση. Το σύστημα τίθεται σε κατάσταση **Γκρίζας Λειτουργικής Οπής (Gray Operational Hole).** Είναι φανερό πως η περίπτωση της Λευκής Οπής είναι μια γενικευμένη περίπτωση της Γκρίζας**.** Η κατάσταση αυτή οδηγεί το σύστημα σε απροσδιόριστες μορφές και δεν είναι σίγουρο αν το σύστημα θα συνεχίσει να λειτουργεί μετά από μια τέτοια κατάσταση. Δυο είναι τα πιθανά μονοπάτια που θα ακολουθήσει. Από τη μια πλευρά, σε καταστάσεις γκρίζας οπής η κατάληξη του συστήματος, μετά την εμφάνιση της κρίσης, εξαρτάται από τους κανόνες που διέπουν την ιεραρχία των δομών του συστήματος και των υποσυστημάτων τους. Έτσι,

λοιπόν, σύμφωνα με το μεταλλακτικό θεώρημα η μερική κατάρρευση των υποσυστημάτων του κυρίως συστήματος οδηγεί στη μετάλλαξη του και στη δημιουργία μιας νέας μορφής υποδεέστερης της αρχικής ή σε σπάνιες περιπτώσεις καλύτερης της αρχικής μορφής του S. Από την άλλη πλευρά, η μερική κατάρρευση των υποσυστημάτων του S μπορεί να οδηγήσει και σε ολική κατάρρευση του συστήματος. Αυτό οφείλεται στο γεγονός ότι τα υποσυστήματα που κατέρρευσαν είναι αναγκαία για τη λειτουργία του S. [Petrantonakis (2006)]

Είναι, λοιπόν, φανερό ότι τα κοινά σημεία των υποσυστημάτων μπορούν να επηρεαστούν τελικά ακόμα και από μια μόνο καταστροφική κατάσταση μέσα στον ορίζοντα προγραμματισμού, οπότε θα οδηγηθούμε σε ολοκληρωτικές καταρρεύσεις. Πρέπει, επομένως, να ενισχυθούν τα σημεία αυτά με το χωρισμό ορισμένων υποσυστήματων του χώρου μας σε περισσότερα, με κύριο στόχο την ελαχιστοποίηση των κοινών σημείων τους. Βέβαια, το να μεταλλάσσει κάποιος το πληροφοριακό σύστημα για να γλιτώσει μια μελλοντική κατάρρευση είναι απλά μια προσωρινή λύση για να δοθεί χρόνος Ανάκτησης από την Πτώση, σύμφωνα με τη γενικευμένη έννοια του Recovery.

3.3.6 ΜΕΘΟΔΟΛΟΓΙΑ ΠΡΟΛΗΨΗΣ- ΑΝΤΙΜΕΤΩΠΙΣΗΣ ΚΙΝΔΥΝΙΚΩΝ ΦΑΙΝΟΜΕΝΩΝ

3.3.6.1 ΕΙΣΑΓΩΓΗ

Η γενική θεωρία διαχείρισης κινδύνου όπως αναλύθηκε αποσκοπεί στη δημιουργία ικανών συνθηκών για τη διαχείριση κρίσεων σε ψηφιακά πληροφοριακά συστήματα τα οποία λειτουργούν σε Ασταθή, Δυναμικά μεταβαλλόμενα Περιβάλλοντα λειτουργίας. Η Μεθοδολογία Πρόληψης – Αντιμετώπισης Κινδυνικών Φαινομένων είναι ένα προκαταρτικό στάδιο, προκειμένου να αποφευχθούν δυσάρεστες καταστάσεις. Η Αστάθεια του χώρου σε συνδυασμό με τη μη ομαλή δυναμικότητα του μπορούν να προκαλέσουν απρόβλεπτα κινδυνικά φαινόμενα. Η άτακτη μεταβλητότητα του **δεν μπορεί να προβλεφθεί με τη χρήση συμβατικών πιθανο-θεωρητικών μοντέλων.** Στην παρούσα μεθοδολογία ισχύει το εξής απλό θεμελιώδες αξίωμα:

«Ο κίνδυνος δεν είναι πιθανότητα, είναι βεβαιότητα.»

Το παραπάνω αξίωμα εξηγεί με απλά λόγια ότι σε ένα μη ομαλό δυναμικό χώρο δεδομένων ένα κινδυνικό φαινόμενο θα εμφανιστεί χωρίς

να γνωρίζουμε το πότε. Το μοντέλο δε στηρίζεται σε εικασίες, αλλά θεωρεί δεδομένο a priori ότι όλοι οι κίνδυνοι που απειλούν το σύστημα μπορούν να συμβούν. Έτσι, η βέλτιστη λύση για ένα πληροφοριακό σύστημα, που λειτουργεί σε ένα τέτοιο περιβάλλον, είναι η πρόληψη όλων των πιθανών καταστροφικών δυνάμεων που μπορεί να εμφανιστούν, ακόμα και αν αυτές οι δυνάμεις θεωρούνται από την κλασσική θεωρία Κινδύνου απίθανο να υλοποιηθούν.

3.3.6.2 ΜΟΝΤΕΛΟΠΟΙΗΣΗ

Ο πρώτος από τους δύο σημαντικούς κλάδους ενός ολοκληρωμένου και επιτυχημένου ψηφιακού συστήματος αποκλιμάκωσης κρίσεων είναι η πρόληψη. Είναι εύκολα αντιληπτό ότι η πρόληψη είναι βαρύνουσας σημασίας, αφού με τη βοήθειά της όχι μόνο αποφεύγεται η κρίση, αλλά το σύστημα καταφέρνει να αποφύγει την ενεργοποίηση καταστροφικών δυνάμεων που θα συντελέσουν σε μερική ή και σε ολοκληρωτική κατάρρευσή του. Επιπρόσθετα, η πρόληψη είναι ο παράγοντας εκείνος που καθορίζει πόσο ‘έξυπνο’ είναι το σύστημα αποκλιμάκωσης. Ένα «έξυπνο» σύστημα στηρίζεται γενικά στην υπάρχουσα γνώση και την εμπειρία. Για να αποκτηθεί, λοιπόν, αυτή η γνώση και η εμπειρία δεν πρέπει να παραβλέψουμε το γεγονός ότι το έξυπνο σύστημα πρόληψης κρίσεων στηρίζεται σε διαδικασίες, όπως το μοντέλο των ερωτήσεων - απαντήσεων και της συνεχούς ανατροφοδότησης και αυτοεκπαίδευσης. Ο ρόλος του είναι καθαρά συμβουλευτικός, με μειωμένη την ικανότητα αυτόματων πρωτοβουλιών. Άρα, το μοντέλο απαιτεί την ύπαρξη ενός διαχειριστή - αναλυτή των απειλών έτσι ώστε να μπορεί με τη σειρά του να προσδιορίσει τους κανόνες αποκλιμάκωσης.

Το βασικό μειονέκτημα που παρουσιάζει ένα τέτοιο σύστημα είναι το γεγονός ότι στηρίζεται σε ένα πλήθος κανόνων και προτάσεων που πολλές φορές δύσκολα καθορίζονται, εξαιτίας της αλλαγής της δομής του χώρου λειτουργίας του. Τα κριτήρια, λοιπόν, ποικίλουν και χρησιμοποιούνται από τον αναλυτή ανάλογα με το βάρος επικινδυνότητάς τους. Το ρίσκο σε ένα τέτοιο σύστημα είναι μεγάλο, εφόσον πρέπει να προβλέπει κρίσεις που οφείλονται στην αστάθεια του χώρου δεδομένων, αλλά και να έχει την ικανότητα αυτόματης επέμβασης αποκλιμάκωσης της κρίσης, με κύριο στόχο την αποφυγή της κατάρρευσης του Συστήματος.

Επομένως, η πρόληψη παραμένει το πρώτο σημαντικό κομμάτι, απόκρουσης κινδυνικών καταστάσεων, με άμεσο στόχο την ομαλή

λειτουργία του συστήματος και την αποφυγή συμπτωμάτων Ολικής ή Μερικής Κατάρρευσής του. Ειδικότερα, επιτυγχάνεται η αποφυγή και πρόληψη Κρίσεων εν τη γενέσει τους, ενώ **διατηρείται η Επιχειρησιακή Ικανότητα της Λειτουργικής Ισορροπίας του συστήματος.** Το θεμελιώδες αξίωμα, που παρουσιάσθηκε σε προηγούμενη παράγραφο, πλαισιώνεται από δυο άλλα σημαντικά αξιώματα τα οποία στηρίζουν όλο το εννοιολογικό δομικό μοντέλο της θεωρίας. Το δεύτερο σημαντικό αξίωμα είναι το ***Αξίωμα του Ελαχίστου Επιπέδου Λειτουργικότητας (Minimum Level of Functionality):***

«Ένα σύστημα λειτουργεί ομαλά, όταν λειτουργεί με τον ελάχιστο αριθμό υποσυστημάτων (modules), που απαιτούνται, προκειμένου να διατηρηθεί αυτή η ομαλότητα.»

Σκοπός της πρόληψης δεν είναι μόνο η αποφυγή μιας κρίσης. Παράλληλα, απαιτείται η προστασία των υποσυστημάτων του κυρίως συστήματος έτσι, ώστε ακόμα και στην περίπτωση κρίσης να παραμείνει ανεπηρέαστος ένας ικανός αριθμός αυτών, για να συνεχίσουν την απρόσκοπτη λειτουργία του συστήματος.

Η πρόληψη και αποφυγή κρίσεων είναι εφικτή με τη χρήση αντιμέτρων. Ένα σύστημα για να μπορεί να λειτουργεί ομαλά σε ένα εχθρικό περιβάλλον οφείλει να είναι προετοιμασμένο να αντιμετωπίσει οποιαδήποτε καταστροφική δύναμη εμφανιστεί. Αυτή η προετοιμασία απαιτεί προσοχή καθώς και επίπονη δουλειά. Έτσι, καταλήγουμε στο επόμενο θεμελιώδες αξίωμα **(Maximum Covering Functionality):**

«Τα αντίμετρα πρέπει να είναι επαρκή τόσο σε ποσότητα όσο και σε ποιότητα.»

Η δυναμικότητα του χώρου είναι ο βασικός παράγοντας εμφάνισης καταστροφικών δυνάμεων στο σύστημα. Παράλληλα, συντελεί στην εμφάνιση και ανάπτυξη μιας πληθώρας απειλών. Ο αριθμός τους αυξάνει με γοργούς ρυθμούς, ενώ τα αντίμετρα δεν εγγυώνται απόλυτη επιτυχία, επειδή υπάρχουν πάντα περιπτώσεις που αποτυγχάνουν και επιτρέπουν την εμφάνιση καταστροφικών φαινομένων. Απορρέει, επομένως, εύλογα το συμπέρασμα ότι απώτερος σκοπός της μεθοδολογίας είναι το σύστημα να παραμείνει σε **Κατάσταση Λειτουργικής Ομαλότητας (State of Operational Normality)**. Η κατάσταση επιτυγχάνεται όσο ο αριθμός των προτεινομένων αντίμετρων ασφαλείας καλύπτει όλες τις δυνατές περιπτώσεις κρίσεων. Αυτό αποτελεί και τη βασική αρχή της προληπτικής θεωρίας. Ένα πολύ χρήσιμο εργαλείο που βοηθάει στην

αποφυγή δυσμενών καταστάσεων με την εμφάνιση Μαύρων Λειτουργικών Οπών είναι ο **Χάρτης Αντιμετώπισης (Confrontation Map).**

K= (k_{ij}), $\forall i \in T^*,\ j \in CM^*$

κ_{ij}=1, αν η ενεργοποίηση του αντίμετρου cm_j αντιμετωπίζει την απειλή t_i.

κ_{ij}=0, διαφορετικά.

Με τη βοήθεια του χάρτη εντοπίζονται όλα τα τρωτά σημεία του συστήματος δίνοντας τη δυνατότητα στον διαχειριστή του συστήματος να προλάβει μια κρίση που εμφανίζεται, όταν σχηματιστεί μια κρίση, όπου το σύστημα καταρρέει και αυτοκαταστρέφεται. Ο χάρτης αντιμετώπισης αποτελεί το υπόβαθρο πάνω στο οποίο οικοδομούνται στη συνέχεια δυο **Βασικές Αρχές** της Μεθοδολογίας Πρόληψης – Αντιμετώπισης Κινδυνικών Φαινομένων. Η πρώτη αρχή είναι η ακόλουθη:

Πρόταση Αντιμετώπισης Απειλών(Threat Confrontation Proposition)

Αν ισχύει:

$$((\forall i) \in \mathbf{T}^*)\ (\sum_j \kappa_{ij} >= 1,\ j \in \mathbf{CM}^*)$$

τότε το σύστημα S αντιμετωπίζει την απειλή T.

Η ισχύς της πρότασης είναι σαφής. Όταν για κάθε απειλή (threat) υπάρχει ένα τουλάχιστον αντίμετρο, τότε το σύστημα **έχει την επιχειρησιακή δυνατότητα να προλάβει την υλοποίηση μιας κρίσης** και της μετεξέλιξης της σε καταστροφικό φαινόμενο. Η ιδανική περίπτωση φυσικά είναι όταν το σύστημα διαθέτει δυο τουλάχιστον αντίμετρα για κάθε απειλή ή σύνολο απειλών, που το θέτουν σε κίνδυνο.

Γενικότερα, οι απειλές δεν παραμένουν σταθερές μέσα στον ορίζοντα χρονοπρογραμματισμού Δt του συστήματος. Συνεχώς μεταλλάσσονται με ταχύτατους ρυθμούς. Η μετάλλαξη τους αποδεικνύει τη δυναμικότητα του περιβάλλοντος λειτουργίας παράλληλα, όμως, ενισχύει την αστάθεια του χώρου, αυξάνοντας σημαντικά την επικινδυνότητα του. Η αλλαγή αυτή δημιουργεί νέους τύπους απειλών που δύσκολα κατηγοριοποιούνται εξαιτίας της πολυπλοκότητας τους. Αυτό έχει σαν αποτέλεσμα τα αντίμετρα να μην μπορούν να αντιμετωπίσουν τη συγκεκριμένη απειλή κατά συνέπεια εμφανίζονται απειλές οι οποίες δεν έχουν αντίμετρο να τις αναχαιτίσει. Η επόμενη

πρόταση εξηγεί τέτοιες καταστάσεις που μέσα από το χάρτη αντιμετώπισης προσδιορίζονται όλα τα τρωτά σημεία ενός συστήματος, όπως για παράδειγμα τα κενά ασφαλείας και οι μικρές ή μεγάλες Κερκόπορτες, που πρέπει να κλείσουν, ώστε το σύστημα να βρίσκεται σε Κατάσταση Λειτουργικής Ομαλότητας.

Πρόταση Τρωτότητας (Vulnerability Proposition)

Αν ισχύει,

$$(\exists i \in \mathbf{T}^*)(\sum_j \kappa_{ij} = 0\ ,\ j \in \mathbf{CM}^*)$$

τότε το Σύστημα S είναι ευάλωτο στην απειλή T.

Η Πρόταση Τρωτότητας χαρτογραφεί όλα τα κενά που έχει το σύστημα. Εμφανίζει τα απευκταία σενάρια και τις τρύπες που πρέπει να καλυφθούν άμεσα, πριν μετατραπούν σε Μαύρες Λειτουργικές Οπές, μέσω των οποίων το σύστημα οδηγηθεί στην ολοκληρωτική κατάρρευση του. Επιπλέον, για να είναι αποτελεσματικά τα αντίμετρα του συστήματος πρέπει να παρακολουθούν τις αλλαγές που συντελούνται μέσα στο περιβάλλον λειτουργίας και να έχουν την ικανότητα να προσαρμόζονται στις απαιτήσεις του περιβάλλοντος, αλλάζοντας και εξελίσσοντας συνεχώς τη δομή τους. Το απόσταγμα των δυο προαναφερθέντων προτάσεων είναι η δημιουργία ενός **ποσοτικού δείκτη,** ο οποίος καθορίζει την τρωτότητα του συστήματος S. Ο **Δείκτης Τρωτότητας (Vulnerability Index)** παρουσιάζει ποσοτικά αυτά που ο χάρτης αντιμετώπισης εμφανίζει εμπειρικά στον διαχειριστή του συστήματος μετά από εκτενή ανάλυση του.

VI=VP/m

Όπου VP είναι ο πληθάριθμος του συνόλου των απειλών T του συστήματος S, οι οποίες ικανοποιούν την Πρόταση Τρωτότητας.

Ο δείκτης τρωτότητας ποσοτικοποιεί τον κίνδυνο σε προληπτικό επίπεδο. Η βεβαιότητα της εμφάνισης κινδυνικών φαινομένων ποσοτικοποιείται σε συνάρτηση με τα αντίμετρα που έχει το σύστημα. Ο διαχειριστής του συστήματος γνωρίζει πόσο τρωτό είναι το σύστημα του και ανάλογα δρα, προκειμένου να ελαττώσει τυχόν μεγάλη τιμή του δείκτη. Παράλληλα, γνωρίζει **ποιοτικά με τη βοήθεια του χάρτη**

κινδύνου και ειδικότερα με την Πρόταση Τρωτότητας πού είναι το σύστημα του ευάλωτο σε απειλές.

3.3.7 ΜΕΘΟΔΟΛΟΓΙΑ ΑΝΑΚΤΗΣΗΣ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

3.3.7.1 ΕΙΣΑΓΩΓΗ

Είναι αναγκαία η δημιουργία μιας μεθοδολογίας αντιμετώπισης καταστροφικών φαινομένων σε ένα σύστημα διαχείρισης κρίσεων των ψηφιακών πληροφοριακών συστημάτων. Πάντα σε μια κρίση υπάρχει απώλεια μεγάλη ή μικρή. Το ύψος της απώλειας εξαρτάται από το μέγεθος του καταστροφικού φαινομένου. Δυστυχώς, δεν είναι πάντοτε εφικτή η αντιμετώπιση μιας απειλής. Όσο έτοιμο και αν είναι ένα σύστημα να αμυνθεί σε ένα αναμενόμενο κίνδυνο πάντα θα υπάρχει μια μεγάλη πιθανότητα να συμβεί το αναπάντεχο. Οι Αστάθμητοι παράγοντες υποσκάπτουν τη διαδικασία πρόληψης και ενδυναμώνονται από τυχαία ή εσκεμμένα γεγονότα, που μπορεί να υλοποιηθούν μέσα στον ορίζοντα λειτουργίας του έργου. Η αστάθεια του χώρου δεδομένων, η έλλειψη αυτοματισμών αλλά και η ολιγωρία των χειριστών και των μελών του συστήματος αποτελούν βασικούς παράγοντες εμφάνισης καταστροφικών φαινομένων.

Η δυναμικότητα του χώρου δεδομένων και οι συνεχείς αλλαγές που προκαλούνται από ασταθείς δομικά παράγοντες του περιβάλλοντος λειτουργίας ενός συστήματος συμβάλλουν στην εμφάνιση κρίσεων. Στην πλειοψηφία των περιπτώσεων, οι κρίσεις προκαλούν εκτεταμένες καταστροφές στα δομικά στοιχεία του συστήματος, προκαλώντας μερικές καταρρεύσεις ή ολικά αυτοκαταστροφικά φαινόμενα.

3.3.7.2 ΜΟΝΤΕΛΟΠΟΙΗΣΗ

«Όταν αποτυγχάνει η πρόληψη ο κίνδυνος μετατρέπεται από θεωρητική προσεγγιστική έννοια σε πραγματική μετρήσιμη καταστροφική δύναμη.»

Αυτή η πρόταση αποτελεί Βασικό Αξίωμα στη μεθοδολογία Ανάκτησης Πληροφοριακών Συστημάτων. Η παραπάνω προσέγγιση αποδεικνύει την εξελικτική πορεία ενός κινδυνικού φαινομένου το οποίο εμφανίζεται, ενεργοποιείται και δρα σε ένα Δυναμικό Χώρο Δεδομένων. Το παραπάνω αξίωμα ενώνει τις δυο μεθοδολογίες κάτω από τη σκέπη της Γενικής Θεωρίας Διαχείρισης Κινδύνου. Παρουσιάζει με απλά λόγια

τη συνοχή των δυο υπό-μεθοδολογιών. Στην ουσία, δεν πρόκειται για δυο ανεξάρτητα τμήματα της γενικής θεωρίας, αλλά για δυο αλληλοεξαρτώμενα κομμάτια που επικοινωνούν μεταξύ τους, ανταλλάσσοντας δεδομένα και συμπληρώνοντας το ένα τις αδυναμίες του αλλού. Η καθεμία από τις μεθοδολογίες αποτελεί τη λογική επέκταση της άλλης, αφού η συνολική επιτυχία της Γενικής Θεωρίας συνεπάγεται **την ομαλή εφαρμογή και λειτουργία και των δυο μεθοδολογιών ταυτόχρονα**.

Η μεθοδολογία Ανάκτησης Πληροφοριακών Συστημάτων αποτελεί το δεύτερο κομμάτι της Γενικής Θεωρίας Διαχείρισης Κινδύνου. Στόχος της είναι η ανάκτηση των δεδομένων του συστήματος, όταν ένα καταστροφικό φαινόμενο λάβει χώρα μέσα στο διάστημα Δt. Για να είναι επιτυχής η μεθοδολογία Ανάκτησης, πρέπει να ακολουθεί μια βασική αρχή, που αποτελεί την ***Πρόταση Επιτυχούς Ανάκτησης*** για τη θεωρία αυτή καθαυτή:

«Η ανάκτηση ενός συστήματος πρέπει να τείνει όσο το δυνατόν περισσότερο στην αρχική κατάσταση του.»

Η πρόταση αυτή αποτελεί τον ακρογωνιαίο λίθο της μεθοδολογίας αφού καθορίζει το ποσοστό επιτυχίας της ανάκτησης του συστήματος. Για να είναι επιτυχής η υλοποίηση της μεθοδολογίας, πρέπει το Σύστημα Ανάκτησης να επαναφέρει το κατεστραμμένο σύστημα στην αρχική του κατάσταση, πριν δηλαδή την υλοποίηση της κρίσης. Ένα απόλυτα επιτυχημένο σύστημα είναι εκείνο που ανακτά πλήρως όλα τα δεδομένα που απωλέσθησαν. Δυστυχώς, όμως, αυτού του είδους η ανάκτηση δεν είναι πάντοτε επιτυχής, αφού απαιτεί μεγάλο κόπο, πολύπλοκα συστήματα ανάκτησης καθώς και μεγάλα ποσά που απαιτείται να επενδυθούν. Υπάρχουν περιπτώσεις όπου το κόστος ανάκτησης είναι μεγαλύτερο από το κόστος ολικής επαναδημιουργίας νέου συστήματος!

Ένα μοντέλο διαχείρισης Κινδύνου δεν πρέπει μονάχα να έχει την ικανότητα να εντοπίζει τα ευάλωτα σημεία ενός συστήματος. Οφείλει να έχει τη δυνατότητα να προσδιορίζει πόσο πετυχημένο είναι το μοντέλο ανάκτησης δεδομένων, το οποίο υποχρεούται να έχει. Είναι ιδιαίτερα σημαντικό το μοντέλο να εξασφαλίζει ένα ολοκληρωμένο και ασφαλές σχέδιο ανάκτησης. Ένα χρήσιμο εργαλείο που βοηθάει σημαντικά στον προσδιορισμό του βαθμού επιτυχίας ενός συστήματος ανάκτησης δεδομένων είναι ο ***Δείκτης Επιτυχούς Ανάκτησης (Final State Recovery Index (FSR)).*** Ο οποίος ορίζεται ως ακολούθως:

$$FSR = min(|P(S)| - |R(S)|)$$

Στην **αρχική κατάσταση (Primitive Situation P(S))**, πριν την πτώση του συστήματος S, τα υποσυστήματα του υπό μελέτη συστήματος λειτουργούν κανονικά. Έστω N1 τα υποσυστήματα του συστήματος S τα οποία λειτουργούν ομαλά. Ως P(S) ορίζεται το σύνολο των υποσυστημάτων του συστήματος S, τα οποία λειτουργούν ομαλά στην χρονική περίοδο, πριν την πτώση του συστήματος. Αντίθετα ως R(S) ορίζεται το σύνολο των υποσυστημάτων που ανακτήθηκαν και λειτουργούν μετά την Πτώση του συστήματος S. Είναι, δηλαδή, **η τελική κατάσταση (Restoration Situation R(S))** του συστήματος μετά την πτώση των υποσυστημάτων του. Έστω N2 τα υποσυστήματα του συστήματος S, τα οποία ανακτήθηκαν.

Ο FSR είναι ένας ποιοτικός δείκτης που προσδιορίζει την ικανότητα του συστήματος να ανακτήσει τα καταστραμμένα δεδομένα. Ο βασικός σκοπός του δείκτη είναι το σύστημα ανάκτησης να αγγίξει την τελειότητα, δηλαδή **FSR=0**, το οποίο σημαίνει την πλήρη ανάκτηση. Αυτή η περίπτωση είναι ιδιαίτερα δύσκολη, όπως αναλύθηκε. Έτσι, ο FSR μέσα από ποιοτικά αποτελέσματα αποσκοπεί σε ικανοποιητικά αποτελέσματα με την ελαχιστοποίηση της διαφοράς ανάμεσα στις δυο καταστάσεις. Βέβαια, εδώ γίνεται η παραδοχή, ότι όλα τα υποσυστήματα του S έχουν την ίδια αξία, το οποίο δεν είναι απόλυτα αληθές.

Ο FSR αποτελείται από τρεις περιπτώσεις κάθε μια από τις οποίες προσδιορίζει ποιοτικά το είδος της ανάκτησης που επιτεύχθηκε. Οι τρεις περιπτώσεις αναλύονται παρακάτω:

3.3.7.2.1 Ολική Ανάκτηση Συστήματος (Total Recovery Situation)

$$FSR = N1-N2 = 0.$$

Αποτελεί την ιδανική περίπτωση, η οποία σπάνια συμβαίνει. Πρόκειται για την **πλήρη ανάκτηση του συστήματος** μετά την πτώση. Με απλά λόγια τα N1 υποσυστήματα που λειτουργούσαν στην αρχική κατάσταση συνεχίζουν να είναι επιχειρησιακά άρτια. Ο κίνδυνος σε αυτή την περίπτωση υπερκεράστηκε χωρίς αύξηση του κόστους ανάκτησης με την προσθήκη επιπλέον μέτρου ανάκτησης.

3.3.7.2.2 Ολική Κατάρρευση Συστήματος (Black Operational Hole)

$$FSR = N1 - N2 = N1$$

Πρόκειται για το χειρότερο σενάριο. Μια περίπτωση που συμβαίνει αρκετά συχνά. Οι αστάθμητοι παράγοντες, η ανικανότητα των χρηστών του συστήματος να λειτουργήσουν την ώρα της κρίσης και η άμεση κατάρρευση βασικών δομών του συστήματος συντελούν στην ολική του πτώση. Στην τελική κατάσταση του συστήματος δε λειτουργεί κανένα υποσύστημα, δηλαδή N2=0. Η πτώση έχει υλοποιηθεί και το σύστημα ανάκτησης απέτυχε να την αντιμετωπίσει, αποκαθιστώντας με επιτυχία καταστροφές και απώλειες δεδομένων βαρύνουσας σημασίας για τη δομή όλου του συστήματος. Το πιο σημαντικό και παράλληλα πιο οδυνηρό συνήθως έρχεται μετά την καταστροφή, αφού το κόστος αποκατάστασης σε τέτοιες περιπτώσεις είναι τεράστιο, δίνοντας το τελικό θανατηφόρο χτύπημα για τη συνολική ύπαρξη του συστήματος μέσα στον ορίζοντα χρονοπρογραμματισμού του.

3.3.7.2.3 Μερική Ανάκτηση Συστήματος (Partial Recovery Situation)

$$0 < FSR < n$$

Η τρίτη περίπτωση είναι στην ουσία η ενδιάμεση κατάσταση. Στην πλειοψηφία των περιπτώσεων εξαρτάται από το κόστος του partial recovery. Στην πραγματικότητα, δεν είναι το χειρότερο σενάριο κρίσης αλλά ούτε και το καλύτερο. Είναι μια ενδιάμεση κατάσταση που χρίζει ιδιαίτερης μελέτης και η οποία εξαρτάται επίσης από το μέγεθος της κρίσης, που έχει λάβει χώρα στο σύστημα.

Επιπλέον, γίνεται εύκολα αντιληπτό ότι η τρίτη περίπτωση είναι στην ουσία μια μεταλλακτική κατάσταση στην οποία τίθεται το σύστημα. Ένα νέο σύστημα προκύπτει μέσα από αυτή τη διαδικασία το οποίο στην πλειοψηφία των περιπτώσεων έχει περιορισμένες ικανότητες και μικρότερο αριθμό διεργασιών σε σύγκριση με το αρχικό σύστημα. Σπάνιες είναι οι περιπτώσεις όπου μια κρίση συνέβαλλε στη μετεξέλιξη και ανάπτυξη ενός πληροφοριακού συστήματος.

Επιπρόσθετα, στην τρίτη περίπτωση μόλις το σύστημα μεταλλαχθεί τότε υπολειτουργεί, αφού μόνο N2 υποσυστήματα λειτουργούν ομαλά και είναι ικανά να έχουν το συνολικό σύστημα σε λειτουργία, αποφεύγοντας την ολική κατάρρευση του. Βέβαια, τα (N1-N2) υποσυστήματα που τέθηκαν εκτός λειτουργίας με την πτώση θα

πρέπει να μην είναι απαραίτητα για την ομαλή λειτουργία του συστήματος.[Petrantonakis (2006)]

Τέλος, συμπεραίνουμε ότι ο FSR είναι περισσότερο ένα μέτρο αναφοράς καταστροφής παρά ένας ποιοτικός δείκτης που προσδιορίζει το είδος αποκατάστασης που υλοποιήθηκε στο σύστημα. Προσδιορίζεται η κατάσταση στην οποία βρίσκεται το σύστημα μετά την εμφάνιση ενός καταστροφικού φαινομένου. Ο FSR δεν προσδιορίζει την δομή του νέου συστήματος μετά την πτώση. Παράλληλα, δεν ποσοτικοποιεί τον κίνδυνο ούτε και την καταστροφή, ενώ δεν προσδιορίζει χρονικά την πτώση και την ολοκλήρωση της αποκατάστασης. Είναι ένα πρώτο βήμα, όμως που βοηθάει σημαντικά τον χειριστή του συστήματος να καταλάβει την κατάσταση στην οποία βρίσκεται το σύστημα και να λάβει δραστικά μέτρα. Είναι όπως η κλίμακα Ρίχτερ για τους σεισμούς όπου το μέγεθος δεν ταυτίζεται και με το μέγεθος της καταστροφής.

3.3.7.3 ΑΝΑΚΤΗΣΗ ΑΠΟ ΠΤΩΣΗ

Είναι φανερό ότι τα κοινά σημεία των υποσυστημάτων μπορούν να επηρεαστούν από μια ή ένα σύνολο καταστροφικών δυνάμεων που εμφανίζονται μέσα στον ορίζοντα χρονοπρογραμματισμού του έργου. Αυτή η καταστροφική δύναμη οδηγεί σε ολική κατάρρευση του συστήματος. Πρέπει, λοιπόν, τα κοινά σημεία των υποσυστημάτων να διασπαστούν σε νέα υποσυστήματα, με σκοπό την ελαχιστοποίηση του κίνδυνου. Φυσικά, η εσκεμμένη μετάλλαξη ενός πληροφοριακού συστήματος για την αποφυγή μιας κρίσης δεν είναι παρά μια προσωρινή λύση στο πρόβλημα, για να εξοικονομηθεί χρόνος μέχρι την οριστική ανάκτηση του.

Υπάρχει μια βασική σκέψη που ταλανίζει κάθε προγραμματιστή. Κατάφερε άραγε να κατασκευάσει το απόλυτο πληροφοριακό σύστημα; Η απάντηση σε αυτή την ερώτηση είναι φυσικά αρνητική. Δεν υπάρχει σύστημα ικανό να αποφύγει κάθε είδος κινδύνου ειδικά αν λειτουργεί μέσα σε ένα εχθρικό περιβάλλον. Εμφανίζονται καταστροφικές δυνάμεις, ενώ σχηματίζονται νέες από μεταλλάξεις παλιών μορφών, θέτοντας το σύστημα σε υψηλό κίνδυνο.

Για να αποφευχθούν δυσάρεστες καταστάσεις οι χρήστες πρέπει να είναι συνεχώς σε εγρήγορση για την αντιμετώπιση οποιασδήποτε κρίσεως. Για να μπορέσουν να μειώσουν τον κίνδυνο θα πρέπει να έχουν τη δυνατότητα να χαρτογραφήσουν το σύστημα τους, γνωρίζοντας ακριβώς τις αδυναμίες του και τις καταστροφικές δυνάμεις που ενδέχεται

να υλοποιηθούν στα υποσυστήματα του. Έστω C το σύνολο των καταστροφικών δυνάμεων που υλοποιούνται στον ορίζοντα χρονοπρογραμματισμού του έργου (2) και R το σύνολο των αντίμετρων που έχει το σύστημα για να αντιμετωπίσει την εκδήλωση ενός τέτοιου καταστροφικού φαινομένου (6). Ένα χρήσιμο εργαλείο βοήθημα για την υλοποίηση στρατηγικών αποκατάστασης ζημιών σε ένα ψηφιακό πληροφοριακό σύστημα είναι ο **Χάρτης Καταστροφών (Catastrophe Matrix).**

M = (m_{ij}), $i \in C^*$, $j \in R^*$, όπου:
m_{ij} = 1, αν η καταστροφή c_i μπορεί να ανακτηθεί από το μέτρο ανάκτησης r_j
m_{ij}= 0, διαφορετικά.

Ο Χάρτης Καταστροφών είναι ένα χρήσιμο εργαλείο από το οποίο εξάγονται χρήσιμα συμπεράσματα για την ομαλή λειτουργία του συστήματος καθώς και τους κινδύνους που πρέπει να αντιμετωπίσει. Παράλληλα, με τη βοήθεια του χάρτη ο διαχειριστής αντιλαμβάνεται και οργανώνει την άμυνα του απέναντι σε μια κρίση που λαμβάνει χώρα σε κάποια στιγμή μέσα στον ορίζοντα χρονοπρογραμματισμού του συστήματος. Η παρακάτω πρόταση προσδιορίζει το βάθος της αμυντικής γραμμής που έχει το κάθε σύστημα αντιμετώπισης καταστροφών.

Πρόταση Ανάκτησης Πρώτου Επιπέδου: Πρώτης τάξεως ανάκτηση ενός συστήματος είναι η κατάσταση κατά την οποία κάθε καταστροφή ενεργοποιεί ακριβώς ένα recovery αυτόματα.

Προσδιορίζεται ποιοτικά το επίπεδο στο οποίο βρίσκεται το σύστημα ανάκτησης από πτώσεις. Όσο μεγαλύτερο το βάθος τόσο μεγαλύτερη και η πιθανότητα επιτυχούς ανάκτησης των δεδομένων. Ειδικότερα, η πρόταση αναλύεται σύμφωνα με τα παρακάτω.

Αν ισχύει:
$(\forall i) \in C^*$: $\sum_j m_{ij} = 1$, $j \in R^*$,

και $(\forall j) \in R^*$: $\sum_i m_{ij} = 1$, $i \in C^*$

τότε το σύστημα S βρίσκεται σε Κατάσταση Ανάκτησης Πρώτου Επιπέδου.

Με τη βοήθεια του χάρτη καταστροφών εξορύσσονται και άλλα χρήσιμα συμπεράσματα ικανά να συντελέσουν στην επιτυχία της

ανάκτησης των δεδομένων, που αποτελεί και απώτερο σκοπό της μεθοδολογίας. Πιο συγκεκριμένα, επισημαίνεται πόσο ικανό είναι το σύστημα να ανακτήσει μια πτώση. Η ικανότητα του αυτή αποτελεί βαρόμετρο για την επιτυχημένη εφαρμογή της μεθοδολογίας ανάκτησης. Η ακόλουθη πρόταση καθορίζει την έλλειψη αμυντικών μηχανισμών που μπορεί ενδεχομένως να υπάρχει και οδηγεί, **στην καταστροφή του συστήματος.**

Πρόταση Καταστροφικότητας:

Αν ισχύει:

$$(\exists i \in C^{*}) : \sum_{j} m_{ij}=0 , \; j \in R^{*}$$

τότε το σύστημα S κινδυνεύει με ολική ή μερική καταστροφή.

Η παραπάνω πρόταση είναι μια ποιοτική ανάλυση της κατάστασης στην οποία βρίσκεται το σύστημα, αφού προσδιορίζονται τα κενά στην ανάκτηση των δεδομένων, που αποτελούν σημεία αυτοκατάρρευσης του συστήματος. Επιπλέον, η παραπάνω πρόταση είναι ο γεννήτορας του **Δείκτη Καταστροφικότητας (Destruction Index).** Ο Δείκτης καταστροφικότητας είναι μια ανάλυση των συμπερασμάτων, που εξάγονται μέσα από την ποιοτική ανάλυση της παραπάνω πρότασης. Ειδικότερα, υπολογίζεται το ποσοστό δυνατής καταστροφής και ανικανότητας του συστήματος ανάκτησης να αντιμετωπίσει την κρίση και ποσοτικοποιείται η πιθανότητα μη ομαλούς αντιμετώπισης της επικείμενης καταστροφής. Με το δείκτη καταστροφικότητας δεν προσδιορίζεται η δομή του νέου πληροφοριακού συστήματος και παράλληλα με το δείκτη, δεν καθορίζεται το είδος του μέτρου ανάκτησης, που θα χρησιμοποιηθεί, ούτε και η χρονική διάρκεια χρήσης του. Έτσι, έχουμε:

$$\mathbf{DI= DP/ |S|}$$

Όπου DP είναι ο πληθάριθμος του συνόλου των πτώσεων C του συστήματος S, οι οποίες ικανοποιούν την Πρόταση Καταστροφικότητας.

Ένα σημαντικό κριτήριο για την ενεργοποίηση του κατάλληλου Recovery είναι το **Κόστος Ενεργοποίησης (Activation Cost)** του. Το κόστος ενεργοποίησης ιδιαίτερα σε Πληροφοριακά Συστήματα Εθνικής Άμυνας απαιτεί ένα τεράστιο προϋπολογισμό. Ο χειριστής του

συστήματος δεν πρέπει απλώς να έχει τη δυνατότητα να διαλέγει ένα αντίμετρο με τη βοήθεια του Χάρτη Καταστροφών, πρέπει ακόμη και να μπορεί να διακρίνει ποιο είναι το καταλληλότερο για κάθε περίσταση. Καταλληλότερο δεν είναι πάντοτε το πιο αποτελεσματικό, αλλά - στην πλειοψηφία των περιπτώσεων - και το λιγότερο ακριβό. Έτσι, έχουμε:

$$\mathbf{C = (c_{ij})},\ \forall i \in C^*,\ j \in R^*$$

c_{ij} = κόστος, αν η πτώση c_i ενεργοποιεί το μέτρο ανάκτησης recovery r_j
$c_{ij} = \infty$, διαφορετικά.

Βασικός σκοπός ενός επιτυχημένου Σχεδίου Ανάκτησης είναι η ενεργοποίηση των recoveries συστημάτων με το ελάχιστο δυνατό κόστος. Η σκέψη αυτή αποτελεί τη θεμελιώδη αρχή της Μεθοδολογίας Ανάκτησης Συστημάτων. Η επιτυχής εφαρμογή αυτής της αρχής αποφέρει διπλό αποτέλεσμα και αποτελεί, παράλληλα, τη βέλτιστη λύση ανάκτησης δεδομένων από πτώση. Δεν επιτυγχάνεται μόνο η ενεργοποίηση του κατάλληλου recovery, προκειμένου να αντιμετωπισθεί η καταστροφή, αλλά ταυτόχρονα ενεργοποιείται το κατάλληλο recovery με το ελάχιστο δυνατό κόστος ενεργοποίησης. Η παρακάτω πρόταση πιστοποιεί τα προαναφερθέντα:

Πρόταση Ενεργοποίησης Ανάκτησης Ελάχιστου Κόστους (Minimum Cost Activation Recovery Proposition):

$$\mathbf{min} \sum_i \sum_j \mathbf{c_{ij} m_{ij}}\ ,\ \mathbf{i \in D^*},\ \mathbf{j \in R^*}$$

Έτσι ώστε,

$$(\forall i \in \mathbf{C^*}): \sum_j \mathbf{m_{ij} = 1},\ \mathbf{j \in R^*}$$

και $(\forall j \in \mathbf{R^*}): \sum_i \mathbf{m_{ij} <= 1}\ ,\ \mathbf{i \in C^*}$

Η παραπάνω πρόταση περιέχει το γνωστό ***Assignment problem (Πρόβλημα Κατανομής)*** με n γραμμές και m στήλες, το οποίο λύνεται σε πολυωνυμικό χρόνο O (nlogm) για μεγάλο αριθμό n και m με τη βοήθεια βέλτιστων μεθόδων επίλυσης του μαθηματικού προγραμματισμού, όπως ο αλγόριθμος επίλυσης του Kuhn. [Petrantonakis (2005β)]

3.3.7.4 ΥΠΟΛΟΓΙΣΜΟΣ ΚΟΣΤΟΥΣ ΑΝΑΚΤΗΣΗΣ

Η ελαχιστοποίηση του κόστους ενεργοποίησης recovery, όπως αναλύθηκε στην προηγούμενη παράγραφο, είναι το κύριο σημείο της Μεθοδολογίας Ανάκτησης Συστημάτων. Με τη βοήθεια του Προβλήματος Κατανομής επιλέγεται το σύνολο συστημάτων ανάκτησης, η επιλογή του οποίου θα πρέπει να στηρίζεται στο γεγονός ότι το κόστος ενεργοποίησης παίζει και αυτό με τη σειρά του πρωτεύοντα ρόλο στη διαδικασία αξιολόγησης.

Υπάρχουν δυο βασικοί άξονες για τον υπολογισμό του κόστους ανάκτησης δεδομένων. Και οι δυο άξονες έχουν ισοδύναμη αξία, αφού ο καθένας από την μεριά του προσδιορίζει το κόστος σε διαφορετικό επίπεδο μέσα στο κατεστραμμένο σύστημα. Ο πρώτος τομέας υπολογίζει τη **χρηματική άξια των κατεστραμμένων υλικών** των υποσυστημάτων του υπό κρίση συστήματος. Ο δεύτερος τομέας υπολογίζει **την άξια των χαμένων δεδομένων** των υποσυστημάτων του πληροφοριακού συστήματος S. Ο υπολογισμός της άξιας αυτής, σε πολλές περιπτώσεις είναι, **υποκειμενικός**. Ο υπολογισμός του κόστους ανάκτησης αποτελεί σημαντικό τμήμα της συνολικής διαδικασίας αποκατάστασης ψηφιακών συστημάτων στα οποία έχει υλοποιηθεί μια καταστροφή. Αυτό οφείλεται στο γεγονός ότι το κόστος ανάκτησης εμμέσως προσδιορίζει με τη σειρά του το μέγεθος της καταστροφής, ενώ από την άλλη μεριά σημαντικό σε ένα σύστημα ανάκτησης από πτώσεις είναι το χρονικό σημείο κατά το οποίο υλοποιείται η καταστροφή. Το timing - όπως αλλιώς λέγεται - αυξάνει σημαντικά το μέγεθος της καταστροφής, αφού προσδιορίζει ποιοτικά τη σημαντικότητα και αναγκαιότητα λειτουργίας των υπό κρίση υποσυστημάτων. Η παραπάνω ποιοτική προσέγγιση των πτώσεων, που υλοποιούνται σε ένα σύστημα, προσδιορίζεται από την ακόλουθη πρόταση:

Το μέγεθος της καταστροφής ενός συστήματος είναι ανάλογο του κόστους ανάκτησης των δεδομένων του και της σημαντικότητας ανάκτησης της καταστροφής.

Ο **Δείκτης Καταστροφικής Βαρύτητας (Catastrophe Weight Index)** προσδιορίζει υποκειμενικά το μέγεθος μιας καταστροφής. Οι τιμές του εξαρτώνται από ποιοτικές εκτιμήσεις των εκάστοτε χειριστών του πληροφοριακού συστήματος. Οι εκτιμήσεις αυτές στηρίζονται στη χρονική στιγμή υλοποίησης των πτώσεων - μέσα στον ορίζοντα χρονοπρογραμματισμού - και καθορίζουν την σημαντικότητα της καταστροφής. Έτσι, έχουμε:

CWI = Cost * timing

Όπου:
Cost: το κόστος της ανάκτησης των δεδομένων.
Timing: η σημαντικότητα ανάκτησης της καταστροφής με τιμές από 0-10, ανάλογα με την αμεσότητα ανάκτησης των δεδομένων.

Το κόστος ανάκτησης εξαρτάται κυρίως από το χρόνο. Μετά την υλοποίηση μιας καταστροφής είναι ζωτικής σημασίας η άμεση και ολοκληρωμένη ανάκτηση του συστήματος. Συνεπώς, όσο μεγαλύτερη είναι η καθυστέρηση ανάκτησης τόσο μεγαλύτερο γίνεται το κόστος αποκατάστασης των ζημιών, όπως για παράδειγμα σε περιπτώσεις άμεσης ανάγκης όπου απαιτείται άμεσα η χρήση των κατεστραμμένων υποσυστημάτων του συστήματος S.

Ένα ολοκληρωμένο σύστημα ανάκτησης πληροφοριακών συστημάτων για να μπορέσει να είναι αποτελεσματικό θα πρέπει να δίνει την δυνατότητα στο χρήστη να προϋπολογίζει το κόστος ανάκτησης για κάθε είδος καταστροφικής δύναμης, που ενδέχεται να λάβει χώρα σε κάποια χρονική στιγμή του ορίζοντα προγραμματισμού Δt. Ο υπολογισμός του κόστους είναι βαρύνουσας σημασίας για τη χάραξη μιας ολοκληρωμένης στρατηγικής Διαχείρισης Κρίσεων. Σε ένα ασταθή χώρο δεδομένων ο υπολογισμός του κόστους ενεργοποίησης του κατάλληλου αντίμετρου στηρίζεται στην απλή ιδέα ότι οι απώλειες είναι διπλές, τόσο σε υλικό όσο και στο λογισμικό.

Ας υποθέσουμε ότι **PRC** είναι το **Προτεινόμενο Κόστος Ανάκτησης (Proposed Recovery Cost)** του συστήματος S, όπως έχει προϋπολογιστεί.

PRC = (DVALUE * DMRT) + (DATAVALUE * DLRT)

Έτσι, ώστε:

DVALUE: Οικονομική Άξια των κατεστραμμένων Υλικών του συστήματος S.
DMRT: Destroyed Materials Recovery Time - Ο απαραίτητος χρόνος έτσι, ώστε να επιτευχθεί η ολική ανάκτηση των κατεστραμμένων υλικών. Υπολογίζεται συνήθως σε εργατοώρες.
DATAVALUE: Η υποκειμενική οικονομική άξια των κατεστραμμένων δεδομένων.

DLRT: Data Lost Recovery Time - Ο απαραίτητος χρόνος έτσι, ώστε να επιτευχθεί η ολική ανάκτηση των κατεστραμμένων δεδομένων του συστήματος S. Υπολογίζεται συνήθως σε εργατοώρες.

Όλοι οι όροι της παραπάνω σχέσης μπορούν εύκολα να υπολογιστούν. Ειδικότερα, οι τιμές των DVALUE, DMRT και DLRT μπορούν να μετρηθούν, αν στηριχθούμε σε πραγματικά αντικειμενικά μετρήσιμα δεδομένα έτσι, ώστε να εξάγουμε χρήσιμες πληροφορίες για την πιθανή οικονομική απώλεια. Ενδιαφέρον παρουσιάζει ο υπολογισμός του Subjective Value of the Lost Data (DATAVALUE). Μετά την υλοποίηση της καταστροφής ένας σημαντικός αριθμός δεδομένων, φακέλων και υποφακέλων είναι πολύ πιθανό να καταστραφεί. Είναι αναγκαία, επομένως, η άμεση και τάχιστη αποκατάσταση τους, ώστε να επιτευχθεί η ολική ανάκτηση του συστήματος S με το ελάχιστο δυνατό κόστος. [Petrantonakis (2005α)]

Είναι, επίσης, ιδιαίτερα σημαντικό να κατηγοριοποιούνται τα δεδομένα και οι φάκελοι ανάλογα με τη σπουδαιότητα τους. Η κατηγοριοποίηση αυτή βοηθάει σημαντικά τόσο στην καλύτερη προστασία των ευαίσθητων δεδομένων, των δεδομένων ύψιστης σημασίας όσο και στο μετέπειτα υπολογισμό του κόστους απώλειας τους. Υπεύθυνο της κατηγοριοποίησης πρέπει να είναι ένα εξειδικευμένο προσωπικό, όπως οι ίδιοι οι διαχειριστές του συστήματος.

Ας υποθέσουμε ότι:

$$\mathbf{DATAVALUE = DATAVOLUME * WEIGHT}$$

όπου,

DATAVOLUME: Το άθροισμα των κατεστραμμένων φακέλων μετρούμενο σε Kilobytes. ($\sum_{i=1}^{|F|} Kb_i$).

WEIGHT: Ένας παράγοντας που δείχνει την σπουδαιότητα του κατεστραμμένου φακέλου (W_i).

Επομένως, η τελική τιμή του DATAVALUE είναι:

$$\mathbf{DATAVALUE= \sum_{i=1}^{|F|} (Kb_i * W_i)}$$

όπου συγκεκριμένα ορίζουμε ως :

Kb_i : Kilobytes του καταστραμμένου i-file.
F : Το σύνολο των φακέλων οι οποίοι έχουν καταστραφεί.

Στην πλειοψηφία των περιπτώσεων δεν είναι σίγουρο ότι επιτυγχάνεται ολικό recovery του συστήματος. Σε έναν Ασταθή Χώρο δεδομένων - ακόμα και στην διαδικασία της ανάκτησης - μπορεί να εμφανιστούν αστάθμητοι παράγοντες ο οποίοι να θέσουν την όλη διαδικασία εν αμφιβόλω. Αυτό σημαίνει ότι μπορεί μεν να είχε προϋπολογισθεί ένα πλήρες recovery (PRC), όμως, τελικά μόνο μερικό recovery να έχει επιτευχθεί, πράγμα που σημαίνει ότι μερικά από τα υποσυστήματα του Συστήματος S έχουν ανακτηθεί. Από την άλλη μεριά, κάποια αλλά υποσυστήματα να παραμένουν κατεστραμμένα, με αποτέλεσμα το συνολικό κόστος ανάκτησης να διαφέρει από το προϋπολογισθέν πόσο. Με άλλα λόγια, επιτεύχθηκε και σε αυτή την περίπτωση **Μερική Ανάκτηση(Partial Recovery Situation)**.

Η **Απόδοση (Performance)** είναι ένα πολύ χρήσιμο εργαλείο το οποίο βοηθάει στην εκτίμηση για την επιτυχία του recovery plan. Στην ουσία, πρόκειται για ένα ποσοτικό δείκτη που προσδιορίζει την επιτυχία του συστήματος ανάκτησης ακόμα και αν εμφανιστούν ασταθείς παράγοντες κατά την διάρκεια λειτουργίας του. Πρόκειται για το κλάσμα του πραγματικού κόστους ανάκτησης προς το προτεινόμενο κόστος ανάκτησης. Έστω ότι **RP** είναι ο **Δείκτης Απόδοσης Ανάκτησης (Recovery Performance Plan):**

$$RP = ARC/PRC$$

όπου ARC είναι το **Πραγματικό Κόστος Ανάκτησης (Actual Recovery Cost**) το οποίο υλοποιήθηκε.

ARC= (DRVALUE * ADMRT) + (NEWDATAVALUE * ADLRT)

Έτσι, ώστε:

DRVALUE: Οικονομική Άξια των ανακτηθέντων Υλικών του συστήματος S.
NEWDATAVALUE: Η υποκειμενική οικονομική άξια των ανακτηθέντων δεδομένων.

Το NEWDATAVALUE στην περίπτωση αυτή είναι:

$$\textbf{NEWDATAVALUE} = \sum_{j=1}^{|\Lambda|} (\mathbf{Kb_j} * \mathbf{W_j}) \qquad |F| \geq |\Lambda|$$

Έτσι, ώστε:

Kb_j: Kilobytes του ανακτηθέντος j-file.
Λ : Το σύνολο των φακέλων, οι οποίοι έχουν ανακτηθεί.

ADMRT: Actual Destroyed Materials Recovery Time: Ο απαραίτητος χρόνος που τελικά χρειάστηκε έτσι, ώστε να επιτευχθεί η ολική ανάκτηση των κατεστραμμένων υλικών. Υπολογίζεται συνήθως σε εργατοώρες.
ADLRT: Actual Data Lost Recovery Time, Ο απαραίτητος χρόνος που τελικά χρειάστηκε έτσι ώστε να επιτευχθεί η ολική ανάκτηση των κατεστραμμένων δεδομένων του συστήματος S. Υπολογίζεται συνήθως σε εργατοώρες.

Η υποκειμενικότητα είναι ένας παράγοντας που χρειάζεται να μειωθεί, προκείμενου να υπολογισθεί με μεγαλύτερη ακρίβεια το κόστος ανάκτησης. Μειώνοντας τον Υποκειμενικό Παράγοντα θα αυξηθεί η Απόδοση, ώστε να επιτευχθεί όσο το δυνατόν περισσότερο η Total Recovery Situation. [Petrantonakis (2005α)]

Γίνεται εύκολα αντιληπτό ότι δεν είναι πάντα εφικτή η ανάκτηση των δεδομένων όπως έχει σχεδιαστεί πριν την υλοποίηση της καταστροφής. Σίγουρα θα υπάρξουν απώλειες. Αυτό έχει σαν αποτέλεσμα να απαιτείται ο **Recovery Performance,** προκειμένου να υπολογιστεί η απόδοση, σε γενικές γραμμές, των χρημάτων που δαπανώνται.

Το μοντέλο αποκλιμάκωσης κρίσεων στην ουσία είναι ένα διπλό πακέτο που στηρίζεται στο δίπτυχο Πρόληψη – Αντιμετώπιση. Στόχος του η πρόληψη Κρίσεων και η ελαχιστοποίηση δυσάρεστων καταστάσεων. Παρόλα αυτά, περιλαμβάνει και την περίπτωση όπου συμβαίνει το αναπόφευκτο και όπου είναι αναγκαία η ανάκτηση του Συστήματος.

ΚΕΦΑΛΑΙΟ 4

ΨΗΦΙΑΚΑ ΣΥΣΤΗΜΑΤΑ ΕΘΝΙΚΗΣ ΑΜΥΝΑΣ

4.ΨΗΦΙΑΚΟΣ ΣΤΡΑΤΙΩΤΗΣ

4.1 ΕΙΣΑΓΩΓΗ

"Λένε πως ο Θεός είναι πάντα με το μέρος των καλύτερα αρματωμένων."

Βολταίρος

Ο εικοστός αιώνας είναι ο αιώνας της τεχνολογικής επανάστασης. Είναι ο αιώνας, κατά την διάρκεια του οποίου ο δυτικός τεχνολογικά ανεπτυγμένος πολιτισμός έκανε το άλμα από τη βιομηχανική στην ψηφιακή εποχή. Η **Πληροφορική** και οι σύγχρονες εφαρμογές της **Ηλεκτρονικής** επηρέασαν σημαντικά πολλούς τομείς της καθημερινότητας του Ανθρώπου. Ειδικότερα, ευαίσθητοι τομείς, όπως της Άμυνας, της Παιδείας και της Υγείας, παρουσιάζουν ραγδαία ανάπτυξη. Έτσι, δημιουργήθηκαν πολύπλοκα Πληροφοριακά Συστήματα, προκείμενου να καλύψουν τις ανάγκες τους και να ελαχιστοποιήσουν το κόστος λειτουργίας τους, διευκολύνοντας βασικές νόρμες της ζωής του Ανθρώπου, όπως είναι ο χρόνος και η απόσταση. Η πολυπλοκότητα αυτή αποτέλεσε σημείο αδυναμίας και Κατάρρευσης πολλών εξ' αυτών των συστημάτων καθώς εμφανίστηκε μεγάλη ποικιλία κινδύνων με ανεξέλεγκτα αποτελέσματα. Τα Πληροφοριακά Συστήματα αναγκάστηκαν να λειτουργούν μέσα σε ένα Εχθρικό περιβάλλον, μέσα σε ένα χώρο δεδομένων, όπου συνεχώς μεταβάλλεται απρόβλεπτα πολλές φορές, προκαλώντας Κρίσεις στο ίδιο το Σύστημα. Δυστυχώς, μέχρι και σήμερα είναι λίγα έως και ανύπαρκτα τα Πληροφοριακά Συστήματα εκείνα που μπορούν να αντιμετωπίσουν όσο και να προβλέψουν ικανοποιητικά μια Κρίση.

Ιδιαίτερα, την τελευταία δεκαετία είναι εμφανή τα σημάδια της αλλαγής που επέφερε η χρήση των υπολογιστών στην καθημερινότητα με την εμφάνιση του Διαδικτύου και των υπηρεσιών που αυτό προσφέρει. Όπως ήταν φυσικό, δεν έμεινε ανεπηρέαστος και ο νευραλγικός τομέας της **Εθνικής Άμυνας**. Οι εφαρμογές της πληροφορικής και η ψηφιοποίηση των οπλικών συστημάτων συνέβαλαν στην αλλαγή του τρόπου διεξαγωγής των στρατιωτικών επιχειρήσεων στα σύγχρονα πεδία μάχης.

Πολλοί τομείς στον αμυντικό κλάδο άλλαξαν ριζικά τη δομή τους, αφού αναγκάστηκαν να χρησιμοποιήσουν νέα αμυντικά μέσα, ενώ παράλληλα υποχρεώθηκαν να προσαρμοστούν στις απαιτήσεις και τη μορφολογία του σύγχρονου θεάτρου επιχειρήσεων. Τρεις είναι οι βασικοί

τομείς της Εθνικής Άμυνας, οι οποίοι αναδιοργανώθηκαν, προκειμένου να υιοθετήσουν το νέο ψηφιακό μοντέλο λειτουργίας:

- **Οι επικοινωνίες και οι πληροφορίες**,
- **Η εφαρμογή τεχνολογίας για προσβολή εντοπιζόμενων στόχων σε ελάχιστο χρόνο,**
- **Η δυνατότητα μεταφοράς όγκου δυνάμεων και πυρός.**

Οι νέες τεχνολογίες στην υπηρεσία της εθνικής άμυνας εξοπλίζουν τον στρατιώτη του μέλλοντος. Μια νέα ψηφιακή μορφή κάνει την εμφάνιση της. Ο **σύγχρονος ψηφιακός στρατιώτης (Land Warrior),** ο οποίος αποτελεί την αιχμή του δόρατος για κάθε σύγχρονο αμυντικό σύστημα, αφού στηρίζεται κατά μεγάλο ποσοστό σε πολύπλοκα τηλεπικοινωνιακά και πληροφοριακά συστήματα. Η νέα αυτή δομή είναι τεχνολογικά εξελιγμένη και παρουσιάζει σημαντικά πλεονεκτήματα σε σχέση με τη συμβατική μορφή του στρατιώτη. Πλεονεκτήματα που γέρνουν την πλάστιγγα της νίκης σημαντικά προς το μέρος της ψηφιακής μονάδας. [Petrantonakis (2004α)]

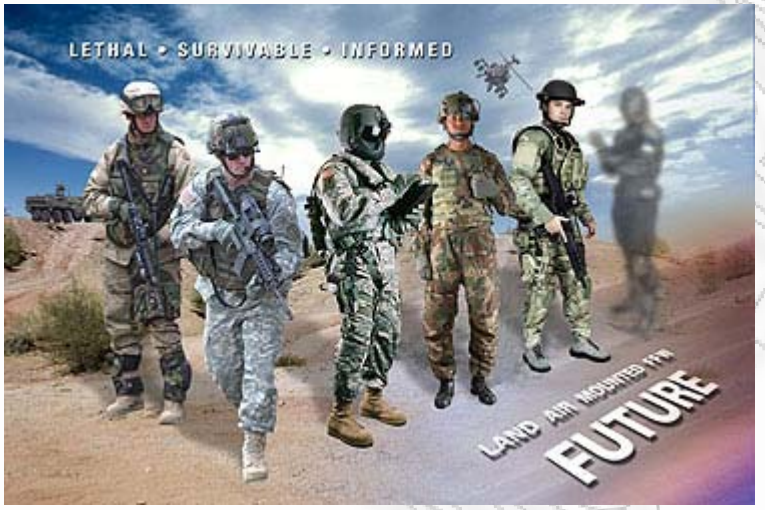
Τα πράγματα, ωστόσο, συνήθως δεν είναι πάντα τόσο ιδανικά κατασκευασμένα. Ο κίνδυνος κατάρρευσης ελλοχεύει παντού. Μια λανθασμένη λειτουργία μπορεί να προκαλέσει βλάβη στο ψηφιακό σύστημα, η οποία να αποβεί μοιραία για τη ζωή του ψηφιακού στρατιώτη ειδικά, όταν αυτός λειτουργεί σε ένα σύγχρονο απαιτητικό πεδίο μάχης με παραμέτρους και δεδομένα που συνεχώς μεταβάλλονται.

Κατά πόσον, όμως, μπορούν αυτά τα συστήματα να είναι αξιόπιστα και ακριβή; Είναι ένα θεμελιώδες ερώτημα, που αποτέλεσε το εφαλτήριο για τη μελέτη της Διοίκησης Κίνδυνου σε στρατιωτικά Ψηφιακά Συστήματα και τη δημιουργία ενός νέου μοντέλου Πρόληψης, Αντιμετώπισης και Ελαχιστοποίησης Καταστροφικών γεγονότων. Ενός μοντέλου πρωτοποριακού που προσαρμόζεται εύκολα σε οποιοσδήποτε Ψηφιακό σύστημα, ενώ με καίριες αλλαγές προσαρμόζεται σε πληθώρα άλλων μη ψηφιακών συστημάτων.

4.2 ΔΟΜΗ ΨΗΦΙΑΚΟΥ ΣΤΡΑΤΙΩΤΗ

Τα επιτεύγματα της Πληροφορικής δεν ήταν δυνατόν να αφήσουν ανεπηρέαστες **τις Ένοπλές Δυνάμεις**. Και οι τρεις κλάδοι των Ενόπλων Δυνάμεων χρησιμοποιούν ευρύτατα **ψηφιακές εφαρμογές,** αλλάζοντας την έως τώρα ροή εργασιών των επιχειρήσεων και των επιτελείων τους.

Σε πολλές περιπτώσεις, οι εφαρμογές αυτές πετυχαίνουν τη **διασυλλογικότητα και διεπικοινωνία** των κλάδων, με σκοπό την άψογη συνεργασία τους και την αύξηση της επιχειρησιακής τους ικανότητας, παρέχοντας τους ένα σημαντικό τακτικό πλεονέκτημα.



Παραλλαγές ψηφιακού Στρατιώτη
Εικόνα 4.1 (Πηγή General Dynamics)

Ο κλάδος που θα εστιάσουμε την μελέτη μας είναι ο Στρατός Ξηράς, καθώς είναι εκείνος που υπάρχει ανεξάρτητα των άλλων. Ο σύγχρονος ψηφιακός στρατιώτης αποτελεί σήμερα μια κινούμενη μάχιμη πληροφοριακή μονάδα, άλλοτε τμήμα ενός ευρύτερου στρατιωτικού σχηματισμού και άλλοτε ως ανεξάρτητο τμήμα.

Ο ψηφιακός στρατιώτης είναι ένα μόνο τμήμα της σύγχρονης δομής των Ένοπλων δυνάμεων. Ιδιαίτερα, στις Ηνωμένες Πολιτείες της Αμερικής, όπου πρωτοϋλοποιήθηκε το σχέδιο, έχουν εμφανιστεί διάφορες παραλλαγές του αρχικού σχεδίου. Οι αποκλίσεις αυτές οφείλονται στις ιδιαιτερότητες που αντιμετωπίζουν τα διάφορα στρατιωτικά τμήματα στο σύγχρονο θέατρο των επιχειρήσεων, πράγμα που αποδεικνύει ότι ο Ψηφιακός στρατιώτης δεν είναι ένα στατικό Ψηφιακό Σύστημα Ενόπλων Δυνάμεων, αλλά ένα σύγχρονο δυναμικά μεταβαλλόμενο και εξελίξιμο σύστημα, το οποίο οφείλει να προσαρμόζεται στις αντίξοες συνθήκες λειτουργίας του εξωτερικού περιβάλλοντος. Οι δυνατότητες που παρέχονται σήμερα σε ένα στρατιώτη είναι εκπληκτικές και πέρα από τα συμβατικά δεδομένα. Αναλύοντας τη δομή του ψηφιακού στρατιώτη, θα γίνει αντιληπτό ότι ξεφεύγει από τα καθιερωμένα στρατιωτικά στερεότυπα. Τα υποσυστήματα του ψηφιακού στρατιώτη είναι τα ακόλουθα:

- **όπλα,**
- **φορητός υπολογιστής,**
- **οθόνη,**
- **ποντίκι,**
- **κάρτα,**
- **μπαταρίες,**
- **μια μονάδα επικοινωνίας και προσανατολισμού,**
- **ένα δέκτη υπολογισμού θέσης (GPS),**
- **μια συσκευή επεξεργασίας ψηφιακών σημάτων,**
- **μια συσκευή με αδρανειακό αισθητήρα,**
- **δίκτυο.**

Αναλυτικότερα, ο οπλισμός του απέχει κατά πολύ από το μέχρι τώρα συμβατικό. Αρχικά, το σύστημα περιλαμβάνει ένα **φορητό υπολογιστή ενσωματωμένο στη στολή** του στρατιώτη. Ο υπολογιστής αυτός αποτελεί το κεντρικό σημείο ελέγχου των λειτουργιών των περιφερειακών ψηφιακών συσκευών, ενώ παράλληλα είναι το κέντρο της επικοινωνίας και της λήψης αποφάσεων. Αυτόματα γίνεται αντιληπτό ότι ο υπολογιστής αυτός αποτελεί **κεντρικό σημείο αυτοκατάρρευσης** του συστήματος του ψηφιακού στρατιώτη και επομένως το πιθανό **μεταλλακτικό του σημείο** για μετατροπή σε συμβατικό στρατιώτη!

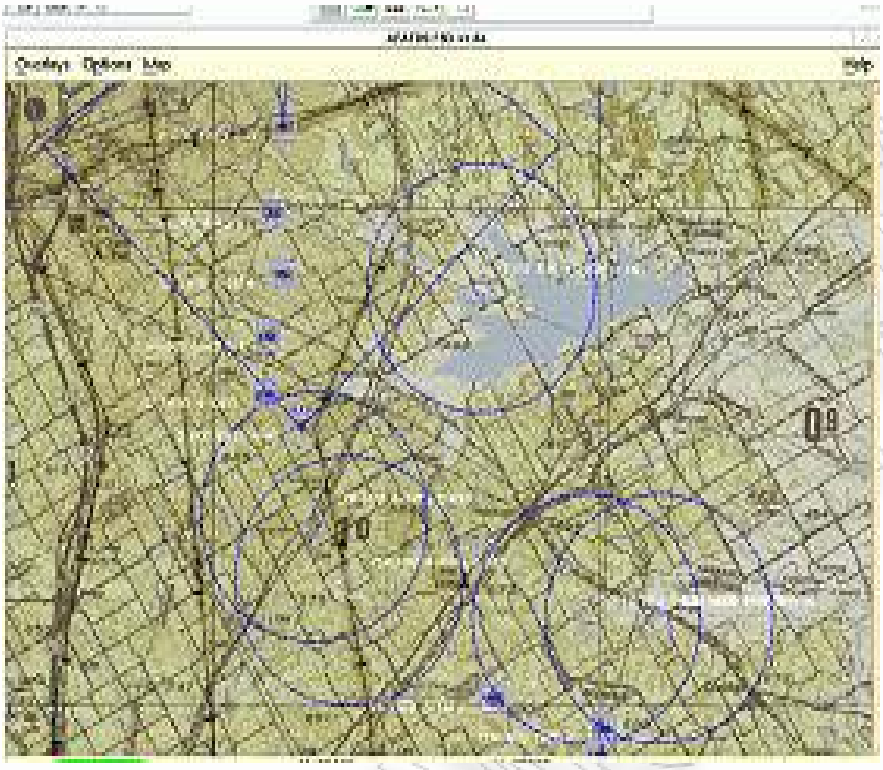
Εικόνα 4.2
Οπλισμός ψηφιακού στρατιώτη

Οι περιφερειακές ψηφιακές συσκευές που φέρει μαζί του ο Land Warrior είναι μια **οθόνη προσαρμοσμένη στο κράνος** μέσα από την οποία ο στρατιώτης έχει τη δυνατότητα να λαμβάνει οπτικοποιημένες τις εντολές δράσης και να παρακολουθεί την εξέλιξη και τη θέση των υπολοίπων φίλιων τμημάτων.

Εικόνα 4.3
Μονάδες σε άσκηση

Παράλληλα, μέσω της οθόνης του έχει τη δυνατότητα εμφάνισης τρισδιάστατων χαρτών, καθώς και πολυμεσικών εφαρμογών. Για τη διαχείριση των γραφικών των ψηφιακών εφαρμογών χρησιμοποιεί **ποντίκι ραμμένο στο στήθος της στολής,** ενώ για τη λειτουργία του υπολογιστή χρησιμοποιεί **κάρτα,** η οποία ενεργοποιεί το φορητό υπολογιστή. Έτσι, επιτυγχάνεται μεγαλύτερη ασφάλεια, αφού ο υπολογιστής δεν μπορεί να τεθεί σε λειτουργία από οποιονδήποτε άλλον καταφέρει να τον αποκτήσει χωρίς την κάρτα. [Macedonia 2002]

Δύο άλλα τμήματα του ψηφιακού στρατιώτη ιδιαίτερα σημαντικά για την ορθή λειτουργία του συστήματος είναι το **σύστημα επικοινωνίας και προσανατολισμού και οι συσσωρευτές του συστήματος.** Το σύστημα επικοινωνίας, το οποίο καθοδηγείται από τον κεντρικό υπολογιστή, περιλαμβάνει **μια μονάδα επικοινωνίας και προσανατολισμού, ένα δέκτη υπολογισμού θέσης (GPS), μια συσκευή επεξεργασίας ψηφιακών σημάτων** και τέλος, και **μια συσκευή με αδρανειακό αισθητήρα για την παροχή της θέσης,** όταν το GPS δεν μπορεί να λειτουργήσει.



Εικόνα 4.4
Είδος λογισμικού του Ψηφιακού Στρατιώτη

Όλα τα παραπάνω για να δουλέψουν χρειάζονται **ένα δίκτυο,** το οποίο χρησιμοποιεί ένα εύκαμπτο πλαστικό κύκλωμα ραμμένο στο υλικό του στρατιώτη. Για να λειτουργήσει το σύστημα χρειάζεται μικρές μπαταρίες με όσο το δυνατό μεγαλύτερη αυτονομία.

Εικόνα 4.5
Οπλισμός του Ψηφιακού Στρατιώτη

Φυσικά, από τον οπλισμό δε λείπουν και τα **συμβατικά όπλα,** τα οποία έχουν τροποποιηθεί, ώστε να χρησιμοποιούν και αυτά με τη σειρά τους **ψηφιακές συσκευές, όπως σκόπευτρα Laser, πυξίδες υπέρυθρες κτλ.** [Rapp 2001].

4.3 ΨΗΦΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

Ο ψηφιακός στρατιώτης είναι η αιχμή του δόρατος των Ενόπλων Δυνάμεων του εικοστού πρώτου αιώνα. Είναι μια εφαρμογή των πλέον σύγχρονων οπλικών συστημάτων που δίνει ένα σημαντικό πλεονέκτημα στο πεδίο της μάχης. Η δομή του Land Warrior συνεχώς μεταβάλλεται και εξελίσσεται. Το πρώτο βήμα έγινε στις Ηνωμένες Πολιτείες με την κατασκευή του πρώτου μοντέλου την έκδοση 1.0 το 2003. Αργότερα νέες βελτιώσεις και δοκιμές μας έφτασαν στην τωρινή έκδοση του οπλικού συστήματος 3.0, η οποία είναι η πιο εξελιγμένη μορφή, ενώ φημολογείται ότι ήδη τέθηκε σε δοκιμαστική επιχειρησιακή λειτουργία στον πόλεμο του Ιράκ.

Η σημαντικότητα αυτού του νέου είδους ‘οπλικού συστήματος’, το οποίο αποτελείται κατά μεγάλο ποσοστό από ψηφιακά υποσυστήματα, είναι μεγάλη. Η κυβέρνηση των Ηνωμένων Πολιτειών αποσκοπεί στην περαιτέρω αναβάθμιση του μέχρι το 2010 στην έκδοση 4.0 όπου πλέον θα γίνει ο εφοδιασμό του στρατεύματος. Μια έκδοση που θα είναι πιο ευέλικτη, με λιγότερο βάρος οπλισμού και μεγαλύτερη αυτάρκεια. Παράλληλα, και άλλες εφαρμογές του ψηφιακού στρατιώτη αναπτύσσονται, προκειμένου να καλύψουν κενά που μπορεί να εμφανιστούν. Κενά που δημιουργούνται από το γεγονός ότι τα σύγχρονα πεδία μαχών είναι ασταθή, πολύπλοκα και το πιο σημαντικό πολύμορφα.Αυτή η πολυμορφία κάνει τον Land Warrior ευάλωτο σε κινδύνους, τους οποίους δεν μπορεί να προβλέψει και να αντιμετωπίσει. Το κενό αυτό έρχεται να καλύψει μια παραλλαγή του ψηφιακού στρατιώτη και συγκεκριμένα ο Urban Warrior, ο οποίος αποτελεί μεταλλαγμένη μορφή του Land Warrior ικανή να αντιμετωπίσει κρίσεις σε πολύπλοκα περιβάλλοντα λειτουργίας, όπως πόλεις ή δύσβατες περιοχές, όπου ο δείκτης επικινδυνότητας και θνησιμότητας του ψηφιακού στρατιώτη αυξάνεται σημαντικά.

Ο εξοπλισμός του Ψηφιακού στρατιώτη είναι ένα ενδιαφέρον, πρωτοποριακό πληροφοριακό σύστημα Εθνικής Άμυνας. Έτσι και άλλες χώρες αναπτύσσουν τα δικά τους συστήματα εξοπλισμού ψηφιακού στρατιώτη, όπως η Αγγλία (FIST), η Γερμανία, η Ολλανδία, η Ισπανία, η Γαλλία (FELIN), η Σουηδία (MARKUS) και η Νότιος Αφρική.

Ο ψηφιακός στρατιώτης δεν είναι το μόνο οπλικό σύστημα το οποίο έχει ψηφιοποιηθεί. Η χρήση και ο χειρισμός ενός τέτοιου συστήματος απαιτεί υψηλές γνώσεις και άρτια κατάρτιση. Η εκπαίδευση

είναι αναπόσπαστο κομμάτι της ολοκληρωμένης και επιτυχημένης λειτουργίας του Συστήματος Land Warrior.

Σήμερα, που η Πληροφορική έχει κατακλύσει όλους τους τομείς των Σύγχρονων Συστημάτων, η στρατιωτική εκπαίδευση αλλάζει μορφή. Γίνεται περισσότερο αυτοματοποιημένη και εύχρηστη για το χρήστη. Έτσι, εξοικονομείται χρήμα, προσομοιώνονται και δημιουργούνται εικονικά πεδία μάχης, ελαχιστοποιείται το κόστος απωλειών σε ανθρώπινο δυναμικό, αλλά παράλληλα και το κόστος χρήσης αληθινού οπλισμού. Τα σύγχρονα συστήματα εξομοίωσης μάχης εκπαιδεύουν τις στρατιωτικές δυνάμεις σε πραγματικά δύσκολες συνθήκες μάχης. Εξομοιώνουν καταστάσεις στις οποίες ο μαχητής καλείται να δοκιμαστεί σωματικά, πνευματικά για την επίτευξη του επιθυμητού νικηφόρου αποτελέσματος.

4.4 ΚΙΝΔΥΝΟΙ ΚΑΤΑΡΡΕΥΣΗΣ ΣΥΣΤΗΜΑΤΟΣ ΨΗΦΙΑΚΟΥ ΣΤΡΑΤΙΩΤΗ

Για να μπορέσουμε να αναλύσουμε καλύτερα τη δομή του ψηφιακού στρατιώτη, αλλά και για να αντιμετωπίσουμε τους κίνδυνους, που τον απειλούν, οφείλουμε να μελετήσουμε το πρόβλημα υπό το πρίσμα της μελέτης **ενός πολύπλοκου, συνεχώς μεταλλασσόμενου και εξελικτικού Πληροφοριακού Συστήματος,** το οποίο αλληλεπιδρά σε ένα δυναμικά μεταβαλλόμενο πεδίο.

4.4.1 ΑΠΕΙΛΕΣ ΔΕΔΟΜΕΝΩΝ

Η σημερινή πολύπλοκη δομή των πληροφοριακών συστημάτων εθνικής άμυνας έχει σαν αποτέλεσμα την αύξηση των απειλών. Έτσι, οι απειλές μπορούν να κατηγοριοποιηθούν σε ένα ευρύτερο σύνολο, αφού η δομή τους, η λειτουργία τους αλλά και τα καταστροφικά τους αποτελέσματα συνεχώς αλλάζουν. Συνεπώς, αυτό συνεπάγεται λήψη επιπλέον αντιμέτρων για την αντιμετώπισή τους. Διαχωρίζουμε, λοιπόν, τις απειλές που δέχεται ένα πληροφοριακό σύστημα σε δύο κατηγορίες. Τις απειλές υλικού και τις απειλές λογισμικού, χωρίς η ύπαρξη του ενός να αποκλείει το άλλο[NIST (2001)].

4.4.2 ΑΠΕΙΛΕΣ ΥΛΙΚΟΥ

Το υλικό (hardware) ενός πληροφοριακού συστήματος είναι ευάλωτο σε απειλές, οι οποίες συχνά είναι απρόβλεπτες, όπως για παράδειγμα οι κίνδυνοι, που μέχρι πριν ένα χρόνο δεν λαμβάνονταν

υπόψη, στα Συστήματα Ανάκτησης από Πτώσεις. Μια βαθύτερη ανάλυση των πτώσεων υλικού μας οδηγεί στο συμπέρασμα ότι αυτές θα πρέπει να κατηγοριοποιηθούν σε δύο επιμέρους κατηγορίες:

- **τις φυσικές,**
- **τις τεχνητές**.

4.4.2.1 ΦΥΣΙΚΕΣ ΠΤΩΣΕΙΣ ΥΛΙΚΟΥ

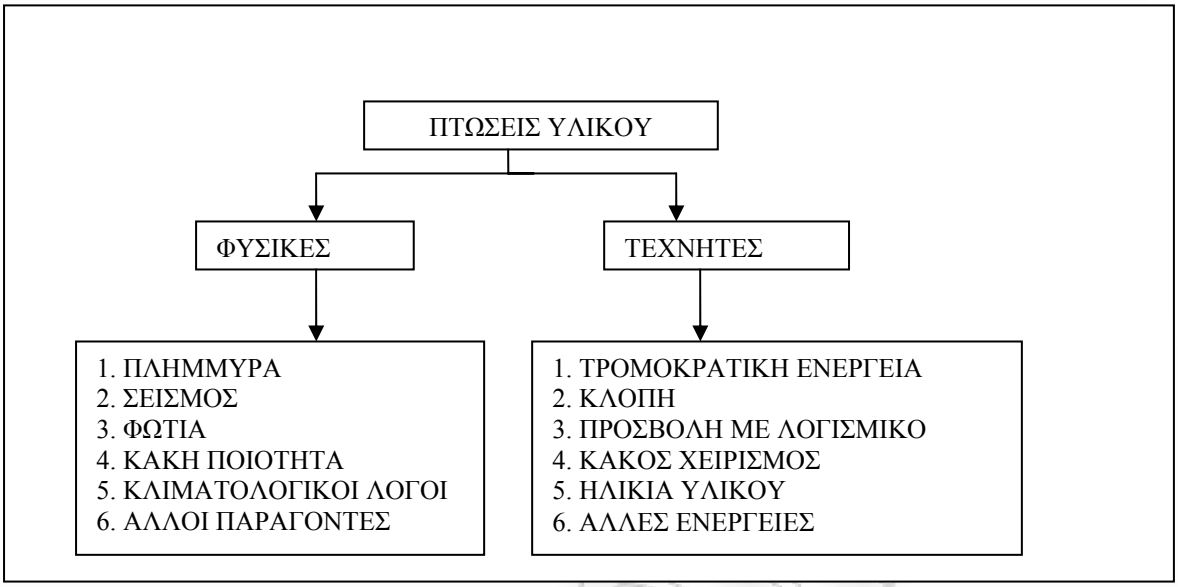
Παρόλο που τα όρια μεταξύ αυτής της κατηγορίας και εκείνης της τεχνητής πτώσης υλικού δεν είναι ξεκάθαρα, θα μπορούσαμε να επισημάνουμε το γεγονός ότι οι φυσικές πτώσεις υλικού είναι αυτές οι οποίες ποτέ δεν λαμβάνουμε σοβαρά υπόψη και πάντα απευχόμαστε να συμβούν.

Τέτοιες είναι συνήθως οι φυσικές καταστροφές των εγκαταστάσεων του συστήματος και οι καταστροφές του υλικού μετά από πλημμύρα, σεισμό ή φωτιά. Ακόμη, σε αυτήν την κατηγορία περιλαμβάνονται και οι περιπτώσεις καταστροφής τμημάτων του εξυπηρετητή από δυσλειτουργία, από κακή εγκατάσταση, κακή ποιότητα υλικού, και αυξομειώσεις της τάσης του ηλεκτρικού ρεύματος.

4.4.2.2 ΤΕΧΝΗΤΕΣ ΠΤΩΣΕΙΣ ΥΛΙΚΟΥ

Συνήθως, αυτού του είδους οι πτώσεις οφείλονται κατά μεγάλο ποσοστό στον ανθρώπινο παράγοντα. Στην πλειονότητα των περιπτώσεων πρόκειται για καταστροφή ενός εξυπηρετητή κόμβου δικτύου από τρομοκρατική ενέργεια είτε με την καταστροφή των εγκαταστάσεων, (ανατίναξη, καταστροφή, κλοπή εξυπηρετητή) είτε με τη χρήση κατάλληλου λογισμικού που προκαλεί ζημιά σε τμήμα του υλικού του εξυπηρετητή (δίσκος, μητρική, τροφοδοτικό).

Μια γενικότερη εικόνα και ολοκληρωμένη άποψη της κατάστασης μπορούμε να έχουμε στο ακόλουθο σχήμα όπου βλέπουμε τις κατηγορίες και την ιεράρχηση τους.



Σχήμα 4.1
Κατηγοριοποίηση πτώσεων Υλικού

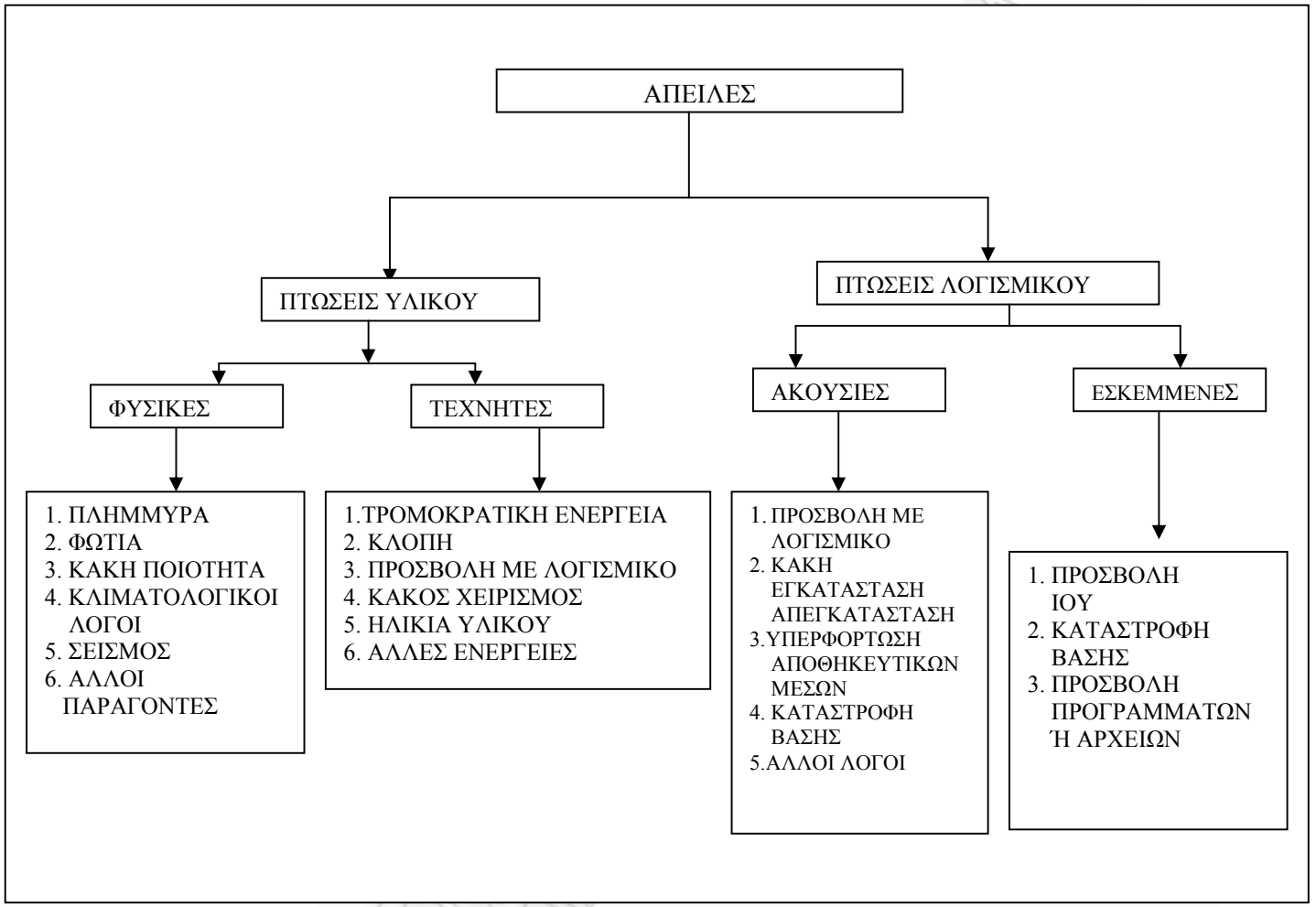
4.4.3 ΠΤΩΣΕΙΣ ΛΟΓΙΣΜΙΚΟΥ

Οι συχνότερες και ευκολότερα υλοποιήσιμες απειλές ενός πληροφοριακού συστήματος είναι αυτές που αποσκοπούν στην πτώση λογισμικού. Η επικινδυνότητα τους είναι μεγάλη, αφού τα αποτελέσματα και η υλοποίησή τους είναι άμεση, με μικρό κόστος, αλλά τεράστιες οικονομικές συνέπειες ιδιαίτερα σε κατανεμημένες διαδικτυακές βάσεις μεγάλων οικονομικών εταιρειών ή κυβερνητικών και στρατιωτικών οργανισμών. Οι πτώσεις αυτές μπορούν να προκληθούν είτε ηθελημένα είτε ακούσια και δομούνται ως εξής:

- Πτώσεις λογισμικού από καταστροφή εξαιτίας ιού.
- Πτώση λογισμικού από καταστροφή αρχείων λόγω κακής εγκατάστασης ή απεγκατάστασης προγραμμάτων.
- Πτώση λογισμικού σε μεγάλες βάσεις δεδομένων, οφειλόμενες σε καταστροφή πινάκων, σχέσεων πινάκων ή διαγραφή κεντρικής βάσης δεδομένων από κακή χρήση του προγράμματος διαχείρισης της βάσης.
- Πτώση λογισμικού από έλλειψη επάρκειας υλικού (υπερφόρτωση αποθηκευτικών μέσων ή ασυμβατότητα υλικού-λογισμικού).

Συνοψίζοντας, λοιπόν, και καταλήγοντας όσον αφορά την κατηγοριοποίηση των απειλών των δεδομένων μας, μπορούμε να σχηματοποιήσουμε τα όσα αναφέραμε μέχρι στιγμής στο ακόλουθο

σχήμα (Σχήμα 4.2). Αξίζει να αναφερθεί ότι πολλά είναι τα σύγχρονα εργαλεία που προσπαθούν να απαλείψουν αυτά τα μειονεκτήματα.

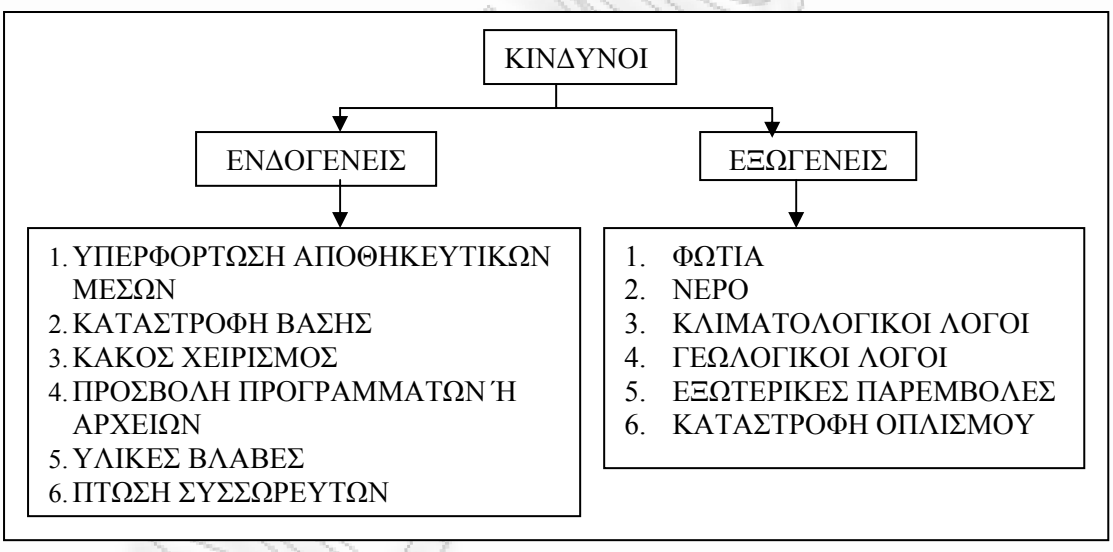


Σχήμα 4.2
Κατηγοριοποίηση Απειλών Υλικού - Λογισμικού

Στην προηγούμενη παράγραφο παρουσιάσαμε τη δομή και ειδικότερα τον οπλισμό του Land Warrior. Περιγράφοντας τον εξοπλισμό, παρατηρούμε τα εξής δύο σημαντικά τμήματα. Πρώτον, την ύπαρξη ενός κεντρικού υπολογιστή με τον οποίο ο ψηφιακός στρατιώτης χειρίζεται τις επιμέρους ψηφιακές συσκευές και δεύτερον, τη χρήση κατάλληλου λογισμικού για την ψηφιοποίηση των λειτουργιών του Πληροφοριακού του συστήματος.

Το ρίσκο κατάρρευσης του συστήματος είναι υπαρκτό. Γίνεται μεγάλο δε, αν λάβουμε υπόψη τους κινδύνους που αντιμετωπίζει ο ψηφιακός στρατιώτης, δρώντας μέσα σε ένα ασταθές, εχθρικό περιβάλλον του οποίου τα δεδομένα μεταβάλλονται συνεχώς.

Αναλυτικότερα, οι κίνδυνοι που απειλούν τον ψηφιακό στρατιώτη χωρίζονται σε δύο μεγάλες κατηγορίες: στους **Ενδογενείς** και στους **Εξωγενείς** κινδύνους, όπως φαίνονται και στο σχήμα που ακολουθεί.



Σχήμα 4.3

Με τον όρο Ενδογενείς, όπως φαίνεται και στο Σχήμα 4.3, εννοούμε τους κινδύνους που απειλούν το Πληροφοριακό Σύστημα και οι οποίοι προέρχονται από **εσωτερικές διεργασίες και μη ομαλές – εσωτερικές και αυτές- μεταλλακτικές καταστάσεις**. Από την άλλη πλευρά, εξωγενείς ονομάζονται οι κίνδυνοι που απειλούν το Πληροφοριακό Σύστημα και που οφείλονται σε **επιδράσεις εξωγενών από το σύστημα παραγόντων**, όπου και αυτοί με τη σειρά τους οδηγούν σε μη ομαλές μεταλλακτικές καταστάσεις. Και οι δύο τύποι κινδύνων αναλύονται στο τρίτο επίπεδο του Σχήματος 4.3.

Παρατηρούμε ότι και οι δύο κατηγορίες κινδύνων με την πραγματοποίησή τους οδηγούν στο ίδιο αποτέλεσμα, δηλαδή στην οριστική κατάρρευση του Πληροφοριακού συστήματος του Ψηφιακού Στρατιώτη. Αυτό έχει σαν αποτέλεσμα τη διάλυση των **δομών συνεκτικότητας** μεταξύ των υποσυστημάτων του. Πολλές φορές, ωστόσο, επιτυγχάνεται εκ παραδρομής ή εσκεμμένα η μερική κατάρρευση του συστήματος και η μετάβασή του σε ένα διαφορετικό επίπεδο δομής και λειτουργίας. Ένα επίπεδο που περιλαμβάνει **Ανώμαλους Χώρους Δεδομένων,** οι οποίοι οδηγούν με την σειρά τους σε ανεξέλεγκτες αλυσιδωτές **χαοτικές καταστάσεις.**

4.5 ΔΙΟΙΚΗΣΗ ΚΙΝΔΥΝΟΥ ΣΕ ΨΗΦΙΑΚΑ ΣΥΣΤΗΜΑΤΑ ΕΘΝΙΚΗΣ ΑΜΥΝΑΣ

Ο σύγχρονος ψηφιακός στρατιώτης είναι ένα ιδιαίτερα εξελιγμένο οπλικό σύστημα, το οποίο λειτουργεί σε οποιοδήποτε θέατρο επιχειρήσεων και αν απαιτηθεί. Το πληροφοριακό αυτό σύστημα, όπως είπαμε και προηγουμένως, λειτουργεί κάτω από αντίξοες συνθήκες. Αντιμετωπίζει τις πιο δύσκολες καταστάσεις, αφού στην συντριπτική πλειοψηφία των περιπτώσεων το περιβάλλον λειτουργίας είναι **ιδιαίτερα απαιτητικό και πολύ εχθρικό**. Εχθρικό και μη στατικό με δεδομένα και λειτουργίες που αλλάζουν συνεχώς το status quo του χώρου, μέσα στον οποίο δρα ο ψηφιακός στρατιώτης. Οι αλλαγές είναι δυναμικές, ασταθείς, διακριτές και διαρκείς. Η συνεχής μεταβλητότητα του χώρου και η αστάθεια του δεν μπορούν να προβλεφθούν, γεγονός που κάνει ακόμα πιο δύσκολη τη λειτουργία του Ψηφιακού Συστήματος Εθνικής Άμυνας.

Οι δυνάμεις που δημιουργούνται και ασκούνται πάνω σε ένα σύστημα, το οποίο δρα μέσα σε ένα τέτοιο περιβάλλον λειτουργίας, δεν μπορούν να υπολογιστούν μέσα από συμβατικά μοντέλα πρόληψης. Τέτοιες χαοτικές καταστάσεις καθιστούν αδύνατο τον υπολογισμό μιας καταστροφικής δύναμης για αυτό το λόγο είναι επιτακτική η ανάγκη δημιουργίας ενός μοντέλου αντιμετώπισης κρίσεων. Ένα σύστημα υποστήριξης άμεσων αποφάσεων, που θα συμβουλεύει το χρήστη για την αποφυγή δυσμενών καταστάσεων.

Τα σύγχρονα συστήματα εκπαίδευσης στρατιωτών στηρίζονται στην εκπαίδευση του μαχητή σε προσομοιωμένες συνθήκες μάχης. Προγράμματα επιβίωσης, χρήσης οπλισμού, τακτικής, εφοδιασμού είναι η βασική θεματολογία. Πιο πολύπλοκα προγράμματα εξομοιώνουν κατάστασης μάχης και παράλληλα διοίκησης στρατιωτικών τμημάτων, μονάδων, λόχων ή ακόμα και ολόκληρου στρατεύματος μέσα από τη

χρήση παιχνιδιών προσομοίωσης μαχών.(Land Warrior, Delta Force 2, DOOM, etc).

Όπως απορρέει από τα προηγούμενα, είναι αναγκαία η δημιουργία ενός προγράμματος το οποίο θα αναλύει το ρίσκο που αντιμετωπίζει ο Ψηφιακός στρατιώτης σε ένα εχθρικό επιχειρησιακό περιβάλλον. Μέσα από σενάρια διοίκησης κίνδυνου ο ψηφιακός στρατιώτης θα μπορεί να γνωρίζει τις δυνατότητες του, τις αδυναμίες του, και τις ικανότητες της μονάδος του. Παράλληλα, ένα τέτοιο πρόγραμμα θα μπορεί να προτείνει λύσεις αντιμετώπισης κρίσεων. Το πιο σημαντικό, όμως είναι το πρόγραμμα να έχει τη δυνατότητα σε κρίσιμα σημεία καταστροφής να προτείνει λύσεις. Όταν, για παράδειγμα, υλοποιούνται καταστροφικές δυνάμεις σε ένα εχθρικό θέατρο επιχειρήσεων είναι αναγκαία η βοήθεια, ώστε να ληφθούν οι καλύτερες δυνατές λύσεις με το ελάχιστο κόστος τόσο σε ανθρώπινο δυναμικό όσο και σε υλικές απώλειες. Η μεθοδολογία Διοίκησης Κινδύνου σε Πληροφοριακά Συστήματα Εθνικής Άμυνας, που παρουσιάστηκε σε προηγούμενο κεφάλαιο, βρίσκει εδώ την εφαρμογή της.

4.6 ΣΥΜΠΕΡΑΣΜΑΤΑ

Ο Ψηφιακός Στρατιώτης είναι ένα υπερσύγχρονο, εξελισσόμενο οπλικό σύστημα. Δημιουργήθηκε για να καλύψει τις σύγχρονες απαιτήσεις των θεάτρων μάχης. Οι εφαρμογές της πληροφορικής εξέλιξαν και μετάλλαξαν το συμβατικό οπλίτη σε ψηφιακό, προσφέροντας τους μεγάλα πλεονεκτήματα στο πεδίο της μάχης.

Η σημαντική αυτή αλλαγή μπορεί να επηρεάσει συνολικά τη δομή των Ένοπλων Δυνάμεων του 21^{ου} αιώνα. Με τις κατάλληλες αλλαγές, ρυθμίσεις και προσθήκες ο ψηφιακός στρατιώτης μπορεί από απλή επιχειρησιακή μονάδα να γίνει μέρος ενός μεγαλύτερου ψηφιακού επιχειρησιακού πληροφοριακού συστήματος και να δρα σε ένα εχθρικό, δυναμικό πεδίο μάχης. Οι προσθήκες αυτές έχουν την δυνατότητα να συμβάλλουν στη δημιουργία ψηφιακών μονάδων, λόχων, ακόμα και ψηφιακής διοίκησης στρατιωτικού τμήματος. **Ψηφιακά επιτελεία** και ειδικά μοντέλα διοίκησης θα μπορούσαν να υλοποιηθούν κάτω από αυστηρές προϋποθέσεις. Παράλληλα, η μοντελοποίηση αυτή θα μπορεί να αλλάξει και τη δομή των άλλων δυο Όπλων. Νέα συστήματα, όπως ο **ψηφιακός ναύτης, η ψηφιακή φρεγάτα, και ανώτερα μοντέλα όπως η ψηφιακή διοίκηση στόλου,** θα μπορέσουν να εκσυγχρονίσουν το Νέο Πολεμικό Ναυτικό. Όμοια, ο **ψηφιακός σμηνίτης, ο ψηφιακός**

αεροπόρος, τα μη επανδρωμένα ψηφιακά σμήνη είναι το μέλλον της Πολεμικής Αεροπορίας.

Όπως γίνεται εύκολα αντιληπτό και όπως συμβαίνει σε κάθε πληροφοριακό σύστημα, υπάρχουν κίνδυνοι που απειλούν την ομαλή και απρόσκοπτη λειτουργία του. Κίνδυνοι που μπορούν να αλλάξουν τη δομή του συστήματος, υπονομεύοντας την επιχειρησιακή του ικανότητα και κίνδυνοι ικανοί να οδηγήσουν ακόμα και στην κατάρρευση του ψηφιακού συστήματος, πράγμα που θα αποβεί μοιραίο για τη ζωή του ίδιου του στρατιώτη.

Η γεννήτρια των κίνδυνων είναι το ίδιο το περιβάλλον λειτουργίας του συστήματος. Ένα περιβάλλον δυναμικά μεταβαλλόμενο, με ασταθή δομή, που επιδρά αρνητικά στη συνεκτικότητα των υποσυστημάτων του Land Warrior. Είναι αναγκαία, λοιπόν, η δημιουργία ενός μοντέλου διαχείρισης κρίσεων. Ενός μοντέλου που θα εκπαιδεύσει τον ψηφιακό στρατιώτη στη σωστή λήψη αποφάσεων σε στιγμές κρίσης. Παράλληλα, θα του δώσει τη δυνατότητα να ελαχιστοποιήσει το κόστος απώλειας των δικών του υποσυστημάτων ή του ευρύτερου συστήματος, τμήμα του οποίου λειτουργεί στο σύγχρονο θέατρο επιχειρήσεων.

ΚΕΦΑΛΑΙΟ 5

ΥΛΟΠΟΙΗΣΗ ΣΕΝΑΡΙΩΝ ΚΡΙΣΕΩΣ ΣΕ ΨΗΦΙΑΚΑ ΣΥΣΤΗΜΑΤΑ ΕΘΝΙΚΗΣ ΑΜΥΝΑΣ

5. ΥΛΟΠΟΙΗΣΗ ΣΕΝΑΡΙΩΝ ΚΡΙΣΕΩΣ

5.1 ΕΙΣΑΓΩΓΗ

" Πείρα είναι το όνομα που δίνει ο καθένας στα λάθη του"

Ο. Ουάιλντ

Τα σύγχρονα πολύπλοκα ψηφιακά συστήματα αντιμετωπίζουν, όπως προαναφέρθηκε, μια πληθώρα κινδύνων οι οποίοι απειλούν την ομαλή λειτουργία τους. Η Μεθοδολογία Διαχείρισης Κινδύνου αντιμετωπίζει με επιτυχία τέτοια κινδυνικά φαινόμενα που μπορεί να εμφανιστούν.

Η αστάθεια του χώρου λειτουργίας των ψηφιακών συστημάτων είναι ο γεννήτορας των καταστροφικών δυνάμεων που εμφανίζονται στον ορίζοντα χρονοπρογραμματισμού του συστήματος S. Η εμφάνιση τους γίνεται σε μη τακτά χρονικά διαστήματα. Είναι, κυρίως, μη προβλέψιμες καταστροφικές δυνάμεις, με συνέπεια αυτή ακριβώς η έλλειψη προβλεψιμότητας να τις καθιστά ιδιαίτερα επικίνδυνες.

Είναι επιτακτική η ανάγκη το σύστημα διαχείρισης κινδύνου να είναι σε ετοιμότητα, καθώς μια επιτυχημένη ανάκτηση, μια επιτυχημένη αποτροπή απαιτεί ένα σύστημα ετοιμοπόλεμο. Άμεση συνέπεια των παραπάνω είναι η ανατροφοδότηση μέσα από συνεχή σενάρια κρίσεως. Η εκπαίδευση του συστήματος αποτελεί προϋπόθεση για την επιτυχημένη λειτουργία του, ενώ η προσομοίωση των σεναρίων κρίσεως είναι απαιτούμενη για την ολοκληρωμένη προστασία του.

5.2 ΠΡΟΒΛΕΠΟΝΤΑΣ ΜΙΑ ΚΡΙΣΗ

Ένα ολοκληρωμένο σύστημα διαχείρισης κρίσεως οφείλει να είναι αποτελεσματικό. Η αποτελεσματικότητα του οφείλεται στην ικανότητα του να μπορεί να αντιμετωπίζει πιθανές απειλές που εμφανίζονται μέσα στον ορίζοντα λειτουργίας του συστήματος. Για να μπορέσει το σύστημα να αντιμετωπίσει τις απειλές που εμφανίζονται μέσα στο διάστημα Δt, πρέπει να γνωρίζει το περιβάλλον λειτουργίας, τη δομή του και τους πιθανούς κινδύνους που μπορεί να εμφανιστούν.

Η δυναμικότητα και η αστάθεια του χώρου των δεδομένων δεν επιτρέπουν πάντοτε στο σύστημα διαχείρισης κρίσεως να αποκτήσει ευρεία γνώση του περιβάλλοντος λειτουργίας. Η συνεχής αλλαγή των δεδομένων σε μη τακτά χρονικά διαστήματα δυσχεραίνει τη δυνατότητα ικανοποιητικής πρόβλεψης. Η μεθοδολογία πρόληψης κινδυνικών φαινομένων, λοιπον, δεν πρέπει να μένει στάσιμη και η επιτυχία της οφείλεται στη δυνατότητα που έχει να παρακολουθεί τη δυναμικότητα του χώρου των δεδομένων και να προλαμβάνει πιθανά κινδυνικά φαινόμενα.

Αυτό επιτυγχάνεται με τη συνεχή **προσομοίωση του συστήματος** με πραγματικά δεδομένα. Όσο πιο αληθοφανή είναι τα δεδομένα τόσο πιο αληθοφανές είναι το αποτέλεσμα και, παράλληλα, η πρόβλεψη γίνεται με μεγαλύτερη επιτυχία. Μέσα από τη συνεχή προσομοίωση βελτιώνεται τόσο η λειτουργική ικανότητα της μεθοδολογίας πρόληψης όσο και ο χρήστης - αναλυτής του συστήματος κρίσης.

Μέσα από τη μεθοδολογία πρόληψης κινδυνικών φαινομένων και τα εργαλεία που παρέχει μπορεί να προβλεφθεί οποιαδήποτε κρίση με τη χρήση του καταλληλότερου αντιμέτρου. Φυσικά αυτό προαπαιτεί το σύστημα να είναι πλήρως ενημερωμένο για τα είδη των κινδύνων που απειλούν το υπό μελέτη σύστημα μας, απαγορεύοντας τυχόν αποκλεισμούς. Σε έναν ασταθή δυναμικό χώρο δεδομένων δεν υπάρχουν περισσότερο ή λιγότερο πιθανά συμβάντα. Η προσομοίωση παρέχει στις περισσότερες των περιπτώσεων χρήσιμα συμπεράσματα για την εξέλιξη ενός πιθανού συμβάντος, προετοιμάζοντας τους χρήστες για την αντιμετώπιση του, ενώ παράλληλα η συνεχής προσομοίωση σε αντίξοες συνθήκες λειτουργίας εκπαιδεύει τους χρήστες και βελτιστοποιεί το χρόνο αντίδρασης αυτοματοποιώντας τις διαδικασίες άμυνας.

Χωρίς βλάβη της γενικότητας, στη συνέχεια, θα προσομοιώσουμε ένα τυχαίο σύστημα S, ώστε να δούμε αναλυτικά τα αποτελέσματα της προσομοίωσης και να εξάγουμε χρήσιμα συμπεράσματα. Παράλληλα, θα παρουσιασθεί ένας αλγόριθμος δημιουργίας και εκτέλεσης σεναρίων πρόληψης κρίσεως σε ψηφιακά πληροφοριακά σύστημα.

5.2.1 ΜΟΝΤΕΛΟΠΟΙΗΣΗ

Η δομή των συστημάτων στον πραγματικό κόσμο είναι πολύπλοκη, αφού τα περισσότερα αποτελούνται από δυο ή περισσότερα υποσυστήματα. Ας υποθέσουμε ένα τέτοιο ψηφιακό πληροφοριακό

σύστημα S και το σύνολο των υποσυστημάτων του όπως ορίσθηκε στην (1).

Όπως αναλύθηκε προηγουμένως, ένα σημαντικό βήμα για την αποφυγή κρίσεων είναι η δημιουργία κατάλληλων σεναρίων αποκλιμάκωσης και πρόβλεψης κρίσεως. Αυτό μπορεί να επιτευχθεί με συστηματική προσέγγιση των δεδομένων του χώρου στον οποίο λειτουργεί το υπό μελέτη σύστημα.

Το πρώτο βήμα για τη δημιουργία ενός σεναρίου αποκλιμάκωσης είναι η ιχνηλάτηση του χώρου δεδομένων και η ανίχνευση των καταστροφικών δυνάμενων που δραστηριοποιούνται μέσα σε αυτόν και είναι πιθανό να εμφανιστούν μέσα στον ορίζοντα προγραμματισμού Δt. Το βήμα αυτό είναι ιδιαίτερα σημαντικό, αφού προσφέρει στη μεθοδολογία δεδομένα για μελέτη και ανάλυση. Η σωστή εφαρμογή του θα οδηγήσει στην ορθή σεναριολογία και θα έχει σαν συνέπεια την εξαγωγή ιδιαίτερα σημαντικών συμπερασμάτων για τη μελλοντική εξέλιξη του συστήματος S. Λόγω της δυναμικότητας του χώρου και της αστάθειας των δεδομένων του δεν είναι πάντοτε δυνατή η απόλυτη ανίχνευση των δυνάμεων που δρουν μέσα στο περιβάλλον λειτουργίας, με αποτέλεσμα να ενεργοποιηθούν εν υπνώσει δυνάμεις στο διάστημα Δt, χωρίς να είναι εφικτή η προβλεψιμότητα τους, και να προκαλέσουν δυσάρεστες καταστάσεις οι οποίες δεν μπορούν να υπολογιστούν από κανένα σενάριο. Με τη χρήση των εργαλείων της μεθοδολογίας πρόληψης κινδυνικών φαινομένων, θα επισημανθούν τυχόν πιθανές μεταλλάξεις, καταστροφές και φυσικά πιθανά σημεία αυτοκατάρρευσης του υπό μελέτη συστήματος. Ας υποθέσουμε ότι C είναι το σύνολο των καταστροφικών δυνάμεων που θα τύχει να εμφανιστούν μέσα στο διάστημα Δt, όπως ορίσθηκε στην (2).

5.2.1.1 ΑΛΓΟΡΙΘΜΟΣ ΣΕΝΑΡΙΩΝ ΚΡΙΣΕΩΣ

Όπως αναλύθηκε σε προηγούμενη παράγραφο, τα ψηφιακά πληροφοριακά συστήματα δρουν σε ασταθή, δυναμικά μεταβαλλόμενα περιβάλλοντα εργασίας. Εντούτοις, μέσα από τη δημιουργία σεναρίων κρίσεως δεν υπολογίζεται η αστάθεια ούτε η δυναμικότητα του χώρου. Έτσι, για τον υπολογισμό των ασταθών καταστροφικών δυνάμεων χρησιμοποιούνται χαοτικά μαθηματικά μοντέλα.

Το επόμενο σημαντικό βήμα της διαδικασίας δημιουργίας σεναρίων κρίσεως είναι η υλοποίηση ενός σημαντικού εργαλείου, καταλυτικού για την μετέπειτα ομαλή και αποτελεσματική προσομοίωση

του συστήματος. Το **Διάνυσμα Σεναρίων (Scenario Vector SV)** περιέχει σύνολο διανυσμάτων το καθένα από τα οποία αποτελεί ξεχωριστή οντότητα δεδομένων προσομοίωσης κινδυνικών φαινομένων τα οποία μπορεί να υλοποιηθούν μέσα στο διάστημα Δt. Για να είναι επιτυχημένη η προσομοίωση του συστήματος διαχείρισης κρίσεως, απαιτείται να εκτελεστεί ένας ικανοποιητικός αριθμός σεναρίων προκειμένου να προσδιοριστούν τυχόν αδυναμίες του ψηφιακού πληροφοριακού συστήματος. Εκτός από τα σύνολα S και C, που αναλύθηκαν προηγουμένως, απαιτείται και ο χάρτης κινδύνου X, που αναλύθηκε λεπτομερώς σε προηγούμενο κεφάλαιο.

Κάθε πιθανό σενάριο έχει το δικό του διάνυσμα. Ένα διάνυσμα σεναρίων SV ορίζεται ως εξής:

$SV = (sv_j), \quad \forall j \in C^*$

sv_j *=1, αν η* c_j *καταστροφή ενεργοποιείται, στο παρόν σενάριο.*
=0, διαφορετικά.

Μια επιτυχημένη συνταγή προσομοίωσης σεναρίων κρίσεως παρουσιάζεται στη συνέχεια με την υλοποίηση του **Αλγόριθμου Σεναρίων Κρίσεως (Crisis Scenarios Algorithm (CSA)).** Μέσα από συγκεκριμένα βήματα επιτυγχάνεται η υλοποίηση σεναρίων πρόληψης κρίσεως με τη βοήθεια της μεθοδολογίας πρόληψης κινδυνικών φαινομένων σε πληροφοριακά συστήματα.

Crisis Scenarios Algorithm (CSA)

Βήμα1: Δημιουργία Διανύσματος Σεναρίων SV.
Βήμα2: Εύρεση των ανταποκρινόμενων Διανυσμάτων Καταστροφών CV.
Βήμα3: Υπολογισμός του Δείκτη Τρωτότητας (VI) του Συστήματος S από το CV.
Βήμα4: Έλεγχος του Δείκτη Τρωτότητας (VI), σύμφωνα με Mutation or Total Destruction or Operation Tranquility Proposition.
Βήμα5: Επανάληψη του παραπάνω συνόλου βημάτων, μέχρις ότου δημιουργηθεί ένα κατάλληλο πλήθος από διαφορετικά πιθανά σενάρια.
Βήμα6: Εμφάνιση με γραφική παράσταση του συνόλου των Δείκτη Τρωτότητας (VI).

Το διάνυσμα καταστροφών CV ορίζεται ως ακολούθως:

$CV = (cv_i),\ \forall i \in S^*$
$cv_i = 1$ αν $\exists j \in C^* : sv_j * k_{ij} = 1,$
$= 0$, διαφορετικά.

Το τρίτο βήμα στον αλγόριθμο, που παρουσιάσθηκε στην προηγούμενη παράγραφο, υπολογίζει το Δείκτη Τρωτότητας (VI) του συστήματος S, ο οποίος αναλύθηκε διεξοδικά σε προηγούμενο κεφάλαιο. Μόλις ο δείκτης υπολογισθεί τότε αναλύεται η τιμή του με βάση τα θεωρήματα, που αναλύθηκαν σε προηγούμενο κεφάλαιο. Η τιμή του ελέγχεται, αν ικανοποιεί τις συνθήκες του Θεωρήματος των Μαύρων, Λευκών και των Γκρίζων Λειτουργικών Οπών.

Ειδικότερα, αν ισχύει ότι VI =1 τότε το σύστημα βρίσκεται σε κατάσταση Μαύρης Λειτουργικής Οπής. Το σύνολο των δυνάμεων που αναπαριστώνται από το διάνυσμα CV είναι ένας καταστροφικός συνδυασμός που οδηγεί σε ολική κατάρρευση των δομών του και σε πλήρη αποδιοργάνωση του, πράγμα που πολλές φορές αποβαίνει θανατηφόρο ιδιαίτερα αν πρόκειται για ψηφιακό σύστημα Εθνικής Άμυνας.

Στη δεύτερη περίπτωση, αν ισχύει ότι VI=0 τότε το σύστημα βρίσκεται σε κατάσταση Λευκής Λειτουργικής Οπής. Το σύνολο των δυνάμεων που αναπαριστώνται από το διάνυσμα CV δεν είναι επικίνδυνο για το σύστημα και δεν προκαλεί κανένα είδος κωλυσιεργίας. Έτσι, το σύστημα S λειτουργεί ομαλά παρέχοντας πλήρεις υπηρεσίες.

Στην τρίτη και τελευταία περίπτωση, αν ισχύει ότι VI<1 $\exists m \in S^*$: VI=0 τότε το σύνολο των καταστροφικών δυνάμεων που μπορεί να υλοποιηθούν στο σύστημα είναι δυνατό να το οδηγήσουν σε κατάσταση Γκρίζας Προγραμματιστικής Οπής. Σε αυτήν εμφανίζονται Ασταθείς καταστάσεις που προκαλούν μερικές καταρρεύσεις υποσυστημάτων. Το αρχικό σύστημα μεταλλάσσεται και ένα νέο απροσδιόριστο σύστημα δημιουργείται. Η μετάλλαξη αυτή μπορεί να είναι εξελικτική ή όχι, ανάλογα με τη δομή του νέου συστήματος και τις απαιτήσεις που καλύπτει μέσα στο περιβάλλον λειτουργίας που δρα.

Αλλάζοντας το αρχικό διάνυσμα σεναρίων SV, εκτελείται ένα νέο σενάριο που με τη σειρά του εμφανίζει νέα αποτελέσματα, τα οποία και αναλύονται. Ο διαχειριστής του συστήματος έχει τη δυνατότητα να συλλέξει τα αποτελέσματα, να τα αξιολογήσει και να εξάγει χρήσιμα

συμπεράσματα, έτσι ώστε να υλοποιήσει τα κατάλληλα αντίμετρα για την αποφυγή μελλοντικών καταστροφικών φαινομένων. Μέσα από τη συνεχή εκτέλεση σεναρίων, λοιπόν, βελτιστοποιεί τόσο τη δική του απόδοση όσο και την απόδοση του ίδιου του αλγορίθμου.

5.2.1.2 ΕΦΑΡΜΟΓΗ ΑΛΓΟΡΙΘΜΟΥ ΣΕΝΑΡΙΩΝ ΚΡΙΣΕΩΣ

Ο παραπάνω αλγόριθμος, όπως παρουσιάσθηκε, μπορεί να εφαρμοστεί σε οποιοδήποτε είδος συστήματος το οποίο δρα μέσα σε ένα ασταθές και δυναμικά μεταβαλλόμενο περιβάλλον λειτουργίας και δεν περιορίζεται από κανενός είδους περιορισμούς που θα μπορούσαν να ελαττώσουν τις δυνατότητες του. Με ελάχιστες τροποποιήσεις, ανάλογα με το είδος του συστήματος, ο αλγόριθμος διαχείρισης κρίσεως μπορεί να εφαρμοστεί και σε ψηφιακά πληροφοριακά συστήματα τα οποία δρουν σε ομαλούς, δυναμικά μεταβαλλόμενους χώρους δεδομένων. Παράλληλα, ο αριθμός των καταστροφικών δυνάμεων, που υλοποιούνται στο σύστημα, και το πλήθος των υποσυνόλων δεν αποτελούν περιορισμό στην ομαλή λειτουργία του αλγόριθμου.

Η πολυπλοκότητα του υπολογισμού όλων των πιθανών σεναρίων κρίσεως σε ένα σύστημα από n υποσυστήματα και m καταστροφές υπολογίζεται σαν το άθροισμα όλων των πιθανών συνδυασμών καταστροφικών κρίσεων.

Μέσα από το παρακάτω παράδειγμα, παρουσιάζεται η λειτουργία και η αμεσότητα του Crisis Scenario Algorithm, ο οποίος αναλύθηκε στην προηγούμενη παράγραφο. Ας υποθέσουμε ένα τυχαίο σύστημα **S με πέντε υποσυστήματα και ένα σύνολο από οχτώ πιθανές καταστροφές,** οι επιπτώσεις των οποίων παρουσιάζονται στον ακόλουθο Χάρτη Κινδύνου Χ με n=5 και m=8:

1	0	0	1	0	0	0	1
1	0	1	0	0	0	0	0
0	1	0	0	0	1	0	0
1	0	0	1	1	0	0	1
0	0	1	0	1	0	1	0

Ο χάρτης κινδύνου εμφανίζει μια τυχαία μορφή σε σχέση με το είδος των απειλών και τη δομή των υποσυστημάτων. Τα πάντα, σε αυτό το πρωταρχικό στάδιο, εναπόκεινται στον αναλυτή – διαχειριστή του συστήματος.

Επιπλέον, δημιουργούνται τυχαία σύνολα Scenarios Vectors. Σύμφωνα με τον παραπάνω τύπο, το σύνολο των πιθανών διανυσμάτων είναι 109600. Τα 109600 αυτά σενάρια θα χρησιμοποιηθούν για την εκτέλεση του CSA, και την εξαγωγή χρήσιμων συμπερασμάτων. Ο χρήστης του συστήματος δεν είναι υποχρεωμένος να εκτελέσει τον αλγόριθμο και για τα 109600 πιθανά σενάρια, αλλά να αποφασίσει το πλήθος και πολύ περισσότερο το είδος των σεναρίων που επιθυμεί να προσομοιώσει, λαμβάνοντας υπόψη τις απαιτήσεις και τις συγκυρίες που εμφανίζονται μέσα σε ένα διάστημα Δt στον ορίζοντα προγραμματισμού του συστήματος. Ένα πλήθος 82 υποθετικών σεναρίων για το υποθετικό μας σύστημα δίνεται παρακάτω:

0	0	0	0	0	0	1	1
0	0	1	0	1	1	0	1
0	1	1	0	1	1	1	0
0	1	0	1	0	0	0	1
1	0	0	0	1	0	0	0
0	0	1	1	1	0	0	1
0	0	1	0	0	1	0	0
0	0	0	0	1	0	1	1
0	0	1	1	0	1	1	0
1	1	1	0	0	1	0	1
0	0	1	0	0	1	0	1
1	1	0	1	0	0	0	0
1	1	1	0	1	0	0	1
0	1	1	0	0	1	0	0
1	1	1	1	0	1	1	0
0	0	0	0	0	0	1	0
1	1	1	1	1	1	0	1
1	0	1	0	0	0	1	1
1	1	0	0	1	0	1	1
0	1	1	0	1	0	0	1
0	0	0	1	1	0	1	1
1	1	0	0	1	0	1	0
0	0	0	0	0	0	0	0
0	0	1	0	1	1	0	0
1	0	1	0	1	1	1	0
0	0	0	0	1	1	0	1
1	1	0	0	1	0	0	0
1	0	0	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	1	0	0	1
0	1	1	1	1	1	0	0
1	1	0	1	0	0	0	1
0	0	0	1	1	1	1	0

0	1	0	1	0	0	1	0
0	1	1	0	1	0	1	0
1	1	0	0	1	1	1	1
1	0	1	0	0	1	1	0
0	0	1	1	1	1	1	1
0	0	0	0	1	0	0	0
1	0	0	0	0	1	1	1
0	1	1	0	0	1	1	0
1	0	0	0	1	1	1	0
0	1	1	0	1	1	1	1
0	1	0	1	0	1	0	1
0	0	1	0	1	1	1	0
1	1	0	1	1	1	0	0
0	0	1	1	1	1	0	0
1	0	0	0	1	0	1	0
0	1	0	0	0	1	0	1
0	0	0	0	0	1	1	0
1	0	0	1	1	1	1	0
1	0	0	0	1	1	1	1
0	1	0	0	0	0	0	0
1	0	0	1	1	0	0	0
0	0	0	1	1	1	1	1
0	1	1	0	1	0	0	0
0	0	1	1	0	0	1	0
1	0	1	0	1	0	1	0
0	0	1	1	1	1	1	0
0	1	1	1	0	0	1	0
0	1	0	0	1	1	1	1
1	0	1	1	0	0	1	1
1	0	0	1	0	1	0	1
0	0	0	0	0	1	0	1
0	0	0	1	1	1	0	1
1	0	0	1	1	1	1	1
1	0	1	1	1	1	0	1
0	1	1	1	0	1	1	0
1	1	0	1	0	0	1	1
1	1	0	1	0	1	1	0
1	1	1	1	1	0	1	1
1	0	0	1	1	0	1	0
1	1	0	0	0	0	1	1
0	0	1	1	1	0	0	0
0	1	0	0	1	0	0	0
1	0	0	1	0	1	0	0
1	1	0	0	1	1	0	0
0	0	0	1	1	0	0	0
0	1	1	0	1	1	0	0
0	0	0	1	1	0	1	0
0	1	1	0	0	0	0	1
0	1	1	0	0	0	0	0

Στην πρώτη γραμμή του παραπάνω συνόλου διανυσμάτων σεναρίων παρατηρείται η υλοποίηση δυο καταστροφικών δυνάμεων. Το διάνυσμα καταστροφών για αυτό το πιθανό σενάριο είναι το παρακάτω:

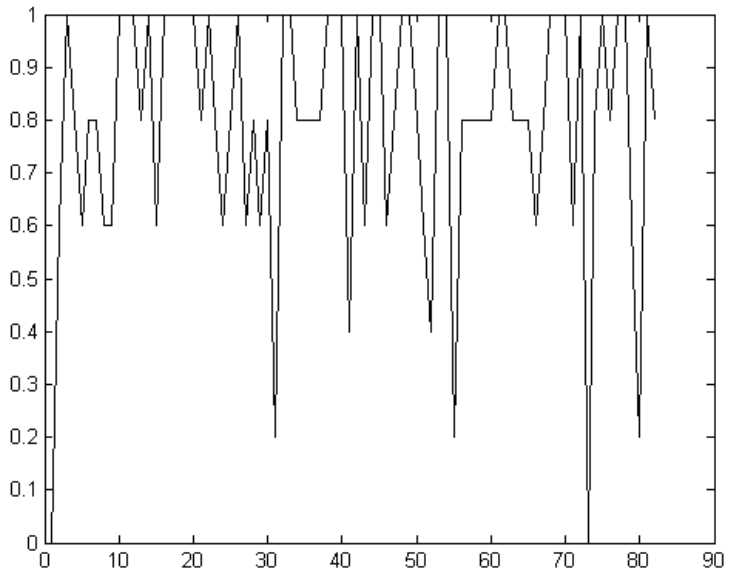
CV= (1 0 0 1 1)

Το CV προκύπτει από το άθροισμα των επιμέρους στηλών του Χάρτη Κινδύνου. Η επιλογή των στηλών υποδεικνύεται από τη στήλη στην οποία το διάνυσμα σεναρίων έχει τιμή 1. Επιπλέον, ο δείκτης Καταστροφικότητας για το παραπάνω επιλεγμένο διάνυσμα είναι VI= 3/5=0.6.

Το επόμενο βήμα είναι ο έλεγχος του δείκτη με βάση τα τρία θεμελιώδη θεωρήματα, που παρουσιάστηκαν σε προηγούμενο κεφάλαιο. Με βάση την τιμή του παραπάνω δείκτη και τον ορισμό του Μεταλλακτικού θεωρήματος παρατηρείται ότι το σύστημα βρίσκεται σε μαύρη προγραμματιστική οπή. Η δομή του έχει μεταλλαχτεί. Δημιουργήθηκε ένα νέο σύστημα με απροσδιόριστη λειτουργικότητα και απροσδιόριστο χρόνο λειτουργίας μέσα στον ορίζοντα χρονοπρογραμματισμού του συστήματος.

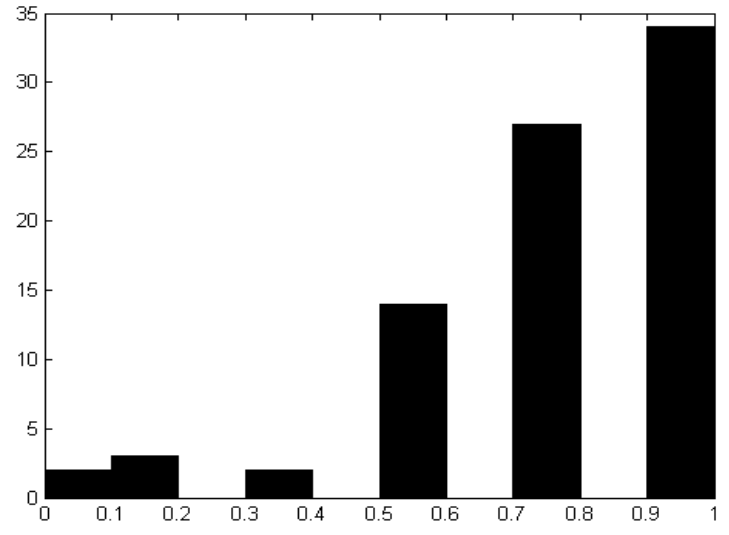
Τα πλήρη αποτελέσματα και των 82 των σεναρίων, που προσομοιώθηκαν με τον CSA, παρουσιάζονται στην επόμενη εικόνα γραφικά (Εικόνα 5.1). Με τη βοήθεια ενός plot-diagram εμφανίζονται οι τιμές που έχει κάθε φορά ο δείκτης καταστροφικότητας για κάθε σενάριο. Όλες οι τιμές ενώνονται με τη βοήθεια γραμμικής παρεμβολής.

Τα αποτελέσματα είναι ποιοτικά. Δεν υπολογίζουν την αστάθεια του χώρου, προσφέρουν, όμως, χρήσιμες πληροφορίες για τη λειτουργία του συστήματος σε περίπτωση που μια κρίση λάβει χώρα. Επίσης, προετοιμάζουν το διαχειριστή του συστήματος για την κατάσταση που πρόκειται να αντιμετωπίσει, ενώ προσδιορίζουν τη μορφή που θα έχει το σύστημα μετά από την εκδήλωση μιας κρίσης με το να εντοπίζουν μεταλλάξεις και τυχόν ολικές καταρρεύσεις. Παράλληλα, προβλέπουν ποια υποσυστήματα κινδυνεύουν με κατάρρευση ή μετάλλαξη και βοηθούν τους χρήστες να προσδιορίσουν σε ποιο σημείο του συστήματος είναι δυνατόν να υλοποιηθεί η κρίση προκειμένου να ληφθούν τα κατάλληλα μέτρα / αντίμετρα.



Εικόνα 5.1
Vulnerability Index

Στη δεύτερη εικόνα, που ακολουθεί, παρουσιάζονται επιπλέον στατιστικά συμπεράσματα με τη βοήθεια του histogram του CI (Εικόνα 5.2).



Εικόνα 5.2
Στατιστικά Αποτελέσματα Vulnerability Index

Με τη βοήθεια του δεύτερου γραφήματος συγκεντρώνονται και ομαδοποιούνται οι τιμές που έχει ο Vulnerability Index. Η ομαδοποίηση των τιμών γίνεται μετά τον υπολογισμό με βάση τα τρία θεμελιώδη θεωρήματα.

Σύμφωνα με τα παραπάνω προκύπτει, όπως φαίνεται και από το δεύτερο γράφημα, ότι η πλειοψηφία των σεναρίων είναι εξαιρετικά επικίνδυνη στα 34 από τα 82 σενάρια. Ένα ποσοστό της τάξης του 40% των σεναρίων έχει σαν αποτέλεσμα την ολική κατάρρευση του συστήματος S με ό,τι αυτό συνεπάγεται για τους χρήστες του.

Επιπλέον, σε 28 από τα 82 σενάρια κρίσεων, σε ποσοστό της τάξης του 34%, των σεναρίων το σύστημα οδηγείται σε κατάσταση Γκρίζας Λειτουργικής Οπής. Συγκεκριμένα, στην κατάσταση αυτή το σύστημα S μεταλλάσσεται σε ένα νέο σύστημα υποδεέστερο του αρχικού με μειωμένες δυνατότητες με συνέπεια, ένα 80% της λειτουργικότητας του συστήματος να έχει καταρρεύσει, αφού 4 στα 5 υποσυστήματα έχουν πληγεί από καταστροφικές δυνάμεις.

Ένα 20% των σεναρίων και ειδικότερα τα 17 από τα 82 σενάρια επίσης οδηγούν σε γκρίζες λειτουργικές οπές. Ωστόσο, το επίπεδο μετάλλαξης είναι μικρότερο σε σχέση με την προηγούμενη περίπτωση. Η μετάλλαξη του συστήματος αγγίζει περίπου το 50% της συνολικής δομής του.

Τέλος, μόνο ένα πολύ μικρό ποσοστό της τάξης του 3.5% και ειδικότερα των 3 από τα 82 σενάρια κρίσης οδηγούν το σύστημα σε κατάσταση Λευκής Λειτουργικής Οπής, αφήνοντας το σύστημα ανεπηρέαστο από την εμφάνιση καταστροφικών δυνάμεων.

Από τα παραπάνω εξάγεται το συμπέρασμα ότι ένα ποσοστό της τάξης του 75% των σεναρίων οδηγούν το σύστημα σε κατάρρευση ή σε μετάλλαξη ανωτάτου επιπέδου. Έτσι, γίνεται εύκολα αντιληπτό ότι το σύστημα είναι ιδιαίτερα ευάλωτο στις δηλωθείσες καταστροφές και απαιτείται ιδιαίτερη προσοχή για την αποφυγή τους.

Μέσα από αυτή την ποιοτική προσέγγιση της Πρόληψης Κινδυνικών φαινομένων ο Διαχειριστής του Συστήματος είναι γνώστης τόσο των κινδύνων, που απειλούν το σύστημα, όσο και της κατάστασης του συστήματος, όταν υλοποιηθεί η καταστροφή. Με τη βοήθεια των εξαγόμενων συμπερασμάτων από την προσομοίωση των σεναρίων αποκτά την ικανότητα να χρησιμοποιήσει το καταλληλότερο αντίμετρο ή, όπως θα παρουσιασθεί παρακάτω, το πιο χρήσιμο σύστημα ανάκτησης αν η καταστροφή υλοποιηθεί. Η ποιοτική ανάλυση, όπως παρουσιάσθηκε, μπορεί εύκολα να γίνει ποσοτική με την προσθήκη βαρών σε κάθε υποσύστημα ανάλογα με τη σημαντικότητα του στο κυρίως σύστημα.

5.3 ΑΝΤΙΜΕΤΩΠΙΖΟΝΤΑΣ ΜΙΑ ΚΑΤΑΣΤΡΟΦΗ

Η πρόληψη δεν είναι πάντα εφικτή. Ασταθείς παράγοντες, ασύμμετρες καταστάσεις κρίσεων, καθώς και τυχόν ολιγωρίες οδηγούν το σύστημα σε δυσμενείς καταστάσεις. Επιπλέον, υλοποιούνται καταστροφικές δυνάμεις, θέτοντας δομικά υποσυστήματα του κυρίως συστήματος εκτός λειτουργίας.

Σε τέτοιες καταστάσεις κρίσεων θα πρέπει το σύστημα να μπορεί να αντιμετωπίζει αυτόματα και με ακρίβεια αυτά τα καταστροφικά φαινόμενα. Παράλληλα, θα πρέπει να μπορεί να ανακτήσει τις απώλειες του. Αυτό απαιτεί όλοι οι φορείς του ψηφιακού πληροφοριακού συστήματος να είναι σε ετοιμότητα, ικανοί να αντιμετωπίσουν οποιαδήποτε κατάσταση ανάγκης με ψυχραιμία και εξυπνάδα.

Για να είναι επιτυχής αυτή η συνταγή άμυνας θα πρέπει να υπάρχει ένα σύστημα αντιμετώπισης κρίσεως το οποίο να στηρίζεται στο δίπολο άνθρωπος – μηχανή. Είναι αναγκαία η παρουσία του ανθρώπινου παράγοντα, ο οποίος θα έχει τη δυνατότητα να αντεπεξέλθει, λαμβάνοντας ζωτικής σημασίας αποφάσεις για την ομαλή λειτουργία του συστήματος. Δεν πρέπει να αφήνονται όλα στα χέρια των μηχανών. Αντίθετα, σε πολλές περιπτώσεις κρίσεων ο ανθρώπινος παράγοντας εμφανίζει συμπτώματα αναξιοπιστίας. Πολλά στελέχη του υπό κρίση ψηφιακού πληροφοριακού συστήματος παρουσιάζουν σημάδια ολιγωρίας, φυγής, αδυναμίας λειτουργίας υπό πίεση. Έτσι, χάνεται πολύτιμος χρόνος, με αποτέλεσμα η καταστροφή στην τελική της μορφή να είναι μη αναστρέψιμη.

5.3.1 ΜΟΝΤΕΛΟΠΟΙΗΣΗ

Ένα σύστημα που δρα σε έναν ασταθή χώρο δεδομένων αντιμετωπίζει καταστροφές οι οποίες εμφανίζονται αναπάντεχα μέσα στον ορίζοντα χρονοπρογραμματισμού του Δt. Οι δυνάμεις αυτές υλοποιούνται ατομικά ή σαν ένα σύνολο δυνάμεων. Και στις δυο περιπτώσεις, το σύστημα οφείλει να βρίσκεται σε ετοιμότητα προκειμένου να ελαχιστοποιήσει τις απώλειες του, καθώς είναι σύνηθες το φαινόμενο του domino. Ειδικότερα, η υλοποίηση μιας καταστροφικής δύναμης μπορεί να προκαλέσει με τη σειρά της αλυσιδωτή αντίδραση ενεργοποιώντας τις εν υπνώσει δυνάμεις ή να ενισχύσει τη δυναμικότητα των υπαρχόντων κρίσεων.

Ας θεωρήσουμε D το σύνολο των δυνατών καταστροφών που μπορούν να ενεργοποιηθούν μέσα στον ορίζοντα χρονοπρογραμματισμού Δt.

$$D= \{d_1 , d_2 ,...,d_n\} , D^*=\{1,2,...,n\}$$

Είναι επιτακτική η ανάγκη εφαρμογής κατάλληλων διαδικασιών ανάκτησης των κατεστραμμένων δεδομένων, ώστε να εξασφαλισθεί η ομαλή λειτουργία του συστήματος. Ας υποθέσουμε ότι R είναι το σύνολο των Δράσεων / αντίμετρων που απαιτούνται για την ανάκτηση των δεδομένων του συστήματος S, που καταστράφηκαν από κάποια καταστροφή D.

$$R= \{r_1 , r_2 ,...,r_m\} , R^*=\{1,2,...,m\}$$

5.3.1.1 ΑΛΓΟΡΙΘΜΟΣ ΑΝΑΚΤΗΣΗΣ ΚΡΙΣΕΩΣ

Όπως αναλύθηκε σε προηγούμενη παράγραφο, τα ψηφιακά πληροφοριακά συστήματα δρουν σε ασταθή δυναμικά μεταβαλλόμενα περιβάλλοντα εργασίας, με επακόλουθο την υλοποίηση καταστροφικών δεδομένων σε απρόσμενα χρονικά διαστήματα. Η αναπάντεχη εμφάνιση τους ενισχύει τον παράγοντα αστάθειας που διέπει τη λειτουργία αυτών των ψηφιακών συστημάτων. Για αυτό το λόγο, οφείλουν όλοι οι εμπλεκόμενοι φορείς να είναι ετοιμότητα έτσι ώστε να αντιμετωπίσουν την καταστροφή άμεσα και τάχιστα. Φυσικά, η ανάκτηση των δεδομένων που καταστράφηκαν προϋποθέτει ένα έμπειρο προσωπικό, το οποίο σε ελάχιστο χρόνο και με το μικρότερο δυνατό κόστος θα αντεπεξέλθει στο δύσκολο αυτό έργο.

Όλα τα παραπάνω προϋποθέτουν ένα έμπειρο προσωπικό διαχείρισης του πληροφοριακού συστήματος, με πολύ καλή γνώση των αδυναμιών του. Για να μπορέσει να αποκτηθεί η εμπειρία που απαιτείται και να γαλουχηθεί το προσωπικό σε καταστάσεις κρίσεων, χρειάζεται συνεχής πρακτική και προσομοίωση κινδυνικών φαινομένων. Οι πραγματικές καταστάσεις προσομοίωσης οδηγούν σε καλύτερα αποτελέσματα και μεγαλύτερη ετοιμότητα που συνεπάγεται βελτιωμένη αντίδραση σε περίπτωση πραγματικής καταστροφής.

Το επόμενο σημαντικό βήμα της διαδικασίας δημιουργίας σεναρίων κρίσεως είναι η υλοποίηση ενός σημαντικού εργαλείου, καταλυτικού για τη μετέπειτα ομαλή και αποτελεσματική προσομοίωση του συστήματος. Το **Διάνυσμα Σεναρίων Κρίσεως (Disaster Scenario**

Vector DSV) περιέχει ένα σύνολο διανυσμάτων το καθένα από τα οποία αποτελεί ξεχωριστή οντότητα δεδομένων προσομοίωσης κινδυνικών φαινομένων που μπορεί να υλοποιηθούν μέσα στο διάστημα Δt.

Για να είναι επιτυχημένη η προσομοίωση του συστήματος διαχείρισης κρίσεων, απαιτείται να εκτελεστεί ένας ικανοποιητικός αριθμός σεναρίων προκειμένου να προσδιοριστούν τυχόν αδυναμίες του ψηφιακού πληροφοριακού συστήματος. Εκτός από τα σύνολα D και R, που αναλύθηκαν προηγουμένως, απαιτείται και ο χάρτης καταστροφών M, που αναλύθηκε λεπτομερώς σε προηγούμενο κεφάλαιο, καθώς και ο πίνακας Κόστους Ενεργοποίησης του κατάλληλου αντιμέτρου.

Κάθε πιθανό σενάριο έχει το δικό του διάνυσμα. Ένα διάνυσμα σεναρίων κρίσεως DSV ορίζεται ως εξής:

$DSV = (dsv_j), \quad \forall j \in D^*$
dsv_j *=1, αν η* d_j *καταστροφή ενεργοποιείται,*
=0, διαφορετικά.

Η επιτυχία στηρίζεται στη συνεχή προσομοίωση του συστήματος κάτω από αντίξοες συνθήκες παραμετροποιώντας αναπάντεχα καταστροφικά φαινόμενα που μπορεί να εμφανιστούν. Έτσι, ένα χρήσιμο εργαλείο για την υλοποίηση σεναρίων κρίσεως σε ένα ψηφιακό πληροφοριακό σύστημα είναι ο **Αλγόριθμος Ανάκτησης Δεδομένων (Data Recovery Algorithm (DRA)).** Ακολουθώντας συγκεκριμένα βήματα επιτυγχάνεται η υλοποίηση της ανάκτησης των δεδομένων του πληροφοριακού συστήματος.

Data Recovery Algorithm (DRA)

Βήμα1: Δημιουργία Διανύσματος Σεναρίων Κρίσεως DSV.
Βήμα2: Εύρεση των ανταποκρινόμενων Διανυσμάτων Κρίσεως στο χάρτη καταστροφών M.
Βήμα3: Ενεργοποίηση του Assignement Problem για την εύρεση της βέλτιστης λύσης ανάκτησης δεδομένων.
Βήμα4: Επανάληψη του παραπάνω συνόλου βημάτων, μέχρις ότου δημιουργηθεί ένα κατάλληλο πλήθος από διαφορετικά πιθανά σενάρια.
Βήμα5: Εμφάνιση των προτεινόμενων λύσεων.

Το πρώτο βήμα του αλγορίθμου είναι ιδιαίτερα σημαντικό. Ο προγραμματιστής – διαχειριστής του συστήματος δημιουργεί το DSV.

Ένα σωστό σύνολο σεναρίων βοηθάει στην εξαγωγή χρήσιμων συμπερασμάτων ικανών να συμβάλλουν στην επιτυχημένη ανάκτηση του συστήματος στην περίπτωση που μια κρίση λάβει χώρα.

Το DSV περιέχει τα πιθανά σενάρια κρίσεως που μπορεί να υλοποιηθούν. Μέσα από μια μεγάλη ποικιλία υποθετικών σεναρίων επιλέγεται αυτό που κρίνεται πιο κρίσιμο την τρέχουσα χρονική στιγμή Δt μέσα στον ορίζοντα χρονοπρογραμματισμού και επαληθεύεται στο Χάρτη Καταστροφών Μ.

Στη συνέχεια, με τη βοήθεια του Προβλήματος Κατανομής επιλέγεται εκείνο το σύνολο αντίμετρων το οποίο ανακτά τα δεδομένα με τον καλύτερο δυνατό τρόπο και φυσικά με το ελάχιστο δυνατό κόστος. Αυτός είναι ένας δυνατός συνδυασμός επιτυχούς ανάκτησης δεδομένων.

Φυσικό επακόλουθο είναι το γεγονός ότι, αλλάζοντας το αρχικό διάνυσμα σεναρίων DSV, εκτελείται ένα νέο σενάριο το οποίο με τη σειρά του εμφανίζει νέα αποτελέσματα που μπορούν να αναλυθούν. Ο διαχειριστής του συστήματος έχει τη δυνατότητα να συλλέξει τα αποτελέσματα, να τα αξιολογήσει και να εξάγει χρήσιμα συμπεράσματα, έτσι ώστε να υλοποιήσει τα κατάλληλα αντίμετρα και να αποφύγει μελλοντικά καταστροφικά φαινόμενα. Παράλληλα, βελτιώνει την ικανότητα του για γρήγορη, έγκαιρη, αξιόπιστη και φυσικά ορθολογική αντιμετώπιση της υλοποιηθείσας κρίσης.

5.3.1.2 ΕΦΑΡΜΟΓΗ ΑΛΓΟΡΙΘΜΟΥ ΣΕΝΑΡΙΩΝ ΚΡΙΣΕΩΣ

Όπως αναλύθηκε και σε προηγούμενη παράγραφο, έτσι και εδώ ο παραπάνω αλγόριθμος – DRA -, όπως παρουσιάσθηκε, μπορεί να εφαρμοστεί σε οποιοδήποτε είδος συστήματος, το οποίο δρα μέσα σε ένα ασταθές και δυναμικά μεταβαλλόμενο περιβάλλον λειτουργίας και δεν περιορίζεται από κανενός είδος περιορισμούς οι οποίοι θα μπορούσαν να ελαττώσουν τις δυνατότητες του. Με ελάχιστες τροποποιήσεις ανάλογα με το είδος του συστήματος, ο αλγόριθμος διαχείρισης κρίσεως μπορεί να εφαρμοστεί και σε ψηφιακά πληροφοριακά συστήματα τα οποία δρουν σε ομαλά, δυναμικά μεταβαλλόμενους χώρους δεδομένων. Παράλληλα, ο αριθμός των καταστροφικών δυνάμεων που υλοποιούνται στο σύστημα και το πλήθος των υποσυνόλων δεν αποτελούν περιορισμό στην ομαλή λειτουργία του αλγόριθμου.

Το παρακάτω παράδειγμα παρουσιάζει τη λειτουργία και την αμεσότητα του *Data Recovery Algorithm,* ο οποίος αναλύθηκε στην

προηγούμενη παράγραφο. Χωρίς βλάβη της γενικότητας, στη συνέχεια, θα παρουσιασθεί ένα παράδειγμα υλοποίησης του παραπάνω αλγορίθμου. Ας υποθέσουμε το τυχαίο σύστημα S, το οποίο αναλύθηκε στην προηγούμενη παράγραφο. **Ας θεωρήσουμε ότι το σύστημα S απειλείται από 6 είδη καταστροφών τα οποία αντιμετωπίζονται από 4 είδη ανάκτησης δεδομένων.** Η σύνδεση των καταστροφών με τα συστήματα ανάκτησης παρουσιάζεται αναλυτικά στον παρακάτω Χάρτη Καταστροφών Μ. Έτσι, έχουμε:

1	1	0	1
1	0	0	0
0	1	1	1
1	1	0	1
0	1	1	0
1	0	1	1

Ο παραπάνω χάρτης καταστροφών εμφανίζει μια τυχαία μορφή που μπορεί να είναι ανάλογη με το είδος των απειλών και τη δομή των υποσυστημάτων. Χωρίς βλάβη της γενικότητας, η μορφή του μπορεί να αλλάξει ανάλογα με το είδος της ανάλυσης που έχει γίνει στο σύστημα. Τα πάντα σε αυτό το πρωταρχικό στάδιο εναπόκεινται στον αναλυτή – διαχειριστή του συστήματος.

Επιπλέον, δημιουργούνται τυχαία σύνολα Διανυσμάτων Σεναρίων Κρίσεως. Σύμφωνα με τον παραπάνω τύπο, το σύνολο των πιθανών διανυσμάτων είναι 1956. Τα 1956 αυτά σενάρια θα χρησιμοποιηθούν για την εκτέλεση του DRA και την εξαγωγή χρήσιμων συμπερασμάτων. Ο χρήστης του συστήματος δεν είναι υποχρεωμένος να εκτελέσει τον αλγόριθμο και για τα 1956 πιθανά σενάρια. Ο ίδιος πάντα ο χρήστης αποφασίζει το πλήθος και πολύ περισσότερο το είδος των σεναρίων που επιθυμεί να προσομοιώσει, λαμβάνοντας υπόψη τις απαιτήσεις και τις συγκυρίες που εμφανίζονται μέσα σε ένα διάστημα Δt στον ορίζοντα προγραμματισμού του συστήματος. Έτσι, ένα υποθετικό πλήθος 53 πιθανών σεναρίων κρίσεως από τα συνολικά 1956 σενάρια του υποθετικού συστήματος S δίνεται παρακάτω:

0	0	0	0	0	0
1	1	1	0	0	1
0	0	1	0	0	0
1	0	0	0	0	1
1	0	1	1	1	1
0	1	0	1	0	0
1	0	0	1	1	0
0	0	1	1	0	1
0	1	1	1	0	1
0	0	1	1	0	0
0	1	0	0	1	0
1	0	0	0	1	1
1	0	1	0	1	0
0	0	1	0	1	1
1	0	1	1	0	0
0	1	1	1	1	1
1	1	0	0	0	1
1	1	0	1	0	0
0	0	1	0	0	1
0	0	0	1	1	1
0	1	0	1	1	1
1	0	0	0	1	0
1	1	0	1	1	1
1	0	0	1	0	1
0	0	1	0	1	0
1	1	1	1	0	0
0	0	0	1	0	0
1	0	0	0	0	0
1	1	1	1	0	1
1	0	1	0	0	0
0	0	1	1	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	0	1	0
0	1	1	0	1	0
1	0	0	1	1	1
0	0	0	1	1	0
1	1	1	1	1	0
1	0	1	0	1	1
1	1	0	1	1	0
0	0	0	0	1	1
0	1	1	0	0	0
0	1	1	0	0	1
1	0	1	0	0	1
1	0	1	1	1	0
0	1	0	1	0	1
0	0	0	1	0	1

0	0	0	0	0	1
0	1	1	1	0	0
0	1	0	0	0	1
0	1	1	0	1	1
1	1	0	0	0	0
0	0	0	0	1	0

Κάθε καταστροφικό φαινόμενο συνήθως έχει και ένα αντίμετρο ανάκτησης των δεδομένων του. Στην πλειοψηφία των περιπτώσεων, ένα καταστροφικό φαινόμενο έχει περισσότερα από ένα συστήματα ανάκτησης δεδομένων και η επιλογή του καταλληλότερου είναι μια ιδιαίτερα δύσκολη διαδικασία. Με τη βοήθεια του Πίνακα Κόστους του κάθε συστήματος ανάκτησης και με τη χρήση του Προβλήματος της Κατανομής, η επιλογή του κατάλληλου Συστήματος Ανάκτησης γίνεται όλο και πιο εύκολη. Έτσι, βελτιστοποιείται η επιλογή του συστήματος ανάκτησης δεδομένων με ό,τι αυτό συνεπάγεται για τη μετέπειτα ομαλή πορεία του υπό απειλή συστήματος.

Ας υποθέσουμε ότι, ο Πίνακας Κόστους έχει την ακόλουθη μορφή:

6	2	2	4
4	1	9	3
6	0	8	2
3	3	5	6
7	4	8	7
8	3	1	6

Η επιλογή της βέλτιστης λύσης γίνεται, στη συνέχεια, με τη βοήθεια του Προβλήματος Κατανομής. Καταλήγουμε, λοιπόν, σε ένα σύνολο αντιμέτρων με συγκεκριμένο κόστος ανάκτησης, που στη συγκεκριμένη περίπτωση είναι 7 μονάδες.

Τα αποτελέσματα σε αυτήν την περίπτωση είναι ταυτόχρονα ποιοτικά και ποσοτικά. Δεν υπολογίζουν την αστάθεια του χώρου. Προσφέρουν, όμως, χρήσιμες πληροφορίες για τη λειτουργία του συστήματος σε περίπτωση που μια κρίση λάβει χώρα. Προετοιμάζουν το διαχειριστή του συστήματος για την κατάσταση που πρόκειται να αντιμετωπίσει και παράλληλα, προβλέπουν ποια υποσυστήματα κινδυνεύουν με κατάρρευση, βοηθώντας, τους χρήστες να προσδιορίσουν επακριβώς την κρίση καθώς το σημείο του συστήματος στο οποίο πρόκειται να υλοποιηθεί. Συνεισφέρουν, επομένως, στη λήψη των

κατάλληλων μέτρων ανάκτησης προσδοκώντας τη βέλτιστη λύση του προβλήματος.

5.4 ΣΥΜΠΕΡΑΣΜΑΤΑ

Μέσα από αυτήν την ποιοτική προσέγγιση της Πρόληψης Κινδυνικών φαινομένων, ο Διαχειριστής του Συστήματος είναι γνώστης τόσο των κίνδυνων που απειλούν το σύστημα όσο και της μετέπειτα κατάστασης του συστήματος, αφού υλοποιηθεί η καταστροφή. Με τη βοήθεια των εξαγομένων συμπερασμάτων από την προσομοίωση των σεναρίων, ο χρήστης αποκτά την ικανότητα να χρησιμοποιήσει το καταλληλότερο αντίμετρο ή το πιο χρήσιμο σύστημα ανάκτησης, αν υλοποιηθεί η καταστροφή.

Τόσο στην περίπτωση της πρόληψης όσο και στην περίπτωση της ανάκτησης δεδομένων παρατηρήθηκε το φαινόμενο ο αριθμός των πιθανών σεναρίων κρίσεως ενός n x m συστήματος S να δίνεται από τον τύπο $\sum_{j=1}^{n} \frac{(j+1)(j+2)...(n-1)n}{(n-j)!}$. Πράγματι, τα πιθανά σενάρια προκύπτουν από το άθροισμα των συνδυασμών των επιμέρους περιπτώσεων υλοποίησης καταστροφικών δυνάμεων. Το ερώτημα που γεννάται είναι αν πράγματι είναι τόσα τα πιθανά σενάρια κρίσεως ή μήπως ο αριθμός τους μπορεί να μειωθεί λαμβάνοντας υπόψη ότι κάποια από αυτά επαναλαμβάνονται δυο φορές.

Είναι αλήθεια ότι, αναλύοντας τη δομή των Χαρτών Κινδύνου και Αντιμετώπισης, παρατηρούμε πως το πλήθος των σεναρίων κρίσεως μπορεί να μειωθεί. Σε πολλές περιπτώσεις, επιτυγχάνεται και μείωση σε ποσοστό 50% του πλήθους των σεναρίων, που εξετάζονται, βελτιστοποιώντας το χρόνο εξαγωγής συμπερασμάτων.

Στην πραγματικότητα, όμως, τα πράγματα δεν είναι έτσι. Η ακολουθία των γεγονότων μπορεί να παίξει σημαντικό ρόλο στην εξέλιξη ενός κινδυνικού φαινομένου. Παράλληλα, η σειρά υλοποίησης των καταστροφικών δυνάμεων μπορεί να δημιουργήσει διαφορετικές καταστάσεις κρίσεως στο σύστημα καθώς λειτουργεί μέσα σε ένα ασταθές περιβάλλον λειτουργίας. Έτσι, λοιπόν, αν η αλληλουχία των γεγονότων διαταραχθεί ή αντιστραφεί μέσα στον ορίζοντα Δt, μπορεί να προκαλέσει εντελώς διαφορετικά αποτελέσματα με αντιδιαμετρικές - πολλές φορές - συνέπειες για το σύστημα S.

ΚΕΦΑΛΑΙΟ 6

ΨΗΦΙΑΚΑ ΠΡΩΤΟΚΟΛΛΑ ΔΙΑΧΕΙΡΙΣΗΣ ΚΡΙΣΕΩΝ

6. ΠΡΩΤΟΚΟΛΛΑ ΔΙΑΧΕΙΡΙΣΗΣ ΚΡΙΣΕΩΣ

6.1 ΕΙΣΑΓΩΓΗ

"Η επιστήμη δημιουργεί το σοφό, η λογική τον άνθρωπο.
Η πρώτη είναι κτήμα μερικών, η δεύτερη όλων."

Λα Κορνταίρ

Ένα σύστημα διαχείρισης κρίσεων όπως προαναφέρθηκε δρα σε ένα ασταθές δυναμικά μεταβαλλόμενο περιβάλλον. Πολλοί αστάθμητοι παράγοντες επιδρούν πάνω του προκαλώντας πληθώρα καταστροφικών φαινομένων με δυσάρεστες συνέπειες για την ομαλή λειτουργία του συστήματος. Σκοπός ενός επιτυχημένου συστήματος πρέπει να είναι η πρόληψη και η αποφυγή τέτοιων καταστροφικών φαινομένων. Η επιτυχημένη πρόληψη τέτοιων καταστροφικών δεδομένων οδηγεί στη σίγουρη αποφυγή μελλοντικών κινδυνικών φαινομένων.

Τα Πρωτόκολλα Διαχείρισης Κρίσεων είναι μια λύση στο πρόβλημα ελέγχου Ασταθών Χώρων Δεδομένων. Η μεταβλητότητα του Χώρου και παράλληλα η αυξημένη επικινδυνότητα του καθιστά αδύνατη την ολοκληρωμένη και απόλυτη πρόβλεψη πιθανών κινδυνικών φαινομένων. Τα Πρωτόκολλα Διαχείρισης Κρίσεων παρέχουν μια συντεταγμένη μεθοδολογία μέσα από την οποία επιτυγχάνεται η αποκλιμάκωση και σε τελική ανάλυση η αντιμετώπιση της κρίσεως. Επομένως ορίζεται:

Πρωτόκολλο Διαχείρισης Κρίσεων είναι ένα πεπερασμένο σύνολο ενεργειών αυστηρά καθορισμένο που στόχο έχει τη βέλτιστη δυνατή ανάκτηση λειτουργίας του συστήματος S μετά από πτώση.

Είναι επιτακτική η ανάγκη δημιουργίας δυναμικών Πρωτοκόλλων Διαχείρισης Κρίσεων προσαρμοσμένων στις απαιτήσεις του χώρου Δεδομένων ικανών να αντιμετωπίσουν ή τουλάχιστον να περιορίσουν κινδινικά φαινόμενα και τις συνέπειες αυτών. Οι μη Ομαλά Δυναμικοί Χώροι Δεδομένων εμπεριέχουν έντονη Ασάφεια και αυξημένη επικινδυνότητα για τα συστήματα που λειτουργούν μέσα σε αυτούς.

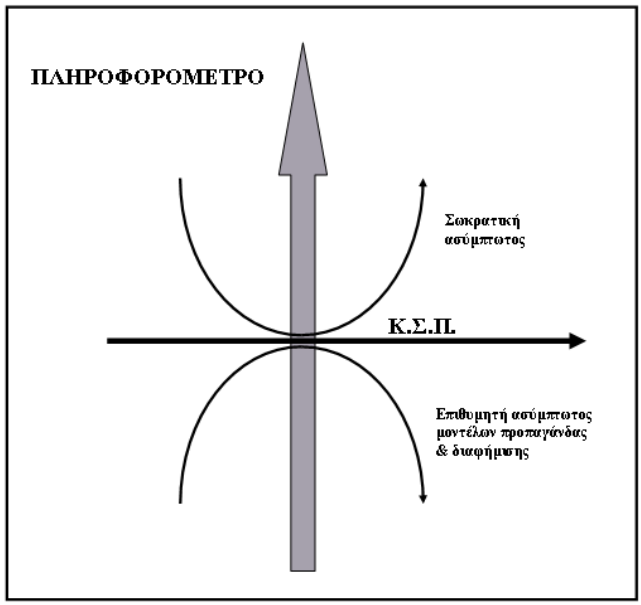
Στόχος των πρωτοκόλλων διαχείρισης κρίσεων είναι η μείωση της ασάφειας των μη Ομαλών Δυναμικών Χώρων Δεδομένων. Δεν πρέπει να ξεχνάμε ότι τα Πρωτόκολλα δεν είναι πανάκεια αφού ισχύει πάντα η

αξιωματική πρόταση ότι η ύπαρξη κινδύνου είναι βεβαιότητα και όχι πιθανότητα.

6.2 ΚΡΙΣΙΜΟ ΣΗΜΕΙΟ ΠΛΗΡΟΦΟΡΗΣΗΣ

Τα τελευταία χρόνια έχει αποδειχθεί, κάτι το οποίο γνώριζαν ήδη καλά από πείρα για χρόνια τώρα, οι αναλυτές που προσπαθούσαν να μηχανογραφήσουν ολοκληρωμένους χώρους (επιχειρήσεις, οργανισμούς, εταιρείες, κλπ). Η αναγκαιότητα συλλογής πληροφοριών για τη μηχανογράφηση των χώρων κατέδειξε ότι δεν φθάνει ο αναλυτής να συγκεντρώσει πληροφορία από κάθε ένα υποχώρο της επιχείρησης με συζητήσεις ή ερωτηματολόγια ή ζώντας για λίγο διάστημα μαζί τους. Απεδείχθη ότι το μέγιστο πλήθος πληροφόρησης δεν ήταν στις κορυφές του γραφήματος των υποχώρων της εταιρείας αλλά στις σχέσεις των υποχώρων μεταξύ τους.

Γενικά η συλλογή πληροφορίας στις ψηφιακές επιστήμες σήμερα δεν ακολουθεί τα παλαιά μοντέλα συλλογής πληροφορίας αλλά γίνεται με ειδικά έξυπνα προγράμματα τα οποία παράλληλα με την απλή πληροφορία προσπαθούν να προσδιορίσουν κανόνες (rules) που τα διέπουν (data mining, knowledge engineering, process management, λογικό προγραμματισμό, κλπ). Μάλιστα, μετρούν σήμερα το μέγεθος της πληροφορίας για κάποιο μελετούμενο σύστημα με το **Εικονικό Πληροφορόμετρο**. Έχει αποδειχθεί ότι για να έχει κανείς **ορθή αντίληψη** ενός συστήματος θα πρέπει η συλλογή πληροφορίας του να βρίσκεται τουλάχιστον από το **Κρίσιμο Σημείο Πληροφόρησης (ΚΣΠ)** και πάνω. [Panayiotopoulos (2007)]



Σχήμα 6.1
Το εικονικό «Πληροφορόμετρο»

Επομένως σαν ΚΣΠ ορίζεται η ελάχιστη πληροφορία που απαιτείται για τη στοιχειώδη αλλά ορθή αντίληψη ενός συστήματος. Στο σημείο αυτό έχουν παρατηρηθεί από διαφόρους κλάδους επιστήμης ότι εάν οι γνώσεις που έχει κάποιος για το μελετούμενο σύστημα κινούνται ασυμπτωτικά κάτω από ΚΣΠ, τότε αυτός όχι μόνο έχει λανθασμένη αντίληψη για το σύστημα αλλά επιπρόσθετα κινείται στα βήματα του **ισχυρογνώμονος οπαδού** κάτι που εφαρμόζουν σήμερα τα **σύγχρονα μοντέλα διαφήμισης και προπαγάνδας** (τα οποία βέβαια είναι κατά κανόνα κοινά).

Συμμετρικά αν η παρεχόμενη πληροφορία για το σύστημα κινείται ασυμπτωτικά πάνω από το ΚΣΠ, τότε ο εκπαιδευόμενος όχι μόνο διαθέτει ήδη ορθή αντίληψη για το υπό μελέτη σύστημα, αλλά νιώθει παράλληλα το μέγεθος της πληροφορίας που ακόμη του λείπει και για το λόγο αυτό ονομάζεται **Σωκρατική ασύμπτωτος**. Το πληροφορόμετρο μοιάζει με το θερμόμετρο πυρετού όπου το 37 ^{o}C είναι κρίσιμο σημείο απόφασης αν κάποιος έχει πυρετό ή όχι, με τις διαφορές ότι το πληροφορόμετρο δεν τελειώνει πουθενά προς τα πάνω, έχει άπειρο μήκος μια που μόνο η φύση γνωρίζει την πλήρη πληροφόρηση για τα υπάρχοντα συστήματα και επίσης ότι το κάθε σύστημα έχει και το δικό του κρίσιμο σημείο πληροφόρησης. Οργανωμένα συστήματα με μικρό δείκτη ασάφειας εμφανίζουν χαμηλό ΚΣΠ, ενώ συστήματα που υπάρχουν μέσα σε ασαφείς και ανώμαλους χώρους δεδομένων (ανώμαλοι

χώροι δεδομένων), εμφανίζουν υψηλό ΚΣΠ και επομένως είναι επιρρεπή σε μεταλλάξεις.

Από τη στιγμή που θα συγκροτηθεί ένα Ολοκληρωμένο Σύστημα Διαχείρισης Κρίσεων, θα πρέπει η παρεχόμενη γνώση από αυτό τουλάχιστον να φτάνει και να ξεπερνά το κρίσιμο σημείο πληροφόρησης του γνωσιολογικού χώρου που πρέπει να αναλυθεί.

6.3 ΜΕΘΟΔΟΛΟΓΙΑ ΑΝΑΛΥΣΗΣ ΚΑΙ ΣΧΕΔΙΑΣΜΟΥ ΠΡΩΤΟΚΟΛΛΩΝ ΔΙΑΧΕΙΡΙΣΗΣ ΚΡΙΣΕΩΝ

Η επιτυχία αποφυγής μιας κρίσης που μπορεί να υλοποιηθεί μέσα στον ορίζοντα χρονοπρογραμματισμού λειτουργίας ενός συστήματος στηρίζεται στην σωστή και έγκαιρη πρόληψη πιθανών κινδυνικών φαινομένων. Το σημαντικό εμπόδιο στην επιτυχημένη εφαρμογή προληπτικών μεθοδολογιών διαχείρισης κρίσεων είναι ο ίδιος ο χώρος δεδομένων. Το περιβάλλον λειτουργίας είναι εχθρικό για το σύστημα που δρα μέσα σε αυτό. Ασταθείς παράγοντες δρουν σε τυχαίες μη προβλέψιμες στιγμές μέσα στον ορίζοντα λειτουργίας του συστήματος S. Οι δυνάμεις αυτές δεν μπορούν να προβλεφθούν μέσα από συγκεκριμένες στατιστικές κατανομές λόγω της μη ομαλής δυναμικότητας του ίδιου του χώρου δεδομένων.

Παράλληλα δεν είναι δυνατή η μακροχρόνια μελέτη και συλλογή στατιστικών δεδομένων της αλληλεπίδρασης του συστήματος με το περιβάλλον του. Ακόμα και αν επανατοποθετούσαμε το σύστημα στο ίδιο περιβάλλον λειτουργάς και κάτω από τις ίδιες συνθήκες η δυναμική του χώρου δεν θα επιτρέψει την εμφάνιση των ίδιων καταστροφικών δυνάμεων την ίδια χρονική στιγμή και με τα ίδια αποτελέσματα.

Με βάση τα παραπάνω η μεθοδολογία που πρέπει να τηρηθεί για τα πρωτόκολλα χωρίζεται σε φάσεις οι οποίες είναι αναγκαίες προκειμένου να επιτευχθεί ο σκοπός τους. Οι φάσεις αυτές είναι:

- **Ανάλυση,**
- **Σχεδιασμός,**
- **Εφαρμογή,**
- **Αξιολόγηση.**

6.4.1 ΦΑΣΗ ΑΝΑΛΥΣΗΣ

Η φάση της ανάλυσης του περιβάλλοντος λειτουργίας ενός ψηφιακού πληροφοριακού συστήματος είναι η πιο σημαντική για την επιτυχημένη αντιμετώπιση πιθανών κινδυνικών φαινομένων. Υπάρχουν πολλές θεωρίες πάνω στο πως θα ήταν δυνατόν να αναλυθεί ένας Ασταθής χώρος δεδομένων προκειμένου να εντοπιστούν οι αδυναμίες του. Μια προσέγγιση γίνεται με τη βοήθεια Ασαφούς Λογικής και έμπειρων συστημάτων κατάλληλων να διαγνώσουν τα ασαφή σύνολα που δημιουργούνται. Η Αστάθεια του χώρου και η μη ομαλή δυναμικότητα του δεν επιτρέπουν τη στατιστική μελέτη του πράγμα που θα διευκόλυνε αφάνταστα τη γρήγορη και αψεγάδιαστη ανάπτυξη λειτουργικά ορθών Πρωτοκόλλων Διαχείρισης Κρίσεων.

Αρχικά θα πρέπει με ελάχιστο πλήθος ερωτήσεων ένα ειδικό έμπειρο σύστημα να καταλήξει σε διάγνωση σε σχέση με το Πρωτόκολλο Διαχείρισης Κρίσεων που πρέπει να σχεδιαστεί. Όπως θα αναλυθεί, το έμπειρο σύστημα διάγνωσης στην ουσία θα λειτουργήσει σε διπλό ρόλο. Ο πρώτος ρόλος του θα είναι ανιχνευτικός, όπου θα εντοπιστεί απλά ο γνωσιολογικός χώρος που θα κινηθεί το Πρωτόκολλο. Ο δεύτερος ρόλος του έχει να κάνει με τον προσδιορισμό του προφίλ του χρήστη αναφορικά με τον γνωσιολογικό χώρο στον οποίο κινείται, δηλαδή τον εντοπισμό των γνώσεων που ήδη έχει αλλά και αυτών που αναζητά από το υπό ανάπτυξη Πρωτόκολλο. Στην πρώτη φάση επίλυσης θεωρείται υπαρκτό ένα **έμπειρο σύστημα διάγνωσης (Diagnosis Expert System)** το οποίο να τηρεί τις κλασσικές προδιαγραφές της τεχνολογίας Γνώσης των έμπειρων συστημάτων τύπου MYCIN αλλά παράλληλα να είναι επανδρωμένο με τα υποσυστήματα διδασκαλίας και αυτοεξήγησης των έμπειρων συστημάτων τύπου EURISCO. [Yu (1984)]

Το υποσύστημα διάγνωσης θα πρέπει να είναι ένα έμπειρο σύστημα το οποίο να αντλεί τις γνώσεις από κλασσικές σχεσιακές ή μη βάσεις δεδομένων αλλά να έχει και τη δυνατότητα άντλησης γνώσεων από ειδικούς. Διαφορετικά θα έχει και αυτό τον κίνδυνο της αδρανοποίησης, της στατικότητας και επομένως της σταδιακής συρρίκνωσης και παροπλισμού του (**συστολικό σημείο αυτοκατάρρευσης**). Επίσης θα πρέπει να διαθέτει υποσύστημα εκπαίδευσης και ανάλυσης του τρόπου σκέψης του (tutor). Συγκεκριμένα οι ερωτήσεις που θα κάνει προς τον τελικό χρήστη θα πρέπει να έχουν τη δυνατότητα των απαντήσεων «ΝΑΙ», «ΟΧΙ» και «ΓΙΑΤΙ». Με τον τρόπο αυτό, τα ερωτήματα θα συγκροτούν ερωτηματολόγιο κλειστό και διχοτομικό το οποίο είναι πάντα δυνατόν να γίνει σύμφωνα με τη θεωρία

της ιεραρχικής ανάλυσης (κλίμακες GOODMAN). Στην περίπτωση που ένας χρήστης αντί να δώσει μία από τις δύο απαντήσεις δώσει το γιατί, τότε το έμπειρο σύστημα θα πρέπει να έχει τη δυνατότητα της καθολικής αντίδρασής του για τη συγκεκριμένη περίπτωση μέχρι εκείνη τη στιγμή. Δηλαδή τι έχει βρει ότι ισχύει μέχρι τώρα, και τι πρόκειται να προσπαθήσει να βρει στις αμέσως επόμενες ερωτήσεις του. Με τον τρόπο αυτό ο Διαχειριστής και η ομάδα αντιμετώπισης κρίσεων θα έχουν τη δυνατότητα να βελτιστοποιήσουν τις αντιδράσεις τους σε περίπτωση εμφάνισης ενός κινδυνικού φαινομένου.

Το πρώτο κύμα ερωτήσεων θα είναι ανιχνευτικό απλά και μόνο για να προσδιοριστεί το σημείο που βρίσκεται το επίπεδο της ομάδας διαχείρισης κρίσεων τόσο γνωσιολογικά όσο και επιστημονικά μέσα στο χώρο της επιστήμης και της τεχνολογίας. Αν το παρομοιάσουμε με την αναζήτηση κάποιου κεφαλαίου σε κάποιο βιβλίο το πρώτο κύμα ερωτήσεων σταματά ακριβώς στον εντοπισμό του βιβλίου. Στη συνέχεια το δεύτερο κύμα ερωτήσεων είναι εξειδικευμένο και θα προσπαθήσει να εντοπίσει το κεφάλαιο και τις παραγράφους εκείνους που πρέπει ο χρήστης να καταλήξει. Συγκεκριμένα θα εντοπιστεί και το χαρακτηριστικό διάνυσμα γνώσεων που απαιτείται για το αντικείμενο και θα εντοπιστούν εκείνες από τις γνώσεις που δεν διαθέτει και επομένως αυτές και μόνο θα πρέπει να συγκροτήσουν το Πρωτόκολλο Διαχείρισης Κρίσεων. Στη συνέχεια βέβαια το έμπειρο σύστημα παραδίδει τη σκυτάλη στο υποσύστημα Σχεδιασμού.

6.4.2 ΦΑΣΗ ΣΧΕΔΙΑΣΜΟΥ

Κατά το σχεδιασμό του Πρωτοκόλλου εντοπίζονται οι ενότητες γνώσης που πρέπει αυτό να καλύψει, κάτω από συγκεκριμένους περιορισμούς που σχετίζονται με το επιθυμητό βαθμό πληρότητας κάλυψης του αντικειμένου από το διαχειριστή και το χρόνο τον οποίο έχει στη διάθεσή του για να το παρακολουθήσει. Το υποσύστημα σχεδιασμού λαμβάνει από το υποσύστημα ανάλυσης της προηγούμενης Φάσης , το **χαρακτηριστικό ψηφιακό Διάνυσμα Γνώσης Κρίσεων (Crisis Knowledge Vector)** που πρέπει να καλύψει. Ακολουθεί έλεγχος για το κατά πόσο οι ενότητες αυτές μπορούν να καλυφθούν από τη Βάση Γνώσης Κρίσεων (**Crisis Knowledge Database).** Το χαρακτηριστικό ψηφιακό διάνυσμα δρομολογείται προς τη βάση γνώσης του συστήματος προκειμένου να επιλεχθεί μέσο βέλτιστου σχεδιασμού το κατάλληλο σενάριο διαχείρισης Κρίσεων. Η επίλυση του προβλήματος βέλτιστου σχεδιασμού που ακολουθεί δύναται να έχει περισσότερες από μία λύσεις.

Πέρα της βέλτιστης λύσης ο χρήστης μπορεί να επιλέξει και από τις υπόλοιπες εναλλακτικές που προτείνονται λαμβάνοντας την οριστική απόφαση για το πόσο χρόνο θα αφιερώσει στη μελέτη του σεναρίου που καλύπτει το υπό δημιουργία Πρωτόκολλο Διαχείρισης Κρίσεων. Η τελική απόφαση του χρήστη δρομολογείται στο υποσύστημα Εφαρμογής της επόμενης Φάσης.

6.4.3 ΦΑΣΗ ΕΦΑΡΜΟΓΗΣ

Κατά τη φάσης σύνθεσης του Πρωτοκόλλου, εντοπίζονται από τη Βάση Γνώσης Κρίσεων του συγκεκριμένου χώρου που εντάσσεται το Σύστημα, όλα τα αντικείμενα που απαιτούνται προκειμένου να ικανοποιηθεί στο μέγιστο δυνατό βαθμό το χαρακτηριστικό ψηφιακό διάνυσμα που προέκυψε από τη φάση του Σχεδιασμού. Αυτά τα αντικείμενα φέρουν όλη την απαραίτητη πληροφορία που θα συγκροτήσει το Πρωτόκολλο. Το υποσύστημα αυτό τροφοδοτείται με το χαρακτηριστικό διάνυσμα Κρίσης. Προκειμένου να επιτευχθεί αυτό, εντοπίζονται από τη κεντρική βάση γνώσης του συγκεκριμένου χώρου που εντάσσεται, όλα τα αντικείμενα γνώσης που απαιτούνται προκειμένου να ικανοποιηθεί το χαρακτηριστικό ψηφιακό διάνυσμα Γνώσης Κρίσεων που προέκυψε από τη φάση του Σχεδιασμού.

6.4.4 ΦΑΣΗ ΑΞΙΟΛΟΓΗΣΗΣ

Κατά τη φάση της αξιολόγησης το περιεχόμενο των αντικειμένων γνώσης που συγκροτούν το Πρωτόκολλο, μετασχηματίζεται κατάλληλα προκειμένου να παρουσιαστεί (πιθανότατα μέσο ενός web interface) στο χρήστη. Η λειτουργία της φάσης αυτής εξαρτάται από δύο κρίσιμους παράγοντες που αναλύονται σε επόμενη παράγραφο. Ο πρώτος, είναι η μέθοδος που έχει επιλεγεί για τη μοντελοποίηση της κεντρικής βάσης γνώσης, ενώ ο δεύτερος έχει να κάνει με το Crisis Gateway που θα χρησιμοποιηθεί για να προβληθεί το περιεχόμενο του Πρωτοκόλλου.

Όπως γίνεται κατανοητό και οι δύο παράγοντες αυτοί εξαρτώνται από τις ιδιαιτερότητες που παρουσιάζουν οι χώροι τους οποίους καλύπτει ένα Πρωτόκολλο, καθώς και από το χρησιμοποιούμενο σύστημα για την εμφάνιση του περιεχομένου στους χρήστες. Η δυσκολότερη περίπτωση που έχουμε να αντιμετωπίσουμε για τη λειτουργία του υποσυστήματος αυτού, αφορά καταστάσεις που το περιεχόμενο ενός διανύσματος Κρίσεων δεν καλύπτεται από ένα υπάρχον πρωτόκολλο διαχείρισης κρίσεων. Σε αυτή την περίπτωση υιοθετούμε μια προσεγγιστική μεθοδολογία με σκοπό να ελαχιστοποιήσουμε πιθανές απώλειες από

τυχόν υλοποίηση καταστροφικών δυνάμεων που δε μπόρεσε το Πρωτόκολλο να προβλέψει. Σε κάθε άλλη περίπτωση απαιτείται μόνο τεχνική υλοποίηση παρουσίασης των Crisis Knowledge Vectors, στο επιθυμητό user interface.

Ένα πρώτο μέτρο αποτελεσματικότητας ενός Πρωτοκόλλου Διαχείρισης Κρίσεων αποτελούν οι διαδικασίες αξιολόγησης του χρήστη επάνω στα σενάρια που υλοποίησε. Επειδή όμως μπορεί να μην υπάρχουν τέτοιες διαδικασίες για όλα τα πιθανά σενάρια που μπορούν να παραχθούν από ένα σύστημα ή να μην επιθυμεί ο χρήστης να αξιολογηθεί λόγω π.χ. έλλειψης χρόνου, προτείνεται η υλοποίηση σύντομης αξιολόγησης του μοντέλου από το χρήστη με τη μέθοδο της έρευνας ερωτηματολογίου (survey) και άλλων δομημένων τρόπων αξιολόγησης έργων. Το ερωτηματολόγιο αυτό αποτελεί αναπόσπαστο κομμάτι κάθε ενός Συστήματος υλοποίησης σεναρίων κρίσεως, συμπληρώνεται ηλεκτρονικά αμέσως μετά το πέρας της εφαρμογης του σεναρίου από το χρήστη και εκμαιεύει κρίσιμες παρατηρήσεις του χρήστη που μπορούν να οδηγήσουν στη βελτίωση της αποτελεσματικότητας όλων των φάσεων. Ο ρόλος του υποσυστήματος της Αξιολόγησης, προορίζεται στην καταγραφή αποτελεσμάτων τα οποία θα πρέπει να αξιολογηθούν και να αξιοποιηθούν από τους τροφοδότες της Βάσης Γνώσης Κρίσεων του Συστήματος και τους Ακαδημαϊκούς Υπευθύνους για την αξιολόγηση όλων των υποσυστημάτων δημιουργίας ενός ολοκληρωμένου Πρωτοκόλλου.

6.5 ΜΕΤΑΛΛΑΚΤΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΠΡΩΤΟΚΟΛΛΩΝ

Η πρώτη φάση δημιουργίας του Πρωτοκόλλου αποτελεί το θεμέλιο λίθο για την επιτυχημένη λειτουργία και δομή του μοντέλου αντιμετώπισης κρίσεων που παρουσιάσθηκε στην προηγούμενη παράγραφο.

Βασική προϋπόθεση της ανάλυσης είναι η ύπαρξη ικανοποιητικού αριθμού δεδομένων. Ο αριθμός των δεδομένων παράγεται μέσα από συνεχή σενάρια προσομοίωσης κρίσεων κινδυνικων φαινομένων. Η λεπτομερής λειτουργία του συστήματος σε αντίξοες συνθήκες και η πρόκληση τεχνιτών κρίσεων, μη προβλέψιμων προσφέρουν μια ικανή βάση πάνω στην οποία μπορεί να στηριχθεί το μοντέλο παραγωγής Πρωτοκόλλων Διαχείρισης Κρίσεων.

Με την ανάλυση του χώρου δεδομένων επιτυγχάνεται η καλύτερη εμβάθυνση των σημείων εκείνων που παρουσιάζουν αδυναμίες. Φυσικά όπως αναλύθηκε σε προηγούμενο κεφάλαιο ο κίνδυνος είναι βεβαιότητα και όχι πιθανότητα πράγμα που σημαίνει ότι κινδυνικά φαινόμενα μπορούν να εμφανιστούν ανά πάσα στιγμή μέσα στον ορίζοντα προγραμματισμού του έργου Δt.

Ο μεταλλακτικός προγραμματισμός είναι η λύση στην ανάλυση Δυναμικά μεταβαλλόμενων Χώρων Δεδομένων και τη δημιουργία ικανών Πρωτοκόλλων Διαχείρισης Κρίσεων. Το πλεονέκτημα του μεταλλακτικού προγραμματισμού είναι η ικανότητα του να παρακολουθήσει τη δυναμικότητα του χώρου και να προλάβει τη γένεση ασταθών φαινομένων ομαλοποιώντας ή στη χειρότερη περίπτωση ελαχιστοποιώντας τον αντίκτυπο που μπορούν να έχουν στη λειτουργία του συστήματος S.

Ο χώρος δεδομένων στον οποίο αναλύεται το περιβάλλον λειτουργίας των Πρωτοκόλλων ορίζεται ως S(D,R,F,Δt) όπου:

D: τα δεδομένα του υπό μελέτη χώρου, τα στοιχεία που απαιτούνται ώστε να καλυφθεί η βάση γνώσης με τον ελάχιστο όγκο που είναι αναγκαίος για την ορθή εκτίμηση και αξιοποίηση της υπάρχουσας κατάστασης.

R: το σύνολο R περιέχει μια ποικιλία δεδομένων που στηρίζονται σε συλλογισμούς, αλγόριθμους εμπλοκής δεδομένων, κανόνες λογικοί και υπολογιστικοί. Παράλληλα περιλαμβάνει σχέσεις αλληλεξάρτησης μεταξύ των δομών του συστήματος S, πράγμα ιδιαίτερα σημαντικό για την ομαλή και απρόσκοπτη λειτουργία του.

F: το σύνολο F χρίζει ιδιαίτερης προσοχής ανάλυσης και μελέτης καθώς περιλαμβάνει τις καταστροφικές δυνάμεις που επιδρούν στο R και το μεταβάλλουν, δυνάμεις που είναι εν υπνώσει στο t_p (present time), αλλά μπορεί να ενεργοποιηθούν μέσα στο διάστημα Δt.

$\Delta t = t_f - t_p$ ορίζοντας χρονοπρογραμματισμού του έργου.

t_f: μελλοντικός χρόνος – το πέρας του χρονικού ορίζοντα.

Ο δυναμικός χώρος δεδομένων ορίζει ένα Δυναμικό πεδίο συμπεριφοράς του προβλήματος (force field behavior). Δυστυχώς όμως στην περίπτωση μας ο Ασταθής Χώρος Δεδομένων τον οποίο μελετάμε δεν συνίσταται για την εξαγωγή, ανάλυση και γνώση της συνάρτησης δυναμικού του. Αντίθετα η συνάρτηση δυναμικού μας είναι άγνωστη, μη υπολογίσιμη και στη θέση της τοποθετείται η **Μεταλλακτική Μήτρα (Mutation Matrix) του συστήματος S.**

$$\mathbf{MM = mm_{ij} \quad i=1,2,3,\ldots,n \quad , \quad j= 1,2,3,\ldots,m}$$

mm_{ij}=1 , αν επιδρά στο $re_i \in R$ η ενεργοποίηση της καταστροφικής κατάστασης $f_j \in F$ μεσα στο Δt
mm_{ij}=0 , διαφορετικά.

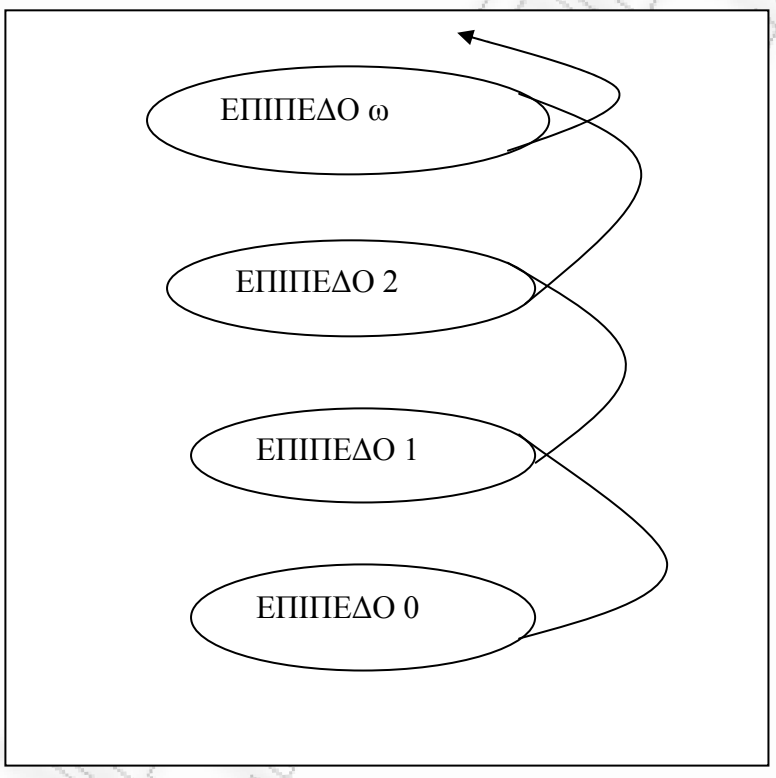
Στόχος του Μεταλλακτκού Προγραμματισμού πρωτοκόλλων διαχείρισης κρίσεων είναι σε πρώτη φάση η επιλογή του κατάλληλου υποσυνόλου $Re^* \subset Re$ έτσι ώστε να υπάρχει ελαχιστοποίηση κινδύνου μέσα στο Δt με παράλληλη ικανοποίηση απαιτήσεων και ιδιαιτεροτήτων του συστήματος S και του χώρου μέσα στον οποίο λειτουργεί.

Ειδικότερα όταν ένα τουλάχιστον στοιχείο του συνόλου F ενεργοποιηθεί μέσα στο διάστημα χρονοπρογραμματισμού του έργου δημιουργείται ένα **νέο Επίπεδο Οργάνωσης του φυσικού προβλήματος** διαφορετικό από το ήδη υπάρχον τη χρονική στιγμή t_p.

Με τη βοήθεια **Ταξονομίας ή με Ιεραρχική Ανάλυση (κλίμακα Goodman)** μπορούν να δομηθούν οι επιδράσεις των f_j στα στοιχεία του R με άμεσο αποτέλεσμα την εμφάνιση πιθανών Επιπέδων Οργάνωσης. Τα επίπεδα οργάνωσης περιέχουν πλήθος μεταβολών που συντελέσθηκαν στο διάστημα Δt. [Goodman (1975)]

Επιτυγχάνεται έτσι μια πρώτη προσέγγιση όπου με την κατάλληλη προσαρμογή το πρωτόκολλο προσδιορίζει το επίπεδο, και ειδικότερα το πλήθος των μεταβολών που έχουν λάβει χώρα στο σύστημα. Η προσέγγιση αυτή γίνεται σε δυο παράλληλα επίπεδα ανάλυσης και σχεδιασμού. Το πρώτο τμήμα περιλαμβάνει την υλοποίηση μεθόδων διαχείρισης κρίσεων στο συγκεκριμένο επίπεδο στο οποίο βρίσκεται αυτή τη δεδομένη χρονική στιγμή η υπάρχουσα κατάσταση το συστήματος S. Η κίνηση με λίγα λόγια στο επίπεδο αυτό είναι οριζόντια μέσα στο πλαίσιο λειτουργίας του συστήματος τη χρονική στιγμή t_p.

Το δεύτερο τμήμα περιλαμβάνει την ανάλυση και το σχεδιασμό μεθόδων ομαλής μετάβασης από το ένα επίπεδο οργάνωσης στο άλλο και ελαχιστοποίησης των απωλειών. Η κίνηση στο επίπεδο αυτό είναι κατακόρυφη ώστε να επιτυγχάνεται η μετάβαση στα διαφορετικά διαστρωματικά επίπεδα οργάνωσης. Η σύνθεση των δυο παραπάνω κινήσεων απεικονίζεται στο παρακάτω σχήμα.(Σχήμα 6.2) Η ταυτόχρονη κίνηση στα δυο επίπεδα που αναλύθηκαν προηγουμένως δημιουργούν **τη Λογισμική Οργάνωση Σπείρας πάνω στην οποία δομείται ο Μεταλλακτικός Προγραμματισμός πρωτοκόλλων διαχείρισης κρίσεων**.



Σχήμα 6.2
Λειτουργία Πρωτοκόλλων Κρίσεων

6.6 ΤΟ ΦΑΙΝΟΜΕΝΟ ΤΟΥ ΝΤΟΜΙΝΟ

Ο Χώρος δεδομένων μέσα στον οποίο δρα ένα Ψηφιακό πρωτόκολλο διαχείρισης κρίσεων είναι ένας ασταθής χώρος δεδομένων. Παράλληλα η δομή του μεταβάλλεται δυναμικά όπως έχει προαναφερθεί. Αυτό είναι και το πιο δύσκολο σημείο ανάπτυξης και λειτουργίας των πρωτοκόλλων. Η δυσκολία αυτή επισημαίνεται κυρίως στην πρώτη φάση της ανάλυσης και λιγότερο στον σχεδιασμό.

Σε ομαλούς χώρους δεδομένων τα πράγματα είναι πολύ απλά. Η ομαλότητα του χώρου δίνει τη δυνατότητα στον αναλυτή- σχεδιαστή του πρωτοκόλλου διαχείρισης κρίσεων να γνωρίζει τη συνάρτηση δυναμικού πράγμα που το βοηθάει να προβλέψει την εξέλιξη μιας κρίσης ώστε να προετοιμάσει το σύστημα κατάλληλα. Οι κρίσεις υλοποιούνται με συγκεκριμένο τρόπο στηριζόμενες σε πιθανοθεωρητικά στατιστικά μοντέλα πρόγνωσης. Έτσι ένα πρωτόκολλο στηριζόμενο σε τέτοια στατιστικά μοντέλα μπορεί να αντεπεξέλθει στις απαιτήσεις του συγκεκριμένου χώρου δεδομένων προστατεύοντας όσο το δυνατόν καλύτερα το σύστημα S το οποίο λειτουργεί μέσα του.

Αντίθετα σε μη ομαλούς δυναμικά μεταβαλλόμενους χώρους δεδομένων τα πράγματα γίνονται ιδιαίτερα πολύπλοκα. Η πολυπλοκότητα αυξάνει αν αναλογιστεί κανείς ότι οι χώροι αυτοί δεν υπόκεινται σε κανένα στατιστικό πιθανοθεωρητικό μοντέλο λειτουργίας. Χαοτικές καταστάσεις οριοθετούν τους κανόνες λειτουργίας τους. Άμεσο αποτέλεσμα είναι οποιοδήποτε πρωτόκολλο αναπτυχθεί να στηρίζεται σε ασύμβατες μεθόδους υλοποίησης οι οποίες πολλές φορές υπερβαίνουν βασικούς στατιστικούς και μαθηματικούς κανόνες. Οι μέθοδοι αυτοί είναι κυρίως στοχαστικοί οι οποίοι προσπαθούν να ισορροπήσουν μεταξύ πραγματικών μαθηματικών μοντέλων και θεωρητικών προσεγγίσεων.

Οι δυνάμεις που υλοποιούνται σε ένα τέτοιο περιβάλλον λειτουργίας είναι απρόβλεπτες με καταστροφικές συνήθως συνέπειες. Στην ουσία πρόκειται για ασύμμετρες απειλές όπως έχει σήμερα ορισθεί η έννοια αυτή. Η μη συμμετρικότητα αυτή καθιστά τα πρωτοκόλλα αδύνατα να προβλέψουν με ασφαλή τρόπο το χρόνο υλοποίησης τους μέσα στο διάστημα χρονοπρογραμματισμού του έργου Δt. Πόσο μάλλον αδύνατη καθίσταται η αντιμετώπιση μιας τέτοιας απειλής η οποία δεν μπορεί να προβλεφθεί.

Η λύση στο πρόβλημα αυτό έρχεται μέσα από στοχαστικά μοντέλα διαχείρισης κρίσεων. Στόχος είναι ο περιορισμός της ασάφειας που υπάρχει και η προσπάθεια συγκεκριμενοποίησης του προβλήματος. Μια λύση που προτείνεται είναι η μετατόπιση του προβλήματος από το πεδίο της πρόβλεψης των καταστροφικών δυνάμεων στο πεδίο των υποσυστημάτων του συστήματος S και τις σχέσεις μεταξύ αυτών. Μεταφέροντας το πρόβλημα σε εκείνο το πεδίο θα μπορούν να επισημανθούν οι αδυναμίες του συστήματος ώστε να δημιουργηθούν κατάλληλα πρωτόκολλα διαχείρισης κρίσεων που θα θωρακίζουν αυτές τις αδυναμίες.

Ειδικότερα έστω S το σύστημα που μελετάται μέσα στον ορίζοντα χρονοπρογραμματισμού Δt και το σύνολο των υποσυστημάτων του:

$$S=\{s_1,s_2,\ldots,s_i,\ldots,s_n\} \quad , \quad S^*=\{1,2,\ldots,n\}$$

Στο σημείο αυτό σημαντικό ρόλο δεν παίζει η υλοποίηση μιας κρίσης σε κάποιο υποσύστημα αλλά οι σχέσεις αλληλεξάρτησης που διέπουν τα υποσυστήματα. Στόχος είναι η ισχυροποίηση του συστήματος μέσο των υποσυστημάτων του που αποτελούν δομικά του συστατικά. Οι σχέσεις αλληλεξάρτησης μεταξύ των υποσυστημάτων είναι ιδιαίτερα σημαντικές καθώς χαρτογραφούν τη δομή του όλου συστήματος προσδιορίζοντας τυχόν αδύναμα σημεία και ειδικότερα κεντρικά σημεία αυτοκατάρρευσης του συστήματος S που μπορεί να ενεργοποιηθούν με τη βοήθεια μιας κρίσης. Η **μήτρα σχέσεων των υποσυστημάτων του συστήματος S (Subsystems Relation Matrix)** είναι ένα σημαντικό εργαλείο που προσδιορίζει τις σχέσεις αλληλεξάρτησης όπως αυτές ορίσθηκαν παραπάνω. Έτσι έχουμε:

RS = (rs$_{ij}$), $\forall i \in S^*,\ j \in S^*$

rs$_{ij}$ = 1, αν το υποσύστημα s$_i$
επηρεάζει το υποσύστημα s$_j$
rs$_{ij}$= 0, διαφορετικά.

Μέσα από την ανάλυση των σχέσεων προκύπτουν ιδιαίτερα σημαντικά συμπεράσματα χρήσιμα για την ανάλυση, το σχεδιασμό και την υλοποίηση ψηφιακών πρωτοκόλλων διαχείρισης κρίσεων σε ασταθής χώρους δεδομένων. Πρέπει να τονισθεί στο σημείο αυτό ότι η εξάρτηση μεταξύ των υποσυστημάτων δεν είναι αμφίδρομη. Θεωρείται αυτονόητο ότι ένα σύστημα μπορεί να εξαρτάται από ένα άλλο χωρίς να ισχύει η ανάποδη σχέση με οτιδήποτε αυτό συνεπάγεται για το όλο σύστημα.

Θεώρημα Εντοπισμού Μαύρων Λειτουργικών Οπών (Black Operational Hole Locating Theorem)

Αν ισχύει:

$$(\forall i \in S^*)(\sum_{j=1}^{n} \text{rs}_{ij}=\text{n} ,\ \forall j \in S^*\ )$$

τότε το υποσύστημα s$_i$ αποτελεί κεντρικό σημείο αυτοκατάρρευσης του συστήματος S.

Η απόδειξη του παραπάνω θεωρήματος είναι απλή και αυτονόητη. Πράγματι από τα δεδομένα προκύπτει ότι αν ένα υποσύστημα επηρεάζει ή εξαρτώνται από αυτό τα υπόλοιπα υποσυστήματα του συστήματος S τότε η κατάρρευση του θα προκαλέσει αλυσιδωτές αντιδράσεις. Τα υπόλοιπα υποσυστήματα που επηρεάζονται από αυτό θα μεταλλαχθούν οδηγώντας το κυρίως σύστημα σε πιθανή αυτοκατάρρευση.

Θεώρημα Εντοπισμού Μεταλλακτικών Σημείων (Gray Operational Hole Locating Theorem)

Αν ισχύει:

$$(\forall i \in S^{*})(\sum_{j=1}^{n} \mathbf{rs_{ij}<n}\ ,\ \forall j \in S^{*}\ )$$

τότε το υποσύστημα s_i αποτελεί σημείο μετάλλαξης του συστήματος S.

Η απόδειξη και σε αυτή την περίπτωση είναι πολύ απλή. Η κατάρρευση ενός υποσυστήματος οδηγεί σε μερική κατάρρευση του συστήματος αφού θα παραμείνουν ενεργά μερικά υποσυστήματα τα οποία δεν επηρεάζονται από τη μεταβολή αυτή. Εντοπίζοντας έτσι τα αδύναμα στοιχεία μπορούν να κατασκευαστούν ψηφιακά πρωτόκολλα διαχείρισης κρίσεων. Σκοπός τους θα είναι η προστασία και ενδυνάμωση των αδύνατων σημείων από κρίσεις που μπορούν να υλοποιηθούν τυχαία μέσα στο διάστημα Δt.

Οι δυο παραπάνω περιπτώσεις είναι οι πιο απλές μορφές τις οποίες μπορεί να επισημάνει ο αναλυτής – σχεδιαστής του πρωτοκόλλου διαχείρισης κρίσεων.

Οι δαιδαλώδης δομές των σύγχρονων κατανεμημένων ψηφιακών συστημάτων κρύβουν παγίδες οι οποίες δεν είναι εμφανής. Ένα επιτυχημένο ολοκληρωμένο σύστημα διαχείρισης κρίσεων οφείλει να μπορεί να επισημάνει αυτές τις παγίδες οι οποίες εγκυμονούν κινδύνους εν υπνώσει και τυχόν ενεργοποίηση τους θα αποδυναμώσει τη δομή του συστήματος. Οι παγίδες αυτές έγκεινται στο γεγονός ότι η κατάρρευση ενός υποσυστήματος μπορεί να προκαλέσει αλυσιδωτή αντίδραση γεγονότων. Μια αντίδραση με απρόβλεπτες διαστάσεις ικανή να οδηγήσει με τη σειρά της στην κατάρρευση και άλλων υποσυστημάτων του κεντρικού συστήματος. Άμεσο αποτέλεσμα τέτοιων καταστάσεων είναι η μετάλλαξη του κεντρικού συστήματος ή στη χειρότερη περίπτωση

αυτό που είναι απευκταίο η ολοκληρωτική κατάρρευση του συστήματος S.

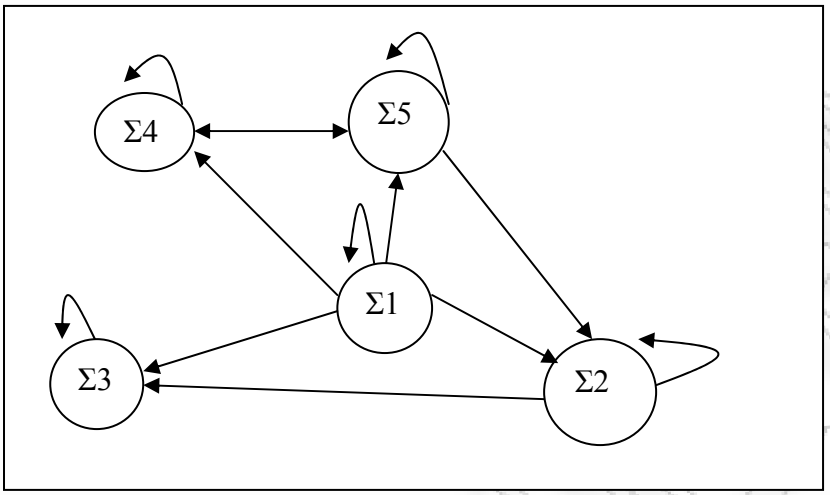
Η αλυσιδωτή πτώση διαδοχικών αλληλεξαρτώμενων υποσυστημάτων οδηγεί με μαθηματική ακρίβεια στη συνολική κατάρρευση του συστήματος. **Υποσυστήματα καταρρέουν αλυσιδωτά σαν ντόμινο προκαλώντας πτώση συστημάτων.**

Οι αλυσιδωτές αυτές πτώσεις είναι το σημείο στο οποίο πρέπει να εστιάσει το σύστημα διαχείρισης κρίσεων προκειμένου να αντιμετωπιστούν πιθανά φαινόμενα κρίσεων. Οι αλυσίδες ποικίλουν ανάλογα με το μέγεθος και την κατανομή του υπό μελέτη πληροφοριακού συστήματος.

6.6.1 ΕΦΑΡΜΟΓΗ ΤΟΥ ΝΤΟΜΙΝΟ

Για να γίνει καλύτερα αντιληπτό το μοντέλο της θεωρίας του ντόμινο θα παρουσιασθεί ένα παράδειγμα πάνω στο οποίο στηρίζεται η προαναφερθείσα θεωρία. Χωρίς βλάβη της γενικότητας ας υποθέσουμε ένα τυχαίο σύστημα το οποίο αποτελείται από 5 υποσυστήματα. Η δομή και αλληλεξάρτηση των υποσυστημάτων παρουσιάζεται στο παρακάτω σχήμα (Σχήμα 6.3)

Είναι, λοιπόν, δυνατό να αντιστοιχισθεί η αλληλεξάρτηση των υποσυστημάτων με ένα προσανατολισμένο γράφημα δεσμών στο οποίο οι κόμβοι αποτελούν τα υποσυστήματα του κάθε επιπέδου, ενώ τα τόξα αντιπροσωπεύουν την εξάρτηση μεταξύ των συστημάτων. (Σχήμα 6.3). Τα τόξα δείχνουν από το υποσύστημα εξάρτησης προς τα εξαρτημένα υποσυστήματα του ιδίου επιπέδου. Ακόμη γίνεται η υπόθεση ότι η λειτουργία του κάθε υποσυστήματος εξαρτάται πάντα από τον εαυτό του και γραφικά αναπαρίσταται από τα κυρτά τόξα του σχήματος.



Σχήμα 6.3
Προσανατολισμένο γράφημα δεσμών για την αναπαράσταση σχέσεων των υποσυστημάτων

Έτσι, για το παραπάνω γράφημα του παραδείγματός και κάθε υποσύστημά του ισχύει:

- Σ1: Η κατάρρευση αυτού του συστήματος έχει σαν συνέπεια σε πρώτο χρόνο την κατάρρευση των συστημάτων Σ2, Σ3, Σ4 και Σ5
- Σ2: Η κατάρρευση αυτού του υποσυστήματος έχει ως συνέπεια την κατάρρευση του Σ3.
- Σ3: Η κατάρρευση αυτού του υποσυστήματος δεν έχει καμία συνέπεια για τα άλλα υποσυστήματα.
- Σ4 : Η κατάρρευση αυτού του υποσυστήματος έχει σαν συνέπεια σε πρώτο την κατάρρευση του συστήματος Σ5 που με τη σειρά του σε δεύτερο χρόνο οδηγεί στην κατάρρευση του Σ2 και σε τρίτο χρόνο τη κατάρρευση του Σ3.
- Σ5: Η κατάρρευση αυτού του υποσυστήματος έχει σαν συνέπεια σε πρώτο την κατάρρευση των συστημάτων Σ4, Σ2 που το τελευταίο με τη σειρά του σε δεύτερο χρόνο οδηγεί στην κατάρρευση του Σ2 και σε τρίτο χρόνο τη κατάρρευση του Σ3.

Εύκολα κανείς, μπορεί να παρατηρήσει ότι το Σ1 αποτελεί ένα κεντρικό σημείο αυτοκατάρρευσης του συστήματος, μια που η κατάρρευση του έχει ως συνέπεια την πλήρη διάλυση όλων των άλλων υποσυστημάτων. Όλες οι υπόλοιπες περιπτώσεις κατάρρευσης υποσυστημάτων οδηγούν το σύστημα σε μετάλλαξη, καθώς κάποια από τα υποσυστήματα του λειτουργούν.

Μια άλλη παρατήρηση που θα μπορούσε κανείς να κάνει, είναι το «φαινόμενο Ντόμινο» που παρατηρείται όταν συγκεκριμένα υποσυστήματα τίθενται εκτός λειτουργίας. Κάτι τέτοιο παρατηρείται στη περίπτωση κατάρρευσης του Σ4 και του Σ5.

Αυτό το «φαινόμενο Ντόμινο», είναι πιο γνωστό ως Δρόμος του Hamilton (Hamiltonian Path). Όπως είναι γνωστό, από τη Θεωρία γραφημάτων, δρόμος ή τροχιά ενός γραφήματος τόξων ονομάζεται κάθε σύνολο διαδοχικών τόξων. Με βάση τον ορισμό αυτό για το γράφημα του παραδείγματος οι τροχιές είναι:

1) Σ1 Σ4 Σ5 Σ2 Σ3 τροχιά μήκους 4
2) Σ1 Σ5 Σ2 Σ3 τροχιά μήκους 3
3) Σ1 Σ2 Σ3 τροχιά μήκους 2
4) Σ1 Σ2 τροχιά μήκους 1
5) Σ1 Σ3 τροχιά μήκους 1
6) Σ1 Σ4 τροχιά μήκους 1
7) Σ1 μηδενική τροχιά
8) Σ1 Σ5 τροχιά μήκους 1
9) Σ2 Σ3 τροχιά μήκους 1
10) Σ2 μηδενική τροχιά
11) Σ3 μηδενική τροχιά
12) Σ4 Σ5 Σ2 Σ3 τροχιά μήκους 3
13) Σ4 μηδενική τροχιά
14) Σ5 Σ2 Σ3 τροχιά μήκους 2
15) Σ5 Σ4 τροχιά μήκους 1
16) Σ5 μηδενική τροχιά

Σκοπός, όπως γίνεται εύκολα αντιληπτό, ενός επιτυχημένου Πρωτοκόλλου Διαχείρισης Κρίσεων είναι η εύρεση των Μονοπατιών Hamilton που προκαλούν κρίση στα υποσυστήματα του κυρίως συστήματος. Προϊόντα των κρίσεων αυτών είναι μεταλλάξεις οι οποίες λαμβάνουν χώρα σε μονοπάτια μικρού μήκους σε σχέση με το συνολικό μέγεθος του μέγιστου μονοπατιού.

Ενδιαφέρον παρουσιάζει η περίπτωση εμφάνισης μονοπατιών μεγίστου μήκους. **Τα μονοπάτια αυτά μπορεί να είναι σημεία αυτοκατάρρευσης του συνόλου του συστήματος** και υλοποιούνται στον ορίζοντα χρονοπρογραμματισμού του έργου Δt.

Τέλος συμπεραίνουμε ότι, όσο πιο μεγάλος είναι ο βαθμός της υπό μελέτης κορυφής τόσο πιθανό είναι η κατάρρευση της να οδηγήσει στην

κατάρρευση εξαρτώμενων κορυφών και συνεπώς στην ολική κατάρρευση του συστήματος S.

6.6.2 ΑΛΓΟΡΙΘΜΟΣ ΝΤΟΜΙΝΟ

Σημαντικό κομμάτι είναι η παρουσίαση του αλγόριθμου πάνω στον οποίο στηρίζεται η θεωρία του ντόμινο. Η θεωρία αυτή αποτελεί τη βάση για τα Πρωτόκολλα Διαχείρισης Κρίσεων. Στη συνέχεια παρατίθεται ο αλγόριθμος εύρεσης Μονοπατιών Hamilton η χρησιμότητα των οποίων αναλύθηκε σε προηγούμενη παράγραφο.

Ο πίνακας αλληλεξάρτησης των υποσυστημάτων του προηγούμενου σχήματος (Σχήμα 6.3) έχει την ακόλουθη μορφή:

$$RS = \begin{matrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \end{matrix}$$

Τα μονοπάτια Hamilton είναι ένα εργαλείο χρήσιμο για την πρόληψη κρίσεων και την αποσόβηση εκτεταμένων καταστροφικών φαινομένων. Στη συνέχεια παρουσιάζεται ο **αλγόριθμος Ενεργοποίησης Πρωτοκόλλου Διαχείρισης Κρίσεων (Crisis Management Protocol Activation Algorithm)(CMPAA)** στηριζόμενος στα προαναφερθέντα. Ο αλγόριθμος λειτουργεί αναδρομικά για κάθε επίπεδο ανάλυσης του συστήματος S καθώς και των υποσυστημάτων αυτού, θεωρώντας κάθε ένα από αυτά ξεχωριστή αυτόνομη οντότητα.

Αλγόριθμος Ενεργοποίησης Πρωτοκόλλου Διαχείρισης Κρίσεων (Crisis Management Protocols Activation Algorithm)(CMPAA)

∀ *Επίπεδο Ανάλυσης κάνε.*
Δημιουργία Πίνακα Subsystems Relations RS.
Εύρεση μονοπατιών Hamilton.
∀ *μέγιστο μονοπάτι κάνε.*
Ενεργοποίηση μεθοδολογίας Πρόληψης Κινδυνικών φαινομένων.
Αν κρίση δεν αντιμετωπιστεί.
Ενεργοποίηση μεθοδολογίας Αντιμετώπισης Κινδυνικών φαινομένων.

Ο παραπάνω αλγόριθμος αποτελεί τη λύση στην ενεργοποίηση των κατάλληλων Πρωτοκόλλων Διαχείρισης Κρίσεων σε ασταθείς χώρους δεδομένων όπου οι συνθήκες λειτουργίας καθιστούν το περιβάλλον ασταθές, εχθρικό και το πιο σημαντικό από όλα μη προβλέψιμο.

6.7 ΣΥΜΠΕΡΑΣΜΑΤΑ

Τα ψηφιακά πρωτόκολλα διαχείρισης κρίσεων είναι η λύση στην αδυναμία πρόληψης κινδυνικών φαινομένων τα οποία λαμβάνουν χώρα σε Δυναμικά μεταβαλλόμενους χώρους δεδομένων. Η λειτουργία τους παρουσιάζει κάποιες αδυναμίες λόγω του γεγονότος ότι αδυνατούν να προβλέψουν μια κρίση στηριζόμενες στις καταστροφικές δυνάμεις που μπορούν να υλοποιηθούν μέσα στον ορίζοντα χρονοπρογραμματισμού του έργου.

Το κενό αυτό καλύπτουν τα μονοπάτια του Hamilton. Τα υποσυστήματα και οι σχέσεις αλληλεξαρτήσεις είναι η λύση στην αστάθεια του χώρου δεδομένων. Οι δομικοί δεσμοί μεταξύ των υποσυστημάτων του συστήματος μπορούν κάτω από ειδική μελέτη και επίβλεψη να συμβάλλουν στην προστασία του από οποιαδήποτε κρίση και οποτεδήποτε αυτή συμβεί.

ΚΕΦΑΛΑΙΟ 7

ΣΥΜΠΕΡΑΣΜΑΤΑ ΔΙΑΤΡΙΒΗΣ

7. ΣΥΜΠΕΡΑΣΜΑΤΑ

7.1 ΣΥΜΠΕΡΑΣΜΑΤΙΚΑ ΣΧΟΛΙΑ ΚΑΙ ΠΑΡΑΤΗΡΗΣΕΙΣ

"Η επιστήμη πάντοτε πέφτει έξω, ποτέ δεν έλυσε ένα πρόβλημα χωρίς να θέσει κάποιο άλλο"

Μπ. Σω

Τα συμπεράσματα και οι παρατηρήσεις, που προέκυψαν από τη βιβλιογραφική μελέτη και την έρευνα, που συντελέστηκε στην παρούσα διδακτορική διατριβή, συνοψίζονται παρακάτω:

- Η μεθοδολογία διαχειρίζεται κρίσεις σε οποιοδήποτε είδος ψηφιακού πληροφοριακού συστήματος.
- Η γεωγραφική διάσπαση του συστήματος δεν επηρεάζει την ομαλή εφαρμογή της μεθοδολογίας.
- Η πολυπλοκότητα του υπό μελέτη συστήματος δεν αποτελεί πρόβλημα για την εφαρμογή της θεωρίας.
- Η γενική θεωρία αντιμετωπίζει όλα τα είδη των κινδύνων δίχως αποκλεισμούς και εξαιρέσεις.
- Η Θεωρία Διοίκησης Κινδύνου δεν εξαλείφει τους κινδύνους, αφού δεν υπόσχεται 100% ασφάλεια. Διαχειρίζεται τις κρίσεις και τις συνέπειες αυτών στο πληροφοριακό σύστημα.
- Η γενική θεωρία και οι υποθεωρίες της εφαρμόζονται σε ασταθείς, δυναμικά μεταβαλλόμενους χώρους δεδομένων.
- Τα υπάρχοντα συστήματα διαχείρισης κινδύνων εστιάζονται στη μελέτη και βαθμολόγηση των υφισταμένων δομών ασφαλείας.
- Η πλειοψηφία των μεθοδολογιών ασχολείται με την πρόληψη κινδύνων.
- Άμεσο αποτέλεσμα του παραπάνω συμπεράσματος είναι το γεγονός ότι οι περισσότερες μεθοδολογίες διαχείρισης κινδύνου

δεν μελετούν την κατάσταση μετά την καταστροφή του συστήματος.

- Τα σενάρια κρίσεως στηρίζονται σε “what if “ ερωτήματα χωρίς εμβάθυνση στις συνέπειες της καταστροφής, την αντιμετώπιση της και το κόστος ανάκτησης των δεδομένων.

- Δεν μελετάται διεξοδικά η ανάκτηση των καταστραμμένων δεδομένων.

- Μέρος των συστημάτων ασφάλειας αντιμετωπίζει συγκεκριμένα είδη κινδύνων.

- Δεν αντιμετωπίζονται όλα τα πιθανά φαινόμενα κρίσεως, που μπορούν να εμφανιστούν.

- Οι κίνδυνοι υπολογίζονται ότι υλοποιούνται με υποκειμενικά, πιθανοθεωρητικά κριτήρια.

- Η πλειοψηφία των συστημάτων δεν υπολογίζει τους κινδύνους που είναι απίθανο να συμβούν θεωρώντας τους αμελητέους.

- Αρκετές μεθοδολογίες βαθμολογούν το επίπεδο των κινδύνων που απειλούν το υπό μελέτη σύστημα κατηγοριοποιώντας τους ποιοτικά ή ποσοτικά.

- Πλήθος μεθοδολογιών στηρίζεται σε γνωστά μαθηματικά και στατιστικά μοντέλα κατανομής και προσδιορισμού της πιθανότητας εμφάνισης κινδυνικών φαινομένων, χωρίς τη δυνατότητα πρόβλεψης του αναπάντεχου.

- Δεν υπολογίζεται το κόστος ενεργοποίησης αντιμέτρων του συστήματος.

- Δεν υπολογίζεται το κόστος ανάκτησης κατεστραμμένων δεδομένων από μια κρίση.

- Δεν υπολογίζεται η αξία των κατεστραμμένων υλικών.

- Δεν υπολογίζεται το κόστος συντήρησης ενός τέτοιου συστήματος.

- Με κατάλληλες μετατροπές η γενική θεωρία διαχείρισης κρίσεων εφαρμόζεται και σε ομαλούς, δυναμικούς χώρους δεδομένων.

- Η γενική θεωρία Κινδύνου δεν υπολογίζει την Αστάθεια του χώρου δεδομένων. Η θεωρία των καταστροφών και γενικότερα τα χαοτικά μαθηματικά μοντέλα είναι κατάλληλα για τέτοιου είδους μετρήσεις.

- Η πλειοψηφία των θεωριών διαχείρισης κινδύνου μελετούν κινδυνικά φαινόμενα με ποιοτικά κριτήρια και κλίμακες στηριζόμενα σε υποκειμενικές εκτιμήσεις.

- Άλλες θεωρίες υπολογίζουν τον κίνδυνο στηριζόμενες σε ποσοτικά χαρακτηριστικά ανάλογα με το κόστος, το χρόνο, ή την πιθανότητα εμφάνισης του κινδύνου.

- Η γενική θεωρία Κινδύνου είναι ανεξάρτητη από το περιβάλλον λειτουργίας του υπό μελέτη συστήματος.

- Η γενική θεωρία διαχείρισης Κινδύνου είναι ένα υποσύνολο της γενικής θεωρίας διαχείρισης Έργου μέσα από την οποία προσεγγίζεται δομημένα η αναγνώριση, η ανάλυση και η αντιμετώπιση κινδυνικών φαινομένων.

- Οι περισσότερες μεθοδολογίες διαχείρισης κινδύνου στηρίζονται στο πλαίσιο της ανάλυσης (Risk Analysis) – αναγνώρισης (Risk Identification) – επιλογής (Risk Resolution) και υλοποίησης (Risk Monitoring). Παραλλαγές στο πλαίσιο παρουσιάζονται σε πληθώρα άλλων μεθοδολογιών.

- Η γενική θεωρία διαχείρισης κινδύνου αλλάζει ριζικά το πλαίσιο ανάπτυξης. Το νέο προτεινόμενο πλαίσιο χωρίζεται σε δυο υποθεωρίες, που περιλαμβάνουν την Πρόληψη Κινδυνικών φαινομένων και την Αντιμετώπιση Καταστροφικών φαινομένων αντίστοιχα.

- Κάθε υποθεωρία με τη σειρά της δημιουργεί ένα νέο πλαίσιο λειτουργίας.

- Η πρόληψη κινδυνικών φαινομένων περιέχει ένα διευρυμένο πλαίσιο ανάπτυξης που περιλαμβάνει την Ανάλυση του

συστήματος, τον Προσδιορισμό των Κινδυνικών φαινομένων, την Ποιοτική μελέτη των αδυναμιών του συστήματος και την ποσοτική ανάλυση του πιθανού κινδυνικού φαινομένου.

- Παρουσιάζεται η έννοια του μεταλλακτικού προγραμματισμού και της μετάλλαξης ενός ψηφιακού πληροφοριακού συστήματος.

- Η διαχείριση του κινδύνου παρουσιάζεται ως ένα ευρύτερο τμήμα της γενικής θεωρίας συστημάτων που είναι συγγενές με το υποσύνολο του μεταλλακτικού προγραμματισμού.

- Προσδιορίζονται και οριοθετούνται οι χώροι δεδομένων μέσα στους οποίους δρα ένα ψηφιακό πληροφοριακό σύστημα. Επιπλέον, παρουσιάζεται η έννοια του ασταθούς χώρου δεδομένων μέσα στον οποίο υλοποιείται το μοντέλο της μεθοδολογίας.

- Παρουσιάζονται νέα και ευέλικτα εργαλεία διαχείρισης κρίσεων, όπως ο Χάρτης Καταστροφών, ο Χάρτης Κινδύνου, Χάρτης Αντιμετώπισης και ο Πίνακας Κόστους Ενεργοποίησης Αντιμέτρων .

- Μέσα από την πρόληψη κινδυνικών φαινομένων, με τη βοήθεια ποιοτικών μετρήσεων, γίνεται χαρτογράφηση του συστήματος και των αδυναμιών του, ώστε να είναι εφικτή η πρόληψη κρίσεων.

- Εισάγονται οι έννοιες των κεντρικών σημείων αυτοκατάρρευσης (Μαύρων Προγραμματιστικών Οπών), των μεταλλακτικών σημείων (Γκρίζες Προγραμματιστικές Οπές) και των Λευκών Προγραμματιστικών Οπών.

- Παρουσιάζεται η έννοια της καταστροφής σαν πιθανό ενδεχόμενο σε αντίθεση με υπάρχουσες μεθοδολογίες, ενώ παράλληλα μελετώνται μέθοδοι ανάκτησης από πτώση.

- Μελετάται διεξοδικά το πλαίσιο ανάκτησης από πτώση στηριζόμενο τόσο στην ύπαρξη κατάλληλων αντιμέτρων όσο και στην ενεργοποίηση των καταλληλότερων με το μικρότερο κόστος λειτουργίας.

- Ειδικότερα, το πλαίσιο λειτουργίας της υπομεθοδολογίας περιέχει το στάδιο της Ανάλυσης των κινδύνων, του Προσδιορισμού των

αντιμέτρων, της Κοστολόγησης ενεργοποίησης του αντιμέτρου, της εύρεσης βέλτιστης λύσης ανάκτησης των δεδομένων, του υπολογισμού καταστροφών και του υπολογισμού απόδοσης αντιμέτρων Ανάκτησης.

- Ποσοτικοποιείται η καταστροφή με τον υπολογισμό του κόστους ανάκτησης ως αναπόσπαστο κομμάτι της υπομεθοδολογίας ανάκτησης καταστροφικών δεδομένων.

- Ενεργοποιείται το καταλληλότερο αντίμετρο με το ελάχιστο δυνατό κόστος ενεργοποίησης, που αποτελεί τη βέλτιστη λύση ανάκτησης δεδομένων.

- Παράλληλα, προσδιορίζεται τόσο ποιοτικά όσο και ποσοτικά η απόδοση της ανάκτησης με τη βοήθεια των αντιστοίχων εργαλείων.

- Προσδιορίζεται τόσο ποιοτικά όσο και ποσοτικά το κόστος ανάκτησης των δεδομένων.

- Τα δεδομένα κατηγοριοποιούνται σε υλικά και ηλεκτρονικά καθένα από τα οποία κοστολογείται αντίστοιχα.

- Προσδιορίζεται το κόστος ανάκτησης των δεδομένων στην περίπτωση μερικής ανάκτησης.

- Υπολογίζεται η απόδοση του σχεδίου ανάκτησης στηριζόμενη τόσο σε ποιοτικά κριτήρια, όπως ο FSR, όσο και ποσοτικά κριτήρια με το λόγο του πραγματικού ποσού ανάκτησης προς το προϋπολογισθέν πόσο.

- Παρουσιάζεται το Ψηφιακό Πληροφοριακό Σύστημα Εθνικής Άμυνας, ο Ψηφιακός Στρατιώτης. Αναλύεται η δομή του, η λειτουργία του και το περιβάλλον λειτουργίας του.

- Αναλύονται οι κίνδυνοι που αντιμετωπίζει στο πεδίο της μάχης το ψηφιακό του σύστημα καθώς και παρουσιάζονται προοπτικές επεκτάσης και ανάπτυξης του συστήματος.

- Προτείνονται λύσεις υλοποίησης σεναρίων κρίσεως και σε άλλους κλάδους των Ενόπλων Δυνάμεων.

- Το κομμάτι των σεναρίων στηρίζεται στην επιλογή κατάλληλων συνθηκών κρίσεων με τη μέθοδο “what if”.

- Δημιουργείται διάνυσμα κρίσεων και προσδιορίζεται ο κίνδυνος ποιοτικά με τη βοήθεια αντίστοιχων θεωρημάτων.

- Εξάγονται στατιστικά αποτελέσματα με τη βοήθεια της ποιοτικής ανάλυσης.

- Καταστροφικά σενάρια υλοποιούνται και τα αποτελέσματα αξιολογούνται αντίστοιχα.

7.2 ΣΥΜΒΟΛΗ ΤΗΣ ΔΙΑΤΡΙΒΗΣ ΣΤΟΝ ΕΠΙΣΤΗΜΟΝΙΚΟ ΧΩΡΟ

Η πρωτότυπη έρευνα, η οποία πραγματοποιήθηκε στο πλαίσιο αυτής της διδακτορικής διατριβής συμβάλει στην προώθηση της επιστήμης της Διαχείρισης Κίνδυνου σε Ψηφιακά Πληροφοριακά Συστήματα αλλά και στο πλαίσιο δημιουργίας, συγκρότησης και λειτουργίας των Πληροφοριακών Συστημάτων Εθνικής Άμυνας στα παρακάτω σημεία:

- Η γενική θεωρία αντιμετωπίζει όλα τα είδη των κινδύνων δίχως αποκλεισμούς και εξαιρέσεις.

- Η Θεωρία Διοίκησης Κινδύνου δεν εξαλείφει τους κινδύνους, αφού δεν υπόσχεται 100% ασφάλεια. Διαχειρίζεται όμως τις κρίσεις και τις συνέπειες αυτών στο πληροφοριακό σύστημα.

- Η γενική θεωρία και οι υποθεωρίες της εφαρμόζονται σε ασταθείς, δυναμικά μεταβαλλόμενους χώρους δεδομένων.

- Άμεσο αποτέλεσμα του παραπάνω συμπεράσματος είναι το γεγονός ότι η μεθοδολογία διαχείρισης κινδύνου μελετά την κατάσταση του συστήματος μετά την καταστροφή.

- Μελετάται διεξοδικά η ανάκτηση των κατεστραμμένων δεδομένων.

- Μέρος των συστημάτων ασφάλειας αντιμετωπίζει όλα τα είδη των κινδύνων.

- Αντιμετωπίζονται όλα τα πιθανά φαινόμενα κρίσεως που μπορούν να εμφανιστούν.

- Οι κίνδυνοι δεν υπολογίζονται με υποκειμενικά, πιθανοθεωρητικά κριτήρια.

- Η μεθοδολογία διαχείρισης κινδύνου υπολογίζει τους κινδύνους που απειλούν το σύστημα θεωρώντας όλα τα ενδεχόμενα ισοπίθανα.

- Η κατηγοριοποίηση αυτή συμβάλει στη δυνατότητα πρόβλεψης του αναπάντεχου.

- Υπολογίζεται το κόστος ενεργοποίησης αντιμέτρων του συστήματος.

- Υπολογίζεται το κόστος ανάκτησης των κατεστραμμένων δεδομένων από μια κρίση.

- Υπολογίζεται η αξία των κατεστραμμένων υλικών.

- Υπολογίζεται το κόστος συντήρησης ενός τέτοιου συστήματος.

- Με κατάλληλες μετατροπές, η γενική θεωρία διαχείρισης κρίσεων εφαρμόζεται και σε ομαλούς, δυναμικούς χώρους δεδομένων.

- Η γενική θεωρία Κινδύνου δεν υπολογίζει την Αστάθεια του χώρου δεδομένων. Η θεωρία των καταστροφών και γενικότερα τα χαοτικά μαθηματικά μοντέλα είναι κατάλληλα για τέτοιου είδους μετρήσεις.

- Δεν υπολογίζει τον κίνδυνο στηριζόμενη σε ποσοτικά χαρακτηριστικά που έχουν σχέση με το κόστος, το χρόνο, η την πιθανότητα εμφάνισης του κινδύνου.

- Η γενική θεωρία Κινδύνου είναι ανεξάρτητη από το περιβάλλον λειτουργίας του υπό μελέτη συστήματος.

- Η γενική θεωρία διαχείρισης Κινδύνου είναι ένα υποσύνολο της γενικής θεωρίας διαχείρισης Έργου μέσα από την οποία

προσεγγίζεται δομημένα η αναγνώριση, ανάλυση και αντιμετώπιση των κινδυνικών φαινομένων.

- Οι περισσότερες μεθοδολογίες διαχείρισης κινδύνου στηρίζονται στο πλαίσιο της ανάλυσης (Risk Analysis) – αναγνώρισης (Risk Identification) – επιλογής (Risk Resolution) και υλοποίησης (Risk Monitoring). Παραλλαγές στο πλαίσιο παρουσιάζονται σε πληθώρα άλλων μεθοδολογιών.

- Η γενική θεωρία διαχείρισης κινδύνου αλλάζει ριζικά το πλαίσιο ανάπτυξης. Το νέο προτεινόμενο πλαίσιο χωρίζεται σε δυο υποθεωρίες που περιλαμβάνουν την Πρόληψη Κινδυνικών φαινομένων και την Αντιμετώπιση Καταστροφικών φαινομένων αντίστοιχα.

- Κάθε υποθεωρία με τη σειρά της δημιουργεί ένα νέο πλαίσιο λειτουργίας.

- Η πρόληψη κινδυνικών φαινομένων περιέχει ένα διευρυμένο πλαίσιο ανάπτυξης, που περιλαμβάνει την Ανάλυση του συστήματος, τον Προσδιορισμό των Κινδυνικών φαινομένων, την Ποιοτική μελέτη των αδυναμιών του συστήματος και την ποσοτική ανάλυση του πιθανού κινδυνικού φαινομένου.

- Παρουσιάζεται η έννοια του μεταλλακτικού προγραμματισμού και της μετάλλαξης ενός ψηφιακού πληροφοριακού συστήματος.

- Η διαχείριση του κινδύνου παρουσιάζεται ως ένα ευρύτερο τμήμα της γενικής θεωρίας συστημάτων που είναι συγγενές με το υποσύνολο του μεταλλακτικού προγραμματισμού.

- Προσδιορίζονται και οριοθετούνται οι χώροι δεδομένων μέσα στους οποίους δρα ένα ψηφιακό πληροφοριακό σύστημα. Επιπλέον, παρουσιάζεται η έννοια του ασταθούς χώρου δεδομένων μέσα στον οποίο υλοποιείται το μοντέλο της μεθοδολογίας.

- Παρουσιάζονται νέα και ευέλικτα εργαλεία διαχείρισης κρίσεων, όπως ο Χάρτης Καταστροφών, ο Χάρτης Κινδύνου, ο Χάρτης Αντιμετώπισης και ο Πίνακας Κόστους Ενεργοποίησης Αντιμέτρων.

- Μέσα από την πρόληψη κινδυνικών φαινομένων, με τη βοήθεια ποιοτικών μετρήσεων, γίνεται χαρτογράφηση του συστήματος και των αδυναμιών του, ώστε να είναι εφικτή η πρόληψη κρίσεων.

- Εισάγονται οι έννοιες των κεντρικών σημείων αυτοκατάρρευσης (Μαύρων Προγραμματιστικών Οπών), των μεταλλακτικών σημείων (Γκρίζες Προγραμματιστικές Οπές) και των Λευκών Προγραμματιστικών Οπών.

- Παρουσιάζεται η έννοια της καταστροφής σαν πιθανό ενδεχόμενο σε αντίθεση με υπάρχουσες μεθοδολογίες, ενώ παράλληλα μελετώνται μέθοδοι ανάκτησης από πτώση.

- Μελετάται διεξοδικά το πλαίσιο ανάκτησης από πτώση στηριζόμενο τόσο στην ύπαρξη κατάλληλων αντιμέτρων όσο και στην ενεργοποίηση των καταλληλότερων με το μικρότερο κόστος λειτουργίας.

- Ειδικότερα το πλαίσιο λειτουργίας της υπομεθοδολογίας περιέχει το στάδιο της Ανάλυσης των κινδύνων, του Προσδιορισμού των αντίμετρων, της Κοστολόγησης ενεργοποίησης του αντιμέτρου, του Εύρεσης βέλτιστης λύσης ανάκτησης των δεδομένων, του υπολογισμού καταστροφών και απόδοσης αντιμέτρων Ανάκτησης.

- Ποσοτικοποιείται η καταστροφή, υπολογίζοντας το κόστος ανάκτησης ως αναπόσπαστο κομμάτι της υπομεθοδολογίας ανάκτησης καταστροφικών δεδομένων.

- Ενεργοποιείται το καταλληλότερο αντίμετρο με το ελάχιστο δυνατό κόστος ενεργοποίησης, που αποτελεί τη βέλτιστη λύση ανάκτησης δεδομένων.

- Παράλληλα, προσδιορίζεται τόσο ποιοτικά όσο και ποσοτικά η απόδοση της ανάκτησης με τη βοήθεια αντίστοιχων εργαλείων.

- Προσδιορίζεται τόσο ποιοτικά όσο και ποσοτικά το κόστος ανάκτησης των δεδομένων.

- Τα δεδομένα κατηγοριοποιούνται σε υλικά και ηλεκτρονικά καθένα από τα οποία κοστολογείται αντίστοιχα.

- Προσδιορίζεται το κόστος ανάκτησης των δεδομένων στην περίπτωση μερικής ανάκτησης.

- Υπολογίζεται η απόδοση του σχεδίου ανάκτησης στηριζόμενη τόσο σε ποιοτικά κριτήρια, όπως ο FSR, όσο και ποσοτικά κριτήρια με το λόγο του πραγματικού ποσού ανάκτησης προς το προϋπολογισθέν πόσο.

- Παρουσιάζεται το Ψηφιακό Πληροφοριακό Σύστημα Εθνικής Άμυνας, ο Ψηφιακός Στρατιώτης. Αναλύεται η δομή του, η λειτουργία του και το περιβάλλον λειτουργίας του.

- Αναλύονται οι κίνδυνοι που αντιμετωπίζει στο πεδίο της μάχης το ψηφιακό του σύστημα καθώς και παρουσιάζονται προοπτικές επέκτασης και ανάπτυξης του συστήματος.

- Προτείνονται λύσεις υλοποίησης σεναρίων κρίσεως και σε άλλους κλάδους των Ενόπλων Δυνάμεων.

- Καταστροφικά σενάρια υλοποιούνται και τα αποτελέσματα αξιολογούνται αντίστοιχα.

- Τα ψηφιακά πρωτόκολλα διαχείρισης κρίσεων είναι η λύση στην πρόληψη κρίσεων σε ασταθή περιβάλλοντα λειτουργίας.

- Η δομή τους είναι ανεξάρτητη από τα υπόλοιπα τμήματα της μεθοδολογίας Διοίκησης Κινδύνου.

- Τα υποσυστήματα του υπό μελέτη συστήματος και οι σχέσεις αλληλεξάρτησης είναι το κλειδί στην πρόληψη κινδυνικών, μη προβλέψιμων φαινομένων.

- Οι δρόμοι Hamilton είναι το βοήθημα που απαιτείται για να προσδιοριστούν οι σχέσεις αλληλεξάρτησης μεταξύ των υποσυστημάτων καθώς και πιθανά μεταλλακτικά ή κεντρικά σημεία αυτοκατάρρευσης.

7.3 ΔΗΜΟΣΙΕΥΣΕΙΣ ΠΟΥ ΥΠΟΣΤΗΡΙΖΟΥΝ ΤΗΝ ΔΙΑΤΡΙΒΗ

Σε αυτό το τμήμα παρατίθενται οι επιστημονικές εργασίες, οι οποίες εξασφαλίζουν την πρωτοτυπία της διδακτορικής διατριβής και αποτελούν το θεωρητικό υπόβαθρο για αυτήν. Επίσης, αποτελούν την απαραίτητη βιβλιογραφική αναφορά και το σημείο αναφοράς για την επιστημονικά ορθή συγγραφή της διατριβής. Οι πρωτότυπες αυτές επιστημονικές εργασίες δημοσιεύθηκαν σε πρακτικά διαφόρων συνεδρίων και σε επιστημονικά περιοδικά, όπως φαίνεται αναλυτικά:

- Καλλιγκάτσης Ι., Μαυρομμάτης Γ., Πετραντωνάκης Π., Παναγιωτόπουλος Ι.-Χ., ***Παραπλάνηση Χρηστών στην Προσπέλαση Πληροφορίας,*** *Πρακτικά 15^{ου} Πανελλήνιου Συνεδρίου Ελληνικής Εταιρίας Επιχειρησιακών Ερευνών, 2002, Τρίπολη.*

- Μαυρομμάτης Γ., Καλλιγκάτσης Ι., Πετραντωνάκης Π., Παναγιωτόπουλος Ι.-Χ., ***Συλλογή Μέγιστης Πληροφόρησης από Πηγές Πληροφορίας,*** *Πρακτικά 15^{ου} Πανελλήνιου Συνεδρίου Ελληνικής Εταιρίας Επιχειρησιακών Ερευνών, 2002, Τρίπολη.*

- Πετραντωνάκης Π., Καλλιγκάτσης Ι., Μαυρομμάτης Γ., Παναγιωτόπουλος Ι.-Χ., ***Θεωρήσεις και Προβληματισμοί σε Συστήματα Ανάκτησης από Πτώσεις,*** *Πρακτικά 15^{ου} Πανελλήνιου Συνεδρίου Ελληνικής Εταιρίας Επιχειρησιακών Ερευνών, 2002, Τρίπολη.*

- Πετραντωνάκης Π., Παναγιωτόπουλος Ι.-Χ., ***ΔΙΟΙΚΗΣΗ ΚΙΝΔΥΝΟΥ ΣΕ ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ ΕΘΝΙΚΗΣ ΑΜΥΝΑΣ,*** *Πρακτικά 16^{ου} Πανελλήνιου Συνεδρίου Ελληνικής Εταιρίας Επιχειρησιακών Ερευνών, 2003, Λάρισα.*

- Petrantonakis P., Panayiotopoulos J.-C., ***BLACK OPERATING HOLE AND LAND WARRIOR INFORMATION SYSTEMS,*** *20th European Conference on Operational Research, 2004, Rhodes.*

- Petrantonakis P., Panayiotopoulos J.-C., ***Diagnosis in Distributed Medical System Basis****, 2th National Conference on Medical Technologies, University of Patras, June 2004.*

- Πετραντωνάκης Π., Παναγιωτόπουλος Ι.-Χ., ***Αντιμετώπιση Κινδύνου με ελάχιστο κόστος σε Εχθρικά Περιβάλλοντα Λειτουργίας,*** *Πρακτικά 1ου Συνεδρίου Εταιρείας Συστημικών Μελετών,2005, Τρίπολη.*

- Petrantonakis P., Panayiotopoulos J.-C., ***Using Assignment Model as an automated Recovery System****, Disaster Prevention and Management. An International Journal, 2005, Volume 14, Number 1, Page 89-96*

- Πετραντωνάκης Π., Παναγιωτόπουλος Ι.-Χ., ***Ψηφιακά συστήματα αυτόματης αποκλιμάκωσης κρίσεων,*** *Γεωστρατηγική, τεύχος 6, τετραμηνιαία επιθεώρησης Σεπτέμβριος – Δεκέμβριος 2004, σελ. 179-188.*

- Petrantonakis P., Panayiotopoulos J.-C., ***Minimizing Recovery Cost in Military Information Systems: the Land Warrior Case.*** *Defensor Pacis, issue 17, July 2005, pp 189-200.*

- Petrantonakis P., Panayiotopoulos J.-C., ***Some Considerations on Crisis Management***, *Defensor Pacis, issue 18, July 2006, pp 7-14.*

- Petrantonakis P.M., Panayiotopoulos J.-C., ***CRISIS MANAGEMENT SCENARIOS IN IRREGULAR DYNAMIC DATA SPACES,*** *European Journal of Operational Research, under Evaluation.*

- Πετραντωνάκης Π, Παναγιωτόπουλος Ι.-Χ., ***Προοπτικές ανάπτυξης πυρηνικών εργοστασίων για την παραγωγή ηλεκτρικής ενέργειας στην Ελλάδα***. *Ινστιτούτο Αμυντικών Αναλύσεων, 2005.*

7.4 ΜΕΛΛΟΝΤΙΚΗ ΕΠΕΚΤΑΣΗ

Η έρευνα αποτελεί αναπόσπαστο κομμάτι της προτεινόμενης μεθοδολογίας. Φυσικά, η έρευνα δεν τελειώνει με την παρουσίαση της παραπάνω διατριβής, αφού παρουσιάσθηκε ένα νέο επιστημονικό πεδίο με ποικίλα θέματα και πλούσιο περιεχόμενο μελέτης. Ειδικότερα, σαν αντικείμενο μελλοντικής έρευνας αποτελεί:

- Η ανάπτυξη και βελτίωση του μοντέλου διαχείρισης κρίσεων με την παρουσίαση και ενσωμάτωση μεταλλακτικών κλάσεων αποκλιμάκωσης κρίσεων.

- Η δημιουργία ψηφιακών πρωτοκόλλων διαχείρισης κρίσεων και η επέκταση τους που θα στηρίζονται σε αρχές και δομές των Ψηφιακών Ιδιαίτερων Μαθημάτων και των μεθόδων κατασκευής εγχειριδίων.

- Η ανάπτυξη και βελτιστοποίηση της μεθόδου υλοποίησης σεναρίων κρίσεως με την προσθήκη έμπειρου συστήματος αποκλιμάκωσης κρίσεων.

- Η Βελτιστοποίηση, σύμφωνα με τα παραπάνω, του λογισμικού διαχείρισης κρίσεων, όπως αυτό παρουσιάσθηκε στην παρούσα διατριβή.

ΚΕΦΑΛΑΙΟ 8

ΚΩΔΙΚΑΣ ΕΦΑΡΜΟΓΗΣ ΔΙΑΧΕΙΡΙΣΗΣ ΚΡΙΣΕΩΝ ΕΦΑΡΜΟΓΗ ΔΙΑΧΕΙΡΙΣΗΣ ΚΡΙΣΕΩΝ ΣΕ ΨΗΦΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

8. ΚΩΔΙΚΑΣ ΕΦΑΡΜΟΓΗΣ ΕΓΧΕΙΡΙΔΙΟ ΧΡΗΣΗΣ

« Άνθρωπος είναι ένα ον που κατασκευάζει και χρησιμοποιεί εργαλεία»

Δαρβίνος

8.1 ΕΓΧΕΙΡΙΔΙΟ ΠΡΟΓΡΑΜΜΑΤΟΣ

Στη παρούσα εργασία έγινε η ανάπτυξη ενός προγράμματος υπολογιστή που έχει σαν σκοπό την υποστήριξη των διαδικασιών αντιμετώπισης κρίσεων και καταστροφών που εμφανίζονται σε ένα σύστημα κατά τη διάρκεια της λειτουργίας του. Η εφαρμογή Διαχείρισης Κίνδυνου σε Πληροφοριακά Συστήματα Εθνικής Άμυνας υλοποιεί με τον καλύτερο δυνατό τρόπο τη μεθοδολογία που αναλύθηκε σε προηγούμενα κεφάλαια.

Το πρόγραμμα δίνει τη δυνατότητα στο χρήστη να αναπαραστήσει το υπό μελέτη σύστημα λαμβάνοντας υπόψη την ιεραρχία και τη δομή των υποσυστημάτων του. Αρχικά ο χρήστης καλείται μέσα από την κεντρική φόρμα του προγράμματος να οριοθετήσει τη δομή του υπό μελέτη πληροφοριακού συστήματος. Στην κεντρική φόρμα του προγράμματος (Εικόνα 1) φαίνεται καθαρά ο σχεδιασμός και η ανάλυση του υπό μελέτη συστήματος.

Σε κάθε Υποσύστημα μπορεί ο χρήστης να αντιστοιχίσει ορισμένες απειλές τις οποίες αντιμετωπίζει το σύστημα ή ενδέχεται να αντιμετωπίσει στο μέλλον. Με βάση αυτές τις αντιστοιχίες το πρόγραμμα δημιουργεί τον **πίνακα αντιμετώπισης (Confrontation Map)**. Στη συνέχεια εξετάζει όλους τους δυνατούς συνδυασμούς και εντοπίζει εκείνα τα μέρη του συστήματος τα οποία βρίσκονται σε απειλή, χωρίς να υπάρχουν μέτρα αντιμετώπισης.

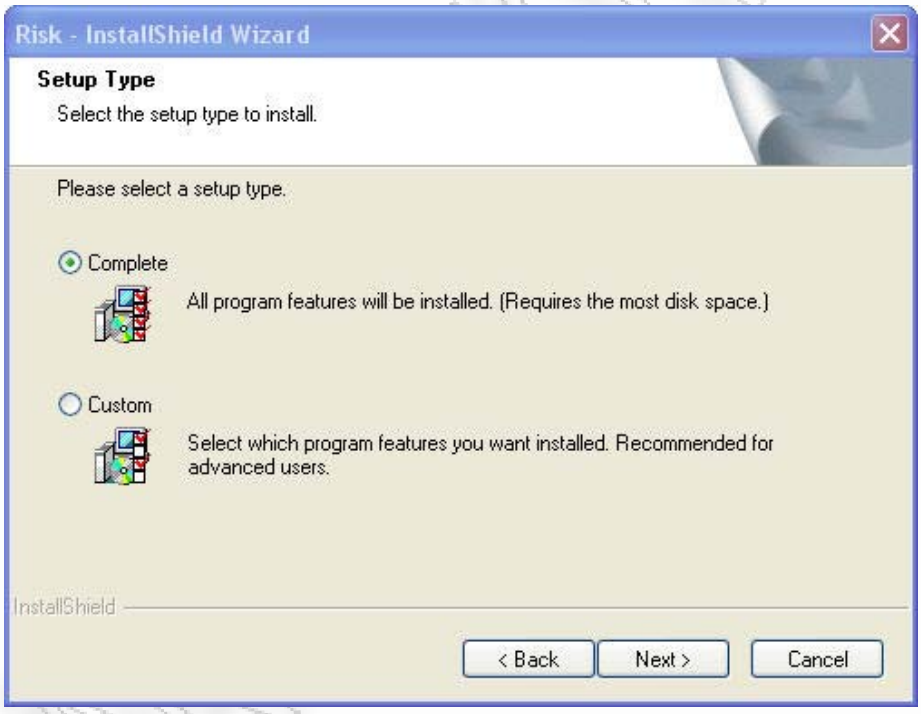
Μία άλλη λειτουργία που επιτελεί η εφαρμογή είναι η διαχείριση των δυνατών **Πλάνων Αποκατάστασης (Recovery Plans)** που μπορούν να εφαρμοστούν εφόσον συμβεί μία καταστροφή ώστε να μπορέσει να επανέλθει το σύστημα στην αρχική του κατάσταση. Κάθε πλάνο έχει κάποιο κόστος και το πρόβλημα που υπάρχει είναι να εντοπιστεί εκείνος ο συνδυασμός ενεργειών που αντιμετωπίζουν του κινδύνους με το μικρότερο κόστος ενεργοποίησης.

8.1.2 ΕΓΚΑΤΑΣΤΑΣΗ - ΑΠΑΙΤΗΣΕΙΣ

Η εφαρμογή έχει τις εξής απαιτήσεις:

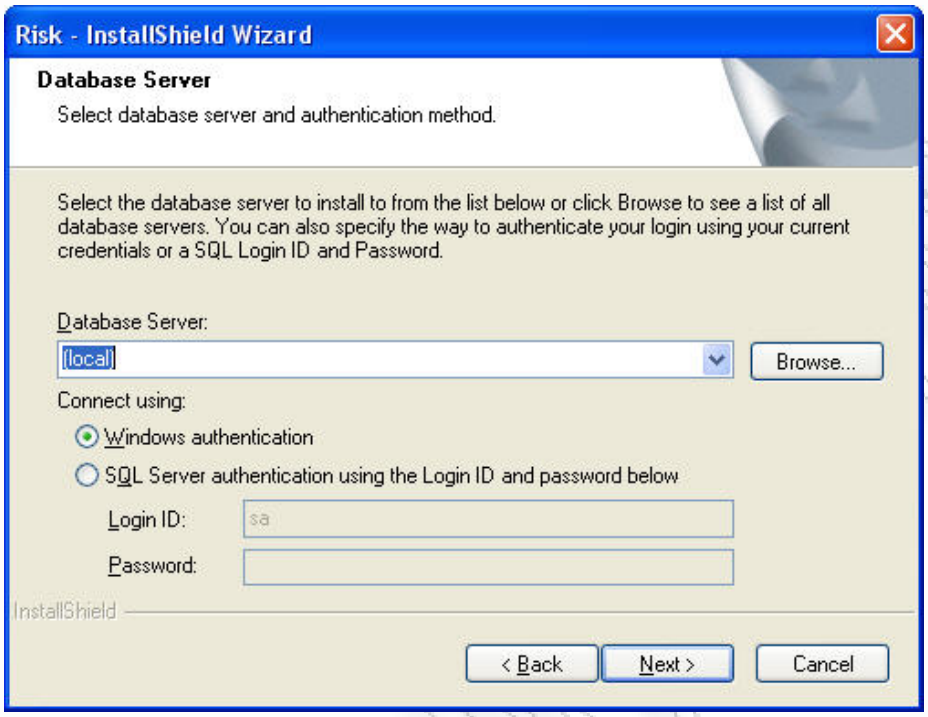
- **Windows XP,2000,2003**
- **.Net Framework 2.0**
- **SQL Server 2000**

Εκτελούμε το αρχείο setup.exe από το CD εγκατάστασης. Στην συνέχεια, επιλέγουμε τον τύπο της εγκατάστασης Πλήρη (Complete) ή Προσαρμοσμένη (Custom). Με τη δεύτερη επιλογή μπορούμε να επιλέξουμε τη διαδρομή στο δίσκο που θα γίνει η εγκατάσταση καθώς και αν θα γίνει η όχι η δημιουργία της βάσης στον SQL SERVER (σε αυτή την περίπτωση πρέπει να δημιουργήσουμε manually την βάση).



Εικόνα 8.1
Εγκατάσταση Εφαρμογής

Πατώντας στην εγκατάσταση θα εμφανιστεί η οθόνη ρυθμίσεων της βάσης δεδομένων που απαιτείται για να λειτουργήσει η εφαρμογή.



Εικόνα 8.2
Συνέχεια εγκατάστασης

Στο πεδίο Database Server εισάγουμε το όνομα του υπολογιστή στον οποίο βρίσκεται εγκατεστημένος ο SQL SERVER (local αν είναι εγκατεστημένος στον ίδιο υπολογιστή με αυτόν στον οποίο εγκαθίσταται και η εφαρμογή).

8.1.3 ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΣΥΣΤΗΜΑΤΟΣ

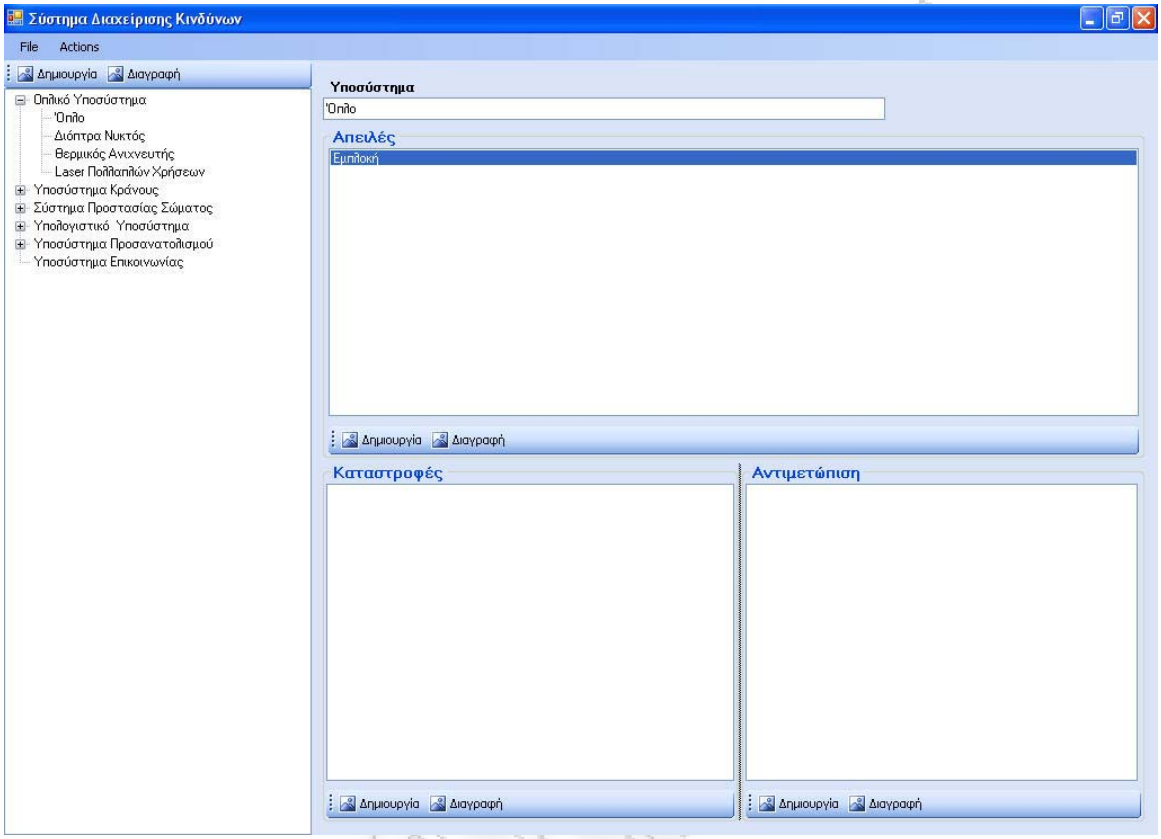
Η εφαρμογή έχει δομηθεί ακολουθώντας το multi-layer μοντέλο έτσι ώστε να είναι δυνατή η ευκολότερη τροποποίηση και περαιτέρω επέκτασή του .Έτσι υπάρχουν τρία επίπεδα οργάνωσης του κώδικα:

1. **Επίπεδο Πρόσβασης Βάσης Δεδομένων**: Σε αυτό το επίπεδο υπάρχουν οι κλάσεις οι οποίες έχουν σαν στόχο την αλληλεπίδραση με τη βάση δεδομένων που υποστηρίζει το πρόγραμμα.

2. **Επίπεδο Μοντελοποίησης Προβλήματος**: Σε αυτό το επίπεδο μοντελοποιούνται οι έννοιες που συναντιούνται στο πρόβλημα όπως στην προκειμένη περίπτωση είναι η απειλή, τα υποσυστήματα και οι καταστροφές.

3. **Επίπεδο Παρουσίασης(GUI)**: Σε αυτό το επίπεδο υπάρχει ο κώδικας που διαχειρίζεται την αλληλεπίδραση του χρήστη με την εφαρμογή.

8.1.4 ΠΕΡΙΓΡΑΦΗ USER INTERFACE



Εικόνα 8.3
Κεντρική Φόρμα Προγράμματος.

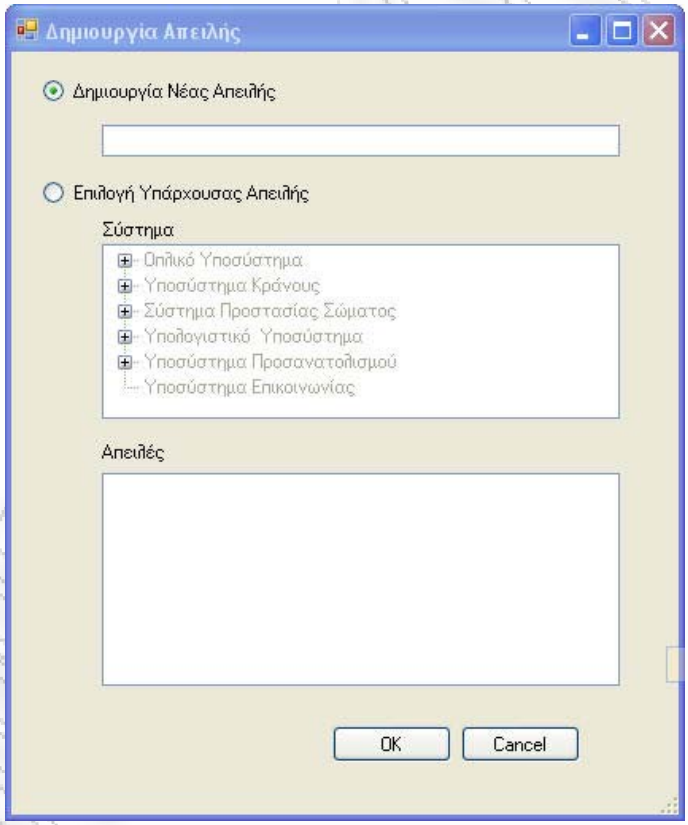
Στο αριστερό μέρος φαίνεται σε δενδροειδή μορφή το σύστημα το οποίο εξετάζεται. Ο χρήστης μπορεί να προσθέτει και να αφαιρεί συστήματα και υποσυστήματα με τα δυο κουμπιά **προσθήκη** και **διαγραφή** στο πάνω μέρος. Επιλέγοντας ένα υποσύστημα στο δεξιό μέρος του παραθύρου εμφανίζονται οι απειλές που αντιμετωπίζει.

Επιλέγοντας μια απειλή στη περιοχή Καταστροφές του παραθύρου κάτω αριστερά εμφανίζονται οι καταστροφές που μπορούν να συμβούν στο υποσύστημα που έχει επιλέγει από την περιοχή υποσυστημάτων, εάν πραγματοποιηθεί η συγκεκριμένη απειλή.

Στη συνέχεια αν επιλέξουμε μια καταστροφή τότε στη περιοχή Αντιμετώπιση κάτω δεξιά εμφανίζονται οι ενέργειες που μπορούν να γίνουν ώστε να αντιμετωπιστεί η συγκεκριμένη καταστροφή.

8.1.5 ΠΡΟΣΘΗΚΗ ΑΠΕΙΛΗΣ

- Επιλέγουμε από το δέντρο του συστήματος ένα υποσύστημα..
- Στην περιοχή Απειλές πατάμε το κουμπί Δημιουργία οπότε εμφανίζεται ένα νέο παράθυρο στο οποίο μας δίνονται δυο επιλογές : είτε να δημιουργήσουμε μια καινούργια απειλή, είτε να προσθέσουμε στη λίστα με τις απειλές του υποσυστήματος μια απειλή που υπάρχει και σε κάποιο άλλο υποσύστημα. Στη δεύτερη περίπτωση εμφανίζεται μια δενδροειδή λίστα με τα υποσυστήματα. Επιλέγουμε ένα υποσύστημα, οπότε στη λίστα που βρίσκεται από κάτω φαίνονται οι απειλές του συγκεκριμένου υποσυστήματος. Επιλέγουμε μια και πατάμε **OK** οπότε η απειλή αυτή προστίθεται στη λίστα των απειλών του άλλου υποσυστήματος που είχαμε επιλέξει αρχικά στην κεντρική φόρμα.

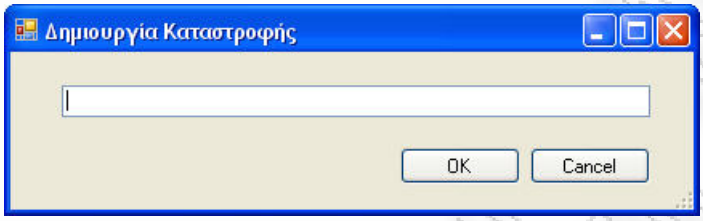


Εικόνα 8.4
Προσθήκη Απειλής

8.1.6 ΠΡΟΣΘΗΚΗ ΚΙΝΔΥΝΟΥ

- Επιλέγουμε ένα Υποσύστημα από το δένδρο του συστήματος.
- Επιλέγουμε μία απειλή.

- Πατάμε το κουμπί Δημιουργία στη περιοχή Κίνδυνοι οπότε εμφανίζεται ένα παράθυρο στο οποίο προσθέτουμε ένα κίνδυνο.
- Πατάμε OK οπότε ο κίνδυνος έχει προστεθεί στη λίστα με τους κινδύνους του υποσυστήματος για μια απειλή.



Εικόνα 8.5
Προσθήκη Καταστροφής

8.1.7 ΠΡΟΣΘΗΚΗ ΑΝΤΙΜΕΤΩΠΙΣΗΣ

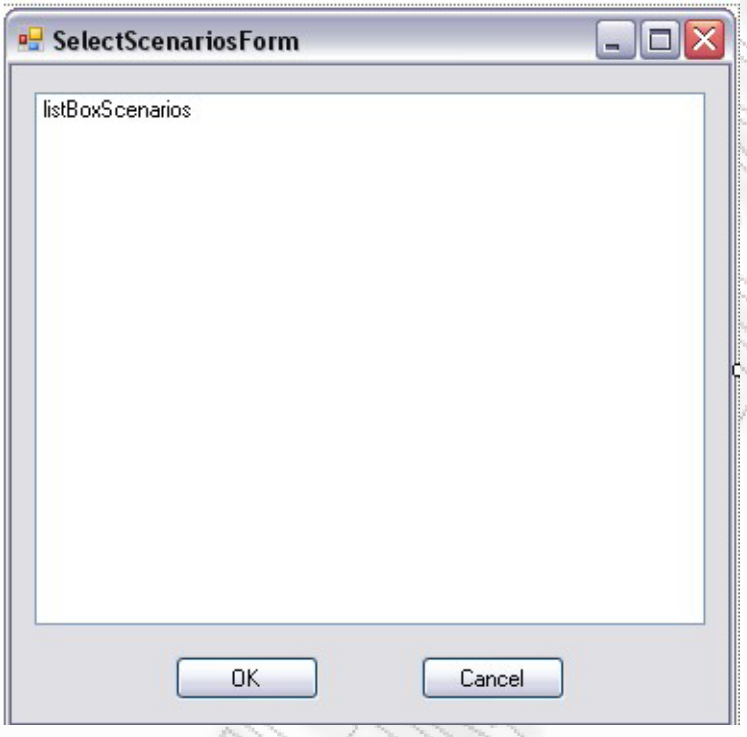
- Επιλέγουμε ένα Υποσύστημα από το δένδρο του συστήματος.
- Επιλέγουμε μία απειλή.
- Επιλέγουμε ένα κίνδυνο.
- Πατάμε το κουμπί Δημιουργία στη περιοχή Αντιμετωπίσεις οπότε εμφανίζεται ένα παράθυρο στο οποίο προσθέτουμε την ονομασία και το κόστος της συγκεκριμένης Αντιμετώπισης.
- Πατάμε OK οπότε ο κίνδυνος έχει προστεθεί στη λίστα με τους κινδύνους του υποσυστήματος για μια απειλή.

Εικόνα 8.6
Προσθήκη Αντιμετώπισης

8.1.8 ΔΗΜΙΟΥΡΓΙΑ ΣΕΝΑΡΙΟΥ

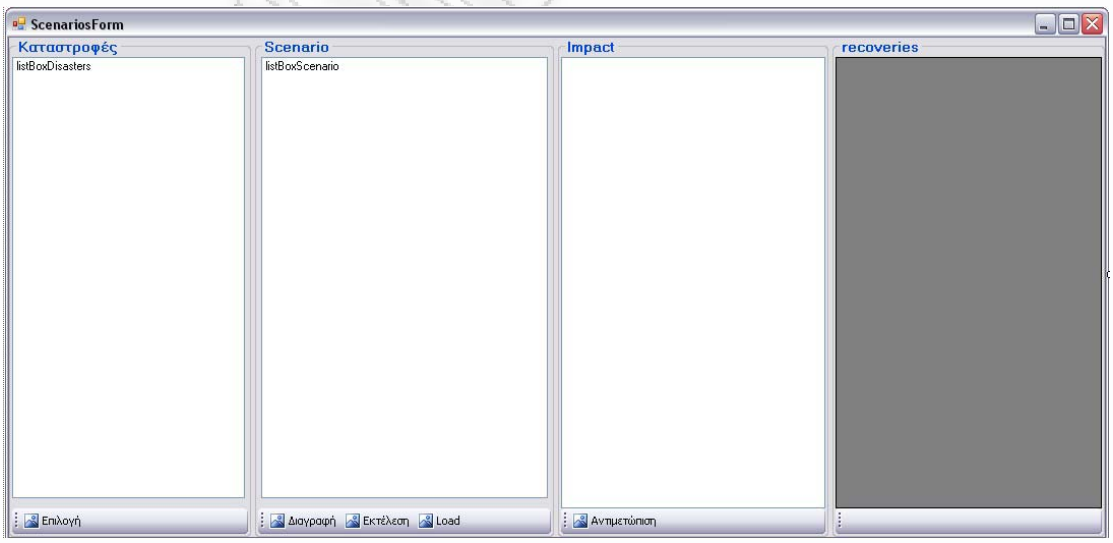
Από την κεντρική φόρμα επιλέγετε στο menu την επιλογή **Action.** Στη συνέχεια, επιλέγοντας το υπομενu **Δημιουργία Σεναρίου** ανοίγουμε τη φόρμα στην οποία ο χρήστης έχει την δυνατότητα να διαλέξει το σενάριο που επιθυμεί να τρέξει προκείμενου να μελετήσει τα

αποτελέσματα του. Παρακάτω παρουσιάζεται η φόρμα επιλογής σεναρίων (Εικόνα 8.7).



Εικόνα 8.7
Επιλογή Σεναρίων

Μόλις επιλέγει το κατάλληλο σενάριο τότε ανοίγει η φόρμα του σεναρίου με φορτωμένα τα χαρακτηριστικά δεδομένα του.



Εικόνα 8.8
Εκτέλεση Σεναρίων

8.2 ΚΩΔΙΚΑΣ ΕΦΑΡΜΟΓΗΣ

Παρακάτω παρατίθεται ο κώδικας της εφαρμογής.

8.2.1 TextInputForm.cs

Φόρμα στην οποία ο χρήστης δίνει δεδομένα για χρήση στην εφαρμογή.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace RiskManagement{
    public partial class TextInputForm : Form
    {
        public TextInputForm()
        {
            InitializeComponent();
        }

        public string InputText
        {
            get
            {
                return textBox1.Text;

      } } } }
```

8.2.2 StatistcstForm.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using Histograma;
using RiskManagement.BusinessLayer;

namespace RiskManagement
{
    public partial class StatistcsForm : Form
    {

        public StatistcsForm()
        {
            InitializeComponent();
        }
```

```
        public struct Value
        {
            public string name;
            public long vi;}

        private void StatistcsForm_Load(object sender, EventArgs e)
        {
            /*
            Value[] values = new Value[11];
            values[0].vi = 1;
            values[0].name = "test1";
            values[1].vi = 3;
            values[1].name = "test2";
            values[2].vi = 10;
            values[2].name = "test3";
            values[3].vi = 4;
            values[3].name = "test3";
            values[4].vi = 10;
            values[4].name = "test3";
            values[5].vi = 4;
            values[5].name = "test3";
            values[6].vi = 6;
            values[6].name = "test3";
            values[7].vi = 3;
            values[7].name = "test3";
            values[8].vi = 2;
            values[8].name = "test3";
            values[9].vi = 0;
            values[9].name = "test3";
            values[10].vi = 4;
            values[10].name = "test3";
            */
            int[] vis = Scenarios.GetViArray();
            // Φόρμα εμφάνισης στατιστικών δεδομένων
            long[] myValues = GetHistogram(vis);
            Histogram.DrawHistogram(myValues);
        }

        public long[] GetHistogram(int[] value)
        {
            long[] myHistogram = new long[value.Length];

            int ComponentsCount =
RiskManagement.BusinessLayer.Application.System.Components.Count;
            for (int i = 0; i < value.Length; i++)
            {
                myHistogram[i] = (long)((float)value[i] /
ComponentsCount *100);
            }

            return myHistogram;
        } }}
```

8.2.3 SqlDBManager.cs

```
using System;
using System.Data;
using System.Data.Odbc;
using System.Data.SqlClient;
using System.Data.OleDb;
```

```
using System.Data.OracleClient;

namespace DataAccessLayer
{

  public sealed class SqlDBManager: DBManager,IDisposable
  {
    private IDbConnection idbConnection;
    private IDataReader idataReader;
    private IDbCommand idbCommand;
    private DataProvider providerType;
    private IDbTransaction idbTransaction =null;
    private IDbDataParameter[]idbParameters =null;
    private string strConnection;

    private static SqlDBManager sqlMgr = null;

    public static SqlDBManager GetInstance( )
    {
        if (sqlMgr == null){
            SqlDBManager.sqlMgr = new SqlDBManager();
        }

    }
// Σύνδεση με SQL SERVER και με τη βάση δεδομένων
idbConnection = new SqlConnection("Server=localhost;" +
                                  "Database=ADONET;" +
                                  "User ID=sa;" +
                                  "password=trustno1");

    public IDbConnection Connection
    {
      get
      {
        return idbConnection;
      }
    }

    public IDataReader DataReader
    {
      get
      {
        return idataReader;
      }
      set
      {
        idataReader = value;
      }
    }

    public DataProvider ProviderType
    {
      get

      {
        return providerType;
      }
      set
      {
        providerType = value;
      }
```

```
    }

    public string ConnectionString
    {

 get
      {
        return strConnection;
      }
      set
      {
        strConnection = value;
      }
    }

    public IDbCommand Command
    {
      get
      {
        return idbCommand;
      }
    }

    public IDbTransaction Transaction
    {
      get
      {
        return idbTransaction;
      }
    }

    public IDbDataParameter[]Parameters
    {
      get
      {
        return idbParameters;
      }
    }

      public IDbConnection createConnection()
      {
          IDbConnection conn =  new SqlConnection("Server=localhost;"
+
                                "Database=ADONET;" +
                                "User ID=sa;password=trustno1");
          return conn;      }  }}
```

8.2.4 SelectScenariosForm.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using RiskManagement.BusinessLayer;

namespace RiskManagement
{
```

```
    public partial class SelectScenariosForm : Form
    {
        public SelectScenariosForm()
        {

            InitializeComponent();
        }

        public Scenario SelectedScenario
        {

            get { return _selectedScenario; }
        }
//Φόρτωμα του σεναρίου που έχει επιλεχθεί
        private void SelectScenariosForm_Load(object sender,
EventArgs e)
        {
            Scenarios scenarios =
BusinessLayer.Application.System.Scenarios;
            scenarios.SelectAll();
            listBoxScenarios.DataSource = scenarios.DataProvider;
            listBoxScenarios.ValueMember = "ID";
            listBoxScenarios.DisplayMember = "ID";
        }

        private void listBoxScenarios_SelectedIndexChanged(object
sender, EventArgs e)
        {

        }

        private void button1_Click(object sender, EventArgs e)
        {
            _selectedScenario =
(Scenario)listBoxScenarios.SelectedItem;
            this.Close();
        }

        private Scenario _selectedScenario;
    }}
```

8.2.5 ScenariosForm2.cs

```
using System;
using System.IO;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using RiskManagement.BusinessLayer;
namespace RiskManagement

{
    public partial class ScenariosForm2 : Form
    {
        public ScenariosForm2()
        {
            InitializeComponent();
```

```
        }

        public static Scenario _scenario;

//Προσθήκη Καταστροφών στο επιλεγέν σενάριο
        private void AddDisasterButton_Click(object sender, EventArgs
e)
        {
            ScenariosForm_SelectDisaster form = new
ScenariosForm_SelectDisaster();
            if (form.ShowDialog() == DialogResult.OK)
            {
                Disaster disaster = form.SelectedDisaster;
            if (listBoxDisasters.Items.Contains(disaster) == false)
                {
                    if (_scenario == null)
                    {
                        _scenario = Scenario.Create();
                    }
                    _scenario.AddDisaster(disaster);
     listBoxDisasters.DataSource = _scenario.Disasters.DataProvider;
                    listBoxDisasters.ValueMember = "ID";
                    listBoxDisasters.DisplayMember = "Name";
          }
            }
        }

        private void openScenarioMenuItem_Click(object sender,
EventArgs e)
        {
            ScenariosForm_SelectScenario form = new
ScenariosForm_SelectScenario();
            if (form.ShowDialog() == DialogResult.OK)
            {
                _scenario = form.SelectedScenario;
                listBoxDisasters.DataSource =
_scenario.Disasters.DataProvider;
                listBoxDisasters.ValueMember = "ID";
                listBoxDisasters.DisplayMember = "Name";
                Text = _scenario.Name;
            }
        }

        private void ScenrtioExecuteMenuItem_Click(object sender,
EventArgs e)
        {
            if (listBoxDisasters.Items.Count > 0)
            {
                int[] disaters = new
int[listBoxDisasters.Items.Count];

                for (int i = 0 ; i< listBoxDisasters.Items.Count;i++)
                {

                    int disasterID =
((Disaster)listBoxDisasters.Items[i]).ID;
                    disaters[i] =
RiskUtils.ComponentsThreatsTable.LocateDisaster(disasterID);
                }
                RiskUtils.Scenario scenario = new
RiskUtils.Scenario(disaters);
```

```
//Υπολογισμός συστημάτων που έχουν επηρρεαστεί απο την καταστροφή
  int[] impactVector = scenario.Execute();
int[] impactVectorFinal = new int[impactVector.Length];

foreach (int compId in
RiskUtils.ComponentsThreatsTable.Components.Keys)
                {
  //αν το διάνυσμα καταστροφών είναι μοναδιαίο και το υποσύστημα
αναγκαίο κοκκινίζει το τμήμα που καταρρέει

int m = RiskUtils.ComponentsThreatsTable.LocateComponent(compId);
                    if (impactVector[m] == 1)
                    {
                        //impactVectorFinal[i] = 1;
BusinessLayer.Component component =
Presentation_Layer.MainForm.Components.GetItem(compId);
                        if (component.IsCritical)
                        {

       TreeNode node = myTreeView1.GetObjectNode(component);
                            node.ForeColor = Color.Red;
                BusinessLayer.Component parent = component.Parent;

                            while (parent != null)
                            {
                                if (component.IsCritical)
                                {
       int m2 =
RiskUtils.ComponentsThreatsTable.LocateComponent(parent.ID);
                                impactVectorFinal[m2] = 1;
                      node = myTreeView1.GetObjectNode(parent);
                                    node.ForeColor = Color.Red;
                                    parent = parent.Parent;
                                }
                                else
                                    break;
                            }

                        }
                    }
                }
                // impactVector[0]++;
                int count = 0;
//Δημιουργία διανύσματος Σεναρίων και υπολογισμός δείκτη τρωτότητας

  for (int j = 0; j < impactVectorFinal.Length; j++)
                    if (impactVectorFinal[j] == 1)
                        count++;
                _scenario.vi = count;
            }
        }

        private void ScenariosForm2_Load(object sender, EventArgs e)
        {
            myTreeView1.RootNode = new TreeNode();
myTreeView1.RootNode.Text = BusinessLayer.Application.System.Name;
myTreeView1.PopulateTreeView(Presentation_Layer.MainForm.Components.D
ataProvider);
        }
```

```
        private void SaveScenarioMenuItem_Click(object sender,
EventArgs e)
        {
//Αποθήκευση εκτελεσθέντος σεναρίου
            TextInputForm form = new TextInputForm();
            if (form.ShowDialog() == DialogResult.OK)
            {
                Disaster[] disaters = new
Disaster[listBoxDisasters.Items.Count];
                int i = 0;
                foreach (object item in listBoxDisasters.Items)
                    disaters[i++] = (Disaster)item;
                _scenario.Name = form.InputText;
                Scenarios.SaveSceanario(disaters, _scenario);
            }
        }

        private void RecoveriesMenuItem_Click(object sender,
EventArgs e)
        {
         int[] disastersIds = new int[listBoxDisasters.Items.Count];
            for (int a = 0; a < listBoxDisasters.Items.Count; a++)
            {
       disastersIds[a] = ((Disaster)(listBoxDisasters.Items[a])).ID;

            }
//Επιλογή του κατάλληλου αντιμέτρου με τη βοήθεια του αλγόριθμου του
Προβλήματος Κατανομής
int[][] prob =
RiskUtils.DisastersRecoveriesTable.SelectDisasters(disastersIds);
Combinatorics.LinearAssignmentProblem problem = new
Combinatorics.LinearAssignmentProblem(prob);
      System.SByte[][] solution = problem.munkres();

            //Pass the filepath and filename to the StreamWriter
Constructor
            StreamWriter sw = new StreamWriter("C:\\solution.txt");

            for (int k = 0; k < solution.Length; k++)
            {
                for (int l = 0; l < solution.Length; l++)
                {
                    sw.Write(solution[k][l].ToString() + " ");

                }
                sw.WriteLine();
            }
            //Close the file
            sw.Close();
            Recoveries recoveries = new Recoveries();
            int i, j, numRows, numCols;
            numRows = solution.Length;
            numCols = solution[0].Length;
            for (i = 0; i < RiskUtils.DisastersRecoveriesTable.n;
i++)
            {
                for (j = 0; j < numCols; j++)
                {
                    if (solution[i][j] == 1)
                    {
```

```
int recID = RiskUtils.DisastersRecoveriesTable.GetRecoveryIdDB(i *
numCols + j);
                        if (recID > 0)
                        {
                    Recovery recovery = Recoveries.FindById(recID);
                        recoveries.DataProvider.Add(recovery);
                        break;
                        }
                     }
                }
            }
//εμφάνιση των κατάλληλων επιλεχθέντων recoveries απο τη λίστα του
αλγορίθμου
dataGridViewRecoveries.DataSource = recoveries.DataProvider;
            dataGridViewRecoveries.Columns.Remove("ID");
            dataGridViewRecoveries.Columns.Remove("DataItem");
        }

        private void DeleteDisasterMenuItem_Click(object sender,
EventArgs e)
        {

        }
//Διαγραφή επιλεχθείσας καταστροφής απο την λίστα.
        private void DeleteDisasterButton_Click(object sender,
EventArgs e)
        {
       if (listBoxDisasters.SelectedItem != null)
            {
 Disaster disaster = (Disaster)listBoxDisasters.SelectedItem;
                _scenario.DeleteDisaster(disaster);
 listBoxDisasters.DataSource = _scenario.Disasters.DataProvider;
               listBoxDisasters.ValueMember = "ID";
               listBoxDisasters.DisplayMember = "Name";
            } } }}
```

8.2.6 ScenariosForm_SelectScenario.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using RiskManagement.BusinessLayer;
namespace RiskManagement

{
    public partial class ScenariosForm_SelectScenario : Form
    {
        public ScenariosForm_SelectScenario()
        {
            InitializeComponent();
        }

        private void ScenariosForm_SelectScenario_Load(object sender,
EventArgs e){
```

```
     scenariosListBox.DataSource =
BusinessLayer.Application.System.Scenarios.DataProvider;
            scenariosListBox.ValueMember = "ID";
            scenariosListBox.DisplayMember = "Name";
        }

        public Scenario SelectedScenario
        {
            get
            {
                Scenario scenario = null;
                if (scenariosListBox.SelectedItem != null)
                {
                    scenario =
(Scenario)scenariosListBox.SelectedItem;
                }
                return scenario;
            } } }}
```

8.2.7 ScenariosForm_SelectDisaster.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using RiskManagement.BusinessLayer;

namespace RiskManagement
{
    public partial class ScenariosForm_SelectDisaster : Form
    {
        public ScenariosForm_SelectDisaster()
        {
            InitializeComponent();
        }

        private void ScenariosForm_SelectDisaster_Load(object sender,
EventArgs e)
        {
Επιλογή σεναρίου και κατάλληλης καταστροφής για το σενάριο αυτό
BusinessLayer.System system = new
RiskManagement.BusinessLayer.System();
            system.LoadSystem(1);
            Disasters disasters = system.Disasters;
            disastersListBox.DataSource = disasters.DataProvider;
            disastersListBox.ValueMember = "ID";
            disastersListBox.DisplayMember = "Name";
        }

        public Disaster SelectedDisaster
        {
            get
            {
                Disaster disaster = null;
                if (disastersListBox.SelectedItem != null)
                {
                 disaster = (Disaster)disastersListBox.SelectedItem;
```

```
                }
                return disaster;
            } } }}
```

8.2.8 ScenariosForm.cs

```
using System;
using System.IO;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using RiskManagement.BusinessLayer;
using RiskManagement.DataAccessLayer;
using RiskManagement.PresentationLayer;

namespace RiskManagement
{
    public partial class ScenariosForm : Form
    {
        public ScenariosForm()
        {
            InitializeComponent();
        }

        private Scenario _loadedScenario;

        private void ScenariosForm_Load(object sender, EventArgs e)
        {

myTreeView1.PopulateTreeView(Presentation_Layer.MainForm.Components.D
ataProvider);

            Disasters disasters = new Disasters();
            disasters.SelectAll();
            listBoxDisasters.DataSource = disasters.DataProvider;
            listBoxDisasters.ValueMember = "ID";
            listBoxDisasters.DisplayMember = "Name";
        }
     private void toolStripButton3_Click(object sender, EventArgs e)
        {
            if (listBoxDisasters.SelectedItem != null)
            {

                Disaster disaster =
(Disaster)listBoxDisasters.SelectedItem;
                if(listBoxScenario.Items.Contains(disaster)==false)
                    listBoxScenario.Items.Add(disaster);
            }
        }
        private void toolStripButton6_Click(object sender, EventArgs
e)
        {
            if (listBoxScenario.SelectedItem != null)
            {
      Disaster disaster = (Disaster)listBoxScenario.SelectedItem;
```

```
                listBoxScenario.Items.Remove(disaster);
            }
        }

      private void toolStripButton1_Click(object sender, EventArgs e)
        {
            if (listBoxScenario.Items.Count > 0)
            {
                int[] disaters = new
int[listBoxScenario.Items.Count];
                int i = 0;
                foreach (object item in listBoxScenario.Items)
                {
                 int disasterID = ((Disaster)item).ID;
disaters[i++] =
RiskUtils.ComponentsThreatsTable.LocateDisaster(disasterID);
                }
 RiskUtils.Scenario scenario = new RiskUtils.Scenario(disaters);
                int[] impactVector = scenario.Execute();
                int[] impactVectorFinal = new
int[impactVector.Length];
foreach (int compId in
RiskUtils.ComponentsThreatsTable.Components.Keys)
                {
int m = RiskUtils.ComponentsThreatsTable.LocateComponent(compId);
                    if (impactVector[m] == 1)
                    {
                        impactVectorFinal[i] = 1;
                        BusinessLayer.Component component =
Presentation_Layer.MainForm.Components.GetItem(compId);
                        if (component.IsCritical)
                        {
//Εμφάνιση κατεστραμμένων υποσυστημάτων
      TreeNode node = myTreeView1.GetObjectNode(component);
                            node.ForeColor = Color.Red;
                BusinessLayer.Component parent = component.Parent;
                            while (parent != null)
                            {
                                if (component.IsCritical)
                                {
                                    int m2 =
RiskUtils.ComponentsThreatsTable.LocateComponent(parent.ID);
                                    impactVectorFinal[m2] = 1;

                                node = myTreeView1.GetObjectNode(parent);
                                    node.ForeColor = Color.Red;
                                    parent = parent.Parent;
                                }
                                else
                                    break;
                            } } } }
                // impactVector[0]++;
                int count = 0;
                for (int j = 0; j < impactVectorFinal.Length;j++)
                    if (impactVectorFinal[j] == 1)
                        count++;
                _loadedScenario.vi = count;}}

      private void toolStripButton2_Click(object sender, EventArgs e)
        {
```

```
            SelectScenariosForm selectScenariosForm = new
SelectScenariosForm();

            // Show the dialog and determine the state of the
            // DialogResult property for the form.
            if (selectScenariosForm.ShowDialog() == DialogResult.OK)
            {
                Scenario scenario =
selectScenariosForm.SelectedScenario;
                _loadedScenario = scenario;
                Disasters disasters = scenario.Disasters;
                listBoxScenario.DataSource = disasters.DataProvider;
                listBoxScenario.ValueMember = "ID";
                listBoxScenario.DisplayMember = "Name";
            }
        }

        private void splitContainer2_Panel2_Paint(object sender,
PaintEventArgs e)
        {

        }

        private void toolStripButtonRecovery_Click(object sender,
EventArgs e)
        {
            int[] disastersIds = new
int[listBoxScenario.Items.Count];

            for (int a=0 ; a < listBoxScenario.Items.Count;a++)
            {

disastersIds[a]=((Disaster)(listBoxScenario.Items[a])).ID;
}

            int[][] prob =
RiskUtils.DisastersRecoveriesTable.SelectDisasters(disastersIds);
            Combinatorics.LinearAssignmentProblem problem = new
Combinatorics.LinearAssignmentProblem(prob);
            System.SByte[][] solution = problem.munkres();
            //Pass the filepath and filename to the StreamWriter
Constructor
            StreamWriter sw = new StreamWriter("C:\\solution.txt");
            for (int k = 0; k < solution.Length; k++)
            {
                for (int l = 0; l < solution.Length; l++)
                {
                    sw.Write(solution[k][l].ToString() + " ");

                }
                sw.WriteLine();
            }

            //Close the file
            sw.Close();

            Recoveries recoveries =new Recoveries();


            int i, j, numRows, numCols;
            numRows = solution.Length;
```

```
            numCols = solution[0].Length;
           for (i = 0; i < RiskUtils.DisastersRecoveriesTable.n; i++)
            {
                for (j = 0; j < numCols; j++)
                {
                    if (solution[i][j] == 1)
                    {
                        int recID =
RiskUtils.DisastersRecoveriesTable.GetRecoveryIdDB(i*numCols + j);
                        if (recID > 0)
                        {
                      Recovery recovery = Recoveries.FindById(recID);
                            recoveries.DataProvider.Add(recovery);
                            break;
                        }
                    }
                }
            }

            dataGridViewRecoveries.DataSource =
recoveries.DataProvider;
            dataGridViewRecoveries.Columns.Remove("ID");
            dataGridViewRecoveries.Columns.Remove("DataItem");
        }
        private void saveScenariosButton_Click(object sender,
EventArgs e)
        {
            TextInputForm form = new TextInputForm();

            if (form.ShowDialog() == DialogResult.OK)
            {
               // form.InputText
                Scenario scenario = new Scenario();
                scenario.Name = form.InputText;
                Disaster[] disasters = new
Disaster[listBoxScenario.Items.Count];
                for (int i = 0; i < listBoxScenario.Items.Count; i++)
                {
                    disasters[i] =
(Disaster)listBoxScenario.Items[i];
                }
                Scenarios.Insert(scenario, disasters);
            } } }}
```

8.2.9 RiskDataset.Designer.cs

```
//------------------------------------------------------------------
-----------
// <auto-generated>
//     This code was generated by a tool.
//     Runtime Version:2.0.50727.42
//
//     Changes to this file may cause incorrect behavior and will be
lost if
//     the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------
-----------
```

```
#pragma warning disable 1591
namespace RiskManagement {
    using System;

[System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.T
ypedDataSetGenerator", "2.0.0.0")]
    [Serializable()]
    [System.ComponentModel.DesignerCategoryAttribute("code")]
    [System.ComponentModel.ToolboxItem(true)]

[System.Xml.Serialization.XmlSchemaProviderAttribute("GetTypedDataSet
Schema")]
    [System.Xml.Serialization.XmlRootAttribute("RiskDataSet")]

[System.ComponentModel.Design.HelpKeywordAttribute("vs.data.DataSet")
]
    public partial class RiskDataSet : System.Data.DataSet {

        private ComponentsDataTable tableComponents;

        private System.Data.SchemaSerializationMode
_schemaSerializationMode =
System.Data.SchemaSerializationMode.IncludeSchema;

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]
        public RiskDataSet() {
            this.BeginInit();
            this.InitClass();


            System.ComponentModel.CollectionChangeEventHandler
schemaChangedHandler = new
System.ComponentModel.CollectionChangeEventHandler(this.SchemaChanged
);
            base.Tables.CollectionChanged += schemaChangedHandler;
            base.Relations.CollectionChanged += schemaChangedHandler;
            this.EndInit();
        }

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]

        protected
RiskDataSet(System.Runtime.Serialization.SerializationInfo info,
System.Runtime.Serialization.StreamingContext context) :
                base(info, context, false) {
            if ((this.IsBinarySerialized(info, context) == true)) {
                this.InitVars(false);
                System.ComponentModel.CollectionChangeEventHandler
schemaChangedHandler1 = new
System.ComponentModel.CollectionChangeEventHandler(this.SchemaChanged
);
                this.Tables.CollectionChanged +=
schemaChangedHandler1;
                this.Relations.CollectionChanged +=
schemaChangedHandler1;
                return;
            }
            string strSchema = ((string)(info.GetValue("XmlSchema",
typeof(string))));
            if ((this.DetermineSchemaSerializationMode(info, context)
== System.Data.SchemaSerializationMode.IncludeSchema)) {
```

```
 System.Data.DataSet ds = new System.Data.DataSet();
                ds.ReadXmlSchema(new System.Xml.XmlTextReader(new
System.IO.StringReader(strSchema)));
                if ((ds.Tables["Components"] != null)) {
                    base.Tables.Add(new
ComponentsDataTable(ds.Tables["Components"]));
                }
                this.DataSetName = ds.DataSetName;
                this.Prefix = ds.Prefix;
                this.Namespace = ds.Namespace;
                this.Locale = ds.Locale;
                this.CaseSensitive = ds.CaseSensitive;
                this.EnforceConstraints = ds.EnforceConstraints;
                this.Merge(ds, false,
System.Data.MissingSchemaAction.Add);
                this.InitVars();
            }
            else {
                this.ReadXmlSchema(new System.Xml.XmlTextReader(new
System.IO.StringReader(strSchema)));
            }
            this.GetSerializationData(info, context);
            System.ComponentModel.CollectionChangeEventHandler
schemaChangedHandler = new
System.ComponentModel.CollectionChangeEventHandler(this.SchemaChanged
);
            base.Tables.CollectionChanged += schemaChangedHandler;
            this.Relations.CollectionChanged += schemaChangedHandler;
        }
        [System.Diagnostics.DebuggerNonUserCodeAttribute()]
        [System.ComponentModel.Browsable(false)]

[System.ComponentModel.DesignerSerializationVisibility(System.Compone
ntModel.DesignerSerializationVisibility.Content)]
        public ComponentsDataTable Components {
            get {
                return this.tableComponents;
            }}

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]

        [System.ComponentModel.BrowsableAttribute(true)]

[System.ComponentModel.DesignerSerializationVisibilityAttribute(Syste
m.ComponentModel.DesignerSerializationVisibility.Visible)]
        public override System.Data.SchemaSerializationMode
SchemaSerializationMode {
            get {
                return this._schemaSerializationMode;
            }
            set {
                this._schemaSerializationMode = value;
            }}

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]

[System.ComponentModel.DesignerSerializationVisibilityAttribute(Syste
m.ComponentModel.DesignerSerializationVisibility.Hidden)]
        public new System.Data.DataTableCollection Tables {
            get {
```

```
                return base.Tables;
            }
        }

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]

[System.ComponentModel.DesignerSerializationVisibilityAttribute(Syste
m.ComponentModel.DesignerSerializationVisibility.Hidden)]
        public new System.Data.DataRelationCollection Relations {
            get {
                return base.Relations;
            }
        }

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]
        protected override void InitializeDerivedDataSet() {
            this.BeginInit();
            this.InitClass();
            this.EndInit();
        }

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]
        public override System.Data.DataSet Clone() {
            RiskDataSet cln = ((RiskDataSet)(base.Clone()));
            cln.InitVars();
            cln.SchemaSerializationMode =
this.SchemaSerializationMode;
            return cln;
        }

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]
        protected override bool ShouldSerializeTables() {
            return false;
        }

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]
        protected override bool ShouldSerializeRelations() {
            return false;
        }

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]

        protected override void
ReadXmlSerializable(System.Xml.XmlReader reader) {
            if ((this.DetermineSchemaSerializationMode(reader) ==
System.Data.SchemaSerializationMode.IncludeSchema)) {
                this.Reset();
                System.Data.DataSet ds = new System.Data.DataSet();
                ds.ReadXml(reader);
                if ((ds.Tables["Components"] != null)) {
                    base.Tables.Add(new
ComponentsDataTable(ds.Tables["Components"]));
                }
                this.DataSetName = ds.DataSetName;
                this.Prefix = ds.Prefix;
                this.Namespace = ds.Namespace;

                this.Locale = ds.Locale;
                this.CaseSensitive = ds.CaseSensitive;
                this.EnforceConstraints = ds.EnforceConstraints;
```

```
this.Merge(ds, false, System.Data.MissingSchemaAction.Add);
                this.InitVars();
            }
            else {
                this.ReadXml(reader);
                this.InitVars();
            }
        }

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]
        protected override System.Xml.Schema.XmlSchema
GetSchemaSerializable() {
            System.IO.MemoryStream stream = new
System.IO.MemoryStream();
            this.WriteXmlSchema(new System.Xml.XmlTextWriter(stream,
null));
            stream.Position = 0;
            return System.Xml.Schema.XmlSchema.Read(new
System.Xml.XmlTextReader(stream), null);
        }

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]
        internal void InitVars() {
            this.InitVars(true);
        }

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]
        internal void InitVars(bool initTable) {
            this.tableComponents =
((ComponentsDataTable)(base.Tables["Components"]));
            if ((initTable == true)) {
                if ((this.tableComponents != null)) {
                    this.tableComponents.InitVars();
                }
            }
        }

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]
        private void InitClass() {
            this.DataSetName = "RiskDataSet";
            this.Prefix = "";
            this.Namespace = "http://tempuri.org/RiskDataSet.xsd";

            this.EnforceConstraints = true;
            this.SchemaSerializationMode =
System.Data.SchemaSerializationMode.IncludeSchema;
            this.tableComponents = new ComponentsDataTable();
            base.Tables.Add(this.tableComponents);
        }

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]
        private bool ShouldSerializeComponents() {
            return false;
        }

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]

        private void SchemaChanged(object sender,
System.ComponentModel.CollectionChangeEventArgs e) {
```

```
if ((e.Action ==
System.ComponentModel.CollectionChangeAction.Remove)) {
                this.InitVars();
            }
        }

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]
        public static System.Xml.Schema.XmlSchemaComplexType
GetTypedDataSetSchema(System.Xml.Schema.XmlSchemaSet xs) {
            RiskDataSet ds = new RiskDataSet();
            System.Xml.Schema.XmlSchemaComplexType type = new
System.Xml.Schema.XmlSchemaComplexType();
            System.Xml.Schema.XmlSchemaSequence sequence = new
System.Xml.Schema.XmlSchemaSequence();
            xs.Add(ds.GetSchemaSerializable());
            System.Xml.Schema.XmlSchemaAny any = new
System.Xml.Schema.XmlSchemaAny();
            any.Namespace = ds.Namespace;
            sequence.Items.Add(any);
            type.Particle = sequence;
            return type;
        }

 public delegate void ComponentsRowChangeEventHandler(object sender,
ComponentsRowChangeEvent e);

[System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.T
ypedDataSetGenerator", "2.0.0.0")]
        [System.Serializable()]

[System.Xml.Serialization.XmlSchemaProviderAttribute("GetTypedTableSc
hema")]
        public partial class ComponentsDataTable :
System.Data.DataTable, System.Collections.IEnumerable {

            private System.Data.DataColumn columncom_sys_id;
            private System.Data.DataColumn columncom_id;
            private System.Data.DataColumn columncom_name;
            [System.Diagnostics.DebuggerNonUserCodeAttribute()]
            public ComponentsDataTable() {
                this.TableName = "Components";
                this.BeginInit();

                this.InitClass();
                this.EndInit();
            }

            [System.Diagnostics.DebuggerNonUserCodeAttribute()]
            internal ComponentsDataTable(System.Data.DataTable table)
{
                this.TableName = table.TableName;
                if ((table.CaseSensitive !=
table.DataSet.CaseSensitive)) {
              this.CaseSensitive = table.CaseSensitive;
                }
                if ((table.Locale.ToString() !=
table.DataSet.Locale.ToString())) {
                    this.Locale = table.Locale;}

                if ((table.Namespace != table.DataSet.Namespace)) {
```

```
 this.Namespace = table.Namespace;
                }
                this.Prefix = table.Prefix;
                this.MinimumCapacity = table.MinimumCapacity;
            }

            [System.Diagnostics.DebuggerNonUserCodeAttribute()]
            protected
ComponentsDataTable(System.Runtime.Serialization.SerializationInfo
info, System.Runtime.Serialization.StreamingContext context) :
                    base(info, context) {
                this.InitVars();
            }
            [System.Diagnostics.DebuggerNonUserCodeAttribute()]
            public System.Data.DataColumn com_sys_idColumn {
                get {
                    return this.columncom_sys_id;
                }
            }

            [System.Diagnostics.DebuggerNonUserCodeAttribute()]
            public System.Data.DataColumn com_idColumn {
                get {
                    return this.columncom_id;
                }
            }

            [System.Diagnostics.DebuggerNonUserCodeAttribute()]
            public System.Data.DataColumn com_nameColumn {
                get {
                    return this.columncom_name;
                } }

            [System.Diagnostics.DebuggerNonUserCodeAttribute()]
            [System.ComponentModel.Browsable(false)]
            public int Count {
                get {
                    return this.Rows.Count;
                }
            }

            [System.Diagnostics.DebuggerNonUserCodeAttribute()]

            public ComponentsRow this[int index] {
                get {
                    return ((ComponentsRow)(this.Rows[index]));
                }
            }

  public event ComponentsRowChangeEventHandler ComponentsRowChanging;
  public event ComponentsRowChangeEventHandler ComponentsRowChanged;
  public event ComponentsRowChangeEventHandler ComponentsRowDeleting;
  public event ComponentsRowChangeEventHandler ComponentsRowDeleted;

            [System.Diagnostics.DebuggerNonUserCodeAttribute()]
            public void AddComponentsRow(ComponentsRow row) {
                this.Rows.Add(row);
            }
```

```
 [System.Diagnostics.DebuggerNonUserCodeAttribute()]
            public ComponentsRow AddComponentsRow(int com_sys_id,
string com_name) {
                ComponentsRow rowComponentsRow =
((ComponentsRow)(this.NewRow()));
                rowComponentsRow.ItemArray = new object[] {
                        com_sys_id,
                        null,
                        com_name};
                this.Rows.Add(rowComponentsRow);
                return rowComponentsRow;
            }

            [System.Diagnostics.DebuggerNonUserCodeAttribute()]
            public ComponentsRow FindBycom_id(int com_id) {
                return ((ComponentsRow)(this.Rows.Find(new object[] {
                            com_id})));
            }

            [System.Diagnostics.DebuggerNonUserCodeAttribute()]
            public virtual System.Collections.IEnumerator
GetEnumerator() {
                return this.Rows.GetEnumerator();
            }

            [System.Diagnostics.DebuggerNonUserCodeAttribute()]
            public override System.Data.DataTable Clone() {
     ComponentsDataTable cln = ((ComponentsDataTable)(base.Clone()));
                cln.InitVars();
                return cln;
            }

            [System.Diagnostics.DebuggerNonUserCodeAttribute()]


          protected override System.Data.DataTable CreateInstance() {
                return new ComponentsDataTable();
            }

            [System.Diagnostics.DebuggerNonUserCodeAttribute()]
            internal void InitVars() {
                this.columncom_sys_id = base.Columns["com_sys_id"];
                this.columncom_id = base.Columns["com_id"];
                this.columncom_name = base.Columns["com_name"];
            }

            [System.Diagnostics.DebuggerNonUserCodeAttribute()]
            private void InitClass() {
                this.columncom_sys_id = new
System.Data.DataColumn("com_sys_id", typeof(int), null,
System.Data.MappingType.Element);
                base.Columns.Add(this.columncom_sys_id);
                this.columncom_id = new
System.Data.DataColumn("com_id", typeof(int), null,
System.Data.MappingType.Element);
                base.Columns.Add(this.columncom_id);
                this.columncom_name = new
System.Data.DataColumn("com_name", typeof(string), null,
System.Data.MappingType.Element);
                base.Columns.Add(this.columncom_name);
```

```
this.Constraints.Add(new System.Data.UniqueConstraint("Constraint1",
new System.Data.DataColumn[] {
                                    this.columncom_id}, true));
                this.columncom_sys_id.AllowDBNull = false;
                this.columncom_id.AutoIncrement = true;
                this.columncom_id.AllowDBNull = false;
                this.columncom_id.ReadOnly = true;
                this.columncom_id.Unique = true;
                this.columncom_name.MaxLength = 255;
            }

            [System.Diagnostics.DebuggerNonUserCodeAttribute()]
            public ComponentsRow NewComponentsRow() {
                return ((ComponentsRow)(this.NewRow()));
            }

            [System.Diagnostics.DebuggerNonUserCodeAttribute()]
            protected override System.Data.DataRow
NewRowFromBuilder(System.Data.DataRowBuilder builder) {
                return new ComponentsRow(builder);
            }

            [System.Diagnostics.DebuggerNonUserCodeAttribute()]
            protected override System.Type GetRowType() {
                return typeof(ComponentsRow);
            }

            [System.Diagnostics.DebuggerNonUserCodeAttribute()]
            protected override void
OnRowChanged(System.Data.DataRowChangeEventArgs e) {
                base.OnRowChanged(e);
                if ((this.ComponentsRowChanged != null)) {
                    this.ComponentsRowChanged(this, new
ComponentsRowChangeEvent(((ComponentsRow)(e.Row)), e.Action));

                }
            }

            [System.Diagnostics.DebuggerNonUserCodeAttribute()]
            protected override void
OnRowChanging(System.Data.DataRowChangeEventArgs e) {
                base.OnRowChanging(e);
                if ((this.ComponentsRowChanging != null)) {
                    this.ComponentsRowChanging(this, new
ComponentsRowChangeEvent(((ComponentsRow)(e.Row)), e.Action));
                }
            }

            [System.Diagnostics.DebuggerNonUserCodeAttribute()]
            protected override void
OnRowDeleted(System.Data.DataRowChangeEventArgs e) {
                base.OnRowDeleted(e);
                if ((this.ComponentsRowDeleted != null)) {
                    this.ComponentsRowDeleted(this, new
ComponentsRowChangeEvent(((ComponentsRow)(e.Row)), e.Action));
                }
            }

            [System.Diagnostics.DebuggerNonUserCodeAttribute()]
```

```
            protected override void
OnRowDeleting(System.Data.DataRowChangeEventArgs e) {
                base.OnRowDeleting(e);
                if ((this.ComponentsRowDeleting != null)) {
                    this.ComponentsRowDeleting(this, new
ComponentsRowChangeEvent(((ComponentsRow)(e.Row)), e.Action));
                }
            }

            [System.Diagnostics.DebuggerNonUserCodeAttribute()]
            public void RemoveComponentsRow(ComponentsRow row) {
                this.Rows.Remove(row);
            }

            [System.Diagnostics.DebuggerNonUserCodeAttribute()]
            public static System.Xml.Schema.XmlSchemaComplexType
GetTypedTableSchema(System.Xml.Schema.XmlSchemaSet xs) {
                System.Xml.Schema.XmlSchemaComplexType type = new
System.Xml.Schema.XmlSchemaComplexType();
                System.Xml.Schema.XmlSchemaSequence sequence = new
System.Xml.Schema.XmlSchemaSequence();
                RiskDataSet ds = new RiskDataSet();
                xs.Add(ds.GetSchemaSerializable());
                System.Xml.Schema.XmlSchemaAny any1 = new
System.Xml.Schema.XmlSchemaAny();
                any1.Namespace = "http://www.w3.org/2001/XMLSchema";
                any1.MinOccurs = new decimal(0);
                any1.MaxOccurs = decimal.MaxValue;
                any1.ProcessContents =
System.Xml.Schema.XmlSchemaContentProcessing.Lax;
                sequence.Items.Add(any1);
                System.Xml.Schema.XmlSchemaAny any2 = new
System.Xml.Schema.XmlSchemaAny();
                any2.Namespace = "urn:schemas-microsoft-com:xml-
diffgram-v1";
                any2.MinOccurs = new decimal(1);

                any2.ProcessContents =
System.Xml.Schema.XmlSchemaContentProcessing.Lax;
                sequence.Items.Add(any2);
                System.Xml.Schema.XmlSchemaAttribute attribute1 = new
System.Xml.Schema.XmlSchemaAttribute();
                attribute1.Name = "namespace";
                attribute1.FixedValue = ds.Namespace;
                type.Attributes.Add(attribute1);
                System.Xml.Schema.XmlSchemaAttribute attribute2 = new
System.Xml.Schema.XmlSchemaAttribute();
                attribute2.Name = "tableTypeName";
                attribute2.FixedValue = "ComponentsDataTable";
                type.Attributes.Add(attribute2);
                type.Particle = sequence;
                return type;
            }
        }

[System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.T
ypedDataSetGenerator", "2.0.0.0")]
        public partial class ComponentsRow : System.Data.DataRow {

            private ComponentsDataTable tableComponents;
```

```
            [System.Diagnostics.DebuggerNonUserCodeAttribute()]
            internal ComponentsRow(System.Data.DataRowBuilder rb) :
                    base(rb) {
                this.tableComponents =
((ComponentsDataTable)(this.Table));
            }

            [System.Diagnostics.DebuggerNonUserCodeAttribute()]
            public int com_sys_id {
                get {
                    return
((int)(this[this.tableComponents.com_sys_idColumn]));
                }
                set {
                    this[this.tableComponents.com_sys_idColumn] =
value;
                }
            }

            [System.Diagnostics.DebuggerNonUserCodeAttribute()]
            public int com_id {
                get {
                    return
((int)(this[this.tableComponents.com_idColumn]));
                }
                set {
                    this[this.tableComponents.com_idColumn] = value;
                }            }

            [System.Diagnostics.DebuggerNonUserCodeAttribute()]
            public string com_name {
                get {
                    try {
                        return
((string)(this[this.tableComponents.com_nameColumn]));
                    }

                    catch (System.InvalidCastException e) {
                        throw new
System.Data.StrongTypingException("The value for column \'com_name\'
in table \'Components\' is DBNull.", e);
                    }
                }
                set {
                    this[this.tableComponents.com_nameColumn] =
value;
                }
            }

            [System.Diagnostics.DebuggerNonUserCodeAttribute()]
            public bool Iscom_nameNull() {
                return
this.IsNull(this.tableComponents.com_nameColumn);
            }

            [System.Diagnostics.DebuggerNonUserCodeAttribute()]
            public void Setcom_nameNull() {
                this[this.tableComponents.com_nameColumn] =
System.Convert.DBNull;}}
```

```
[System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.T
ypedDataSetGenerator", "2.0.0.0")]
        public class ComponentsRowChangeEvent : System.EventArgs {

            private ComponentsRow eventRow;

            private System.Data.DataRowAction eventAction;

            [System.Diagnostics.DebuggerNonUserCodeAttribute()]
            public ComponentsRowChangeEvent(ComponentsRow row,
System.Data.DataRowAction action) {
                this.eventRow = row;
                this.eventAction = action;
            }

            [System.Diagnostics.DebuggerNonUserCodeAttribute()]
            public ComponentsRow Row {
                get {
                    return this.eventRow;
                }
            }

            [System.Diagnostics.DebuggerNonUserCodeAttribute()]
            public System.Data.DataRowAction Action {
                get {
                    return this.eventAction;
                } } } }}

namespace RiskManagement.RiskDataSetTableAdapters {

[System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.T
ypedDataSetGenerator", "2.0.0.0")]
    [System.ComponentModel.DesignerCategoryAttribute("code")]
    [System.ComponentModel.ToolboxItem(true)]
    [System.ComponentModel.DataObjectAttribute(true)]
[System.ComponentModel.DesignerAttribute("Microsoft.VSDesigner.DataSo
urce.Design.TableAdapterDesigner, Microsoft.VSDesigner" +
        ", Version=8.0.0.0, Culture=neutral,
PublicKeyToken=b03f5f7f11d50a3a")]
[System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdap
ter")]
    public partial class ComponentsTableAdapter :
System.ComponentModel.Component {

        private System.Data.SqlClient.SqlDataAdapter _adapter;

        private System.Data.SqlClient.SqlConnection _connection;

        private System.Data.SqlClient.SqlCommand[]
_commandCollection;

        private bool _clearBeforeFill;

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]
        public ComponentsTableAdapter() {
            this.ClearBeforeFill = true;
        }
```

```
  [System.Diagnostics.DebuggerNonUserCodeAttribute()]

        private System.Data.SqlClient.SqlDataAdapter Adapter {
            get {
                if ((this._adapter == null)) {
                    this.InitAdapter();
                }
                return this._adapter;
            }
        }

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]
        internal System.Data.SqlClient.SqlConnection Connection {
            get {
                if ((this._connection == null)) {
                    this.InitConnection();
                }
                return this._connection;
            }
            set {
                this._connection = value;
                if ((this.Adapter.InsertCommand != null)) {
                    this.Adapter.InsertCommand.Connection = value;
                }
                if ((this.Adapter.DeleteCommand != null)) {
                    this.Adapter.DeleteCommand.Connection = value;
                }
                if ((this.Adapter.UpdateCommand != null)) {

                this.Adapter.UpdateCommand.Connection = value;
}
    for (int i = 0; (i < this.CommandCollection.Length); i = (i + 1))
{
                    if ((this.CommandCollection[i] != null)) {
((System.Data.SqlClient.SqlCommand)(this.CommandCollection[i])).Conne
ction = value;
                    }
                }
            }
        }

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]
        protected System.Data.SqlClient.SqlCommand[]
CommandCollection {
            get {
                if ((this._commandCollection == null)) {
                    this.InitCommandCollection();
                }
                return this._commandCollection;
            }
        }

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]
        public bool ClearBeforeFill {
            get {
                return this._clearBeforeFill;
            }
            set {
                this._clearBeforeFill = value;
            }}
```

```
        [System.Diagnostics.DebuggerNonUserCodeAttribute()]
        private void InitAdapter() {
            this._adapter = new
System.Data.SqlClient.SqlDataAdapter();
            System.Data.Common.DataTableMapping tableMapping = new
System.Data.Common.DataTableMapping();
            tableMapping.SourceTable = "Table";
            tableMapping.DataSetTable = "Components";
            tableMapping.ColumnMappings.Add("com_sys_id",
"com_sys_id");
            tableMapping.ColumnMappings.Add("com_id", "com_id");
            tableMapping.ColumnMappings.Add("com_name", "com_name");
            this._adapter.TableMappings.Add(tableMapping);
            this._adapter.DeleteCommand = new
System.Data.SqlClient.SqlCommand();
            this._adapter.DeleteCommand.Connection = this.Connection;
            this._adapter.DeleteCommand.CommandText = "DELETE FROM
[dbo].[Components] WHERE (([com_sys_id] = @Original_com_sys_id) AND
(" +
                "[com_id] = @Original_com_id) AND ((@IsNull_com_name
= 1 AND [com_name] IS NULL) " +
                "OR ([com_name] = @Original_com_name)))";
            this._adapter.DeleteCommand.CommandType =
System.Data.CommandType.Text;
            this._adapter.DeleteCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@Original_com_sys_id",

System.Data.SqlDbType.Int, 0, System.Data.ParameterDirection.Input,
0, 0, "com_sys_id", System.Data.DataRowVersion.Original, false, null,
"", "", ""));
            this._adapter.DeleteCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@Original_com_id",

System.Data.SqlDbType.Int, 0, System.Data.ParameterDirection.Input,
0, 0, "com_id", System.Data.DataRowVersion.Original, false, null, "",
"", ""));
            this._adapter.DeleteCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@IsNull_com_name",
System.Data.SqlDbType.Int, 0, System.Data.ParameterDirection.Input,
0, 0, "com_name", System.Data.DataRowVersion.Original, true, null,
"", "", ""));
            this._adapter.DeleteCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@Original_com_name",
System.Data.SqlDbType.VarChar, 0,
System.Data.ParameterDirection.Input, 0, 0, "com_name",
System.Data.DataRowVersion.Original, false, null, "", "", ""));
            this._adapter.InsertCommand = new
System.Data.SqlClient.SqlCommand();
            this._adapter.InsertCommand.Connection = this.Connection;
            this._adapter.InsertCommand.CommandText = "INSERT INTO
[dbo].[Components] ([com_sys_id], [com_name]) VALUES (@com_sys_id,
@c" +
                "om_name);\r\nSELECT com_sys_id, com_id, com_name
FROM Components WHERE (com_id = S" +
                "COPE_IDENTITY())";
            this._adapter.InsertCommand.CommandType =
System.Data.CommandType.Text;
            this._adapter.InsertCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@com_sys_id",
System.Data.SqlDbType.Int, 0, System.Data.ParameterDirection.Input,
```

```
0, 0, "com_sys_id", System.Data.DataRowVersion.Current, false, null,
"", "", ""));
            this._adapter.InsertCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@com_name",
System.Data.SqlDbType.VarChar, 0,
System.Data.ParameterDirection.Input, 0, 0, "com_name",
System.Data.DataRowVersion.Current, false, null, "", "", ""));
            this._adapter.UpdateCommand = new
System.Data.SqlClient.SqlCommand();
            this._adapter.UpdateCommand.Connection = this.Connection;
            this._adapter.UpdateCommand.CommandText = @"UPDATE
[dbo].[Components] SET [com_sys_id] = @com_sys_id, [com_name] =
@com_name WHERE (([com_sys_id] = @Original_com_sys_id) AND ([com_id]
= @Original_com_id) AND ((@IsNull_com_name = 1 AND [com_name] IS
NULL) OR ([com_name] = @Original_com_name)));
SELECT com_sys_id, com_id, com_name FROM Components WHERE (com_id =
@com_id)";
            this._adapter.UpdateCommand.CommandType =
System.Data.CommandType.Text;
            this._adapter.UpdateCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@com_sys_id",
System.Data.SqlDbType.Int, 0, System.Data.ParameterDirection.Input,
0, 0, "com_sys_id", System.Data.DataRowVersion.Current, false, null,
"", "", ""));
            this._adapter.UpdateCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@com_name",
System.Data.SqlDbType.VarChar, 0,

System.Data.ParameterDirection.Input, 0, 0, "com_name",
System.Data.DataRowVersion.Current, false, null, "", "", ""));
            this._adapter.UpdateCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@Original_com_sys_id",
System.Data.SqlDbType.Int, 0, System.Data.ParameterDirection.Input,

0, 0, "com_sys_id", System.Data.DataRowVersion.Original, false, null,
"", "", ""));
            this._adapter.UpdateCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@Original_com_id",
System.Data.SqlDbType.Int, 0, System.Data.ParameterDirection.Input,
0, 0, "com_id", System.Data.DataRowVersion.Original, false, null, "",
"", ""));
            this._adapter.UpdateCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@IsNull_com_name",
System.Data.SqlDbType.Int, 0, System.Data.ParameterDirection.Input,
0, 0, "com_name", System.Data.DataRowVersion.Original, true, null,
"", "", ""));
            this._adapter.UpdateCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@Original_com_name",
System.Data.SqlDbType.VarChar, 0,
System.Data.ParameterDirection.Input, 0, 0, "com_name",
System.Data.DataRowVersion.Original, false, null, "", "", ""));
            this._adapter.UpdateCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@com_id",
System.Data.SqlDbType.Int, 4, System.Data.ParameterDirection.Input,
0, 0, "com_id", System.Data.DataRowVersion.Current, false, null, "",
"", ""));
        }

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]
        private void InitConnection() {
```

```
 this._connection = new System.Data.SqlClient.SqlConnection();
            this._connection.ConnectionString =
global::RiskManagement.Properties.Settings.Default.RiskConnectionStri
ng;
        }

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]
        private void InitCommandCollection() {
            this._commandCollection = new
System.Data.SqlClient.SqlCommand[1];
            this._commandCollection[0] = new
System.Data.SqlClient.SqlCommand();
            this._commandCollection[0].Connection = this.Connection;
            this._commandCollection[0].CommandText = "SELECT
com_sys_id, com_id, com_name FROM dbo.Components";
            this._commandCollection[0].CommandType =
System.Data.CommandType.Text;
        }

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]

[System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdap
ter")]

[System.ComponentModel.DataObjectMethodAttribute(System.ComponentMode
l.DataObjectMethodType.Fill, true)]
        public virtual int Fill(RiskDataSet.ComponentsDataTable
dataTable) {

            this.Adapter.SelectCommand = this.CommandCollection[0];
            if ((this.ClearBeforeFill == true)) {
                dataTable.Clear();
            }

            int returnValue = this.Adapter.Fill(dataTable);
            return returnValue;
        }

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]

[System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdap
ter")]

[System.ComponentModel.DataObjectMethodAttribute(System.ComponentMode
l.DataObjectMethodType.Select, true)]
        public virtual RiskDataSet.ComponentsDataTable GetData() {
            this.Adapter.SelectCommand = this.CommandCollection[0];
            RiskDataSet.ComponentsDataTable dataTable = new
RiskDataSet.ComponentsDataTable();
            this.Adapter.Fill(dataTable);
            return dataTable;
        }

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]

[System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdap
ter")]
        public virtual int Update(RiskDataSet.ComponentsDataTable
dataTable) {
            return this.Adapter.Update(dataTable);        }
```

```
        [System.Diagnostics.DebuggerNonUserCodeAttribute()]

[System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdap
ter")]
        public virtual int Update(RiskDataSet dataSet) {
            return this.Adapter.Update(dataSet, "Components");
        }

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]

[System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdap
ter")]
        public virtual int Update(System.Data.DataRow dataRow) {
            return this.Adapter.Update(new System.Data.DataRow[] {
                        dataRow});
        }

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]

[System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdap
ter")]
        public virtual int Update(System.Data.DataRow[] dataRows) {
            return this.Adapter.Update(dataRows);
        }

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]

[System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdap
ter")]

[System.ComponentModel.DataObjectMethodAttribute(System.ComponentMode
l.DataObjectMethodType.Delete, true)]

        public virtual int Delete(int Original_com_sys_id, int
Original_com_id, string Original_com_name) {
            this.Adapter.DeleteCommand.Parameters[0].Value =
((int)(Original_com_sys_id));
            this.Adapter.DeleteCommand.Parameters[1].Value =
((int)(Original_com_id));
            if ((Original_com_name == null)) {
                this.Adapter.DeleteCommand.Parameters[2].Value =
((object)(1));
                this.Adapter.DeleteCommand.Parameters[3].Value =
System.DBNull.Value;
            }
            else {
                this.Adapter.DeleteCommand.Parameters[2].Value =
((object)(0));
                this.Adapter.DeleteCommand.Parameters[3].Value =
((string)(Original_com_name));
            }
            System.Data.ConnectionState previousConnectionState =
this.Adapter.DeleteCommand.Connection.State;
            if (((this.Adapter.DeleteCommand.Connection.State &
System.Data.ConnectionState.Open)
                        != System.Data.ConnectionState.Open)) {
                this.Adapter.DeleteCommand.Connection.Open();
            }
            try {
```

```
    int returnValue = this.Adapter.DeleteCommand.ExecuteNonQuery();
                return returnValue;
            }
            finally {
                if ((previousConnectionState ==
System.Data.ConnectionState.Closed)) {
                    this.Adapter.DeleteCommand.Connection.Close();
                }
            }
        }

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]

[System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdap
ter")]

[System.ComponentModel.DataObjectMethodAttribute(System.ComponentMode
l.DataObjectMethodType.Insert, true)]
        public virtual int Insert(int com_sys_id, string com_name) {
            this.Adapter.InsertCommand.Parameters[0].Value =
((int)(com_sys_id));
            if ((com_name == null)) {
                this.Adapter.InsertCommand.Parameters[1].Value =
System.DBNull.Value;
            }
            else {
                this.Adapter.InsertCommand.Parameters[1].Value =
((string)(com_name));

            }
            System.Data.ConnectionState previousConnectionState =
this.Adapter.InsertCommand.Connection.State;


            if (((this.Adapter.InsertCommand.Connection.State &
System.Data.ConnectionState.Open)
                        != System.Data.ConnectionState.Open)) {
                this.Adapter.InsertCommand.Connection.Open();
            }
            try {
                int returnValue =
this.Adapter.InsertCommand.ExecuteNonQuery();
                return returnValue;
            }
            finally {
                if ((previousConnectionState ==
System.Data.ConnectionState.Closed)) {
                    this.Adapter.InsertCommand.Connection.Close();
                }
            }
        }

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]

[System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdap
ter")]

[System.ComponentModel.DataObjectMethodAttribute(System.ComponentMode
l.DataObjectMethodType.Update, true)]
```

```
        public virtual int Update(int com_sys_id, string com_name,
int Original_com_sys_id, int Original_com_id, string
Original_com_name, int com_id) {
            this.Adapter.UpdateCommand.Parameters[0].Value =
((int)(com_sys_id));
            if ((com_name == null)) {
                this.Adapter.UpdateCommand.Parameters[1].Value =
System.DBNull.Value;
            }
            else {
                this.Adapter.UpdateCommand.Parameters[1].Value =
((string)(com_name));
            }
            this.Adapter.UpdateCommand.Parameters[2].Value =
((int)(Original_com_sys_id));
            this.Adapter.UpdateCommand.Parameters[3].Value =
((int)(Original_com_id));
            if ((Original_com_name == null)) {
                this.Adapter.UpdateCommand.Parameters[4].Value =
((object)(1));
                this.Adapter.UpdateCommand.Parameters[5].Value =
System.DBNull.Value;
            }
            else {
                this.Adapter.UpdateCommand.Parameters[4].Value =
((object)(0));
                this.Adapter.UpdateCommand.Parameters[5].Value =
((string)(Original_com_name));
            }
            this.Adapter.UpdateCommand.Parameters[6].Value =
((int)(com_id));

            System.Data.ConnectionState previousConnectionState =
this.Adapter.UpdateCommand.Connection.State;


            if (((this.Adapter.UpdateCommand.Connection.State &
System.Data.ConnectionState.Open)
                        != System.Data.ConnectionState.Open)) {
                this.Adapter.UpdateCommand.Connection.Open();
            }
            try {
                int returnValue =
this.Adapter.UpdateCommand.ExecuteNonQuery();
                return returnValue;
            }
            finally {
                if ((previousConnectionState ==
System.Data.ConnectionState.Closed)) {
                    this.Adapter.UpdateCommand.Connection.Close();
                } }}}}
                                        #pragma warning restore 1591
```

8.2.10 Program.cs

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;
using RiskManagement.Presentation_Layer;
```

```
namespace RiskManagement
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
//Κλήση κεντρικής Φόρμας
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Form form = new Form1();
            Presentation_Layer.Forms.RegisterMainForm(form);
            Application.Run(form);
        } }}
```

8.2.11 LoadSystemForm.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

using RiskManagement.DataAccessLayer;

namespace RiskManagement
{
    public partial class LoadSystemForm : Form
    {
        public LoadSystemForm()

        {
            InitializeComponent();
        }
//Επιλογή συστημάτων απο τη βάση
        private void LoadSystemForm_Load(object sender, EventArgs e)
        {
            Data data = DBManager.DataAccesor.read("Select * from
Systems");
            systemsListBox.DataSource = data.ds.Tables[0];
            systemsListBox.ValueMember = "sys_id";
            systemsListBox.DisplayMember = "sys_name";

        }

        public int SelectedSystem
        {
            get
            {
                if (systemsListBox.SelectedIndex > -1)
                    return (int)systemsListBox.SelectedValue;
                else
                    return -1;
            } }}}
```

8.2.12 HistogramControl.cs

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Windows.Forms;

namespace Histograma
{
	/// <summary>
	/// Summary description for HistogramaDesenat.
	/// </summary>
	public partial class HistogramControl :
System.Windows.Forms.UserControl
	{
		/// <summary>
		/// Required designer variable.
		/// </summary>
		private System.ComponentModel.Container components =
null;

	  private float space;

		public HistogramControl()
		{
			// This call is required by the Windows.Forms Form
Designer.
			InitializeComponent();
			// TODO: Add any initialization after the
InitializeComponent call
		//Ζωγράφισμα στατιστικών Ιστογραμμάτων σεναρίων
			this.Paint += new
PaintEventHandler(HistogramaDesenat_Paint);
	this.Resize+=new EventHandler(HistogramaDesenat_Resize);
		}
		/// <summary>
		/// Clean up any resources being used.
		/// </summary>
		protected override void Dispose( bool disposing )
		{
			if( disposing )
			{
				if(components != null)
				{
					components.Dispose();
				}
			}
			base.Dispose( disposing );
		}

		#region Component Designer generated code
		/// <summary>
		/// Required method for Designer support - do not modify
		/// the contents of this method with the code editor.
		/// </summary>
		private void InitializeComponent()
		{
```

```
                    // HistogramaDesenat
                    // Μορφοποίηση
this.Font = new System.Drawing.Font("Tahoma", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((System.Byte)(0)));
                    this.Name = "HistogramaDesenat";
                    this.Size = new System.Drawing.Size(208, 176);
              }
              #endregion

              private void HistogramaDesenat_Paint(object sender,
PaintEventArgs e)
              {
              if (myIsDrawing)
                    {
                          Graphics g = e.Graphics;
                          Pen myPen = new Pen(new
SolidBrush(myColor),myXUnit);
                          //The width of the pen is given by the XUnit
for the control.
                          for (int i=0;i<myValues.Length;i++)
                          {
                                //We draw each line
                                g.DrawLine(myPen,
                                      new PointF(myOffset +
(i*(myXUnit+space)), this.Height - myOffset),
                          new PointF(myOffset + (i * (myXUnit +
space)), this.Height - myOffset - myValues[i] * myYUnit));

                                //We plot the coresponding index for
the maximum value.
                    if (myValues[i] > myMaxValue)
                    {

                          myMaxValue = myValues[i];
                    }
                          }
     SizeF mySize = g.MeasureString(myMaxValue.ToString(), myFont);
      g.DrawString(myMaxValue.ToString(), myFont, new
SolidBrush(myColor),
                    new PointF(myOffset, this.Height - myOffset -
myMaxValue * myYUnit - 5 * space),
                    System.Drawing.StringFormat.GenericDefault);

                //g.DrawString(i.ToString(), myFont, new
SolidBrush(myColor),
                //    new PointF(myOffset + (i * myXUnit) -
(mySize.Width / 2), this.Height - myFont.Height),
                //    System.Drawing.StringFormat.GenericDefault);

                          //We draw the indexes for 0 and for the
length of the array beeing plotted
                //    g.DrawString("0",myFont, new
SolidBrush(myColor),new PointF(myOffset,this.Height -
myFont.Height),System.Drawing.StringFormat.GenericDefault);
              //          g.DrawString((myValues.Length-
1).ToString(),myFont,
              //                new SolidBrush(myColor),
```

```
//new PointF(myOffset + (myValues.Length * myXUnit) -
g.MeasureString((myValues.Length-1).ToString(),myFont).Width,
            //                  this.Height - myFont.Height),
            //
      System.Drawing.StringFormat.GenericDefault);

                        //We draw a rectangle surrounding the
control.
                        g.DrawRectangle(new System.Drawing.Pen(new
SolidBrush(Color.Black),1),0,0,this.Width-1,this.Height-1);
            }
            }

            long myMaxValue;
            private long[] myValues;
            private bool myIsDrawing;

            private float myYUnit; //this gives the vertical unit
used to scale our values
            private float myXUnit; //this gives the horizontal unit
used to scale our values
            private int myOffset = 20; //the offset, in pixels, from
the control margins.

            private Color myColor = Color.Black;
            private Font myFont = new Font("Tahoma",10);

            [Category("Histogram Options")]
            [Description ("The distance from the margins for the
histogram")]
            public int Offset
            {
                  set
                  {
                        if (value>0)

                              myOffset= value;
                  }
                  get
                  {
                        return myOffset;
                  }
            }

            [Category("Histogram Options")]
            [Description ("The color used within the control")]
            public Color DisplayColor
            {
                  set
                  {
                        myColor = value;
                  }
                  get
                  {
                        return myColor;
                  }
            }

            /// <summary>
            /// We draw the histogram on the control
```

```
/// </summary>
            /// <param name="myValues">The values beeing draw</param>
            public void DrawHistogram(long[] Values)
            {
                  myValues = new long[Values.Length];
                  Values.CopyTo(myValues,0);
                  myIsDrawing = true;
                  myMaxValue = getMaxim(myValues);
                  ComputeXYUnitValues();
                  this.Refresh();
            }

            /// <summary>
            /// We get the highest value from the array
            /// </summary>
            /// <param name="Vals">The array of values in which we
look</param>
            /// <returns>The maximum value</returns>
            private long getMaxim(long[] Vals)
            {
                  if (myIsDrawing)
                  {
                        long max = 0;
                        for (int i=0;i<Vals.Length;i++)
                        {
                              if (Vals[i] > max)
                                    max = Vals[i];
                        }

                        return max;
                  }
                  return 1;
            }

            private void HistogramaDesenat_Resize(object sender,
EventArgs e)

            {
                  if (myIsDrawing)
                  {
                        ComputeXYUnitValues();
                  }
                  this.Refresh();
            }

            private void ComputeXYUnitValues()
            {
            int numValues = myValues.Length;
            float space = 3.0F;
            int componentWidth = this.Width;

            float columnWidth = (componentWidth-(2 * myOffset)) /
numValues - space;
            if (columnWidth > 20.0F)
            {
                columnWidth = 20.0F;
            }
            else if (columnWidth <= 1)
            {
                space = 0.0F;            }
```

```
            myXUnit = columnWidth;
            this.space = space;
myYUnit = (float) (this.Height - (2 * myOffset)) / myMaxValue;
//myXUnit = (float) (this.Width - (2 * myOffset)) / (myValues.Length-
1);
            }}}
```

8.2.13 Global.cs

```
using System.Collections;
using System.Collections.Generic;
using System.Text;
using System.Text.RegularExpressions;

namespace RiskManagement
{
    public class SqlString
    {
        private StringBuilder _input;
        private string _output;
        private Regex theRegex;
        MatchCollection theMatches;
        private Dictionary<string, string> _params;
        public SqlString(string sql)
        {
            _input = new StringBuilder(sql);

            theRegex = new Regex(@"#(\w*)[0-9]");
            _params = new Dictionary<string, string>();
            theMatches = theRegex.Matches(_input.ToString());
            foreach (Match theMatch in theMatches)
            {

                string paramName = theMatch.Groups[1].ToString();
                _params.Add(paramName, "");
            }
        }

        public Dictionary<string, string> Parameters
        {
            get { return _params; }
            set { _params = value; }
        }

        public void AppendText(string str)
        {
            _input.AppendLine(str);
        }
        public string Text
        {
            get
            {
                _output = string.Copy(_input.ToString());
                foreach (string key in _params.Keys)
                {
                    _output = Regex.Replace(_output, "#" + key,
_params[key]);
                }
                return _output;          }
```

```
            set
            {
                _input = new StringBuilder(value);
            } } }}
```

8.2.14 Form1.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using RiskManagement.DataAccessLayer;
using RiskManagement.BusinessLayer;
using Combinatorics;
using RiskManagement.Presentation_Layer;
namespace RiskManagement
{
    public partial class Form1 : Form    {

        public Form1()
        {
            InitializeComponent();
                  DBManager.Initialize(new SqlDbKernel());
        }

        private static MyTreeView.MyTreeView m_tree;


        private IDataAccessor DataAccessor
        {
            get
            {
                        return DBManager.DataAccesor;
            }

        }

        private void LoadSystem(int systemID)
        {
            string title = BusinessLayer.Application.System.Name;

//Δημιουργία δενδροειδούς μορφής παρουσίασης δομής Συστήματος

Presentation_Layer.MainForm.Components.LoadSystemComponents(systemID)
;
            componentsTree.ClearNodes();
            componentsTree.Nodes.Clear();
            System.Windows.Forms.TreeNode rootNode= new
System.Windows.Forms.TreeNode();
            rootNode.Text = title;
            componentsTree.RootNode = rootNode;
componentsTree.PopulateTreeView(Presentation_Layer.MainForm.Component
s.DataProvider);
            componentsTree.SelectedNode = rootNode;
```

```
            this.Text = title;
        }

        public MyTreeView.MyTreeView ComponentsTree
        {
            get
            {
                return componentsTree;
            }
        }

        private void Form1_Load(object sender, EventArgs e)
        {
//Εμφάνιση Δένδρου Συστήματος - Υποσυστημάτων

            Presentation_Layer.MainForm.Create(this);
            threatsListBox.DataSource =
Presentation_Layer.MainForm.Threats.DataProvider;
            threatsListBox.ValueMember = "ID";
            threatsListBox.DisplayMember = "Name";
            listBoxDisasters.DataSource =
Presentation_Layer.MainForm.Disasters.DataProvider;
            listBoxDisasters.ValueMember = "ID";
            listBoxDisasters.DisplayMember = "Name";
            dataGridViewRecoveries.DataSource =
Presentation_Layer.MainForm.Recoveries.DataProvider;
            dataGridViewRecoveries.Columns.Remove("ID");
            dataGridViewRecoveries.Columns.Remove("DataItem");
            ComponentNameTextBox.DataBindings.Add("Text",
Presentation_Layer.MainForm.Components.DataProvider, "Name");
            checkBoxIsCritical.DataBindings.Add("Checked",
Presentation_Layer.MainForm.Components.DataProvider, "IsCritical");
        }
  public ArrayList FindColumnCompinations(object[] threats,int num)
        {
            ArrayList combinations = new ArrayList();
            try
                {

                Combinations c = new Combinations(threats, num);

                      while (c.hasMoreElements())
                      {
                            Object[] combo =
(Object[])c.nextElement();
                    combinations.Add(combo);
                }
                }
                catch (CombinatoricException ex){
                //Console.ReadLine();
            }
            return combinations;
        }
        public void RefreshComponentsTree()
        {

componentsTree.PopulateTreeView(Presentation_Layer.MainForm.Component
s.DataProvider);
        }

      private void toolStripButton1_Click(object sender, EventArgs e)
```

```
        {
            //BusinessLayer.Components.Insert(new
BusinessLayer.Component());
            CreateThreatForm createThreatForm = new
CreateThreatForm();
            createThreatForm.ShowDialog();
        }
        private void label4_Click(object sender, EventArgs e)
        {

        }

        private void threatsListBox_SelectedIndexChanged(object
sender, EventArgs e)
        {

        }

      private void toolStripButton3_Click(object sender, EventArgs e)
        {
            //BusinessLayer.Components.Insert(new
BusinessLayer.Component());
   CreateDisasterForm createDisasterForm = new CreateDisasterForm();
            createDisasterForm.ShowDialog();
        }

        private void threatsListBox_Click(object sender, EventArgs e)
        {
            if((sender as ListBox).SelectedItem!=null)
                Presentation_Layer.MainForm.Threats.Selected =
((sender as ListBox).SelectedItem as Threat);
        }

        private void listBoxDisasters_Click(object sender, EventArgs
e)
        {
            if ((sender as ListBox).SelectedItem != null)

    Presentation_Layer.MainForm.Disasters.Selected = ((sender as
ListBox).SelectedItem as Disaster);
        }
     private void toolStripButton5_Click(object sender, EventArgs e)
        {
         CreateRecoveryForm createRecoveryForm = new
CreateRecoveryForm();
            createRecoveryForm.ShowDialog();
        }
     private void listBoxRecoveries_Click(object sender, EventArgs e)
        {
            Presentation_Layer.MainForm.Recoveries.Selected =
((sender as ListBox).SelectedItem as Recovery);
        }

        private void DeleteThreatsButton_Click(object sender,
EventArgs e)
        {

BusinessLayer.Threats.DeleteThreat(Presentation_Layer.MainForm.Threat
s.Selected);

            Presentation_Layer.MainForm.Threats.Refresh();
```

```
        }
        private void ScenarioCreateToolStripMenuItem_Click(object
sender, EventArgs e)
        {
            ScenariosForm2 form = new ScenariosForm2();
            form.ShowDialog();
        }

        private void tbtnAddComponent_Click(object sender, EventArgs
e)
        {
            if (Presentation_Layer.MainForm.Components.Selected !=
null)
            {

Presentation_Layer.MainForm.Components.Selected.AddSubcomponent();

      Presentation_Layer.MainForm.Components.Refresh();
componentsTree.PopulateTreeView(Presentation_Layer.MainForm.Component
s.DataProvider);
            }
            else
            {

Presentation_Layer.MainForm.Components.AddNewComponent();
                Presentation_Layer.MainForm.Components.Refresh();
            } }
        public void DoNodeMouseClick(System.Windows.Forms.TreeNode
node)
        {
            if (node.Tag != null)
            {
                Presentation_Layer.MainForm.Components.Selected =
(BusinessLayer.Component)node.Tag;



int index =
Presentation_Layer.MainForm.Components.DataProvider.IndexOf(Presentat
ion_Layer.MainForm.Components.Selected);
 Binding binding = ComponentNameTextBox.DataBindings["Text"];
 BindingManagerBase manager = binding.BindingManagerBase;
                manager.Position = index;
            }
            else
                Presentation_Layer.MainForm.Components.Selected =
null;
        }

        private void componentsTree_NodeMouseClick(object sender,
TreeNodeMouseClickEventArgs e)
        {
            //DoNodeMouseClick(e.Node);
        }
        private void DeleteComponentButton_Click(object sender,
EventArgs e)
        {
            if (Presentation_Layer.MainForm.Components.Selected !=
null)
            {
```

```
BusinessLayer.Components.DeleteComponent(Presentation_Layer.MainForm.
Components.Selected);
                Presentation_Layer.MainForm.Components.Refresh();

componentsTree.PopulateTreeView(Presentation_Layer.MainForm.Component
s.DataProvider);
            }
        }
 private void StatisticsToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            StatistcsForm form = new StatistcsForm();
            form.ShowDialog();
        }
 private void toolStripButton4_Click(object sender, EventArgs e) {

          if (Presentation_Layer.MainForm.Disasters.Selected != null)

BusinessLayer.Disasters.DeleteDisaster(Presentation_Layer.MainForm.Di
sasters.Selected);
            Presentation_Layer.MainForm.Disasters.Refresh();
        }
        private void toolStripButton6_Click(object sender, EventArgs
e)
        {
            if (Presentation_Layer.MainForm.Recoveries.Selected !=
null)
BusinessLayer.Recoveries.DeleteRecovery(Presentation_Layer.MainForm.R
ecoveries.Selected);
            Presentation_Layer.MainForm.Recoveries.Refresh();
        }
        private void componentsTree_Click(object sender, EventArgs e)
       {
        }
private void systemToolStripMenuItem_Click(object sender, EventArgs
e)

        {
            Presentation_Layer.MainForm.Components.AddNewComponent();
            Presentation_Layer.MainForm.Components.Refresh();
componentsTree.PopulateTreeView(Presentation_Layer.MainForm.Component
s.DataProvider);
        }
        private void openToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            LoadSystemForm form = new LoadSystemForm();
            if (form.ShowDialog() == DialogResult.OK)
            {
                if (form.SelectedSystem > -1)
                {
    BusinessLayer.Application.System.LoadSystem(form.SelectedSystem);
                    LoadSystem(form.SelectedSystem);
                }
            }
        }
        private void ComponentNameTextBox_TextChanged(object sender,
EventArgs e)
        {
            global::System.Windows.Forms.MessageBox.Show("changes");
```

```
        }

        private void componentsTree_AfterSelect(object sender,
TreeViewEventArgs e)
        {
            DoNodeMouseClick(e.Node);
        }

        private void newToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            TextInputForm form = new TextInputForm();
            if (form.ShowDialog() == DialogResult.OK)
           {
               //form.InputText

int newSystemId =
RiskManagement.BusinessLayer.Application.System.CreateSystem(form.Inp
utText);
  BusinessLayer.Application.System.LoadSystem(newSystemId);
                LoadSystem(newSystemId);
            }
        }
        private void myTreeView1_AfterSelect(object sender,
TreeViewEventArgs e)
        {
                    } }}
```

8.2.15 DBManagerFactory.cs

```
using System;
using System.Data;
using System.Data.Odbc;
using System.Data.SqlClient;
using System.Data.OleDb;
using System.Data.OracleClient;

 namespace DataAccessLayer
{
//Διαχείρηση Λειτουργίας Βάσης Δεδομένων

  public abstract class DBManager
  {
    public static IDbConnection GetConnection(DataProvider
providerType)
    {
      IDbConnection iDbConnection = null;
      switch (providerType)
      {
        case DataProvider.SqlServer:
          iDbConnection = new SqlConnection();
          break;
        case DataProvider.OleDb:
          iDbConnection = new OleDbConnection();
          break;
        case DataProvider.Odbc:
          iDbConnection = new OdbcConnection();
          break;
        case DataProvider.Oracle:
          iDbConnection = new OracleConnection();
```

```
            break;
          default:
            return null;
        }
        return iDbConnection;
      }

      public static IDbCommand GetCommand(DataProvider providerType)
      {
        switch (providerType)
        {
          case DataProvider.SqlServer:
            return new SqlCommand();
          case DataProvider.OleDb:
            return new OleDbCommand();
          case DataProvider.Odbc:
            return new OdbcCommand();

    case DataProvider.Oracle:
            return new OracleCommand();
          default:
            return null;
        }
      }

      public static IDbDataAdapter GetDataAdapter(DataProvider
      providerType)
      {
        switch (providerType)
        {
          case DataProvider.SqlServer:
            return new SqlDataAdapter();
          case DataProvider.OleDb:
            return new OleDbDataAdapter();
          case DataProvider.Odbc:
            return new OdbcDataAdapter();
           case DataProvider.Oracle:
            return new OracleDataAdapter();
          default:

            return null;
        }
      }

      public static IDbTransaction GetTransaction(DataProvider
       providerType)
      {
        IDbConnection iDbConnection =GetConnection(providerType);
        IDbTransaction iDbTransaction
=iDbConnection.BeginTransaction();
        return iDbTransaction;
      }

      public static IDataParameter GetParameter(DataProvider
       providerType)
      {
        IDataParameter iDataParameter = null;
        switch (providerType)
        {
          case DataProvider.SqlServer:
            iDataParameter = new SqlParameter();
```

```
        break;
      case DataProvider.OleDb:
        iDataParameter = new OleDbParameter();
        break;
      case DataProvider.Odbc:
        iDataParameter = new OdbcParameter();
        break;
      case DataProvider.Oracle:
        iDataParameter = newOracleParameter();
       break;
    }
    return iDataParameter;
  }

  public static IDbDataParameter[] GetParameters(DataProvider
   providerType,
    int paramsCount)    {
    IDbDataParameter[]idbParams = newIDbDataParameter[paramsCount];

    switch (providerType)
    {
      case DataProvider.SqlServer:
        for (int i = 0; i < paramsCount;++i)
        {
          idbParams[i] = new SqlParameter();
        }
        break;
      case DataProvider.OleDb:
        for (int i = 0; i < paramsCount;++i)
        {
          idbParams[i] = new OleDbParameter();
        }
        break;
      case DataProvider.Odbc:
        for (int i = 0; i < paramsCount;++i)
        {
          idbParams[i] = new OdbcParameter();
        }
        break;

      case DataProvider.Oracle:
        for (int i = 0; i <intParamsLength; ++i)
        {
          idbParams[i] = newOracleParameter();
        }
        break;
      default:
        idbParams = null;
        break;
    }
    return idbParams;}}}
```

8.2.16 CreateThreatForm.cs

Φόρμα δημιουργίας απειλών και αποθήκευσης τους στην εφαρμογή.

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using RiskManagement;
using RiskManagement.DataAccessLayer;
using RiskManagement.BusinessLayer;
using RiskManagement.PresentationLayer;

namespace RiskManagement
{
    public partial class CreateThreatForm : Form
    {
        public CreateThreatForm()
        {
            InitializeComponent();
            Presentation_Layer.CreateThreatForm.Create(this);

        }
        private void button1_Click(object sender, EventArgs e){
            Threat threat;
```

//Ελεγχος αν η το υποσύστημα ειναι σημαντικό

```
            if (NewRadio.Checked)
            {
                threat = new Threat();
                threat.Name = textBox1.Text;
                BusinessLayer.Component component =
Presentation_Layer.MainForm.Components.Selected ;
                BusinessLayer.Threats.Insert(component, threat);
                Presentation_Layer.MainForm.Threats.Refresh();
            }
            else
            {
                BusinessLayer.Component component =
Presentation_Layer.MainForm.Components.Selected;
                threat =
Presentation_Layer.CreateThreatForm.Threats.Selected;
                BusinessLayer.Threats.InsertExistingThreat(component,
threat);
                Presentation_Layer.MainForm.Threats.Refresh();
            }

            Close();
        }
    private void CreateThreatForm_Load(object sender, EventArgs e)
        {
            componentsView.Enabled = false;
            threatsList.Enabled = false;
            componentsView.RootNode = new TreeNode();


componentsView.RootNode.Text =
RiskManagement.BusinessLayer.Application.System.Name;
componentsView.PopulateTreeView(Presentation_Layer.MainForm.Component
s.DataProvider);
   threatsList.DataSource =
Presentation_Layer.CreateThreatForm.Threats.DataProvider;
```

```
            threatsList.ValueMember = "ID";
            threatsList.DisplayMember = "Name";
        }
        private void componentsTree_NodeMouseClick(object sender,
TreeNodeMouseClickEventArgs e)
        {
            if ((e.Node != null) && (e.Node.Tag != null))
            {
 Presentation_Layer.CreateThreatForm.Components.Selected =
(BusinessLayer.Component)e.Node.Tag;
            }
            else
 Presentation_Layer.CreateThreatForm.Components.Selected = null;
        }
       private void threatsList_SelectedIndexChanged(object sender,
EventArgs e)
        {
            if(((ListBox)sender).SelectedItem != null)
            Presentation_Layer.CreateThreatForm.Threats.Selected =
(Threat)((ListBox)sender).SelectedItem;
        }
        private void NewRadio_CheckedChanged(object sender, EventArgs
e)
        {
            if (((RadioButton)sender).Checked)

                textBox1.Enabled = true;
            else
                textBox1.Enabled = false;
        }

        private void existingRadio_CheckedChanged(object sender,
EventArgs e)
        {
            if (((RadioButton)sender).Checked)
            {
                componentsView.Enabled = true;
                threatsList.Enabled = true;
            }
            else
            {
                componentsView.Enabled = false;
                threatsList.Enabled = false;
            }
        }
private void button2_Click(object sender, EventArgs e)

        {
            Close();
        } }}
```

8.2.17 CreateRecoveryForm.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
```

```
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using RiskManagement;
using RiskManagement.DataAccessLayer;
using RiskManagement.BusinessLayer;
using RiskManagement.PresentationLayer;

namespace RiskManagement
{
    public partial class CreateRecoveryForm : Form
    {
        public CreateRecoveryForm()
        {
            InitializeComponent();
        }
//Δημιουργία φόρμας Ανακτήσεων
        private void button1_Click(object sender, EventArgs e)
        {
            Recovery recovery = new Recovery();
            recovery.Name = textBox1.Text;
            recovery.Cost = Int32.Parse(textBox2.Text);
            Recoveries.Insert(recovery,
Presentation_Layer.MainForm.Disasters.Selected);
            Close();
        } }}
```

8.2.18 CreateDisasterForm.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using RiskManagement;
using RiskManagement.DataAccessLayer;
using RiskManagement.BusinessLayer;
using RiskManagement.PresentationLayer;

namespace RiskManagement
{
    public partial class CreateDisasterForm : Form
    {
        public CreateDisasterForm()
        {
            InitializeComponent();

        }

        private void button1_Click(object sender, EventArgs e)
        {
            Disaster disaster=new Disaster();
            disaster.Name=textBox1.Text;
           //Εισαγωγή Καταστροφών
Disasters.Insert(Presentation_Layer.MainForm.Components.Selected,
Presentation_Layer.MainForm.Threats.Selected, disaster);
            Presentation_Layer.MainForm.Disasters.Refresh();
```

```
            Close();
        }
        private void button2_Click(object sender, EventArgs e)
        {
            Close();
        } }}
```

8.2.19 Algorithms.cs

```
using System;
using System.Collections;

namespace Algorithms
{
      public interface IDataContainer : IComparable,IEnumerable
      {
      }
//Δημιουργία δομών δεδομένων για την υλοποίηση της εφαρμογής και
// παρουσίαση των αποτελεσμάτων
      public interface ITree : IDataContainer
      {
            ITree GetSubtree(int i);
            bool IsLeaf { get;}
      }
    public class Node<T>{

                  // Private member-variables
                  private T data;

                  private NodeList<T> neighbors = null;
                  public Node() {}
                  public Node(T data) : this(data, null) {}
                  public Node(T data, NodeList<T> neighbors)
                  {
                        this.data = data;
                        this.neighbors = neighbors;
                  }

                  public T Value
                  {
                        get
                        {
                              return data;
                        }
                        set
                        {
                              data = value;
                        }
                  }
                  protected NodeList<T> Neighbors

                  {
                        get
                        {
                              return neighbors;
                        }
                        set
                        {
                              neighbors = value;
                        }}}
```

```
        public class NodeList<T> : CollectionNode<T>
        {
              public NodeList() : base() { }

              public NodeList(int initialSize)
              {
                    // Add the specified number of items
                    for (int i = 0; i < initialSize; i++)
                          base.Items.Add(default(Node<T>));
              }

              public Node<T> FindByValue(T value)
              {
                    // search the list for the value
                    foreach (Node<T> node in Items)
                          if (node.Value.Equals(value))
                                return node;

              // if we reached here, we didn't find a matching node
                    return null;
              }
        }
        public class Arrays
        {
              public byte[,] C;

              public Arrays()
              {
                    C=new byte[5,5];} }}
```

8.2.20 Scenario.cs

```
using System;
using System.Collections.Generic;
using System.Text;
namespace RiskManagement.RiskUtils
{
    public class Scenario
    {
        int[] _disasters;
        public Scenario(int[] disasters)
        {
            _disasters = disasters;
            ComponentsThreatsTable.Create();
        }

        public int[] Execute()
        {
//Δημιουργία Διανύσματος Σεναρίων

int[] returnVector = new
int[ComponentsThreatsTable.Components.Count];
            for (int j = 0; j <
ComponentsThreatsTable.Components.Count; j++)
            {
                for (int i = 0; i < _disasters.Length; i++)
                {
                    int disasterIndex = _disasters[i];
```

```
                    returnVector[j] = returnVector[j] +
ComponentsThreatsTable.S[j, disasterIndex];
                }
                if (returnVector[j] > 1)
                    returnVector[j] = 1;
            }
            return returnVector;
        } }}
```

8.2.21 DisastersRecoveriesTable.cs

```
using System;
using System.IO;
using System.Collections.Generic;
using System.Text;
using System.Data;
using RiskManagement.DataAccessLayer;
using RiskManagement.BusinessLayer.Core;

//Κώδικας αντιστοίχισης καταστροφών με Ανακτήσης

namespace RiskManagement.RiskUtils
{
    public class DisastersRecoveriesTable
    {
        public static int[][] R;
        public static Dictionary<int, int> Disasters;
        public static Dictionary<int, int> Recoveries;
        public static int n;
        public static int m;
        private static int[] _recoveries;

        static DisastersRecoveriesTable()
        {
            Recoveries = new Dictionary<int, int>();
            Disasters = new Dictionary<int, int>();
           // Create();
        }

        public static int[][] SelectDisasters(int[] ids)
        {
            Create(ids);
            return R;
        }

        public static int LocateDisaster(int dbID)
        {
            return Disasters[dbID];
        }

        public static int LocateRecovery(int dbID)
        {
            return Recoveries[dbID];        }


        public static int GetRecoveryIdDB(int recoveryIndex)
        {

            return _recoveries[recoveryIndex];
        }
```

```
        public static void Create(int[] ids)
        {
            string selection = "";
            if (ids.Length > 0)
            {
                selection = " where dis_id in (";
                foreach (int id in ids)
                {
                    selection = selection + id.ToString() + ",";

                }
   selection = selection.Substring(0, selection.Length - 1) + ")"; }
//Αντιστοίχιση των Δεδομένων Καταστροφών - ανακτήσεων στη Βάση
Δεδομένων

int recoveriesCount = (int)DBManager.DataAccesor.ReadValue(@"Select
count(dis_id) From Recoveries
Inner Join DisasterRecoveries on  drc_rec_id=rec_id
Inner join Disasters on dis_id=drc_dis_id " + selection+
" group by dis_id");
            m = recoveriesCount;
            string query = "Select count(*) From Disasters";
int disastersCount = (int)DBManager.DataAccesor.ReadValue(query+"
"+selection);
           n = disastersCount;
            DisastersRecoveriesTable.Recoveries.Clear();
            DisastersRecoveriesTable.Disasters.Clear();
            Data data = DBManager.DataAccesor.read(@"Select * from
Recoveries
Inner Join DisasterRecoveries on  drc_rec_id=rec_id
Inner join Disasters on dis_id=drc_dis_id " + selection);
int dim = (disastersCount > recoveriesCount) ? disastersCount :
recoveriesCount;
            int[][] C = new int[dim][];
            for (int k = 0; k < dim; k++)
                C[k] = new int[dim];

            _recoveries = new int[data.Rows.Count];

            int num = 0;
            foreach (DataRow row in data.Rows)
            {
                Recoveries.Add((int)row["rec_id"], num);
                _recoveries[num] = (int)row["rec_id"];
                num++;
            }

data = DBManager.DataAccesor.read("Select * from Disasters
"+selection);
            num = 0;
            foreach (DataRow row in data.Rows)
            {
                Disasters.Add((int)row["dis_id"], num++);
            }

data = DBManager.DataAccesor.read(@"Select
drc_rec_id,drc_dis_id,rec_cost from DisasterRecoveries
Inner Join Recoveries On drc_rec_id=rec_id
Inner join Disasters on dis_id=drc_dis_id "+selection);
            for (int i = 0; i < data.Rows.Count; i++)
```

```
            {
                int disasterIdDB = (int)data.Rows[i]["drc_dis_id"];
                int disasterIndex=Disasters[disasterIdDB];
                int recoveryIdDB = (int)data.Rows[i]["drc_rec_id"];
                int recoveryIndex =Recoveries[recoveryIdDB];
                recoveryIndex = recoveryIndex % disastersCount;
                int CostValue = (int)data.Rows[i]["rec_cost"];
                C[disasterIndex][recoveryIndex] = CostValue;
            }
            //Pass the filepath and filename to the StreamWriter
Constructor
            StreamWriter sw = new StreamWriter("C:\\matrix.txt");
           for (int i = 0; i < dim; i++)
            {
                for (int j = 0; j < dim; j++)
                {
                    sw.Write(C[i][j].ToString() + " ");
                    if (C[i][j] == 0)
                        C[i][j] = Int32.MaxValue;
                }
                sw.WriteLine();
            }
            R = C;
            //Close the file
            sw.Close();
        } }}
```

8.2.22 ComponentsthreatsTable.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Data;
using RiskManagement.DataAccessLayer;
using RiskManagement.BusinessLayer.Core;

namespace RiskManagement.RiskUtils
{

//Κώδικας αντιστοίχισης Απειλών με υποσυστήματα με τη βοήθεια της
δομής Dictionary
    public class ComponentsThreatsTable
    {
        public static int[,] S;
        public static Dictionary<int, int> Disasters;
        public static Dictionary<int, int> Components;
        public static int n;
        public static int m;
        static ComponentsThreatsTable()
        {
            Components = new Dictionary<int, int>();
            Disasters = new Dictionary<int, int>();
            Create();
        }

        public static int LocateDisaster(int dbID)
        {

            return Disasters[dbID];
        }
```

```
        public static int LocateComponent(int dbID)
        {
            return Components[dbID];
        }

        public static void Create()
        {
int compCount =
RiskManagement.BusinessLayer.Application.System.Components.Count;
int disastersCount =
RiskManagement.BusinessLayer.Application.System.Disasters.Count;
            ComponentsThreatsTable.Components.Clear();
            ComponentsThreatsTable.Disasters.Clear();
            int num = 0;
            foreach (BusinessLayer.Component component in
RiskManagement.BusinessLayer.Application.System.Components.Items)
            {
                Components.Add(component.ID, num++);
            }
            num = 0;
foreach (BusinessLayer.Disaster disaster in
RiskManagement.BusinessLayer.Application.System.Disasters.Items)
            {
                Disasters.Add(disaster.ID, num++);
            }
            int[,] C = new int[compCount, disastersCount];
            Data data = DBManager.DataAccesor.read("Select * from
DisasterThreats");
            for (int i = 0; i < data.Rows.Count; i++){
  int componentIndex = Components[(int)data.Rows[i]["dtc_com_id"]];
  int disasterIndex = Disasters[(int)data.Rows[i]["dtc_dis_id"]] ;
                C[componentIndex, disasterIndex] = 1;
            }
            RiskUtils.ComponentsThreatsTable.S = C;
        } }}
```

8.2.23 TreeView.cs

```
using System;
using System.ComponentModel;
using System.Collections;
using System.Collections.Generic;
using System.Text;
using System.Windows.Forms;
namespace MyTreeView
{
//Κατασκευή Δένδρου Παρουσίασης δομής Συστήματος

    public class MyTreeView : System.Windows.Forms.TreeView
    {
 private Dictionary<Object, System.Windows.Forms.TreeNode>
_nodesDictionary;
        ArrayList prevKeys;
        Queue queue;

        public MyTreeView()
            : base()
        {
 _nodesDictionary = new Dictionary<Object,
System.Windows.Forms.TreeNode>();
```

```
 queue = new Queue();
        }
        protected override void
OnNodeMouseClick(TreeNodeMouseClickEventArgs e)
        {
            _selectedNode = e.Node;
            base.OnNodeMouseClick(e);
        }
        private System.Windows.Forms.TreeNode _selectedNode;
        private System.Windows.Forms.TreeNode _rootNode;
        public System.Windows.Forms.TreeNode RootNode
        {
            get
           {
                return _rootNode;
            }
            set
            {
                _rootNode = value;
                if (value != null)
                {
                    Nodes.Clear();
                    Nodes.Add(_rootNode);
                         } } }

        private bool _isRefreshing;
        public void ClearNodes()        {

_nodesDictionary.Clear();
        }

      public System.Windows.Forms.TreeNode GetObjectNode(object obj)
        {
            if (_nodesDictionary.ContainsKey(obj))
                return _nodesDictionary[obj];
            else
                return null;
        }

        public void Update(Object obj, string name)
        {
            if (_nodesDictionary.ContainsKey(obj))
                _nodesDictionary[obj].Text = name;
        }

        public void PopulateTreeView(IEnumerable collection)
        {
          prevKeys = new ArrayList(_nodesDictionary.Keys);
            _isRefreshing = _nodesDictionary.Keys.Count > 0;
            IEnumerator enumerator = collection.GetEnumerator();
            while (enumerator.MoveNext())
            {

       if (((IHierarchyData)enumerator.Current).GetParent() == null)

           queue.Enqueue(enumerator.Current);
            }
            while (queue.Count > 0)
            {
                object item = queue.Dequeue();
```

```
                if (prevKeys.Contains(item))
                    prevKeys.Remove(item);
  object parentItem = ((IHierarchyData)item).GetParent();
  IEnumerable childItems = ((IHierarchyData)item).GetChildren();
              IEnumerator enumerator2 = childItems.GetEnumerator();
                while (enumerator2.MoveNext())
                {
                    queue.Enqueue(enumerator2.Current);
                }

                if (_nodesDictionary.ContainsKey(item) == false)
//Προσθήκη υποσυστημάτων στην δενδροειδή μορφή                  {
System.Windows.Forms.TreeNode node = new
System.Windows.Forms.TreeNode();
                    node.Text = item.ToString();
                    node.Tag = item;
                    if (parentItem == null)
                    {
                        RootNode.Nodes.Add(node);
                        RootNode.Expand();
                        if (_isRefreshing == true)
                        {
                            _selectedNode = RootNode;
                       }
                    }
                    else
                    {
                        System.Windows.Forms.TreeNode parentNode;

  parentNode = _nodesDictionary[((IHierarchyData)parentItem).Item];
                        parentNode.Nodes.Add(node);
                        if (_isRefreshing == true)
                        {
                            parentNode.Expand();
                            _selectedNode = parentNode;
                        }
                    }
                    _nodesDictionary.Add(item, node);
                }
                else
                {
                    _nodesDictionary[item].Text = item.ToString();
                    _nodesDictionary[item].Tag = item;
                }
            }

            this.SelectedNode = _selectedNode;
            foreach (object obj in prevKeys)
            {
                TreeNode deleted = _nodesDictionary[obj];
                if (deleted.Parent != null)
                    deleted.Parent.Nodes.Remove(deleted);
                _nodesDictionary.Remove(obj);
            }} }}
```

8.2.24 QueryObjects.cs

```
using System;
using System.Collections.Generic;
```

```
using System.Text;
using System.Data;

namespace RiskManagement.DataAccessLayer
{

//Συναρτήσεις χειρισμού SQL εντολών και Query στην Βάση
 public class QueryObject
    {
        public string sql;
        public IDbCommand selectCommand;
        public IDbCommand insertCommand;
        public IDbCommand updateCommand;
        public IDbCommand deleteCommand;
    }}
```

8.2.25 KeyGenerator.cs

```
using System;
using System.Collections.Generic;
using System.Text;
namespace RiskManagement.DataAccessLayer
{
    public class KeyGenerator
    {
        public static Key getNext(string table)
        {
string sql = "SELECT tbl_next_key FROM _tables " +
                  "WHERE tbl_name='" + table + "'";


int value=(int) DBManager.DataAccesor.ReadValue(sql);
sql = "Update _tables SET tbl_next_key = " +(value+1)+" "+
                              "WHERE tbl_name='" + table + "'";
            DBManager.DataAccesor.execute(sql);
            Key key =new Key(value);
            return  key;
        } }}
```

8.2.26 Key.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Data;

namespace RiskManagement.DataAccessLayer
{
    public class Key
    {
        private Object[] _fields;

        public Key(DataColumn[] columns)
        {
            _fields = new object[columns.Length];

        }
        public Key(Object[] fields)
        {
```

```
            this._fields = fields;        }

        public Key(object field)
        {
            this._fields = new object[] { field };
        }

        public object this[int index]
        {
            get { return _fields[index]; }
            set { _fields[index] = value; }
        }

        public bool equals(Object obj)
        {
            return true;
        } }}
```

8.2.27 DBManager.cs

```
using System;
using System.Data;
using System.Data.Odbc;
using System.Data.SqlClient;
using System.Data.OleDb;
namespace RiskManagement.DataAccessLayer{

      public interface IDbKernel
      {
            IDbConnection createConnection();
            IDbCommand createCommand();
            IDbDataAdapter createDbAdapter();
      }

    public class DBManager
    {

        //singleton pattern
        private static IDbKernel dbKernel;
        private static _DataAccessor dataAccessor;

        public static void Initialize(IDbKernel kernel)
        {
            dbKernel = kernel;
            dataAccessor = new _DataAccessor();
        }

        public static IDataAccessor DataAccesor
        {
            get
            {
                return dataAccessor;
            }
        }
            private class _DataAccessor :IDataAccessor
        {
//Σύνδεση με την βάση δεδομένων

            public _DataAccessor() {
                conn = DBManager.dbKernel.createConnection();
```

```
                cmd = DBManager.dbKernel.createCommand();
                cmd.Connection = conn;
                adapter = DBManager.dbKernel.createDbAdapter();

                adapter.SelectCommand = cmd;
            }
//διάβασμα sql query
           public Data read(String sqlStatement)
            {
                cmd.CommandText = sqlStatement;
                DataSet ds = new DataSet();
                adapter.FillSchema(ds, SchemaType.Source);
                adapter.Fill(ds);
                return new Data(ds);
            }
            public object ReadValue(string sqlStatement)
            {
                cmd.Connection.Open();
                cmd.CommandText = sqlStatement;
                object value =cmd.ExecuteScalar();
                cmd.Connection.Close();
                return value;
            }

//εκτέλεση query

       public void execute(String sqlStatement)
            {
                try
                {
                    cmd.CommandText = sqlStatement;
                    cmd.Connection.Open();
                    cmd.ExecuteNonQuery();
                    cmd.Connection.Close();}

                catch (Exception ex)
                {
                    System.Windows.Forms.MessageBox.Show(ex.Message);
                } }
            public void update(String table,
                        String[][] rowSelectionFilter,
                        String[][] data)
            { }
    public void delete(String table, String[][] rowSelectionFilter)
            { }
            public IDbConnection conn;
            public IDbCommand cmd;
            public IDbDataAdapter adapter;
            } }}
```

8.2.28 Data.cs

```
using  System;
using System.Collections.Generic;
using System.Collections;
using System.Data;
using System.Data.Sql;
using RiskManagement;

namespace RiskManagement.DataAccessLayer
```

```
{
    public class Data :IEnumerable
    {
        public DataSet ds;
        Dictionary<Key, DataItem> index =
           new Dictionary<Key, DataItem>();
        private DataColumn[] _primaryKey;
        public Data(DataSet ds)
        {
            this.ds = ds;
            _primaryKey = ds.Tables[0].PrimaryKey;
            DataRowCollection rows = ds.Tables[0].Rows;
            foreach (DataRow row in rows)
            {
                Key key = new Key(_primaryKey);
                for (int i = 0; i < _primaryKey.Length; i++)
                    key[i] = row[_primaryKey[i]];
                index.Add(key, new DataItem(row,this));
            }
        }
        public Data.DataItemCollection selectAll()
        {
            DataRow[] rows = new DataRow[ds.Tables[0].Rows.Count];
            ds.Tables[0].Rows.CopyTo(rows,0);
            Data.DataItemCollection selection = new
Data.DataItemCollection(rows,this);
            return selection;
        }
       public DataItemCollection select(string filterExpression,
string sort)        {
   DataRow[] rows = ds.Tables[0].Select(filterExpression, sort);
DataItemCollection selection = new DataItemCollection(rows,this);
            return selection;
        }

        public DataItem getDataItem(int item)
        {
            return new DataItem(ds.Tables[0].Rows[item],this);
        }

        public DataRowCollection Rows
        {
            get
            {
                return ds.Tables[0].Rows;
            }
        }

        public Data.DataItem getByID(int ID)
        {
            return new
Data.DataItem(ds.Tables[0].Rows.Find(ID),this);
        }
        public virtual IEnumerator GetEnumerator()
        {
            return new Data.Enumerator(this);
        }
        private class Enumerator : IEnumerator
        {
            private Data outer;
            private int currentIndex = -1;
```

```
            internal Enumerator(Data outer)
            {
                this.outer = outer;        }

public object Current
            {
                get
                {
                    if (currentIndex ==
outer.ds.Tables[0].Rows.Count)
                      throw new InvalidOperationException();
 return new DataItem(outer.ds.Tables[0].Rows[currentIndex],outer);
                }
            }
            public bool MoveNext()
            {
              if (currentIndex > outer.ds.Tables[0].Rows.Count)
                    throw new InvalidOperationException();
                return ++currentIndex <
outer.ds.Tables[0].Rows.Count;
            }

            public void Reset()
            {
                currentIndex = -1;
            } }
        public class DataItem
        {
            private DataRow _dataRow;
            private Data _data;

            public DataItem(DataRow dataRow,Data data)
            {
                _dataRow=dataRow;
                _data = data;

            }
            public object this[int i]
            {
                get
                {
                    object obj = _dataRow[i];
                    if (obj != DBNull.Value)
                        return obj;
                    else
                       return null;
                }
                set
                {
                    _dataRow[i] = value;

                }
            }

            public object this[string columnName]
            {
                get
                {
                    object obj = _dataRow[columnName];
                    if (obj != DBNull.Value)
                        return obj;
```

```
                    else
                        return null;
                }

                set
                {
                    _dataRow[columnName] = value;

               }}
            public override bool Equals(object obj)
            {
                DataColumn[] primaryKey = _data._primaryKey;

      //iterate over primary key kolumns and check foe equality
              for (int i = 0; i < primaryKey.Length; i++)
                {
                    object thisValue =
_dataRow[primaryKey[i].ColumnName];
object objValue = (obj as
DataItem)._dataRow[primaryKey[i].ColumnName];
                    if (thisValue.Equals(objValue) ==false )
                        return false;
                }
                return true;
            }}

        public class DataItemCollection
        {
            private DataRow[] _dataRows;
            private DataItem[] _dataItems;

            public DataItemCollection(DataRow[] rows,Data data)
            {
                _dataRows = rows;
                _dataItems = new DataItem[_dataRows.Length];
                for (int i = 0; i < _dataRows.Length; i++)
                    _dataItems[i] = new DataItem(rows[i],data);
            }
            public DataItem this[int index]
            {
                get { return _dataItems[index]; }
            }
            public int Length
            {
                get
                {
                    return _dataItems.Length;
                }}}}
    public interface IDataAccessor
    {
        Data read(String SqlStatement);
        object ReadValue(string sqlStatement);
      void execute(String SqlStatement);
void update(String table,
                    String[][] rowSelectionFilter,
                    String[][] data);
       void delete(String table, String[][] rowSelectionFilter);
    }}
```

8.2.29 CustomTreeView.cs

```
using System;
using System.Windows.Forms;
using System.Collections.Generic;
using System.Text;

namespace CustomTreeView
{
    public class CustomTreeView<Item, Collection> : TreeView
        where Collection : IEnumerable<Item>
        where Item : IHierarchyData<Item>

    {
        private IDictionary<Item, System.Windows.Forms.TreeNode>
_nodesDictionary;
        Queue<Item> queue;

        public CustomTreeView():base()
        {
            _nodesDictionary = new Dictionary<Item,
System.Windows.Forms.TreeNode>();
            queue = new Queue<Item>();
        }

        public  void PopulateTreeView(Collection collection)
        {
            IEnumerator<Item> enumerator =
collection.GetEnumerator();
            while (enumerator.MoveNext())
            {

                if (enumerator.Current.GetParent() == null)
                    queue.Enqueue(enumerator.Current);
            }

            while (queue.Count > 0)
            {
                Item item = queue.Dequeue();
                IHierarchyData<Item> parentItem = item.GetParent();
                IEnumerable<Item> childItems = item.GetChildren();
         IEnumerator<Item> enumerator2 = childItems.GetEnumerator();
                while (enumerator2.MoveNext())
                {
                    queue.Enqueue(enumerator2.Current);
                }
System.Windows.Forms.TreeNode node = new
System.Windows.Forms.TreeNode();
                node.Text = item.ToString();
                node.Tag = item;
                if (parentItem == null)
                {
                    Nodes.Add(node);
                }
                else
                {
                    System.Windows.Forms.TreeNode parentNode;
                    parentNode = _nodesDictionary[parentItem.Item];
                    parentNode.Nodes.Add(node);
```

```
                }
                _nodesDictionary.Add(item, node);
            } } }
    public interface IHierarchyData<T>
    {
        IEnumerable<T> GetChildren();
        IHierarchyData<T> GetParent();
        bool HasChildren { get; }
        T Item { get; }
    }
    public interface IHierarchicalEnumerable<T>
    {
        IEnumerator<T> GetEnumerator();

    }}
```

8.2.30 CustomControl1.cs

```
using System;
using System.ComponentModel;
using System.Collections;
using System.Collections.Generic;
using System.Text;
using System.Windows.Forms;
namespace Controls
{
    public class MyTreeView1 : System.Windows.Forms.TreeView
    {
private Dictionary<Object, System.Windows.Forms.TreeNode>
_nodesDictionary;
ArrayList prevKeys;

        Queue queue;
        public MyTreeView1()
            : base()
        {
 _nodesDictionary = new Dictionary<Object,
System.Windows.Forms.TreeNode>();
            queue = new Queue();
        }
protected override void OnNodeMouseClick(TreeNodeMouseClickEventArgs
e)
        {
            _selectedNode = e.Node;
           base.OnNodeMouseClick(e);
        }
        private System.Windows.Forms.TreeNode _selectedNode;
        private System.Windows.Forms.TreeNode _rootNode;
        public System.Windows.Forms.TreeNode RootNode
        {
            get
            {
                return _rootNode;
            }
           set
            {
                _rootNode = value;
                if (value != null)
                {
                    Nodes.Add(value);
```

```
                } }         }
        private bool _isRefreshing;
        public void ClearNodes()

        {
            _nodesDictionary.Clear();
        }
       public System.Windows.Forms.TreeNode GetObjectNode(object obj)
        {
            if (_nodesDictionary.ContainsKey(obj))
                return _nodesDictionary[obj];
            else
                return null;
        }
        public void Update(Object obj, string name)
        {
            if (_nodesDictionary.ContainsKey(obj))
                _nodesDictionary[obj].Text = name;
        }
        public void PopulateTreeView(IEnumerable collection)
        {
            prevKeys = new ArrayList(_nodesDictionary.Keys);
            _isRefreshing = _nodesDictionary.Keys.Count > 0;
            IEnumerator enumerator = collection.GetEnumerator();
    while (enumerator.MoveNext())
            {
      if (((IHierarchyData)enumerator.Current).GetParent() == null)
                    queue.Enqueue(enumerator.Current);
            }
            while (queue.Count > 0)
            {
                object item = queue.Dequeue();

                if (prevKeys.Contains(item))
                    prevKeys.Remove(item);
  object parentItem = ((IHierarchyData)item).GetParent();
  IEnumerable childItems = ((IHierarchyData)item).GetChildren();
              IEnumerator enumerator2 = childItems.GetEnumerator();
                while (enumerator2.MoveNext())
                {
                    queue.Enqueue(enumerator2.Current);
                }
                if (_nodesDictionary.ContainsKey(item) == false)
                {
System.Windows.Forms.TreeNode node = new
System.Windows.Forms.TreeNode();
                node.Text = item.ToString();
                    node.Tag = item;
                    if (parentItem == null)
                    {
                        RootNode.Nodes.Add(node);
                        RootNode.Expand();
                        if (_isRefreshing == true)
                        {
                            _selectedNode = RootNode;
                        }
                    }
                    else
                    {
                        System.Windows.Forms.TreeNode parentNode;
 parentNode = _nodesDictionary[((IHierarchyData)parentItem).Item];
```

```
                        parentNode.Nodes.Add(node);
                        if (_isRefreshing == true)
                        {

                            parentNode.Expand();
                            _selectedNode = parentNode;
                        }
                    }
                    _nodesDictionary.Add(item, node);
                }
                else
                {
                    _nodesDictionary[item].Text = item.ToString();
                    _nodesDictionary[item].Tag = item;
                }
            }
            this.SelectedNode = _selectedNode;
            foreach (object obj in prevKeys)
            {
                TreeNode deleted = _nodesDictionary[obj];
                if (deleted.Parent != null)
                    deleted.Parent.Nodes.Remove(deleted);
                _nodesDictionary.Remove(obj);
            } }}}
```

8.2.31 LinearAssignmentProblem.java

Κώδικας υλοποίησης Προβλήματος Κατανομής σύμφωνα με τον αλγόριθμο του Kuhn.

```
package Combinatorics;
// This is an implementation of the Hungarian algorithm
// that solves the assignment problem in polynomial time.
// Taken from a website I found through wikipedia's
// entry on the "Hungarian algorithm".

// then replaced doubles with ints.

import com.ryanm.util.geom.*;
public final class LinearAssignmentProblem{
    int[][] cost;    //COST MATRIX
    int numCol;
    int numRow;
      public LinearAssignmentProblem(int[][] cost)
      {
        numCol=cost[0].length;
        numRow=cost.length;
        if (numRow>numCol) throw new
IllegalArgumentException("Expecting more rows than columns");
        this.cost=cost;
    }//method
    public byte[][] munkres(){
        Munkres m=new Munkres();
        return m.solve();
    }//method
    //MUNKRES MARKINGS
    private static byte NOT_MARKED=0;
    private static byte STARRED_ZERO=1;
    private static byte PRIMED_ZERO=2;
```

```
    public class Munkres{
        byte[][] mark;        //MARK ZEROS
        int[] rowIsCovered;
        int[] colIsCovered;
        public byte[][] solve(){
            //MAKE THE MARKING ARRAY
            mark=new byte[numRow][];

            for(int i=0;i<numRow;i++){
                mark[i]=new byte[numCol];
            }//for
            rowIsCovered=new int[numRow];
            colIsCovered=new int[numCol];
            reduceRows();
            starSomeZeros();
            while(true){
                if (isDone()) break;;
                pair p=findAndPrimeUncoveredZero();
                new step5().main(p);
            }//loop
            return mark;
        }//method
    public void reduceRows(){
        int minval;
        for(int i=0;i<numRow;i++){
            minval=cost[i][0];
            for(int j=1;j<numCol;j++){
                if (minval>cost[i][j]){
                    minval=cost[i][j];
                }// if;
            }//for
            for(int j=0;j<numCol;j++){
                cost[i][j]-=minval;
            }//for
        }//for
    }// stepone;
    public void starSomeZeros(){
    //ONLY ONE STAR ALLOWED PER ROW AND ONE PER COLUMN

 boolean[] colIsCovered=new boolean[numCol];
A:        for(int i=0;i<numRow;i++){
            for(int j=0;j<numCol;j++){
                if (!colIsCovered[j] && cost[i][j]==0){
                    mark[i][j]=STARRED_ZERO;
                    colIsCovered[j]=true;
                    continue A;        //DO NOT FINISH THIS ROW
                }// if;
            }//for
        }//for
    }// steptwo;
    public boolean isDone(){
    //RETURN true IF DONE
    //ALSO COVER ROWS AND COLUMNS IF NOT DONE
        int count=0;
        for(int j=0;j<numCol;j++){
            for(int i=0;i<numRow;i++){
                if (mark[i][j]==STARRED_ZERO){
                    colIsCovered[j]=1;
                    count++;
                    break;
                }// if;
```

```
            }//loop;
        }//loop;
        if (count>=numRow) return true;
        return false;
    }// stepthree;
    public pair findAndPrimeUncoveredZero(){
    //FIND A UNCOVERED ZERO AND PRIME IT

s4:         while(true){
            pair p=findUncoveredZero();
            if (p==null){
            //CAN'T FIND A ZERO
                makeSomeZeros();        //MAKE IT
                continue s4;        //TRY AGAIN
            }//endif

            mark[p.row][p.col]=PRIMED_ZERO;

            //find_star_in_row
            for(int j=0;j<numCol;j++){
                if (mark[p.row][j]==STARRED_ZERO) {
                    rowIsCovered[p.row]=1;
                    colIsCovered[j]=0;
                    continue s4;
                }// if;
            }//for

            return p;
        }//for
    }// stepfour;

    public pair findUncoveredZero(){
        for(int i=0;i<numRow;i++){
            for(int j=0;j<numCol;j++){
                 if (cost[i][j]==0 && rowIsCovered[i]==0 &&
colIsCovered[j]==0){
                     return new pair(i, j);
                 }//if;
             }//for

        }//for
        return null;
    }//method
    public void makeSomeZeros(){
        //find_smallest
        int minval=Integer.MAX_VALUE;
        for(int i=0;i<numRow;i++){
            if (rowIsCovered[i]==1) continue;
            for(int j=0;j<numCol;j++){
             if (colIsCovered[j]==0){
                 if (minval>cost[i][j]) minval=cost[i][j];
             }// if;
            }//for
        }//for

        for(int i=0;i<numRow;i++){
            for(int j=0;j<numCol;j++){
                if (rowIsCovered[i]==1) cost[i][j]+=minval;
                if (colIsCovered[j]==0) cost[i][j]-=minval;
            }//for
        }//for
```

```
    }//method

   class step5{
    //Construct a series of alternating primed and starred zeros as
follows.

        pair[] path=new pair[numRow*2];

        public int find_star_in_col(int c){
            for(int i=0;i<numRow;i++){
                if (mark[i][c]==STARRED_ZERO) {
                    return i;
                }//if;
            }//for
            return -1;
        }// find_star_in_col;

        public int find_prime_in_row(int r){
            for(int j=0;j<numCol;j++){
                if (mark[r][j]==PRIMED_ZERO){
                    return j;
                }// if;
            }//for
            return -1;
        }// find_prime_in_row;

        public void clear_covers(){
            for(int i=0;i<numRow;i++){
                rowIsCovered[i]=0;
                colIsCovered[i]=0;              //SQUARE ASSUMPTION
MADE HERE
            }//for
        }// clear_covers;
        public void clear_primes(){
            for(int i=0;i<numRow;i++){
                for(int j=0;j<numCol;j++){
                    if (mark[i][j]==PRIMED_ZERO){
                        mark[i][j]=NOT_MARKED;
                    }//if;
                }//for

            }//for
        }// erase_primes;
        public void main(pair p){
            int count=0;
            path[count]=p;
            while(true){
                int r=find_star_in_col(path[count].col);
                if (r==-1) break;
                count++;
                path[count]=new pair(r, path[count-1].col);
                int c=find_prime_in_row(path[count].row);
                count++;
                path[count]=new pair(path[count-1].row, c);
            }//for
            //convert_path(){
            for(int i=0;i<=count;i++){
                if (mark[path[i].row][path[i].col]==STARRED_ZERO){
                    mark[path[i].row][path[i].col]=NOT_MARKED;
                }else{
                    mark[path[i].row][path[i].col]=STARRED_ZERO;
```

```
                }// if;
            }//for
            clear_covers();
            clear_primes();
        }// stepfive;
    }//class
    public String toString(){

        StringBuffer output=new StringBuffer();
        for(int j=0;j<numCol;j++){
            output.append("\t").append(colIsCovered[j]);
        }//for
        output.append("\n");
        for(int i=0;i<numRow;i++){
            output.append(rowIsCovered[i]);
            for(int j=0;j<numCol;j++){
                output.append("\t").append(cost[i][j]);
                if (mark[i][j]==STARRED_ZERO){
                    output.append("*");
                }else if (mark[i][j]==PRIMED_ZERO){
                    output.append("'");
                }//endif
            }//for
            output.append("\n");
        }//for
        return output.toString();
    }//method
    }//class
}//class
class pair{
    int row;
    int col;
   public pair(int row, int col){
        this.row=row;
        this.col=col;
    }//method
}//class
```

8.2.32 Theats.cs

Κώδικας διαχείρισης απειλών. Εισαγωγή – επιλογή απειλών με βάση τις προτιμήσεις του χρήστη.

```
using System;
using System.Collections.Generic;
using System.Text;
using RiskManagement;
using RiskManagement.DataAccessLayer;
using RiskManagement.BusinessLayer.Core;


namespace RiskManagement.BusinessLayer
{
    public class Threat:BusinessBase<Threat>._Item
    {
        private string _name;
        public Threat()
            : base()
        {
```

```
        }

        public string Name
        {
            get
            {
                if (_name == null)
                    return this.DataItem["thr_name"].ToString();
                else
                    return _name;
            }
            set { _name = value; }        }

        public int ID
        {
            get { return (int)DataItem["thr_id"]; }
        }

        public string ToString()
        {
            return Name;
        }
    }
    public class Threats:BusinessBase<Threat>
    {
        public Threats()
            : base()
        { }
       protected override string SelectStatement
        {
            get { return "Select * from Threats"; }
        }
        private static  SqlString InsertStatement =new SqlString(
            @"INSERT INTO Threats (thr_id,thr_name)
                VALUES (#Key,'#Name');
              INSERT INTO ComponentsThreats(cth_com_id,cth_thr_id)
                VALUES ( #ComponentKey , #ThreatKey )");
        private static SqlString InsertExistingStatement = new
SqlString(
            @"INSERT INTO ComponentsThreats(cth_com_id,cth_thr_id)
                VALUES ( #ComponentKey , #ThreatKey )");

        private static SqlString DeleteStatement = new SqlString(
            @"DELETE FROM Threats
                WHERE  thr_id = #Key;");
            //  INSERT INTO ComponentsThreats(cth_com_id,cth_thr_id)
            //    VALUES ( #ComponentKey , #ThreatKey )");

        public static void Insert(Component component,Threat threat)
        {
            string ID=KeyGenerator.getNext("Threats")[0].ToString();
            InsertStatement.Parameters["Key"]=ID;

            InsertStatement.Parameters["Name"] = threat.Name;
InsertStatement.Parameters["ComponentKey"] =
component.ID.ToString();;
            InsertStatement.Parameters["ThreatKey"] = ID;
            DBManager.DataAccesor.execute(InsertStatement.Text);
        }
```

```
        public static void InsertExistingThreat(Component component,
Threat threat)
        {
            InsertExistingStatement.Parameters["ComponentKey"] =
component.ID.ToString();
            InsertExistingStatement.Parameters["ThreatKey"] =
threat.ID.ToString();
             DBManager.DataAccesor.execute(InsertExistingStatement.Text);
        }
        public static void DeleteThreat(Threat threat)
        {
            DeleteStatement.Parameters["Key"] = threat.ID.ToString();
            DBManager.DataAccesor.execute(DeleteStatement.Text);
        } }}
```

8.2.33 System.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using RiskManagement.DataAccessLayer;
namespace RiskManagement.BusinessLayer
{
    public class System
    {
        public System()
        {
            _disasters = new _Disasters(this);
            _scenarios = new _Scenarios(this);
            _components = new _Components(this);
        }

        private int _ID;
        private _Disasters _disasters;
        private _Scenarios _scenarios;
        private _Components _components;
        public int ID
        { get { return _ID; } }
        public string Name
        {
            get
            {
//επιλογή συστήματος απο το χρήστη για μελέτη

return
(string)DataAccessLayer.DBManager.DataAccesor.ReadValue("Select
sys_name FROM Systems where sys_id=" + _ID);
            } }
        public Disasters Disasters
        {
            get
            {

                _disasters.FetchAll();
                return _disasters;
            }
        }
        public Scenarios Scenarios
        {
            get
```

```
            {
                _scenarios.FetchAll();
                return _scenarios;
            }
        }
        public Components Components
        {
            get
            {
                _components.FetchAll();
                return _components;
            }
        }
       public void LoadSystem(int sysid)
        {
            _ID = sysid;
        }
//Δημιουργία νέου συστήματος

        public int CreateSystem(string name)
        {
            Key key = KeyGenerator.getNext("Systems");
            DataAccessLayer.DBManager.DataAccesor.execute("INSERT
into Systems (sys_id,sys_name) values
("+key[0].ToString()+",'"+name+"')");
            return (int)key[0];
        }
        private class _Disasters : Disasters
        {
            public _Disasters(System system)
            {
                _system = system;
            }

            private System _system;

            protected override string SelectStatement
            {
get { return @"Select * from Disasters
  INNER JOIN DisasterThreats ON dis_id = dtc_dis_id
  INNER JOIN Components ON com_id = dtc_com_id
 WHERE com_sys_id = "+ _system._ID.ToString(); }
            } }

        private class _Scenarios : Scenarios
        {
            public _Scenarios(System system)
            {
                _system = system;
            }

            private System _system;

            protected override string SelectStatement
            {
                get
                {
return @"select Distinct sce_id,sce_name,vi from scenarios
inner join DisastersScenarios on dsc_sce_id = sce_id
inner join Disasters on dsc_dis_id = dis_id
inner join DisasterThreats on dtc_dis_id= dis_id
```

```
inner join Components on com_id= dtc_com_id
where com_sys_id =" + _system._ID.ToString();
                } } }

        private class _Components : Components
        {
            public _Components(System system):base(system)
            {}
            protected override string SelectStatement
            {
                get
                {
                    if (System == null)

                        throw new Exception("System Not Loaded");
                    return @"SELECT * FROM Components
                         WHERE com_sys_id =" + System._ID.ToString();
                }}}
        private class _Threats : Threats
        {
            public _Threats(System system)
                : base()
            { _system = system; }

            private System _system;

            protected override string SelectStatement
            {
                get
                {
                    return @"SELECT * FROM Components
                         WHERE com_sys_id =" +
_system._ID.ToString();
                }}}}}
```

8.2.34 Recoveries.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Text;
using RiskManagement;
using RiskManagement.DataAccessLayer;
using RiskManagement.BusinessLayer.Core;
using RiskManagement.PresentationLayer;
namespace RiskManagement.BusinessLayer
{
    public class Recovery : BusinessBase<Recovery>._Item
    {
        private string _name;
        private int _cost;

        public Recovery()
            : base()
        {
        }
        public string Name
        {
            get
            {
```

```
                if (_name == null)
                    return this.DataItem["rec_name"].ToString();
                else
                    return _name;
            }
            set { _name = value; }
        }
        public int ID
        {
            get { return (int)DataItem["rec_id"]; }
        }
       public int Cost
        {

            get
            {
                if (_cost == 0)
                    return (int)this.DataItem["rec_cost"];
                else
                    return _cost;
            }
            set { _cost = value; }
        }

        public string ToString()
        {
            return Name;
        } }

    public class Recoveries : BusinessBase<Recovery>
    {
        public Recoveries()
            : base()
        {
        }
//Επιλογή ανακτήσεων και αντιστοίχιση τους με το κόστος τις καθεμίας
        protected override string SelectStatement
        {
            get { return "Select * from Recoveries"; }
        }
        private static SqlString InsertStatement = new SqlString(
@"INSERT INTO Recoveries(rec_id,rec_name,rec_cost)
VALUES (#Key,'#Name',#Cost);
INSERT INTO DisasterRecoveries(drc_rec_id,drc_dis_id)
VALUES ( #aek ,#DisasterKey )");
//Διαφραφή ανακτήσεων και αντιστοίχων κόστων
private static SqlString DeleteStatement = new SqlString(
@"DELETE FROM Recoveries WHERE  rec_id = #Key;");
public static void Insert(Recovery recovery,Disaster disaster)
 {
string ID = KeyGenerator.getNext("Recoveries")[0].ToString();
       InsertStatement.Parameters["Key"] = ID;
            InsertStatement.Parameters["aek"] = ID;

InsertStatement.Parameters["Name"] = recovery.Name;
InsertStatement.Parameters["DisasterKey"] = disaster.ID.ToString();
InsertStatement.Parameters["Cost"] = recovery.Cost.ToString();
DBManager.DataAccesor.execute(InsertStatement.Text);
Presentation_Layer.MainForm.Recoveries.Refresh();}
        public static Recovery FindById(int id)
        {
```

```
string stmt = "Select * from Recoveries where rec_id=" +
id.ToString();
            Data data = DBManager.DataAccesor.read(stmt);
            Recovery recovery = new Recovery();
            recovery.DataItem = data.getDataItem(0);
            return recovery;
        }
        public static void DeleteRecovery(Recovery recovery)
        {
DeleteStatement.Parameters["Key"] = recovery.ID.ToString();
DBManager.DataAccesor.execute(DeleteStatement.Text);}}}
```

8.2.35 Mapper.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using RiskManagement.DataAccessLayer;
namespace RiskManagement.BusinessLayer
{
 public abstract class AbstractMapper<T>
       where T:BusinessObject  {
        /*
**
**Den xreiazetai reflection
**
Type t = this.GetType();
Type t2 = t.BaseType;
Type[] typeParameters = t2.GetGenericArguments();
Type t3 = (Type)typeParameters.GetValue(0);
foreach (DataItem item in data)
{
  this.Add((T)Activator.CreateInstance(t3, new object[] { item }));
}
*/
    }}
```

8.2.36 HierarchicalBussinessBase.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using RiskManagement;
using RiskManagement.DataAccessLayer;
using RiskManagement.BusinessLayer.Core;
namespace RiskManagement.BusinessLayer
{
    public abstract class HierarchicalBusinessBase<T> :
BusinessBase<T>
where T : HierarchicalBusinessBase<T>._Item, new()
    {
public HierarchicalBusinessBase()
 {
 // SelectStatement = "SELECT * FROM Components WHERE com_sys_id=1";
 init();
 }
public override void init() {
            DataProvider = new Collection(this);
        }
```

```
        public  BusinessBase<T>.Collection Select(string
filterExpression)
        {
Data.DataItemCollection collection = _data.select(filterExpression,
"");
  return new Collection(collection, this);} }}
```

8.2.37 DomainStore.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using RiskManagement.DataAccessLayer;
using System.Reflection;
namespace RiskManagement.BusinessLayer
{
    public static class Singleton<T>
        where T : class
    {
        static Singleton()
        {
            _instance =typeof(T).InvokeMember(typeof(T).Name,
                                BindingFlags.CreateInstance |
                                BindingFlags.Instance |
                                BindingFlags.NonPublic,
                                null, null, null) as T;
        }

        public static readonly T _instance;
    }

    public class DomainManager
    {
        public static Components Components
        {
            get { return Singleton<Components>._instance; }
        }

        public static Threats Threats
        {
            get { return Singleton<Threats>._instance; }
        }

        //public static T Select<T>()
        //    where T : IBusinessCollection, new()
        //{
        //    Data data = DBManager.DataAccesor.read("Asd");
        //    T collection = new T();
        //    collection.SetData(data);
        //    return collection;
        //}
        //public static T Select<T>(string sql)
        //    where T : IBusinessCollection, new()
        //{
        //    Data data = DBManager.DataAccesor.read(sql);
        //    T collection = new T();
        //    collection.SetData(data);
        //    return collection;
        //}
    }}
```

8.2.38 Core.cs

```
using System;
using System.Collections;

namespace RiskManagement.BusinessLayer.Core
{
    public interface IBusinessObject
    {
    }
    public abstract class BusinessBase
    {
    }
    public abstract class BindingList
    {
    }
    public interface ITreeView<T>
    {
        T getRoot();
        T getNext();
        bool IsLeaf();
    }}
```

8.2.39 Application.cs

```
using System;
using System.Collections.Generic;
using System.Text;
namespace RiskManagement.BusinessLayer
{
    public class Application
    {
        public Application()
        {
            _system = new System();
            //_system.LoadSystem(1);
        }
        private static Application _instance;
        private System _system;
        private static Application Instance{
            get{

 if (_instance == null)
                    _instance = new Application();
                return _instance;
            }
        }
        public static System System
        { get { return Instance._system; } }}}
```

8.2.40 Disasters.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Text;
```

```
using RiskManagement;
using RiskManagement.DataAccessLayer;
using RiskManagement.BusinessLayer.Core;
using RiskManagement.PresentationLayer;

namespace RiskManagement.BusinessLayer
{
    public class Disaster : BusinessBase<Disaster>._Item
    {
        private string _name;
        public Disaster()
            : base()
        {
        }
        public string Name
        {
            get
            {
                if (_name == null)
                    return this.DataItem["dis_name"].ToString();
                else
                    return _name;
            }
            set { _name = value; }
        }
        public int ID
        {
            get { return (int)DataItem["dis_id"]; }
        }
        public override string ToString()
        {
            return Name;
        }
    }
    public class Disasters : BusinessBase<Disaster>
    {
       public Disasters() :base()
        {
        }
//Εισαγωγή Καταστροφών - αντιστοίχιση με απειλές
        protected override string SelectStatement
        {
            get { return "Select * from Disasters"; }
        }
        private static  SqlString InsertStatement =new SqlString(
            @"INSERT INTO Disasters(dis_id,dis_name)
                VALUES (#Key,'#Name');
              INSERT INTO
DisasterThreats(dtc_dis_id,dtc_thr_id,dtc_com_id)
                VALUES ( #aek ,#ThreatKey ,#ComponentKey )");

//διαγραφή καταστροφών
        private static SqlString DeleteStatement = new SqlString(

@"DELETE FROM Disasters
                WHERE  dis_id = #Key;");
        public static void Insert(Component component,Threat
threat,Disaster disaster)
        {
            string
ID=KeyGenerator.getNext("Disasters")[0].ToString();
```

```
            InsertStatement.Parameters["Key"]=ID;
            InsertStatement.Parameters["aek"] = ID;
            InsertStatement.Parameters["Name"] = disaster.Name;
InsertStatement.Parameters["ComponentKey"] = component.ID.ToString();
InsertStatement.Parameters["ThreatKey"] = threat.ID.ToString();
DBManager.DataAccesor.execute(InsertStatement.Text);

Presentation_Layer.MainForm.Disasters.Refresh();
        }
        public static void DeleteDisaster(Disaster disaster)
        {
 DeleteStatement.Parameters["Key"] = disaster.ID.ToString();
            DBManager.DataAccesor.execute(DeleteStatement.Text);
        }

        public void  SelectAll()
        {
            FetchAll();} }}
```

8.2.41 Scenarios.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Text;
using RiskManagement;
using RiskManagement.DataAccessLayer;
using RiskManagement.BusinessLayer.Core;
using RiskManagement.PresentationLayer;
namespace RiskManagement.BusinessLayer
{
    public class Scenario : BusinessBase<Scenario>._Item
    {
        private string _id;
        private string _name;
        private int _vi;

//Επιλογή Καταστροφικού Σεναρίου

private static SqlString SelectDisastersStatement = new SqlString(
@"SELECT * FROM Disasters INNER JOIN DisastersScenarios on
dsc_dis_id=dis_id WHERE dsc_sce_id=#ScenarioID");
        public Scenario()
            : base()
        {
        }
        public static Scenario Create()
        {
            Scenario scenario = new Scenario();

int scenarioID = (int)KeyGenerator.getNext("Scenarios")[0];
            scenario.ID = scenarioID;

            scenario.Name = "Unamed";
string sql = "Insert INTO Scenarios (sce_id,sce_name) values (" +
scenarioID.ToString() + ",'Unamed')";
           DBManager.DataAccesor.execute(sql);
           return scenario;
        }
        public string Name
```

```
        {
            get
            {
                if (_name == null)
                    return this.DataItem["sce_name"].ToString();
                else
                    return _name;
            }
            set { _name = value; }
        }

        public int ID
        {
            get
            {
                if (_id == null || _id == "")
                    return (int)DataItem["sce_id"];
                else
                    return Int32.Parse(_id);
            }

            set { _id = value.ToString(); }
        }
        public int vi
        {
    get { return (int)DataItem["vi"]; }
    set
    {
//Επαναϋπολογισμός Δείκτη τρωτότητας και αποθήκευση στη βάση
δεδομένων
                _vi = value;
string sql = "Update Scenarios set vi=" + _vi + " where sce_id = " +
ID ;
   DBManager.DataAccesor.execute(sql);
   }}
        public override string ToString()
        {
            return Name;
        }
        public Disasters Disasters
        {
            get
            {
BusinessLayer.Disasters disasters = new Disasters();
SelectDisastersStatement.Parameters["ScenarioID"] =
this.ID.ToString();
disasters.FetchAll(SelectDisastersStatement.Text);
              return disasters;
            } }
        public void AddDisaster(Disaster disaster)
        {
            string disID = disaster.ID.ToString();
 string sql = "Insert INTO DisastersScenarios (dsc_dis_id,dsc_sce_id)
values ("+disID+","+ID+")";

            DBManager.DataAccesor.execute(sql);
        }
        public void DeleteDisaster(Disaster disaster)
        {
            string disID = disaster.ID.ToString();
```

```
string sql = "DELETE FROM DisastersScenarios WHERE  dsc_dis_id =" +
disID + " AND dsc_sce_id=" + ID ;
DBManager.DataAccesor.execute(sql);
        } }
    public class Scenarios : BusinessBase<Scenario>
    {
        public Scenarios()
            : base()
        {
        }
        protected override string SelectStatement
        {
            get { return "Select * from Scenarios"; }

        }
private static SqlString InsertStatement = new SqlString(
 @"INSERT INTO Scenarios(sce_id,sce_name) VALUES (#Key,'#Name')");
private static SqlString InsertSecondaryStatement = new SqlString(
@"INSERT INTO DisastersScenarios(dsc_dis_id,dsc_sce_id) ");
public static void Insert(Scenario scenario, Disaster[] disasters)
        {
string ID = KeyGenerator.getNext("Scenarios")[0].ToString();
          InsertStatement.Parameters["Key"] = ID;
            InsertStatement.Parameters["Name"] = scenario.Name;
            InsertSecondaryStatement.Parameters["ScenarioKey"] = ID;
            foreach (Disaster disaster in disasters)
            {
 InsertSecondaryStatement.AppendText(" Select
"+disaster.ID.ToString()+" ,"+ID+"  UNION ALL ");
            }
    string str = InsertSecondaryStatement.Text;
   InsertSecondaryStatement.Text = str.Substring(0, str.Length - 10);
string sqlStr = InsertStatement.Text + ";" +
InsertSecondaryStatement.Text;
            DBManager.DataAccesor.execute(sqlStr);
        }
        public void SelectAll()
        {
            //SelectStatement = " Select * From Scenarios";
           FetchAll();
        }
        public static int[] GetViArray()
        {
            string sql = "Select vi from Scenarios";
            Data data = DBManager.DataAccesor.read(sql);
            int[] vis = new int[data.Rows.Count];
            for (int i = 0; i < data.Rows.Count; i++)
            {
                if (data.Rows[i][0]!= null)
                    vis[i] = (int)data.Rows[i][0];
            }
            return vis;
        }
        public static void SaveSceanario(Disaster[]
disasters,Scenario scenario)
        {
string sql = "DELETE FROM  DisastersScenarios WHERE dsc_sce_id= " +
scenario.ID ;
sql = sql + ";Update Scenarios SET sce_name= '" + scenario.Name + "'
WHERE sce_id = "+scenario.ID.ToString();
           for (int i = 0; i < disasters.Length; i++)
```

```
            {
sql = sql + ";Insert Into DisastersScenarios (dsc_dis_id,dsc_sce_id)
values(" + disasters[i].ID.ToString() + "," + scenario.ID + ")";
            }
            DBManager.DataAccesor.execute(sql);}}}
```

8.2.42 Combinatoric.jsl

```
package Combinatorics;

import java.math.*;
/**

 * The class Combinatoric contains methods for performing basic
combinatoric operations
 * such as counting numbers of permutations and combinations.
 *
 * @author dave_hoag
 * @version $Id: Combinatoric.java,v 2.1 2002/08/10 15:12:24
dave_hoag Exp $
 */
class CombinatoricException extends Exception
{
      public CombinatoricException(String str)
      {
            super(str);
      }
}
public class Combinatoric
{
      /**
       * @param n int
       * @param m int
       * @return BigInteger, the number of unordered subsets of m
objects chosen from a group
       *      of n objects.
       * @exception CombinatoricException unless n >= m >= 0
       */
public static BigInteger c(int n, int m) throws CombinatoricException
      {
            check(n, m);
            int r = Math.min(m, n - m);
            return p(n, r).divide(factorial(r));
      }
      /**
       * Check that 0 <= m <= n
       *
       * @param n int
       * @param m int
       * @exception CombinatoricException unless n >= m >= 0
       */
static void check(int n, int m) throws CombinatoricException
      {
            if (n < 0)
            {
throw new CombinatoricException("n, the number of items, must be
greater than 0");
            }
            if (n < m)
            {
```

```
throw new CombinatoricException("n, the number of items, must be >=
m, the number selected");
            }
            if (m < 0)
            {
throw new CombinatoricException("m, the number of selected items,
must be >= 0");}}
      /**
       * @param n int
       * @return BigInteger, the product of the numbers 1 ... n
       * @exception CombinatoricException unless n >= 0
       */


public static BigInteger factorial(int n) throws
CombinatoricException
      {
            if (n < 0)
            {
                  throw new CombinatoricException("n must be >= 0");
            }
            BigInteger factorial = new BigInteger(new byte[] { 1 });
            for (int i = n; i > 1; i--)
            {
                  factorial =
factorial.multiply(new BigInteger(new byte[] { (byte)i }));
            }
            return factorial; }
      /**
       * @param n int
       * @return BigInteger, the number of possible ways of ordering
n objects
       * @exception CombinatoricException unless n >= 0
       */
      public static BigInteger p(int n) throws CombinatoricException
      {
            return factorial(n);
      }
      /**
       * @param n int
       * @param m int
       * @return BigInteger, the number of possible arrangements, or
orderings, of m objects
       *       chosen from a group of n objects.
       * @exception CombinatoricException unless n >= m >= 0
       */
public static BigInteger p(int n, int m) throws CombinatoricException
      {
            check(n, m);
            BigInteger product = new BigInteger(new byte[] { 1 });
            for (int i = n; i > n - m; i--)
            {
product =product.multiply(new BigInteger(new byte[] { (byte)i }));
            }

      return product;   }}
```

8.2.43 Combinations.jsl

```
package Combinatorics;

/**
 * Summary description for Class1
 */
public class Combinations implements java.util.Enumeration
{
	private Object[] inArray;
	private int n, m;
	private int[] index;
	private boolean hasMore = true;
	/**
	* Create a Combination to enumerate through all subsets of the
	* supplied Object array, selecting m at a time.

	*
	* @param Object[] inArray the group to choose from
	* @param m int the number to select in each choice
	* @exception CombinatoricException if m is greater than
	* the length of inArray, or less than 0.
	*/
public Combinations(Object[] inArray, int m) throws
CombinatoricException
	{
		this.inArray = inArray;
		this.n = inArray.length;
		this.m = m;
		// throw exception unless n >= m >= 0
		Combinatoric.check(n, m);
		/**
		* index is an array of ints that keep track of the next
combination to return.
		* For example, an index on 5 things taken 3 at a time
might contain {0 3 4}.
		* This index will be followed by {1 2 3}. Initially, the
index is {0 ... m - 1}.
		*/
		index = new int[m];
		for (int i = 0; i < m; i++)
			index[i] = i;
	}
	/**
	* @return true, unless we have already returned the last
combination.
	*/
	public boolean hasMoreElements()
	{
		return hasMore;
	}
	/**
	* Move the index forward a notch. The algorithm finds the
rightmost
	* index element that can be incremented, increments it, and
then
	* changes the elements to the right to each be 1 plus the
element on their left.
```

```
	* <p>
	* For example, if an index of 5 things taken 3 at a time is at
{0 3 4}, only the 0 can
	* be incremented without running out of room. The next index is
{1, 1+1, 1+2) or
	* {1, 2, 3}. This will be followed by {1, 2, 4}, {1, 3, 4}, and
{2, 3, 4}.
	* <p>
	* The algorithm is from Applied Combinatorics, by Alan Tucker.
	*
	*/
	private void moveIndex()
	{
		int i = rightmostIndexBelowMax();
		if (i >= 0)
		{
			index[i] = index[i] + 1;
			for (int j = i + 1; j < m; j++)

			index[j] = index[j - 1] + 1;
		}
		else
			hasMore = false;
	}
	/**
	* @return java.lang.Object, the next combination from the
supplied Object array.
	* <p>
	* Actually, an array of Objects is returned. The declaration
must say just Object,
	* because the Combinations class implements Enumeration, which
declares that the
	* nextElement() returns a plain Object. Users must cast the
returned object to (Object[]).
	*/
	public Object nextElement()
	{
		if (!hasMore)
			return null;
		Object[] out = new Object[m];
		for (int i = 0; i < m; i++)
			out[i] = inArray[index[i]];
		moveIndex();
		return out;
	}
	/**
	* @return int, the index which can be bumped up.
	*/
	private int rightmostIndexBelowMax()
	{
		for (int i = m - 1; i >= 0; i--)
		{
			if (index[i] < n - m + i)
				return i;
		}
		return -1;  }}
```

Aladwani A.M. (2002), **An Integrated Performance Model of Information Systems Projects,** *Journal of Management Information Systems, vol.19(1).*

Alexander D. (2000), **Scenario methodology for teaching principles of emergency management,** *Disaster Prevention and Management, vol.9(2),89-97.*

Alexander D. (2005), **Towards the development of a standard in emergency planning,** *Disaster Prevention and Management, vol.14 (2), 158 -175.*

Alford K.L., Dunn C., Ruocco A.S. (2001), **Information Assurance Pedagogy,** *Proceedings, IEEE Workshop on Information Assurance and Security, West Point.*

Anderson T., Knight J. (1983), **A Framework for Software Fault Tolerance in Real-Time Systems,** *IEEE Transactions on Software Engineering, vol.SE-9 (3), 355-364.*

Austin C.J., Trimm J.M., Sobzak M.E. (1995), **Information systems and strategic management,** *Health Care Management Review, vol.20(3), 26-33.*

Badenhorst K.P, Eloff J.H.P. (1989), **Framework of a methodology for the life cycle of computer security in an organization,** *Computers & Security, vol.8, 433-442.*

Badenhorst K.P, Eloff J.H.P. (1990), **Computer security methodology: risk analysis and project definition,** *Computers & Security, vol.9, 339-346.*

Baker S., Ponniah D., Smith S. (1999), **Risk response techniques employed currently for major projects,** *Construction Management and Economics, vol.17(2), 205-213.*

Ball L.D, Dentch G, Emerson M. et al. (1982), **Disaster Recovery Services,** *Computers & Security, vol.1, 216-225.*

Barki H., Rivard S., Talbot J. (1993), **Toward an Assessment of Software Development Risk,** *Journal of Management Information Systems, vol.10, 203-225.*

Barki H., Rivard S., Talbot J. (2001), **An Integrative Contingency Model of Software Project Risk Management,** *Journal of Management Information Systems, vol.17(4), 37-69.*

Bell J. (1993), **Why business recovery planning is an absolutely necessary competitive strategy… getting top executives to support your effort,** *Disaster Recovery Journal, vol.6(1), 10-11.*

Berny J. (1989), **A new distribution function for risk analysis,** *Journal of Operational Research Society, vol.40, 1121-1127.*

Bettis R.A. (1982), **Risk considerations in modelling corporate strategy,** *Academy of Management Proceedings, vol.22(5).*

Beynon-Davies P. (1995), **Information systems failure: the case of the London Ambulance Service's Computer Aided Despatch Project**, *European Journal of Information Systems, Vol.4, 171-184.*

Bier V.M. (1999), **Challenges to the Acceptance of Probabilistic Risk Analysis,** *Risk Analysis, vol.19(4),703-710.*

Biswanath M., Heberlein T., Levitt K. (1994), **Network Intrusion Detection,** *IEEE Network, vol.8, 26-41.*

Blake W.F., (1994), **Making recovery a priority,** *Security Management, vol.36(4), 71-74.*

Bodnar G.H. (1993), **Data security and contingency planning,** *Internal Auditing, vol.8(3),74-80.*

Boehm B.W. (1988), **A spiral Model of Software Development and Enhancement,** *Computer, 61-72.*

Boehm B.W. (1991), **Software Risk Management: Principles and Practices,** *IEEE Software, vol.8, 32-41.*

Bonnal P., Gourc D., Lacoste G. (2002), **The life cycle of technical projects**, *Project Management Journal, vol.33(1), 12-19.*

Bonner J.M., Ruekert R.W., Walker O.C. (2001), **Upper management control of new product development projects and project performance,** *The Journal of Product Innovation Management, vol.19 (3), 233-245.*

Borba G. (1996), **The internet and disaster response,** *Australian Journal of Emergency Management, vol.10, 42.*

Bostrom A., Atman C.J, Fischhoff B., Morgan M.G. (1994), **Evaluating risk communications – completing and correcting mental models of hazardous processes,** *Risk Analysis, vol.14(5), 789-798.*

Botterell A. (1996), **Network technology in the practise of emergency management,** *Australian Journal of Emergency Management, vol.10, 43.*

Brown M. (1993), **The Disaster Business,** *Management Today, 42-48.*

Breslau L., Estrin D., Fall K., et al. (2000), **Advances in Network Simulation,** *IEEE Computer, vol.33(5),59-67.*

Burling W.K., Hyle A.E. (1997), **Disaster preparedness planning: policy and leadership issues**, *Disaster Prevention and Management, vol.6(4), 234-244.*

Carver C.A., Hill J.M.D., Surdu J.R. (2000), **A Methodology for Using Intelligent Agents to provide Automated Intrusion Response,** *Proceedings IEEE Workshop on Information Assurance and Security, West Point.*

Carver C.A., Hill J.M.D., Pooch U.W. (2001), **Limiting Uncertainty in Intrusion Response,** *Proceedings, IEEE Workshop on Information Assurance and Security, West Point.*

Castrigiano D.P.L., Hayes S.A. (2004), **Catastrophe Theory,** *WestView Press, Colorado.*

Cervone F.H. (2006), **Disaster Recovery and continuity planning for digital library systems,** *OCLC Systems & Services: International digital library perspectives, vol.22(3), 173-178.*

Chapman C., Ward S. (1997), **Project Risk Management. Processes, Techniques and Insights,** *J. Wiley, England.*

Chapman C.B. (1997), **Project risk analysis and management- PRAM the generic process,** *International Journal of Project Management, vol.53(5), 273-281.*

Chapman C.B., Cooper D.F. (1983), **Risk analysis: testing some prejudices,** *European Journal of Operational Research, vol.14,238-247.*

Chapman R. (1998), **The effectiveness of working group risk identification and assessment techniques,** *International Journal of Project Management, vol.16(6), 333-343.*

Chung-Jen C., Bou-Wen L. (2004), **The effects of environment, knowledge attribute, organizational climate, and firm characteristics on knowledge sourcing decision,** *R&D Management, vol.34(2), Blackwell Publishing Ltd, 137-146.*

Ciechanowicz Z. (1997), **Risk analysis: requirements, conflicts and problems,** *Computers and Security, vol.16(3), 223-232.*

Cooper D.F., MacDonald D.H., Chapman C.B. (1985), **Risk analysis of a construction cost estimate,** *International Journal of Project Management, vol.3(3), 141-149.*

Cornell-Pate E. (1999), **Conditional Uncertainty Analysis and Implications for Decision Making: The Case of WIPP,** *Risk Analysis, Vol.19(5),995-1002.*

Cosgrave J. (1996), **Decision making in emergencies**, *Disaster Prevention and Management, vol.5(4).*

Couillard J. (1995), **The role of project risk in determining project management approach,** *Project Management Journal, vol. 26(4), 3-15.*

Courtney R.H Jr. (1982), **A systematic approach to data security,** *Computers & Security, vol.1, 99-112.*

Courtney R Jr. (1988), **Proper Assignment of Responsibility for Data Security,** *Computers & Security, vol.7(1).*

Cristian F. (1993), **Automatic Reconfiguration in the Presence of Failures,** *Software Engineering Journal, Vol. 8(2), 53-60.*

Davoudian K., Wu J.- S., Apostolakis G., (1994), **The work Process Analysis Model (WPAM),** *Reliability Eng. Syst. Safety, vol.45, 107-125.*

Davoudian K., Wu J.- S., Apostolakis G., (1994), **Incorporating Organizational Factors into Risk Assessment through the Analysis of Work Processes,** *Reliability Eng. Syst. Safety, vol.45, 85-105.*

Denning D.E., (1987), **An Intrusion Detection Model,** *IEEE Transactions on Software Engineering, vol.13, 222-232.*

Dey P.K. (1999), **Process reengineering for effective implementation of projects,** *International Journal of Project Management, vol.17(3),147-159.*

Dey P.K. (2001), **Decision Support system for risk management: a case study,** *Management Decision, vol. 39(8), 634-649.*

Dey P.K. (2002), **Project risk management: A combined analytic hierarchy process and decision tree approach,** *Cost Engineering, vol. 44(3), 13-26.*

Dorofee A.J., Walker J.A. et al. (1996), **Continuous Risk Management Guidebook,** *Carnegie Mellon University, Software Engineering Institute.*

Doyle J.C. (1996), **Improving performance in emergency management,** *Disaster Prevention and Management, vol.5(3).*

Elkington P. & Smallman C. (2000), **Managing project risks: a case study from the utilities sector,** *International Journal of Project Management, vol.20 (1), 49-57.*

Eloff J.H.P., Labuschagne L., Badenhorst K.P. (1993), **A comparative framework for risk analysis methods,** *Computers & Security, vol.12(6), 597-603.*

Fairley W.B., (1981), **Assessment for Catastrophic Risks,** *Risk Analysis, vol.1(3), 197-204.*

Fairley R. (1994), **Risk management for Software Projects,** *IEEE Software, 57-67.*

Fischer G.W., Morgan M.G. et al. (1991), **What risks are people concerned about?,** *Risk Analysis, vol.11(2), 303-314.*

Fischhoff B., Watson S.R., Hope C. (1984), **Defining risk,** *Policy Sciences, vol.17, 123-139.*

Fishwick P.A., Kim G., Lee J.J. (1996), **Improved Decision Making Through Simulation Based Planning,** *Simulation, vol.67(5), 315-327.*

Flin R., Slaven G. (1996), **Personality and emergency command ability,** *Disaster Prevention and Management,vol.5(1).*

Frosdick S. (1997), **The techniques of risk analysis are insufficient in themselves,** *Disaster Prevention and Management, vol.6(3).*

Frost C. (1994), **Effective responses for proactive enterprises: business continuity planning,** *Disaster Prevention and Management, vol.3(1), 7-15.*

Galloway I. (1996), **Design for support and support the design: integrated logistic support – the business case,** *Disaster Prevention and Management, vol.9(1), 24-31.*

Garrity E., Sanders G.L. (1998), **Introduction to Information Systems-Success Measurement,** *IDEA Group Publishing, Hersley USA.*

Gigliotti R., Ronald J. (1991), **Emergency Planning for maximum protection,** *Butterworth-Heinemann, New York.*

Gilmore R. (1981), **Catastrophe Theory for Scientists and Engineers**, *Dover Publications Inc., New York.*

Gillingham D.W., Blanco J., Lewko J.H. (1997), **An integrated model of error management,** *Disaster Prevention and Management, vol.6(3).*

Goodman L.A., Kruskal W.H. (1975), **A new model for scaling response patterns: An application of the quasi-independence concept**." *Journal of American Statistical Association, 755-768.*

Granot H. (1997), **Emergency inter-organizational relationships,** *Disaster Prevention and Management, vol.6(5).*

Guarro S.B. (1987), **Principles and procedures of the ALRAM approach to information system risk analysis and management,** *Computers & Security, vol.6, 493-504.*

Gupta U.G., Clarke R.E. (1996), **Theory and applications of the Delphi technique: a bibliography (1975-1994),** *Technological Forecasting and Change, vol.53, 185-211.*

Haimes Y.Y., Li D., Tulsiani V. (1990), **Multi-objective Decision Tree Analysis,** *Risk Analysis, vol.10(1), 111-127.*

Halman J.I.M., Keizer J.A. (1998), **Risk management in product innovation projects,** *International Journal of Project and Business Risk Management, vol.2(2).*

Hallikas J., Virolainen V.-M., Tuominen M. (2002), **Risk analysis and assessment in network environments: A dyadic case study,** *International Journal of Production Economics, vol.78(1).*

Hayes-Roth B. (1995), **An architecture for adaptive Intelligent Systems,** *Artificial Intelligence, vol.72(1), 329-365.*

Heemstra F., Kusters R. (1996), **Dealing with risk; a practical approach,** *Journal of Information Technology, vol.11.*

Helman P., Liepins G.E. (1993), **Statistical Foundations of Audit Trail Analysis for the Detection of Computer Misuse,** *IEEE Transactions on Software Engineering, vol.19, 886-901.*

Higuera R.P., Dorofee A.J., Walker J.A., Williams R.C. (1994), **Team Risk Management: A New Model for Customer-Supplier Relationships,** *Software Engineering Institute, Carnegie Mellon University.*

Hill J.M.D., Carver C.A. (2000), **A Tactical Data Network Bandwidth Design and Simulation Tool,** *in Proceedings of the Advanced Simulation Technology Conference: Military, Government, and Aerospace Symposium, Washington, 43-48.*

Hill J.M.D., Miller M.S. (2000), **Tactical Event Resolution Using Software Agents, Crisp Rules, and a Genetic Algorithm,** *Advanced Simulation Technology Conference, Washington, 16-20.*

Hill J.M.D., Surdu J.R., Pooch U.W. (2001), **Implementation of the Anticipatory Planning Support System,** *Advanced Simulation Technology Conference: Military, Government, and Aerospace Symposium, Seattle.*

Hoffman T. (1998), **Denial stalls disaster recovery plans,** *Computerworld, vol.32(8), 10.*

Hopfl H., (1994), **Safety culture, corporate culture: organizational transformation and the commitment to safety,** *Disaster Prevention and Management, vol.3(3), 49-58.*

(IRM). (1998), **Risk Assessment and Management Process (Ramp),** *Information Resource Management, 1-44.*

Issac I. (1995), **Training in risk management***, International Journal of Project Management, vol.13(4), 225-229.*

Iyer R.K., Bandyopadhyay K. (2000), **Managing technology risks in the health care sector: disaster recovery and business continuity planning,** *Disaster Prevention and Management, vol.9(4), 257-270.*

James J.R., Ragsdale D., Schafer J., Presby T. (2000), **Performance Modeling of AFATDS and Other Applications: Implications for Information Assurance and Security,** *Proceedings IEEE Workshop on Information Assurance and Security, West Point.*

Johnson W.G. (1975), **Management oversight risk tree (MORT),** *Journal of Safety Research, vol.7(1), 4-15.*

Kara-Zaitri C. (1996), **Disaster Prevention and limitation: state of the art; tools and technologies,** *Disaster Prevention and Management, vol.5(1), 30-39.*

Keil M., Cule P., Lyytinen K., Schmidt R. (1998), **A framework for identifying software project risks,** *Communications of the ACM, vol.14(11).*

Kelly C. (1995), **A framework for improving operational effectiveness and cost effiency in emergency planning and response,** *Disaster Prevention and Management, vol.4(3), 25-31.*

Kent E. (2001), **Continuous Risk Management Plan,** *Geostationary Imaging Fourier Transform Spectrometer (GIFTS)- Indian Ocean METOC Imager(IOMI) Mission, 1-24.*

Kletz T.A. (1983), **The mathematics of risk,** *Interdisciplinary Science Reviews, vol.8(2), 114.*

Kletz T.A. (1996α), **Risk – two views: the public's and the experts',** *Disaster Prevention and Management, vol.5(4), 41-46.*

Kletz T.A. (1996β), **Disaster prevention: current topics,** *Disaster Prevention and Management, vol.5(2),36-41.*

Kliem R., Ludin I. (1997), **Reducing Project Risks,** *Gower Publishing, U.K., Hampshire.*

Kloman H.F. (1990), **Risk Management Agonists,** *Risk Analysis, vol.10(2), 201-205.*

Kosoresow A.P., Hofmeyr S.A. (1997), **Intrusion Detection via System Call Traces,** *IEEE Software, vol.14, 24-42.*

Kremer C., Sosa G. (1991), **Systems development risks in strategic information systems,** *Information and Software, vol.33(3).*

Lamm G., Falauto G., Estrada J., Gadiyaram J. (2001), **Bluetooth Wireless Networks Security Features,** *Proceedings of the IEEE Workshop on Information Assurance and Security, West Point.*

Lauer T. (1996), **Software project managers' risk preferences,** *Journal of Information Technology, vol.11.*

Leopoulos V., Kirytopoulos K., Malandrakis C. (2003), **An applicable methodology for strategic risk management during the bidding process,** *International Journal of Risk Assessment and Management, vol.4(1), 67-80.*

Leopoulos V., Kirytopoulos K. (2004), **Risk Management: A competitive advantage in the purchasing function,** *Production Planning and Control vol.15(7), 678-687.*

Lister T. (1997), **Risk management is project management for adults**, *IEEE Software*, *vol. 14(3), 20-21.*

Lipovetsky S., Tishler A., Dvir D., Shenhar A.J. (1997), **The relative importance of project success dimensions,** *R&D Management, vol.27(1), 97-106.*

Lyytinen K., Mathiassen L., Ropponnen J. (1996), **A framework for software risk management,** *Journal of Information Technology, vol.11(4).*

Machlis G.E., Rosa E.A. (1990), **Desired risk: broadening the social amplification of risk framework,** *Risk Analysis, vol.10(1), 161-180.*

Macedonia M. (2002), **Games Soldiers Play,** *IEEE Spectrum, vol.39(3), 32-37.*

Madan B., Popstojanova K., Vaidyanathan K., Trivedi K., (2004), **A Method for Modeling and Quantifying the Security Attributes of Intrusion Tolerant Systems,** *Performance Evaluation, vol.56(1), 167-186.*

Mader D.P. (2002), **Design for six sigma,** *Quality Progress, vol.35(7), 82-86.*

March J.G. (1987), **Managerial perspectives on risks and risk taking,** *Management Science vol.33, 1404-1418.*

March R.A. (1995), **Is your recovery plan done?,** *Disaster Recovery Journal, vol.8(4), 75-78.*

Mavrommatis G., Kalligatsis J., Panayiotopoulos J.-C. (2005), **MOP : Mutation - Oriented Programming,** *Foundations of Computing and Decision Sciences, Institute of Computing Science, Poznan University of Technology, vol.30(3), 215-226.*

McEntire D.A. (2001), **Triggering agents, vulnerabilities and disaster reduction: towards a holistic paradigm,** *Disaster Prevention and Management, vol.10(3).*

McEntire D.A., Fuller C. (2002), **The need for a holistic theoretical approach: an examination from the El Nino disasters in Peru**, *Disaster Prevention and Management, vol.11(2).*

McGaughey R.E., Snyder C.A., Carr H.H. (1994), **Implementing information technology for competitive advantage: risk management issues,** *Information and Management, vol.26(5), 273-280.*

Meade P., (1993), **Taking the risk out of disaster recovery services,** *Risk Management, 20-26.*

Mick S., Wallace W.A. (1985), **Expert Systems as Decision Aid – For Disaster Management.** *Disasters, vol. 9(2), 98-101.*

Mosleh A., Bier V.M., Apostolakis G.E. (1988), **Methods for the Elicitation and Use of Expert Opinion in Risk Assessment,** *Reliability Eng. Syst. Safety, vol.20, 63-85.*

Moe T.L., Pathranarakul P. (2006), **An integrated approach to natural disaster management,** *Disaster Prevention and Management, vol.15 (3), 396-413.*

Moynihan T. (1997), **How Experienced Project Managers Assess Risk,** *IEEE Software, vol.14, 35-41.*

Murfy D.M., Pate- Cornell M.E. (1996), **The SAM Framework: Modelling the Effects of Management Factors on Human Behavior in Risk Analysis,** *Risk Analysis, vol.16, 501-515.*

Mustafa M.A., Al-Bahar J.F. (1991), **Project risk assessment using the Analytic Hierarchy Process,** *IEEE Trans. Eng. Manag., vol.38(1), 46-52.*

Nash D.A., Ragsdale D.J. (2000), **Simulation of self-similarity in network utilization patterns as a precursor to automated testing of intrusion detection systems,** *Proceedings IEEE Workshop on Information Assurance and Security, West Point.*

National Institute of Standards and Technology U.S. (2001), **Contingency Planning Guide for Information Technology Systems,** *Special Publication, 800-834.*

Neo B., Leong K. (1994), **Managing risks in information technology projects: a case study of TradeNet,** *Journal of Information Technology, vol.5(3).*

Norris C., Perry J., Simon P. (1992), **Project Risk Analysis and Management,** *A Guide by APM, The Association for Project Management.*

Nidimolu S. (1995), **The effect of Coordination and Uncertainty on Software Project Performance: Residual Performance Risk as an Intervening Variable,** *Information Systems Research, vol.6(3).*

Otway H., Thomas K. (1982), **Reflections on risk perception and policy,** *Risk Analysis, vol.2(2), 69-82.*

Oztas A., Okmen O. (2005), **Judgemental risk analysis process development in construction process,** *Building and Environment, vol.40, 1244-1254.*

Panayiotopoulos J.-C. (1979α), **Catastrophe Transportation Programming,** *Industrial Mathematics, vol.29(2), 77-88.*

Panayiotopoulos J.-C. (1979β), **Integer Programming with Irregular Data Space,** *The Journal of Operational Research Society of Japan, vol.22(4), 274-287.*

Panayiotopoulos J.-C. (1980), **The Multi-Dimensional Assignment Model with Provision for an Uncertain Future,** *Journal of Information and Optimization Sciences, vol.1(1), 94-102.*

Panayiotopoulos J.-C. (1981), **Irregular Gaming Areas,** *European Journal of Operational Research,* vol.7, 61-68.

Panayiotopoulos J.-C. (1992), **White, grey and black operational holes: An artificial intelligence approach,** *Journal of Information and Optimization Sciences, vol.13(3), 407-425.*

Panayiotopoulos J-C. (2007), **Satellite C4I within the United Nations Responsibilities :"Can anything on Earth be private anymore",** *Defensor Pacis, vol.19, special issue, 2007.*

Panayiotou N., Gayialis S., et al. (2004), **Risk management issues in the implementation of an ERP system for a large Greek company,** *WSEAS Transactions on Computers, vol.3(4), 1005-1012.*

Parnaby J. (1979), **Concept of Manufacturing System,** *International Journal of Production Research, vol.17(2), 123-135.*

Paton D. (1992), **International Disasters: issues in the management and preparation of relief workers,** *Disaster Management, vol.4, 183-190.*

Paton D. (1999), **Disaster Business Continuity: promoting staff capability,** *Disaster Prevention and Management, vol.8, 127-133.*

Payne F.C. (1999), **Contingency plan exercises,** *Disaster Prevention and Management, vol.8(2), 111-117.*

Pearce M.B. (1994), **Disaster recovery – a key component in any quality program,** *Disaster Recovery Journal, vol.7(3), 26-32.*

Pearson C.M., Mitroff I.I., (1993), **From crisis prone to crisis prepared: a framework for crisis management,** *Academy of Management Executive, vol.7(1), 48-59.*

Peters E.E. (1996), **Chaos and Order in the Capital Markets,** *John wiley, Chichester.*

Petrantonakis P., Panayiotopoulos J.-C. (2004α), **BLACK OPERATING HOLE AND LAND WARRIOR INFORMATION SYSTEMS,** *20th European Conference on Operational Research, Rhodes.*

Petrantonakis P., Panayiotopoulos J.-C. (2004β), **Diagnosis in Distributed Medical System Basis,** *2th National Conference on Medical Technologies, University of Patras.*

Petrantonakis P., Panayiotopoulos J.-C. (2005α), **Minimizing Recovery Cost in Military Information Systems: the Land Warrior Case,** *Defensor Pacis, issue 17,189-200.*

Petrantonakis P., Panayiotopoulos J.-C. (2005β), **Using Assignment Model as an automated Recovery System,** *Disaster Prevention and Management. An International Journal, Volume 14(1), 89-96.*

Petrantonakis P., Panayiotopoulos J.-C. (2006), **Some Considerations on Crisis Management,** *Defensor Pacis, issue 18, 7-14.*

Petroni A. (1999), **Managing information systems' contingencies in banks: a case study,** *Disaster Prevention and Management, vol.8(2),101-110.*

Pinto J.K. & Mantel S.J. (1989), **The causes of Project Failure,** *IEEE Transactions On Engineering Management, vol.37(4), 269-275.*

PMI (2000), **A Guide to the Project Management Body of Knowledge,** *Project Management Institute, USA.*

Poston T. & Stewart I. (1978), **Catastrophe Theory and its Applications, Dover** *Publications Inc., New York.*

Powell P., Klein J. (1996), **Risk management for information systems development,** *Journal of Information Technology, vol.11(4).*

Quarantelli E.L. (1997), **Problematical aspects of the information/ communication revolution for disaster planning and research: ten non-technical issues and questions,** *Disaster Prevention and Management, vol.6(2), 94-106.*

Ragsdale D., Maconachy V. M., Schou C., Welch D. (2001), **A Model for Information Assurance: An Integrated Approach,** *Proceedings of the 2001 IEEE Workshop on Information Assurance and Security, United States Military Academy, West Point.*

Rapp C., Nishikawa D., Bukowczyk J. (2001), **Developing Platform Independent Tactical Data Entry Devices,** *Proceeding of The National Conference On Undergraduate Research (NCUR), University of Kentucky, Lexington Kentucky.*

Rayner S., Cantor R. (1987), **How fair is safe enough? The cultural approach to social technology choice,** *Risk Analysis, vol.7, 3-9.*

Raz T., Michael E. (2000), **Use and benefits of tools for project risk management,** *International Journal of Project Management, vol.19, 9-17.*

Records Management (2003), **Information Disaster Prevention and Recovery: A guide for developing a disaster plan,** *http://www.system.missouri.edu/records/disaster.html*

Ren C.H. (2000), **Understanding and managing the dynamics of linked crisis events,** *Disaster Prevention and Management, vol.9(1),12-17.*

Robinson C.A. (1998), **Warfighter Information Network Harnesses Simulation Validation,** *Signal Magazine, vol.52 (5), 27-31.*

Ronan K.R., Paton D., Johnston D.M., Houghton B.F. (2000), **Managing societal uncertainty in volcanic hazards: a multidisciplinary approach,** *Disaster Prevention and Management, vol.9(5).*

Rosser J.B. Jr. (1994), **From Catastrophe to Chaos: A general Theory of Economic Discontinuities,** *Kluver, Boston.*

Royer P.S. (2000), **Risk Management: The Undiscovered Dimension of Project Management,** *Project Management Journal, vol.31(1), 6-13.*

Ruocco A., Buchheit N., Ragsdale D. (2000), **A Combined Offensive/ Defensive Network Model,** *Proceedings of the 2000 IEEE Workshop on Information Assurance and Security, West Point, NY.*

Shaluf I.M. (2007), **An overview on the technological disasters,** *Disaster Prevention and Management, vol.16(3), 380-390.*

Sheng-Hsien T., Shin-Yann H., (1994), **Failure mode effect analysis: An integrated approach for product design and process control,** *International journal of Quality and Reliability Management, vol.13(5), 8-26.*

Shenhar A.J., Dvir D., Levy O. (1997), **Mapping the dimensions of project success,** *Project Management Journal, vol.28(2), 5-13.*

Shrivastava P, Mitroff I.I., Miller D., Miglani A. (1988), **Understanding industrial crises,** *Journal of Management Studies, vol.25(4), 283-303.*

Smallman C. (1996), **Risk and organizational behaviour: a research model,** *Disaster Prevention and Management, vol.5(2),12-26.*

Smith H.J., Keil M. (2003), **The reluctance to report bad news on troubled software projects: A Theoretical Model,** *Information Systems Journal, vol.13(1), 69-95.*

Song M. (2001), **From experience: applying the risk diagnosing methodology,** *The Journal of Product Innovation Management, vol.19(3), 213-232.*

Souder W.E. & Bethay D. (1993), **The Risk Pyramid for New Product Development: An Application to Complex Aerospace Hardware,** *The Journal of Product Innovation Management, vol.10 (2), 181-194.*

Stephenson R., Anderson P.S. (1997), **Disasters and the Information Technology Revolution,** *Disasters,Vol.21(4),305-334.*

Stewart R.W. & Fortune J. (1995), **Application of systems thinking to the identification, avoidance and prevention of risk,** *International Journal of Project management, vol.13(5), 279-286.*

Strain J.D., Preece D.A. (1999), **Project management and the integration of human factors in military system procurement,** *International Journal of Project Management, vol.17(5), 283-292.*

Sun Zu. (2003), **Η τέχνη του πόλεμου,** *Εκδόσεις Βάνιας, Θεσσαλονίκη.*

Suppes P. (1972), **Axiomatic Set theory,** *Dover Publications Inc, New York*

Surdu J.R., Haines G.D., Pooch U.W. (1999), **OpSim: a Purpose –built Distributed Simulation for the Mission Operational Environment,** *Proceedings International Conference on Web – Based Modelling and Simulation, San Francisco, 69-74.*

Surdu J.R., Maymi F., Hall A., Beltramini R. (2002), **Modeling the wireless network architecture of Land Warrior,** *Proceedings of the 2002 Winter Simulation Conference.*

Surdu J.R., Maymi F., Hall A., Deb A., Freberg K., (2001), **Modeling the Communications Capabilities of the Infantry Soldier,** *Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation Conference.*

Surdu J.R., Pooch U.W. (1998), **A Methodology for Applying Simulation Technologies in the Mission Operational Environment,** *Proceedings IEEE Information Technology Conference, Syracuse, NY, 45-48.*

Surdu J.R., Pooch U.W. (1999α), **Connecting the Operational Environment to Simulation,** *Proceedings Advanced Simulation Technology Conference: Military, Government and Aerospace Simulation, San Diego, 94-99.*

Surdu J.R., Pooch U.W. (1999β), **A Methodology to Using Intelligent Agents to Apply Simulation Technologies to the Mission Operational Enviroment,** *Proceedings Enabling Technologies for Simulation Science III, Orlando, 56-64.*

Surdu J.R., Pooch U.W. (2000α), **Simulation Technologies in the Mission Operational Environment,** *Simulation, vol.74(3), 138-160.*

Surdu J.R., Pooch U.W. (2000β), **A Methodology to Support Anticipatory Planning,** *Advanced Simulation Technologies Conference: Military, Government and Aerospace Simulation, Washington, 22-28.*

Surdu J.R., Hill J.M.D., Pooch U.W. (2000), **Anticipatory Planning Support System,** *Proceedings of the Winter Simulation Conference, Orlando, 950-957.*

Tan J.H.M. & Carr V. (2001), **Towards a framework for project risk knowledge management in the construction supply chain,** *Advances in Engineering Software, vol. 32(10-11), 835 - 846.*

Tate G., Verner J. (1996), **Case study of risk management, development and evolutionary prototyping,** *Information and Software Technology, vol.23(3).*

Tatikonda M.V. & Rosenthal S.R. (2000), **Technology Novelty, Project Complexity, and Product Development Project Execution Success: A Deeper Look at Task Uncertainty in Product Innovation,** *IEEE Transactions On Engineering Management, Vol.47 (1), 74-85.*

Taylor-Adams S., Kirwan B. (1997), **Human reliability data requirements,** *Disaster Prevention and Management, vol.6(5).*

Terry R.J. (1995), **Organizing a contingency plan for success,** *Disaster Recovery Journal, vol.8(2), 43-46.*

Thompson J.M.T. (1986), **Nonlinear Dynamics and Chaos,** *J. Wiley & Sons, New York.*

Totozian V., Nikitakos N., Platis A. (2005), **Hybrid model for shipping risk management,** *Πρακτικά 17ου συνεδρίου Ε.Ε.Ε.Ε., Πάτρα, 185-197.*

Tummala V.M.R., Leung Y.H., (1999), **Applying a risk management process (RMP) to manage cost risk for an EHV transmission line project,** *International Journal of Project Management, vol.17(4), 223-235.*

Tzvi R., Shenhar A.J., Dvir D. (2002), **Risk management, project success, and technological uncertainty,** *R&D Management, vol.32(2), 101-109.*

Violanti J., Paton D., Smith L. (2000), **Disaster response: risk, vulnerability and resilience,** *Disaster Prevention and Management vol.9 (3), 173-179.*

Von Bertanlaffy L. (1962), **General Systems Theory – a critical view,** *General Systems, vol.7, 1-20.*

Von Solms R., Van de Haar H, Von Solms S.H. (1994), **A framework for information security evaluation,** *Information and Management, vol.26(3), 143-153.*

Wack J.P. (1991), **Establishing a Computer Security Incident Response Capability,** *National Institute of Standards and Technology, Special Publication 800-3.*

Ward S. (1999), **Requirements for an effective project risk management process,** *Project Management Journal, vol.30(3), 37-43.*

Ward S.C. & Chapman C.B. (1995), **Risk-management perspective on the project lifecycle,** *International Journal of Project management, vol.13 (3), 145-149.*

Weichselgartner J. (2001), **Disaster mitigation: the concept of vulnerability revisited,** *Disaster Prevention and Management, vol.10(2).*

Welch D., Conti G. (2001), **A framework for the Information Warfare Simulation**, *Proceedings, IEEE Workshop on Information Assurance and Security, West Point.*

Welch D., Ragsdale D., Wayne S. (2002), **Training for Information Assurance,** *IEEE Computer, 2-9.*

White G.B., Fisch E.A., Pooch U.W. (1996), **Cooperating Security Managers: A peer-based Intrusion Detection System,** *IEEE Network, vol.10(1), 20-23.*

Wideman R.M. (1992), **Project and Program Risk Management: A Guide to Managing Project Risks and Opportunities,** *Project Management Institute, USA.*

Wiggins S. (1990), **Introduction to Applied Dynamical Systems and Chaos,** *Spinger, New York.*

Walker P. (1983), **Smart Weapons in Naval Walfare,** *Scientific American, Vol.248(5), 31-39.*

William R.C., Walker J.A., Dorofee A.J. (1997), **Putting risk management into practice,** *IEEE Software, vol.14(3),75-81*

William T. (1995), **A classified bibliography of recent research relating to project management,** *European Journal of Operational Research, vol. 85(1), 18-38*

Williams T.M. (1994), **Using a risk register to integrate risk management in project definition,** *International Journal of Project Management, vol.12(1), 17-22.*

Winsor R.D. (1996), **Military perspectives of organizations,** *Journal of Organizational Change Management, vol.9(4), 34-42.*

Winters P. (1990), **Secure systems design: An evolving national strategy,** *Computers and Security, 379-389.*

Wold G.H., Shriver R.F. (1994), **Risk analysis techniques,** *Disaster Recovery Journal, vol.7(3), 46-52.*

Wold G.H. (2002), **Disaster Recovery Planning Process,** *Disaster Recovery Journal, http://www.drj.com/new2dr/w2_002.htm*

Wood A. (1996), **Integrating Risk Assessment into the Enterprise Information Management Strategy,** *6th International Pipeline Reliability Conference November 19-22, Houston TX.*

Wood C. (1990), **Principles of secure information systems design,** *Computers and Security, 13-24.*

Woodcock A., Davis M. (1988), **Θεωρία των Καταστροφών,** *Εκδόσεις Δαίδαλος, Αθήνα.*

Yacoub S.M., Ammar H.H., (2001), **A Methodology for Architectural – Level Reliability Risk Analysis,** *Publishing Systems and Solutions Laboratory, HP Laboratories Palo Alto.*

Yeo K.T. (1990), **Risks classification of estimates and contingency,** *Journal of Management Engineering, vol.6(4),458-470.*

Yu V.L. et al. (1984), **An Evaluation of MYCIN's Advice, In Rule Based Expert Systems**, Reading, MA: Addison-Wesley.

Zadeh L.A. (1996), **The evolution of systems analysis and control: A personal Perspective,** *IEEE Control Systems, vol.16(3), 95-98.*

Zeeman E.C. (1977), **Catastrophe Theory,** *Addison-Wesley Publishing Company Inc., London.*

Βαρελάς Τ., (2005), **Ολιστικό μοντέλο εκτίμησης κινδύνου του λειτουργικού πλαισίου της ναυτιλίας,** *Πρακτικά 17ου συνεδρίου Ε.Ε.Ε.Ε., Πάτρα, 199-204.*

Γιαννούτσος Θ. (2002), **Προγράμματα εξοπλισμού του στρατιώτη πεζικού με νέα συστήματα για το σύγχρονο πεδίο μάχης,** *Στρατιωτική Επιθεώρηση, Νοέμβριος-Δεκέμβριος, ΓΕΣ.*

Καλλιγκάτσης Ι., Μαυρομμάτης Γ., Πετραντωνάκης Π., Παναγιωτόπουλος Ι.-Χ. (2002), **Παραπλάνηση Χρηστών στην Προσπέλαση Πληροφορίας,** *Πρακτικά 15ου Πανελλήνιου Συνεδρίου Ελληνικής Εταιρίας Επιχειρησιακών Ερευνών, Τρίπολη.*

Κάτσικας Σ. (1995), ***Διαχείριση Κινδύνων Πληροφοριακών Συστημάτων,*** *Ασφάλεια Πληροφοριών Ελληνική Εταιρεία Επιστημόνων Ηλεκτρονικών Υπολογιστών και Πληροφορικής, έκδοση 1η .*

Κηρυττόπουλος Κ., Διαμάντας Β., Λεώπουλος Β., (2005), **Μέθοδοι εντοπισμού κινδύνων σε έργα: Πρακτικές και Συμβουλές,** *Πρακτικά 17ου Πανελλήνιου Συνεδρίου Ελληνικής Εταιρίας Επιχειρησιακών Ερευνών, Πάτρα, 403-413.*

Κιουντούζης Ε. (1995), **Μοντέλα Ασφάλειας Πληροφοριακού Συστήματος,** *Ασφάλεια Πληροφοριών Ελληνική Εταιρεία Επιστημόνων Ηλεκτρονικών Υπολογιστών και Πληροφορικής, έκδοση 1η .*

Μαυρομμάτης Γ., Καλλιγκάτσης Ι., Πετραντωνάκης Π., Παναγιωτόπουλος Ι.-Χ. (2002), **Συλλογή Μέγιστης Πληροφόρησης από Πηγές Πληροφορίας,** *Πρακτικά 15ου Πανελλήνιου Συνεδρίου Ελληνικής Εταιρίας Επιχειρησιακών Ερευνών, Τρίπολη.*

Μπέλλιας Ν. Σ. (2002), **Πολεμικά παίγνια στο στρατιωτικό γίγνεσθαι,** *Στρατιωτική Επιθεώρηση, Νοέμβριος-Δεκέμβριος, ΓΕΣ.*

Μπουντής Α. (1988), **Δυναμικά Συστήματα και Χάος,** *Παπασωτηρίου Α.Ε., Αθήνα.*

Πετραντωνάκης Π., Καλλιγκάτσης Ι., Μαυρομμάτης Γ., Παναγιωτόπουλος Ι.-Χ. (2002), **Θεωρήσεις και Προβληματισμοί σε Συστήματα Ανάκτησης από Πτώσεις,** *Πρακτικά 15ου Πανελλήνιου Συνεδρίου Ελληνικής Εταιρίας Επιχειρησιακών Ερευνών, Τρίπολη.*

Πετραντωνάκης Π., Παναγιωτόπουλος Ι.-Χ. (2003), **ΔΙΟΙΚΗΣΗ ΚΙΝΔΥΝΟΥ ΣΕ ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ ΕΘΝΙΚΗΣ ΑΜΥΝΑΣ,** *Πρακτικά 16ου Πανελλήνιου Συνεδρίου Ελληνικής Εταιρίας Επιχειρησιακών Ερευνών, Λάρισα.*

Πετραντωνάκης Π., Παναγιωτόπουλος Ι.-Χ. (2004), **Ψηφιακά συστήματα αυτόματης αποκλιμάκωσης κρίσεων,** *Γεωστρατηγική, τεύχος 6, 179-188.*

Πετραντωνάκης Π., Παναγιωτόπουλος Ι.-Χ. (2005), **Αντιμετώπιση Κινδύνου με ελάχιστο κόστος σε Εχθρικά Περιβάλλοντα Λειτουργίας,** *Πρακτικά 1ου Συνεδρίου Εταιρείας Συστημικών Μελετών, Τρίπολη.*

Πετραντωνάκης Π, Παναγιωτόπουλος Ι.-Χ. (2005), **Προοπτικές ανάπτυξης πυρηνικών εργοστάσιων για την παραγωγή ηλεκτρικής ενέργειας στην Ελλάδα,** *Ινστιτούτο Αμυντικών Αναλύσεων*.

Χαΐνης Ι. Θ. (2005), **Εισαγωγή στη Θεωρία Καταστροφών,** *Εκδόσεις Φούντας, Αθήνα.*