ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΗΣ (OBJECT ORIENTED) ΑΝΑΛΥΣΗ ΚΑΙ ΣΧΕΔΙΑΣΜΟΣ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΜΕΛΕΤΗ ΣΥΣΤΗΜΑΤΟΣ ΚΥΡΙΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΠΑΡΑΓΩΓΗΣ (MPS) & ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΑΠΑΙΤΗΣΕΩΝ ΥΛΙΚΩΝ (MRP)

Η εργασία υποβάλλεται για την μερική κάλυψη των απαιτήσεων με στόχο την απόκτηση του διπλώματος

ΜΕΤΑΠΤΥΧΙΑΚΟ ΔΙΠΛΩΜΑ ΕΙΔΙΚΕΥΣΗΣ : ΕΦΟΔΙΑΣΜΟΣ & ΔΙΑΚΙΝΗΣΗ ΠΡΟΪΟΝΤΩΝ (LOGISTICS)

από

ΤΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ ΚΑΙ ΤΟ ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΤΑΥΡΟΡΡΑΒΔΗΣ ΧΡΗΣΤΟΣ

ΤΜΗΜΑ ΒΙΟΜΗΧΑΝΙΚΗΣ ΔΙΟΙΚΗΣΗΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ, ΟΚΤΩΒΡΙΟΣ 2002

ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΗΣ (OBJECT ORIENTED) ΑΝΑΛΥΣΗ ΚΑΙ ΣΧΕΔΙΑΣΜΟΣ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΜΕΛΕΤΗ ΣΥΣΤΗΜΑΤΟΣ ΚΥΡΙΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΠΑΡΑΓΩΓΗΣ (MPS) & ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΑΠΑΙΤΗΣΕΩΝ ΥΛΙΚΩΝ (MRP)

Η εργασία υποβάλλεται για την μερική κάλυψη των απαιτήσεων με στόχο την απόκτηση του διπλώματος

ΜΕΤΑΠΤΥΧΙΑΚΟ ΔΙΠΛΩΜΑ ΕΙΔΙΚΕΥΣΗΣ : ΕΦΟΔΙΑΣΜΟΣ & ΔΙΑΚΙΝΗΣΗ ΠΡΟΪΟΝΤΩΝ (LOGISTICS)

από

ΤΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ ΚΑΙ ΤΟ ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΤΑΥΡΟΡΡΑΒΔΗΣ ΧΡΗΣΤΟΣ

ΤΜΗΜΑ ΒΙΟΜΗΧΑΝΙΚΗΣ ΔΙΟΙΚΗΣΗΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ, ΟΚΤΩΒΡΙΟΣ 2002

ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΗΣ (OBJECT ORIENTED) ΑΝΑΛΥΣΗ ΚΑΙ ΣΧΕΔΙΑΣΜΟΣ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΜΕΛΕΤΗ ΣΥΣΤΗΜΑΤΟΣ ΚΥΡΙΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΠΑΡΑΓΩΓΗΣ (MPS) & ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΑΠΑΙΤΗΣΕΩΝ ΥΛΙΚΩΝ (MRP)

Η εργασία υποβάλλεται για την μερική κάλυψη των απαιτήσεων με στόχο την απόκτηση του διπλώματος

ΜΕΤΑΠΤΥΧΙΑΚΟ ΔΙΠΛΩΜΑ ΕΙΔΙΚΕΥΣΗΣ : ΕΦΟΔΙΑΣΜΟΣ & ΔΙΑΚΙΝΗΣΗ ΠΡΟΪΟΝΤΩΝ (LOGISTICS)

από

ΤΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ ΚΑΙ ΤΟ ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΤΑΥΡΟΡΡΑΒΔΗΣ ΧΡΗΣΤΟΣ

ΤΜΗΜΑ ΒΙΟΜΗΧΑΝΙΚΗΣ ΔΙΟΙΚΗΣΗΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ, ΟΚΤΩΒΡΙΟΣ 2002

Πρόλογος

Με την ολοκλήρωση της Διπλωματικής μου Εργασίας, επιθυμώ να ευχαριστήσω θερμά :

- Τον Επίκουρο Καθηγητή του Τμήματος Βιομηχανικής Διοίκησης και Τεχνολογίας του Πανεπιστημίου Πειραιώς Κ° Γρηγόρη Χρονδροκούκη, για την εμπιστοσύνη που έδειξε στο πρόσωπό μου με την ανάθεση της εργασίας αυτής και για την πολύτιμη υποστήριξη που μου παρείχε καθ' όλη τη διάρκεια της εκπόνησής της.

- Το Διευθυντή Μηχανογράφησης της Βιομηχανίας Παρασκευής Ορών ΒΙΟΣΕΡ Α.Ε. Κ° Σταμάτη Μπίλα, για τις αξιόλογες επιστημονικές του παρατηρήσεις, το ενδιαφέρον του και την αμέριστη βοήθειά του σε όλα τα στάδια της Διπλωματικής Εργασίας.

- Τέλος, τους συναδέλφους και συνεργάτες μου στην Εταιρεία P.I.Systems A.E., για τις υποδείξεις και τις χρήσιμες συμβουλές τους κατά τη συγγραφή του πρακτικού μέρους, συμβάλλοντας έτσι ουσιαστικά ώστε η εργασία να πάρει την τελική της μορφή.

Περίληψη

Η παρούσα Διπλωματική Εργασία εκπονήθηκε στα πλαίσια των ακαδημαϊκών υποχρεώσεων του 4ου εξαμήνου του Μεταπτυχιακού Προγράμματος Σπουδών στην Οργάνωση και Διοίκηση Βιομηχανικών Συστημάτων με ειδίκευση στον Εφοδιασμό & Διακίνηση Προϊόντων (LOGISTICS), του Τμήματος Βιομηχανικής Διοίκησης και Τεχνολογίας του Πανεπιστημίου Πειραιώς, υπό την επίβλεψη του Επίκουρου Καθηγητή Κου Γ. Χονδροκούκη.

Όπως αναφέρεται και στον τίτλο, αντικείμενο της Διπλωματικής Εργασίας είναι η Αντικειμενοστραφής Ανάλυση και ο Σχεδιασμός (Object Oriented Analysis and Design – OOA/D) ενός πληροφοριακού συστήματος για την Κατάστρωση του Κύριου Προγράμματος Παραγωγής (Master Production Schedule – MPS) και τον Προγραμματισμό Απαιτήσεων Υλικών (Material Requirements Planning – MRP) Βιομηχανικών Επιχειρήσεων.

Η εργασία αποτελείται από τρία μέρη :

Στο πρώτο μέρος, εξετάζεται η μεθοδολογία Unified Process, μια ανοιχτή πρότυπη μέθοδος η οποία αποτελεί την πιο διαδεδομένη Object Oriented μεθοδολογία ανάπτυξης πληροφοριακών συστημάτων. Δίνεται έμφαση στην περιγραφή των δύο πρώτων χρονικά φάσεων της μεθοδολογίας (Inception και Elaboration), δεδομένου ότι στις φάσεις αυτές λαμβάνουν χώρα κατά κύριο λόγο οι διαδικασίες της ανάλυσης και του σχεδιασμού, που αποτελούν και το αντικείμενο της εργασίας.

Στο πρώτο αυτό μέρος, δίνονται επίσης τα βασικά σημεία της συμβολικής γλώσσας UML (Unified Modeling Language). Η UML αποτελεί ένα ανοιχτό βιομηχανικό πρότυπο και έχει πλέον καθιερωθεί σαν η βασική σημειογραφία για τη διαγραμματική απεικόνιση των μοντέλων και την τεκμηρίωση των παραδοτέων ενός Object Oriented πληροφοριακού συστήματος.

Στο δεύτερο μέρος, γίνεται μια σύντομη ανασκόπηση της εξέλιξης των πληροφοριακών συστημάτων διοίκησης παραγωγής και στη συνέχεια παρουσιάζεται το θεωρητικό υπόβαθρο των Πινάκων Υλικών (Bill of Materials), του Κύριου Προγραμματισμού Παραγωγής (MPS) και του Προγραμματισμού Απαιτήσεων Υλικών (MRP). Ουσιαστικά, οι έννοιες που αναλύονται στο δεύτερο μέρος, αποτελούν το Επιχειρηματικό Μοντέλο (Business Model), πάνω στο οποίο βασίζεται η ανάπτυξη του συστήματος.

Στο τρίτο και τελευταίο μέρος της εργασίας, αναλύεται και σχεδιάζεται το υπό ανάπτυξη σύστημα, μέσα στα πλαίσια που θέτει η μεθοδολογία Unified Process, ενώ για την αποτύπωση των διαγραμμάτων και των μοντέλων χρησιμοποιείται η σημειογραφία UML. Τα παραδοτέα που παρατίθενται είναι κατά σειρά : ο Σκοπός του συστήματος (Vision), οι Λειτουργικές Απαιτήσεις του συστήματος – Use Cases, τα Πρωτότυπα των Οθονών, το Γλωσσάρι Όρων, το Domain Model, τα Διαγράμματα Αλληλεπίδρασης των Αντικειμένων (Sequence Diagrams), το Διάγραμμα Κλάσεων (Class Diagram) του συστήματος και η Γραμμογράφηση της Βάσης Δεδομένων (Database Schema). Τέλος, στα Παραρτήματα δίνεται το Script δημιουργίας της βάσης δεδομένων και ο σκελετός του κώδικα της εφαρμογής.

Περιεχόμενα

ΜΕΡΟΣ 3ο

ΜΕΡΟΣ 1ο

1.1. Object Oriented Ανάλυση και Σχεδιασμός Πληροφοριακών Συστημάτων

Η τεχνική της Object Oriented Ανάλυσης και Σχεδιασμού Πληροφοριακών Συστημάτων (Object Oriented Analysis and Design – OOA/D) έκανε ουσιαστικά την εμφάνισή της στις αρχές της δεκαετίας του 1990. Οι Object Oriented (Αντικειμενοστραφείς) μεθοδολογίες που αναπτύχθηκαν από τότε, ενσωματώνουν και οργανώνουν μαζί, τόσο τις απαιτούμενες πληροφορίες όσο και τις διαδικασίες που είναι απαραίτητες για τη διαχείριση των πληροφοριών, βασιζόμενες στα αντικείμενα (objects) του πραγματικού κόσμου που οι πληροφορίες αυτές περιγράφουν. Παρά το ότι η OOA/D αποτελεί σχετικά νέα τεχνική, έχει ήδη υιοθετηθεί από τη βιομηχανία της πληροφορικής, σαν η νέα κατεύθυνση για την ανάπτυξη των πληροφοριακών συστημάτων. Δύο είναι οι βασικότεροι λόγοι που συνετέλεσαν στο γεγονός αυτό :

- Η επιτυχία και η ταχύτητα με την οποία διαδόθηκε ο αντικειμενοστραφής προγραμματισμός.
- Η αναγνώριση ότι οι Object Oriented μεθοδολογίες παράγουν πιο αποτελεσματικά, αποδοτικά, ευέλικτα και σταθερά πληροφοριακά συστήματα.

Η Object Oriented τεχνική, δεν είναι μία ακόμα μεθοδολογία που ήρθε να προστεθεί στις ήδη υπάρχουσες, αλλά αποτελεί μια νέα φιλοσοφία και έναν νέο τρόπο σκέψης. Η αποτελεσματική χρήση της προϋποθέτει πέρα από τη γνώση μιας object oriented γλώσσας προγραμματισμού, την κατανόηση σε βάθος των objects και του τρόπου λειτουργίας τους.

Για την επιτυχημένη εφαρμογή της Object Oriented ανάλυσης και του σχεδιασμού πληροφοριακών συστημάτων, τρεις είναι οι παράγοντες πάνω στους οποίους βασίζεται η επιτυχία ενός Project :

- Η **Σημειογραφεία** (Notation), η οποία είναι η «συμβολική» γλώσσα επικοινωνίας μεταξύ των εμπλεκομένων στο έργο και χρησιμοποιείται για τη διαγραμματική απεικόνιση των μοντέλων. Αποτελεί επίσης ένα άριστο μέσο τόσο για την κατανόηση όσο και την τεκμηρίωση των παραδοτέων του έργου.
- Η **Μεθοδολογία** (Process). Η μεθοδολογία ανάπτυξης πληροφοριακών συστημάτων (Software Development Process ή Software Engineering Process) περιγράφει τη διαδικασία μέσω της οποίας οι απαιτήσεις ενός συστήματος, υλοποιούνται σε Λογισμικό.
- Το **Εργαλείο Ανάπτυξης** (CASE Tool = Computer Assisted Software Engineering Tool). Το εργαλείο ανάπτυξης είναι ένα εμπορικό λογισμικό, η χρήση του οποίου διευκολύνει τα μέγιστα στην κατασκευή των διαγραμμάτων, των μοντέλων και γενικότερα όλων των παραδοτέων (κάθε Εργαλείο Ανάπτυξης υποστηρίζει μια συγκεκριμένη Σημειογραφία). Πολύ γνωστά CASE Tools είναι το Rational Rose της Rational, ο Power Designer της Sybase, το Erwin της Computer Associations, ο Visual Modeler της Microsoft κ.α.

Τι εννοούμε όμως με τους όρους Ανάλυση και Σχεδιασμός :

Η **Ανάλυση** εστιάζεται στη διερεύνηση του υπό εξέταση προβλήματος και όχι στον τρόπο επίλυσής του. Είναι ένας αρκετά γενικός όρος και περιλαμβάνει την Ανάλυση Απαιτήσεων (Requirements Analysis), όπου γίνεται μελέτη και καταγραφή των απαιτήσεων του συστήματος, και την Ανάλυση των Objects, η έμφαση της οποίας είναι στον εντοπισμό και την περιγραφή των αντικειμένων (εννοιών) του προβλήματος, των ιδιοτήτων αυτών (attributes) και των μεταξύ τους συσχετίσεων (associations).

Ο **Σχεδιασμός** αποσκοπεί στη σχεδίαση μιας λύσης για την ικανοποίηση των απαιτήσεων. Περιλαμβάνει το Σχεδιασμό των Objects, όπου καθορίζονται τα προγραμματιστικά αντικείμενα (software objects) και ο τρόπος που αλληλεπιδρούν μεταξύ τους, και το Σχεδιασμό της Βάσης Δεδομένων για την αποθήκευση των πληροφοριών του συστήματος. Ο σχεδιασμός αποτελεί το σκελετό πάνω στον οποίο τελικά κατασκευάζεται το πληροφοριακό σύστημα με τη συγγραφή του κώδικα.

Εν συντομία η διαφορά μεταξύ Ανάλυσης και Σχεδιασμού συνοψίζεται στην εξής φράση :

"Do the right thing (Analysis), and Do the thing right (Design)".

1.2. Η Σημειογραφία UML

Στη δεκαετία του 1990, στο χώρο της Πληροφορικής έκαναν την εμφάνισή τους αρκετές μεθοδολογίες. Κάθε μεθοδολογία είχε δυνατά και αδύνατα σημεία και έδινε έμφαση σε έναν συγκεκριμένο τομέα, ενώ συνοδευόταν και από τη δική της σημειογραφία.

Η χρήση πολλών και διαφορετικών μεταξύ τους σημειογραφιών, είχε σαν αποτέλεσμα τη δημιουργία σύγχυσης στον κόσμο της Πληροφορικής, καθότι κάθε σύμβολο είχε διάφορες ερμηνείες, ανάλογα με τη μεθοδολογία στην οποία αναφερόταν. Ο όρος που χρησιμοποιήθηκε για να περιγράψει την κατάσταση εκείνης της περιόδου, ήταν «ο πόλεμος των μεθοδολογιών».

Το τέλος της περιόδου αυτής σηματοδοτήθηκε με τη δημιουργία της σημειογραφίας UML (Unified Modeling Language), η πρώτη draft μορφή της οποίας εκδόθηκε τον Οκτώβριο του 1995, ενώ το Νοέμβριο του 1997, υιοθετήθηκε από τον οργανισμό OMG (Object Management Group). Ξεκίνησε από την προσπάθεια των Booch και Rumbaugh το 1994 για την ενοποίηση της σημειογραφίας που χρησιμοποιούσαν ο καθένας στη δική του μέθοδο (Booch και OMT αντίστοιχα), ενώ στη συνέχεια συνεισέφερε στην προσπάθεια αυτή και ο Jacobson (δημιουργός της μεθοδολογίας Objectory). Από τότε, η ομάδα αυτή έγινε γνωστή ως "the three amigos".

Η UML είναι ένα ανοιχτό βιομηχανικό πρότυπο και αποτελεί αυτή τη στιγμή την πιο διαδεδομένη και καθιερωμένη γλώσσα-σημειογραφία object oriented μοντελοποίησης. Είναι μια συμβολική γλώσσα η οποία τυποποιεί και προδιαγράφει, απεικονίζει και τεκμηριώνει τα παραδοτέα ενός υπό ανάπτυξη object oriented συστήματος (όχι απαραίτητα πληροφοριακού συστήματος).

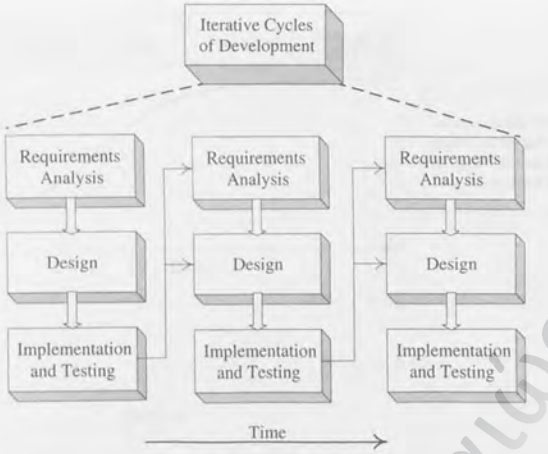
1.3. Η Μεθοδολογία Unified Process (UP)

Η ανάπτυξη ενός πληροφοριακού συστήματος είναι μια πολύπλοκη διαδικασία και περιλαμβάνει ένα πλήθος ενεργειών και δραστηριοτήτων που πρέπει να εκτελεστούν μεθοδικά και συντονισμένα. Η επιτυχία ενός τέτοιου εγχειρήματος εξαρτάται σε πολύ μεγάλο βαθμό από τη μεθοδολογία ανάπτυξης που θα ακολουθηθεί, η οποία θέτει το πλαίσιο μέσα στο οποίο θα κινηθεί η ομάδα ανάπτυξης του λογισμικού καθ' όλη τη διάρκεια κατασκευής του συστήματος.

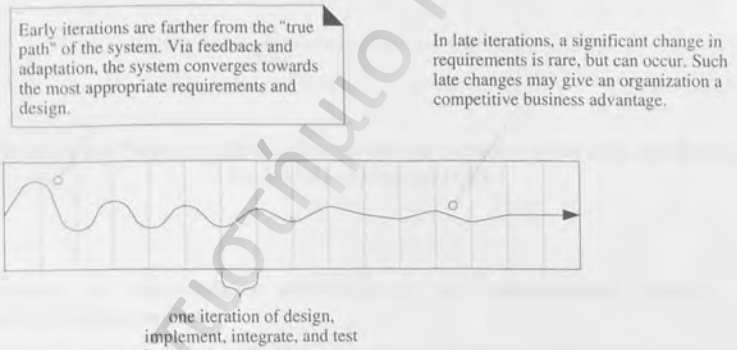
Η μεθοδολογία Unified Process (Unified Software Development Process) είναι μια ανοιχτή πρότυπη μέθοδος ανάπτυξης πληροφοριακών συστημάτων. Προήλθε από το ίδιο project από το οποίο γεννήθηκε και η UML, ενώ αυτή τη στιγμή αποτελεί την πιο διαδεδομένη object oriented μεθοδολογία για την ανάπτυξη πληροφοριακών συστημάτων. Συνδυάζει πολλές κοινά αποδεκτές άριστες πρακτικές (best practices), σημαντικότερη των οποίων θεωρείται η σταδιακή ανάπτυξη του συστήματος μέσα από έναν αριθμό επαναλήψεων (iterations).

Μέσα στα πλαίσια μιας επαναληπτικής μεθοδολογίας ανάπτυξης, η υλοποίηση του συστήματος επιτελείται σε έναν αριθμό επαναλήψεων. Κάθε μια από τις επαναλήψεις αυτές, περιλαμβάνει τις δικές της διαδικασίες καταγραφής απαιτήσεων, ανάλυσης και σχεδιασμού, κατασκευής και ελέγχου (Testing) και έτσι από κάθε επανάληψη προκύπτει ένα μέρος του ολοκληρωμένου συστήματος. Ύστερα από κάθε επανάληψη υπάρχει σημαντική ανατροφοδότηση πληροφορίας (feedback), η οποία χρησιμοποιείται στην επόμενη επανάληψη (Σχήμα 1). Η χρονική διάρκεια κάθε επανάληψης προτείνεται να είναι 2 έως 6 εβδομάδες, το πολύ 8. Μόνο για μεγάλες ομάδες ανάπτυξης που ξεπερνούν τα 100 άτομα και για έργα που πρόκειται να διαρκέσουν πολλά χρόνια, η διάρκεια κάθε επανάληψης μπορεί να είναι μεγαλύτερη.

Το αποτέλεσμα κάθε επανάληψης δεν είναι ένα δοκιμαστικό πρωτότυπο του συστήματος, το οποίο θα είναι άχρηστο στη συνέχεια, αλλά αποτελεί πραγματική, λειτουργική έκδοση ενός τμήματος του τελικού συστήματος. Έτσι σε κάθε επανάληψη υπάρχει ένα έτοιμο τμήμα του συστήματος το οποίο μπορεί να δοκιμαστεί και να αξιολογηθεί από τους χρήστες και όλους τους άλλους ενδιαφερόμενους. Υπάρχει λοιπόν σημαντικό feedback το οποίο λαμβάνεται υπόψη στην επόμενη επανάληψη για περαιτέρω ανάλυση και βελτίωση του σχεδιασμού. Με τον τρόπο αυτό το σύστημα χτίζεται και αναπτύσσεται σταδιακά μέσα από έναν αριθμό επαναλήψεων, βελτιώνεται και ραφινάρεται από επανάληψη σε επανάληψη, συγκλίνοντας ομαλά προς την τελική του μορφή (Σχήμα 2).



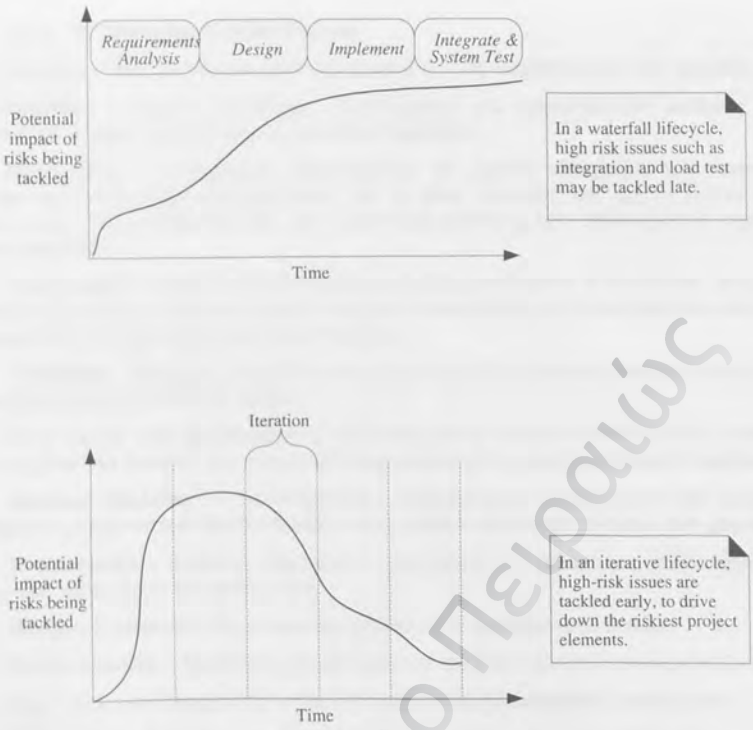
Σχήμα 1. Σταδιακή Ανάπτυξη ενός Συστήματος σε Διαδοχικές Επαναλήψεις.



Σχήμα 2. Ομαλή Σύγκλιση του Συστήματος προς την Τελική του Μορφή, μέσα από έναν Αριθμό Επαναλήψεων.

Αντιμετωπίζοντας μάλιστα τα πιο κρίσιμα θέματα στις πρώτες επαναλήψεις, ελαχιστοποιείται ο κίνδυνος να προκύψουν σημαντικά προβλήματα προς το τέλος του έργου, όταν το κόστος για την αντιμετώπισή τους θα είναι πολύ μεγάλο ή μπορεί να οδηγήσουν σε αποτυχία του συστήματος.

Το γεγονός αυτό αποτελεί το σημαντικότερο ίσως πλεονέκτημα μιας επαναληπτικής μεθόδου έναντι της κλασικής μεθόδου ανάπτυξης πληροφοριακών συστημάτων, γνωστή σαν μέθοδος «Καταρράκτης» (Waterfall). Η μέθοδος αυτή βασίζεται στη σχετικά αυστηρή, σειριακή εφαρμογή των κύριων διαδικασιών ανάπτυξης ενός συστήματος, δηλαδή καταγραφή απαιτήσεων, ανάλυση, σχεδιασμός, κατασκευή και έλεγχος, με την προϋπόθεση και τη δέσμευση ότι καθ' όλη τη διάρκεια της ανάπτυξης, οι απαιτήσεις του συστήματος που καταγράφηκαν στο πρώτο στάδιο, παραμένουν σταθερές και αμετάβλητες (Σχήμα 3).



Σχήμα 3. Σύγκριση του Ρίσκου μεταξύ μιας Επαναληπτικής Μεθόδου Ανάπτυξης ενός Συστήματος και της Μεθόδου «Καταρράκτης».

Συνοπτικά, τα σημαντικότερα πλεονεκτήματα μιας επαναληπτικής μεθόδου ανάπτυξης πληροφοριακών συστημάτων είναι :

- Εντοπισμός και αντιμετώπιση των θεμάτων που εμπεριέχουν υψηλό ρίσκο στην αρχή παρά στο τέλος της ανάπτυξης του συστήματος (τεχνικά θέματα, απαιτήσεις, περιορισμοί κτλ).
- Ορατή πρόοδος στο σύστημα από τις πρώτες φάσεις, εφόσον από την πρώτη κιόλας επανάληψη υπάρχουν αρκετά παραδοτέα αλλά και πραγματικό, έστω και μικρό, τμήμα του συστήματος που λειτουργεί.
- Ανάμιξη των χρηστών και feedback από τις πρώτες κιόλας φάσεις, που οδηγεί σε σταδιακό ραφινάρισμα του συστήματος και συνεχή προσαρμογή στις πραγματικές ανάγκες των χρηστών.
- Έλεγχος και αποτελεσματική διαχείριση της πολυπλοκότητας που εμπεριέχει η ανάπτυξη ενός μεγάλου συστήματος, μέσω του διαχωρισμού της ανάπτυξης σε μικρά και καλά καθορισμένα βήματα.
- Εκμάθηση και εμπειρία στην ίδια τη μεθοδολογία, μέσα από την επανάληψη των ίδιων βημάτων, διαδικασιών και παραδοτέων από iteration σε iteration.

1.3.1. Η Δομή της Unified Process

Ένα project που υλοποιείται μέσα στο πλαίσιο της UP, ολοκληρώνεται σε 4 χρονικές φάσεις :

Inception – Αρχική Διερεύνηση. Περιλαμβάνει μια προκαταρκτική ανάλυση και αρχικές εκτιμήσεις για το έργο, κάτι σαν αρχική μελέτη σκοπιμότητας.

Elaboration – Επεξεργασία. Περιλαμβάνει τη βασική καταγραφή των απαιτήσεων, τον εντοπισμό των απαιτήσεων υψηλού ρίσκου για το έργο, εκτίμηση και αρχικό έλεγχο της βασικής αρχιτεκτονικής του συστήματος και πιο ρεαλιστική εκτίμηση των απαιτούμενων πόρων και του χρονοδιαγράμματος.

Construction – Κατασκευή. Στη φάση αυτή πραγματοποιείται η κατασκευή του μεγαλύτερου τμήματος του συστήματος και ραφινάρονται και οριστικοποιούνται οι λεπτομέρειες των απαιτήσεων, του σχεδιασμού και της αρχιτεκτονικής του συστήματος.

Transition – Εφαρμογή. Στη φάση αυτή ολοκληρώνεται ο έλεγχος (testing) του συστήματος και υλοποιείται η εφαρμογή του στον πελάτη.

Όσον αφορά στις δραστηριότητες που απαιτούνται για την ανάπτυξη ενός συστήματος στα πλαίσια της Unified Process, ένα project περιλαμβάνει τις εξής κύριες διαδικασίες (disciplines) :

Business Modeling – Μοντελοποίηση Προβλήματος. Αναγνώριση και καταγραφή των επιθυμητών χαρακτηριστικών και δυνατοτήτων του συστήματος και των αναγκών των χρηστών.

Requirements – Ανάλυση Απαιτήσεων. Καθορισμός του σκοπού και των λειτουργικών και μη λειτουργικών απαιτήσεων του συστήματος.

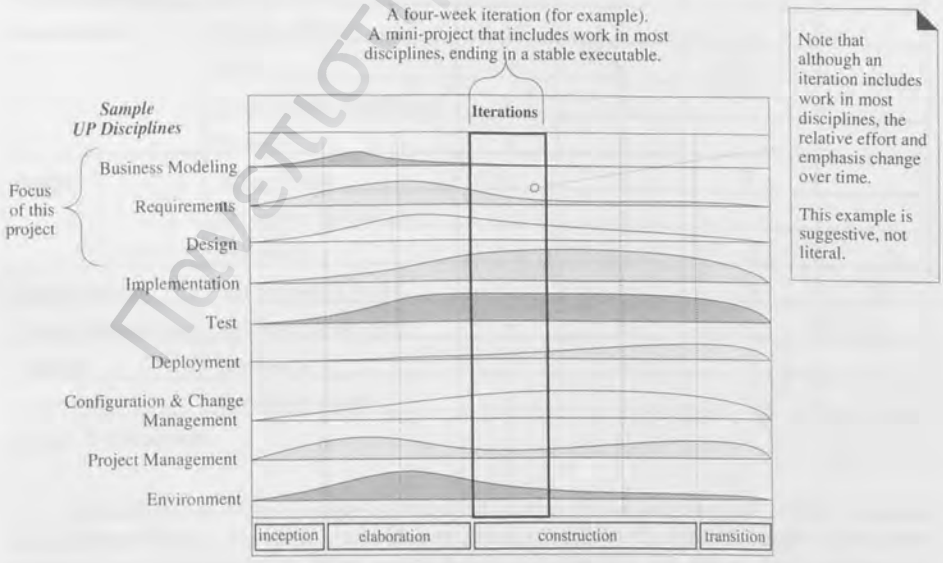
Design – Σχεδιασμός. Περιγραφή του τρόπου κατασκευής του συστήματος.

Implementation – Κατασκευή. Κατασκευή του συστήματος με τη συγγραφή κώδικα.

Test – Έλεγχος. Έλεγχος της ορθής λειτουργίας του ολοκληρωμένου συστήματος.

Deployment – Εγκατάσταση. Εγκατάσταση του συστήματος και εκπαίδευση των χρηστών.

Στο Σχήμα 4 που ακολουθεί, παρουσιάζεται η συσχέτιση των διαδικασιών – φάσεων στη UP.



Σχήμα 4. Συσχέτιση Διαδικασιών – Φάσεων της Unified Process.

Από το διάγραμμα αυτό γίνεται φανερό ότι κάθε διαδικασία, τυπικά εφαρμόζεται σε κάθε φάση υλοποίησης του έργου ανάπτυξης. Βέβαια ο βαθμός και η ένταση στην οποία μια συγκεκριμένη διαδικασία εφαρμόζεται, μεταβάλλεται στη διάρκεια του χρόνου και εξαρτάται άμεσα από τη φάση ανάπτυξης μέσα στην οποία διεξάγεται. Για παράδειγμα, η φάση της Επεξεργασίας εστιάζεται κυρίως στις διαδικασίες ανάλυσης των απαιτήσεων και του σχεδιασμού, ενώ στη φάση της Κατασκευής δίνεται έμφαση στη συγγραφή του κώδικα.

Στη συνέχεια της διπλωματικής εργασίας θα ασχοληθούμε με τις δύο πρώτες φάσεις της UP (Inception και Elaboration), δεδομένου ότι στις φάσεις αυτές λαμβάνουν χώρα κατά κύριο λόγο οι διαδικασίες της ανάλυσης και του σχεδιασμού, που αποτελούν και το αντικείμενο της εργασίας αυτής.

1.3.2. Τα Παραδοτέα της Unified Process

Στη Unified Process, με τον όρο παραδοτέα (artifacts), εννοούμε το αποτέλεσμα κάθε δραστηριότητας, όπως κείμενα, διαγράμματα, μοντέλα, κώδικας κτλ. Το σύνολο των παραδοτέων για ένα έργο είναι ένα υποσύνολο του συνόλου των παραδοτέων που προτείνει η UP. Στον Πίνακα 1 που ακολουθεί, παρατίθενται τα βασικότερα παραδοτέα της UP.

Κάθε έργο έχει τις δικές του ανάγκες και διάφορα παραδοτέα είναι κατάλληλα για διάφορους τύπους συστημάτων προς ανάπτυξη. Το σύνολο των παραδοτέων που η ομάδα ανάπτυξης αποφασίζει να κατασκευάσει για ένα συγκεκριμένο σύστημα περιλαμβάνεται σε ένα έγγραφο που ονομάζεται Development Case. Το παραδοτέο αυτό, συντάσσεται στη διαδικασία Environment, στην οποία η όλη μεθοδολογία προσαρμόζεται στις ανάγκες του εκάστοτε έργου και γίνεται επιλογή των εργαλείων ανάπτυξης. Το μόνο παραδοτέο που δεν είναι προαιρετικό είναι βέβαια το εκτελέσιμο σύστημα.

Πίνακας 1. Βασικά Παραδοτέα της UP ανά Διαδικασία.

Discipline	Παραδοτέο Επανάληψη→	Inception I1	Elaboration E1…En	Construction C1…Cn	Transition T1…Tn
Business Modeling	Domain Model		A		
Requirements	Use Case Model	A	E		
	Vision	A	E		
	Supplementary Specification	A	E		
	Glossary	A	E		
Design	Design Model		A	E	
	SW Architecture Document		A		
	Data Model		A	E	
Implementation	Implementation Model		A	E	E
Project Management	SW Development Plant	A	E	E	E
Testing	Test Model		A	E	
Environment	Development Case	A	E		

A=Αρχή, E=Επεξεργασία

Είναι ιδιαίτερα χρήσιμο, τα διάφορα παραδοτέα ενός έργου να βρίσκονται σε ηλεκτρονική μορφή και με υπεσυνδέσμους (hyperlinks) να συνδέονται μεταξύ τους. Για να επιτευχθεί αυτό, καλό είναι τα παραδοτέα να συντηρούνται σε ειδικά για το σκοπό αυτό εργαλεία και όχι σε απλά έγγραφα ενός επεξεργαστή κειμένου. Για παράδειγμα το Γλωσσάρι μπορεί να είναι αποθηκευμένο σε μια Βάση Δεδομένων και τα Use Cases σε εργαλεία ανάπτυξης λογισμικού (CASE Tools).

1.4. Η Φάση Αρχικής Διερεύνησης (Inception)

Η Αρχική Διερεύνηση (Inception) είναι η πρώτη φάση της UP. Ο βασικός της σκοπός είναι να καθορίσει το στόχο του συστήματος, τη θεματική και επιχειρηματική του περιοχή, να καταγράψει και να αναλύσει τις βασικές απαιτήσεις του συστήματος, να εξετάσει το πόσο εφικτός είναι ο στόχος και η ολοκλήρωση του έργου και να κάνει μια πολύ χονδρική εκτίμηση των πόρων, του κόστους και του χρόνου που θα απαιτηθούν. Είναι μια πολύ μικρή φάση χρονικά, καθώς η διάρκειά της για συνήθη έργα κυμαίνεται από 1 έως 3 το πολύ εβδομάδες.

Πολλά από τα παραδοτέα της UP αρχίζουν στην φάση της Inception (Πίνακας 2) αλλά συμπληρώνονται και ολοκληρώνονται πολύ αργότερα, στη φάση του Elaboration ή του Construction. Για παράδειγμα στο μοντέλο των Use Cases, αναφέρονται τα ονόματα των περισσότερων Use Cases, συνήθως όμως μόνο το 10% αυτών κατασκευάζεται αναλυτικά. Τα Use Cases που αναλύονται στην αρχική αυτή φάση, είναι αυτά που βοηθούν στην κατανόηση του όλου προβλήματος και ενέχουν το μεγαλύτερο ρίσκο.

Στις επόμενες ενότητες θα αναφερθούμε στα τέσσερα πρώτα παραδοτέα της φάσης Inception.

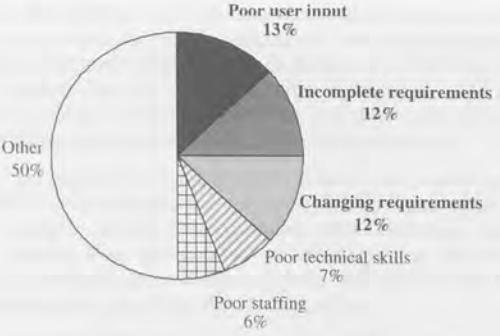
Πίνακας 2. Βασικά Παραδοτέα της φάσης Inception.

Παραδοτέο	Σχόλια
Vison (Σκοπός)	Περιγραφή των υψηλού επιπέδου στόχων και περιορισμών του συστήματος.
Use Case Model	Καταγραφή των Λειτουργικών και των σχετιζόμενων μη Λειτουργικών Απαιτήσεων.
Supplementary Specification (Συμπληρωματικές Απαιτήσεις)	Καταγραφή των Υπόλοιπων Απαιτήσεων.
Glossary (Γλωσσάρι Όρων)	Γλωσσάρι όρων της θεματικής περιοχής του προβλήματος.
Iteration Plan	Περιγραφή των ενεργειών της πρώτης επανάληψης της επόμενης φάσης (Elaboration).
Phase Plan & Software Development Plan	Εκτίμηση της διάρκειας της φάσης Elaboration που ακολουθεί, και των πόρων που θα απαιτηθούν για την υλοποίησή της.
Development Case	Προσαρμογή της UP στις ανάγκες του έργου, επιλογή Παραδοτέων.

1.5. Απαιτήσεις (Requirements)

Η βασικότερη ίσως διαδικασία για την ανάπτυξη ενός συστήματος είναι να βρεθούν και να τεκμηριωθούν, με ολοκληρωμένο και εύληπτο για όλους τους ενδιαφερόμενους τρόπο, οι διάφορες απαιτήσεις του συστήματος (Requirements). Σε μελέτη σχετικά με τους παράγοντες που συντελούν στην επιτυχημένη ανάπτυξη λογισμικού, διαπιστώθηκε ότι το 37% των προβλημάτων που ανέκυψαν, είχε άμεση σχέση με τις απαιτήσεις του πληροφοριακού συστήματος (Σχήμα 5).

Με τον όρο απαιτήσεις, εννοούμε όλες τις δυνατότητες, τα χαρακτηριστικά και τους περιορισμούς του υπό ανάπτυξη συστήματος. Η διαδικασία εντοπισμού και καταγραφής των απαιτήσεων είναι συνεχής καθ' όλη τη διάρκεια ανάπτυξης του συστήματος (αλλά και μετά από αυτήν!). Αυτό συμβαίνει διότι δεν είναι δυνατόν να ανακαλύψει κανείς και να παγιώσει όλες τις απαιτήσεις του συστήματος στις πρώτες φάσεις. Ακόμα και αν το καταφέρει, οι απαιτήσεις ενός συστήματος αλλάζουν διαρκώς, τόσο κατά τη διάρκεια όσο και μετά την ανάπτυξή του, καθώς αυτή είναι η πραγματικότητα των επιχειρήσεων. Πρόκειται για ζωντανούς, εξελισσόμενους οργανισμούς που συνεχώς μεταλλάσσονται και εξελίσσονται. Κάθε προσπάθεια αυστηρού και άκαμπτου προκαθορισμού όλων των απαιτήσεων ενός συστήματος αποτελεί πηγή προβλημάτων και πιθανής αποτυχίας του έργου.



Σχήμα 5. Παράγοντες Επίδρασης στην Ανάπτυξη Επιτυχημένου Λογισμικού.

Οι βασικές κατηγορίες απαιτήσεων ενός συστήματος, όπως ομαδοποιούνται στη μεθοδολογία Unified Process, είναι οι παρακάτω :

Functional Requirements – Λειτουργικές Απαιτήσεις. Λειτουργικότητα του συστήματος, δυνατότητες, χαρακτηριστικά, ασφάλεια συστήματος.

Usability Requirements – Απαιτήσεις Ευχρηστίας. Ανθρώπινος παράγοντας, user interface, πλοήγηση εφαρμογής (navigation), βοήθεια προς το χρήστη, τεκμηρίωση, εγχειρίδια (manuals).

Reliability Requirements – Απαιτήσεις Αξιοπιστίας. Συχνότητα αστοχίας/κατάρρευσης του συστήματος, ευκολία και αμεσότητα επαναλειτουργίας του συστήματος και ανάκτησης δεδομένων, προβλεψιμότητα προβλημάτων και αστοχίας.

Performance Requirements – Απαιτήσεις Απόδοσης. Χρόνοι απόκρισης του συστήματος σε απαιτήσεις του χρήστη, ταχύτητα υπολογισμών, ακρίβεια, διαθεσιμότητα, χρήση και κατανάλωση πόρων συστήματος.

Supportability Requirements – Απαιτήσεις Συντηρησιμότητας. Ευελιξία, προσαρμοστικότητα, προσαρμογή και παραμετροποίηση, εύκολη συντήρηση και διόρθωση, διεθνοποίηση.

Τα αρχικά των παραπάνω κατηγοριών στην αγγλικά γλώσσα, σχηματίζουν τον όρο FURPS+ που αποτελεί ένα μοντέλο καταγραφής και τεκμηρίωσης των απαιτήσεων ενός συστήματος. Το σύμβολο + υποδηλώνει κάποιους συμπληρωματικούς περιορισμούς στις απαιτήσεις, όπως η χρήση/επικοινωνία με εξωτερικά συστήματα, νομικοί περιορισμοί, περιορισμοί στις γλώσσες προγραμματισμού και στα εργαλεία γενικότερα, περιορισμοί στην διαχείριση του συστήματος κτλ. Γενικότερα βέβαια, η σημαντικότερη διάκριση των απαιτήσεων και πιο ευρέως χρησιμοποιούμενη, είναι σε **Λειτουργικές** (Functional) και **Μη Λειτουργικές** (Non Functional) Απαιτήσεις.

Στις επόμενες ενότητες θα αναλυθούν οι τρόποι εύρεσης, καταγραφής και τεκμηρίωσης των απαιτήσεων του συστήματος. Θα ασχοληθούμε κυρίως με τις Λειτουργικές Απαιτήσεις που καταγράφονται με την βοήθεια των Use Cases.

1.6. Use Cases – Λειτουργικές Απαιτήσεις

Τα Use Cases, είναι ένα από τα σημαντικότερα παραδοτέα της UP και όλων των Object Oriented (αλλά και μη Object Oriented) μεθοδολογιών. Αποτελούν το βασικότερο μηχανισμό με τον οποίο βρίσκουμε και καταγράφουμε τις απαιτήσεις του συστήματος και αποτελούν οδηγό για πολλά άλλα μοντέλα της μεθοδολογίας UP (ουσιαστικά αποτελούν τη βάση της μεθοδολογίας, "UP is Use Case Driven"). Μέσα στα Use Cases είναι καταγεγραμμένη ολόκληρη η λειτουργικότητα (functionality) του συστήματος, δηλαδή οι λειτουργικές απαιτήσεις, αλλά και κάποιες μη λειτουργικές απαιτήσεις.

Τα Use Cases δεν είναι τίποτε άλλο από κείμενα, τα οποία περιγράφουν σε γλώσσα απλή και κατανοητή, τις ενέργειες που επιτελούν οι χρήστες/πελάτες ενός πληροφοριακού συστήματος για την επίτευξη των καθημερινών εργασιακών τους στόχων. Ο σκοπός των Use Cases δεν είναι η δημιουργία μιας λίστας η οποία θα περιέχει όλες τις λειτουργίες ενός συστήματος, αλλά ο προσδιορισμός και η τεκμηρίωση των εργασιών εκείνων, οι οποίες δίνουν αξία στη χρήση του και προσφέρουν όφελος στο χρήστη, δηλαδή η πλήρης καταγραφή της λειτουργικότητας του συστήματος.

Οι κύριες έννοιες που περιέχονται στα Use Cases είναι ο Actor, ο οποίος αποτελεί μια εξωτερική οντότητα που αλληλεπιδρά με το σύστημα (π.χ. ο χρήστης ή ένα εξωτερικό σύστημα που επικοινωνεί με το δικό μας) και το σενάριο χρήσης. Με τον όρο αυτό εννοούμε μια συγκεκριμένη σειρά βημάτων/ενεργειών που εκτελεί ένας Actor ώστε να επιτελέσει μια εργασία χρησιμοποιώντας το σύστημα. Το Use Case δεν ταυτίζεται με το σενάριο χρήσης, καθ' ότι ένα Use Case μπορεί να περιέχει εκτός από το βασικό σενάριο χρήσης, πολλά εναλλακτικά σενάρια.

Η λογική πάνω στην οποία βασίζεται η συγγραφή των Use Cases, είναι αυτή του «Μαύρου Κουτιού» (Black Box Use Case). Αυτό σημαίνει ότι η προσπάθεια επικεντρώνεται στον εντοπισμό και την αποτύπωση των εργασιών που θέλει να κάνει ένας χρήστης με το σύστημα και των ευθυνών (responsibilities) του συστήματος, δηλαδή πως πρέπει να αποκρίνεται και να αντιδρά το σύστημα σε κάθε αίτημα του χρήστη. Δεν ενδιαφέρει στη φάση της ανάλυσης των απαιτήσεων το **πως** ένα σύστημα λειτουργεί εσωτερικά, αλλά το **τι** κάνει το σύστημα, δηλαδή η εξωτερική συμπεριφορά του. Αυτή άλλωστε είναι και η βασική διαφορά μεταξύ ανάλυσης και σχεδιασμού ("what" versus "how").

1.6.1. Μορφές και Δομή των Use Cases

Υπάρχουν πολλές διαφορετικές μορφές με τις οποίες μπορούμε να γράψουμε ένα Use Case. Μπορούμε να γράψουμε ένα Use Case με τη μορφή μιας απλής παραγράφου (brief Use Cases), όπου με απλά λόγια περιγράφεται συνοπτικά, συνήθως το βασικό σενάριο χρήσης του συστήματος ή να χρησιμοποιήσουμε αρκετές παραγράφους (Casual μορφή), όπου περιλαμβάνονται και τα εναλλακτικά σενάρια ενός Use Case. Μια ακόμα συχνά χρησιμοποιούμενη μορφή των Use Cases είναι αυτή των 2 στηλών (2 column variation), στην οποία έχουμε στην $1^η$ στήλη τις ενέργειες του Actor και στη $2^η$ τις αντιδράσεις/αποκρίσεις του συστήματος.

Η πιο διαδεδομένη όμως μορφή των Use Cases είναι η πλήρης μορφή (fully dressed Use Case), στην οποία καταγράφονται σε μορφή αριθμημένων βημάτων, οι ενέργειες που πρέπει να ακολουθηθούν/εκτελεστούν για να ολοκληρωθεί ένα Use Case, μαζί με τα εναλλακτικά του σενάρια. Είναι στη διακριτική ευχέρεια της κάθε ομάδας ανάλυσης να επιλέξει ποια μέθοδο θα χρησιμοποιήσει για τη συγγραφή των Use Cases.

Ένα Use Case, γραμμένο στην πλήρη του μορφή, περιλαμβάνει διάφορα τμήματα, τα σημαντικότερα των οποίων είναι τα εξής :

- **Επικεφαλίδα**

Στο τμήμα αυτό καθορίζουμε το βασικό (primary) Actor που λαμβάνει μέρος στο Use Case, δηλαδή την κύρια οντότητα που καλεί το σύστημα για να εκτελέσει τη συγκεκριμένη εργασία την οποία περιγράφει το Use Case. Στη συνέχεια καταγράφουμε όλους τους άλλους ενδιαφερόμενους (χρήστες, actors, πελάτες, εξωτερικούς οργανισμούς κτλ), οι οποίοι μπορεί να αποκομίσουν κάποιο όφελος από τη χρήση του Use Case, καθώς και τα οφέλη που αναμένουν από τη χρήση του.

- **Προϋποθέσεις Χρήσης (Preconditions)**

Αναφέρονται οι συνθήκες που πρέπει να ισχύουν για να μπορεί να εκτελεστεί το Use Case. Τυπικό παράδειγμα είναι να έχει κάνει login ο χρήστης και να έχει δικαίωμα να εκτελέσει την εργασία που περιγράφεται στο Use Case.

- **Postconditions ή Success Guarantees**

Αναφέρεται τι πρέπει να έχει συμβεί στο τέλος του Use Case, ώστε να θεωρηθεί ότι το Use Case ολοκληρώθηκε με επιτυχία και εκπλήρωσε το στόχο του.

- **Βασικό Σενάριο (Main Success Scenario ή Basic Flow)**

Στη συνέχεια ακολουθεί το βασικό σενάριο, που είναι και η ουσία του Use Case. Το βασικό σενάριο είναι μια σειρά από αριθμημένα βήματα στα οποία καταγράφονται οι ενέργειες του Actor και οι αποκρίσεις του συστήματος σε αυτές, μέχρι να ολοκληρωθεί το Use Case. Αρχικά, γράφουμε το κύριο σενάριο χωρίς να λαμβάνουμε υπόψη ειδικές περιπτώσεις, εξαιρέσεις, σφάλματα κτλ που μπορεί να συμβούν.

- **Εναλλακτικά Σενάρια (Extensions ή Alternate Flows)**

Τα εναλλακτικά σενάρια, πάλι με τη μορφή αριθμημένων βημάτων περιγράφουν τι θα συμβεί σε περίπτωση που υπάρξει κάποια εξαίρεση κατά τη χρήση του βασικού σεναρίου του Use Case, αν δημιουργηθεί κάποιο πρόβλημα ή υπάρξει κάποια διακλάδωση (if condition).

Τα βασικά βήματα τα οποία καταγράφονται τόσο στο κύριο σενάριο όσο και στα εναλλακτικά, συνήθως είναι 3 ειδών :

- Αλληλεπίδραση μεταξύ Actor και συστήματος

- Επικύρωση (Validation), συνήθως του συστήματος

- Αλλαγή κατάστασης (state change) από το σύστημα, π.χ. αποθήκευση δεδομένων

Εξαίρεση αποτελεί συνήθως το 1ο βήμα ενός Use Case το οποίο δεν ανήκει σε μια από τις 3 παραπάνω κατηγορίες, αλλά συνήθως περιγράφει το γεγονός εκείνο με το οποίο κάποιος actor ξεκινά το Use Case, δηλαδή σηματοδοτεί την έναρξή του.

Συνήθως το τμήμα που περιλαμβάνει τα εναλλακτικά σενάρια είναι πολύ μεγαλύτερο και πιο πολύπλοκο από το βασικό σενάριο και αυτό γιατί πρέπει να καταγραφούν όλες οι πιθανές περιπτώσεις όπου το Use Case μπορεί να παρεκλίνει από τη βασική του πορεία. Η ιδέα είναι ότι το βασικό σενάριο μαζί με τα εναλλακτικά θα πρέπει να περιγράψει όλες τις δυνατές περιπτώσεις ή σχεδόν όλες, γιατί κάποιοι στόχοι των Actors/χρηστών θα καταγραφούν αναλυτικότερα στις συμπληρωματικές απαιτήσεις (supplementary specifications). Οι περισσότερες όμως και ειδικά αυτές που εκφράζουν λειτουργικές απαιτήσεις, στόχους και ενέργειες που θέλουν να πραγματοποιήσουν οι actors, πρέπει να καταγραφούν μέσα στα Use Cases, τόσο στο βασικό όσο και στα εναλλακτικά σενάρια.

Επειδή κάθε βήμα στο εναλλακτικό σενάριο, αποτελεί παρακλάδι του βασικού σεναρίου, χρησιμοποιούμε μια αρίθμηση όπου κάθε εναλλακτικό σενάριο αριθμείται με το βήμα του βασικού σεναρίου στο οποίο αποτελεί εναλλαγή. Αν υπάρχουν πολλά εναλλακτικά σενάρια στο ίδιο βήμα του βασικού σεναρίου, π.χ. στο βήμα 3, τότε τα αριθμούμε ως 3α (πρώτο εναλλακτικό σενάριο), 3β (δεύτερο εναλλακτικό σενάριο) κτλ. Για κάθε εναλλακτικό σενάριο καταγράφεται κατ' αρχήν η συνθήκη ή η εξαίρεση που δημιουργεί την ανάγκη για το εναλλακτικό σενάριο και στη συνέχεια ο τρόπος με τον οποίον αντιμετωπίζεται το εναλλακτικό σενάριο.

Συνήθως όταν τελειώνει ένα εναλλακτικό σενάριο, ξαναγυρίζουμε στο βασικό σενάριο, στο σημείο που έγινε διακλάδωση, όμως πολλές φορές (π.χ. αν υπάρχει κόλλημα του συστήματος) μπορεί στο εναλλακτικό σενάριο να υπάρχει τερματισμός του Use Case. Επίσης μπορεί κάποια εναλλακτικά σενάρια να μπορούν να συμβούν όχι μέσα σε συγκεκριμένο βήμα του βασικού σεναρίου αλλά σε οποιοδήποτε βήμα του Use Case. Αυτή η περίπτωση συμβολίζεται με ένα αστερίσκο (*) μπροστά από το εναλλακτικό σενάριο. Αν κάποιο εναλλακτικό σενάριο είναι εξαιρετικά πολύπλοκο, τότε είναι υποψήφιο να γίνει ένα ανεξάρτητο Use Case.

- **Ειδικές Απαιτήσεις (Special Requirements)**

Όταν τελειώσει η ενότητα των εναλλακτικών σεναρίων, μπορούμε να γράψουμε κάποιες ειδικές απαιτήσεις (special requirements) των Use Cases. Βέβαια αυτές συνήθως περιλαμβάνονται σε ένα άλλο μεγάλο κείμενο - παραδοτέο της Unified Process, που λέγεται Supplementary Specification, αλλά πολλοί συγγραφείς προτείνουν κάποιες ειδικές απαιτήσεις που σχετίζονται άμεσα με το συγκεκριμένο Use Case να γράφονται μαζί με το Use Case. Τέτοια απαιτήσεις είναι π.χ. θέματα απόδοσης του συστήματος, ευκολία στη χρήση, διάφοροι περιορισμοί σχεδιασμού, αξιοπιστία του συστήματος κτλ.

- **Τεχνολογικές Διαφοροποιήσεις (Technology Variations)**

Ένα τελευταίο τμήμα στην δομή ενός Use Case μπορεί να είναι μια λίστα από τεχνολογικές διαφοροποιήσεις ή διαφοροποιήσεις δεδομένων. Όλα αυτά βέβαια συνήθως σχετίζονται περισσότερο με

το σχεδιασμό της εφαρμογής και όχι με την ανάλυση, είναι όμως χρήσιμο να διευκρινίσουμε ορισμένα σημεία για ένα Use Case ακόμα και αν έχουν σχέση με το σχεδιασμό, ειδικά τα θέματα που αναφέρονται στην Είσοδο/Έξοδο (I/O) του συστήματος (π.χ. ανάγκη χρήσης barcode reader).

1.6.2. Προσδιορισμός των Use Cases

Ένα πολύ μεγάλο και σημαντικό θέμα αποτελεί ο προσδιορισμός και η καταγραφή όλων των Use Cases που είναι απαραίτητα για την ανάπτυξη ενός πληροφοριακού συστήματος. Αυτό ίσως είναι το κρισιμότερο αλλά και δυσκολότερο σημείο στην ανάλυση των Use Cases και την ανάλυση του συστήματος γενικότερα. Τίθεται λοιπόν το ερώτημα : «ποια είναι τα Use Cases του συστήματός μας»;

Η απάντηση στο ερώτημα μπορεί να δοθεί αν λάβουμε υπόψη το γεγονός ότι τα Use Cases σχετίζονται άμεσα με τους εργασιακούς στόχους των Actors/χρηστών του συστήματος, περιγράφοντας με απλό τρόπο τις ενέργειες που απαιτούνται για την επίτευξη των στόχων αυτών. Η διαδικασία είναι σχετικά απλή :

1. Προσδιορίζουμε τους στόχους των Actors/χρηστών του συστήματος. Συνήθως δημιουργούμε μια λίστα δύο στηλών, όπου αριστερά αναφέρονται οι Actors και δεξιά οι στόχοι τους (Actors – Goal List).
2. Δημιουργούμε ένα Use Case για κάθε στόχο.

Το παραπάνω ερώτημα τίθεται πλέον διαφορετικά : «ποιοι είναι οι στόχοι των Actors/χρηστών και τι θέλουν να επιτύχουν με τη χρήση του συστήματος»; Από τις απαντήσεις στο ερώτημα αυτό μπορούμε μάλιστα πιο εύκολα να εντοπίσουμε επιχειρηματικές λύσεις με πραγματική αξία για τους χρήστες/πελάτες του συστήματος, και όχι απλά να κάνουμε μια καταγραφή της υφιστάμενης κατάστασης (π.χ. κάτι τέτοιο θα συνέβαινε σε ενδεχόμενη ερώτησή μας σχετικά με το τι ακριβώς κάνουν οι χρήστες/πελάτες αυτή τη στιγμή για να διεκπεραιώσουν τις εργασίες τους).

Οι στόχοι βέβαια που καθορίζουν οι χρήστες/πελάτες δεν είναι όλοι κατάλληλοι για τη δημιουργία Use Cases, καθότι δεν εκφράζουν πάντοτε μια συγκεκριμένη επιχειρηματική ανάγκη. Για την επιλογή των στόχων οι οποίοι είναι υποψήφιοι να αποτελέσουν Use Cases, πρέπει να λάβουμε υπόψη μας το επίπεδο λεπτομέρειας στο οποίο αναφέρονται, απορρίπτοντας πολύ γενικούς και αόριστους στόχους (high level goals), καθώς και στόχους οι οποίοι είναι δευτερεύοντες ή δρουν υποστηρικτικά για την υλοποίηση άλλων βασικών στόχων (subfunction goals).

Ο γενικός κανόνας είναι ότι, στη φάση της ανάλυσης των απαιτήσεων μιας εφαρμογής, θα πρέπει να επικεντρωθούμε στο επίπεδο των βασικών επιχειρηματικών διαδικασιών (elementary business processes). Ένα elementary business process είναι : μια εργασία που εκτελείται από ένα άτομο σ' ένα συγκεκριμένο χώρο και χρονική στιγμή, η οποία γίνεται σε αντιστοιχία με ένα γεγονός μέσα στην επιχείρηση (business event) και το γεγονός αυτό έχει ως αποτέλεσμα συγκεκριμένη αξία για την επιχείρηση, αφήνοντας τα δεδομένα σε ολοκληρωμένη και σταθερή κατάσταση.

Ο ορισμός του elementary business process δεν χρειάζεται να τηρείται πολύ αυστηρά. Το θέμα είναι να καταλάβουμε ότι ένα Use Case (και κατ' επέκταση ο στόχος που πρόκειται να υλοποιήσει με την εκτέλεσή του) δεν είναι μια απλή γραμμή, μια πολύ απλή ενέργεια όπως : σβήστε μια γραμμή ή εισάγετε έναν κωδικό ή εκτυπώστε ένα έγγραφο. Συνήθως είναι μια εργασία που πιθανόν αποτελείται από 5 ως 10 διαφορετικά βήματα και δεν είναι κάτι που διαρκεί μέρες, αλλά πρέπει να ολοκληρώνεται σε μια φάση (single session), και διαρκεί από λίγα λεπτά έως λίγες ώρες. Κάτι επίσης πολύ σημαντικό στον παραπάνω ορισμό είναι ότι, το Use Case πρέπει να δίνει κάποια συγκεκριμένη αξία σε κάποιο business event, δηλαδή να προσφέρει όφελος στο χρήστη ή στην επιχείρηση.

Ο παραπάνω κανόνας πρέπει να τηρείται στο μεγαλύτερο ποσοστό των Use Cases, αλλά υπάρχουν φυσικά και εξαιρέσεις. Π.χ. σε πολλές περιπτώσεις εντοπίζουμε έναν αριθμό βημάτων που αποτελούν κομμάτι ενός μεγαλύτερου Use Case και τα οποία επαναλαμβάνονται σε ένα μεγάλο αριθμό Use Cases. Για να αποφύγουμε αυτή την επανάληψη, δημιουργούμε ένα βοηθητικό Use Case που αποτελείται από τα κοινά βήματα, το οποίο καλείται από τα άλλα Use Cases με απλή αναφορά στο όνομά του (π.χ. Include Use Case xxx).

1.6.3. Η Θέση των Use Cases στη Unified Process

Η μεθοδολογία Unified Process, θέτει σαν βάση για την ανάπτυξη ενός πληροφοριακού συστήματος την ύπαρξη των Use Cases. Το γεγονός αυτό υπονοεί τα εξής :

- Οι απαιτήσεις του συστήματος καταγράφονται κατά κύριο λόγο στα Use Cases (Use Case Model). Άλλες τεχνικές καταγραφής απαιτήσεων (εάν χρησιμοποιηθούν), είναι δευτερεύουσας σημασίας.
- Ο ρόλος των Use Cases στην ανάπτυξη ενός συστήματος μέσω διαδοχικών επαναλήψεων είναι καταλυτικός. Οι ενέργειες που περιλαμβάνονται σε μια επανάληψη, καθορίζονται σε ένα μεγάλο βαθμό από την επιλογή κάποιων σεναρίων χρήσης ή ολόκληρων Use Cases προς επεξεργασία και υλοποίηση.
- Η υλοποίηση των Use Cases, καθοδηγεί τη διαδικασία του σχεδιασμού, δηλαδή τη δημιουργία αλληλοσυσχετιζόμενων αντικειμένων για την εκτέλεση των Use Cases.
- Τα Use Cases, συνήθως χρησιμοποιούνται σαν βάση για την οργάνωση των τεχνικών εγχειριδίων (manuals) της εφαρμογής.

Η θέση των Uses Case στις διάφορες φάσεις της Unified Process έχει ως εξής :

- Inception : Στη φάση αυτή ξεκινά η κατασκευή του Use Case Model. Προσδιορίζεται η πλειοψηφία των Use Cases, συνήθως όμως μόνο το 10% αυτών γράφονται στην πλήρη τους μορφή. Άλλωστε, ο σκοπός της φάσης αυτής είναι να αποτελέσει μια αρχική προσέγγιση του έργου ανάπτυξης, μέσα από τη μελέτη ενός μικρού και αντιπροσωπευτικού δείγματος των Use Cases, που παρουσιάζουν ενδιαφέρον από πλευράς πολυπλοκότητας και ρίσκου.
- Elaboration : Το μεγαλύτερο μέρος των απαιτήσεων καταγράφεται στη φάση του Elaboration. Σε κάθε επανάληψη της φάσης αυτής, γράφονται νέα Use Cases, ενώ τα ήδη υφιστάμενα επεξεργάζονται και βελτιώνονται σταδιακά. Στο τέλος της φάσης αυτής, το 80-90% περίπου του συνόλου των Use Cases, έχουν κατασκευαστεί στην πλήρη τους μορφή.
- Construction : Στη φάση του Construction, έχει απομείνει ένα μικρό ποσοστό των Use Cases που πρέπει να γραφτούν αναλυτικά. Προς το τέλος της φάσης αυτής, το σύνολο σχεδόν των Λειτουργικών και μη Λειτουργικών απαιτήσεων έχει ήδη καταγραφεί και σταθεροποιηθεί.

Στο Σχήμα 6 που ακολουθεί, φαίνεται η θέση των Use Cases μέσα στη Unified Process και η σχέση τους με τα άλλα παραδοτέα της μεθοδολογίας UP.

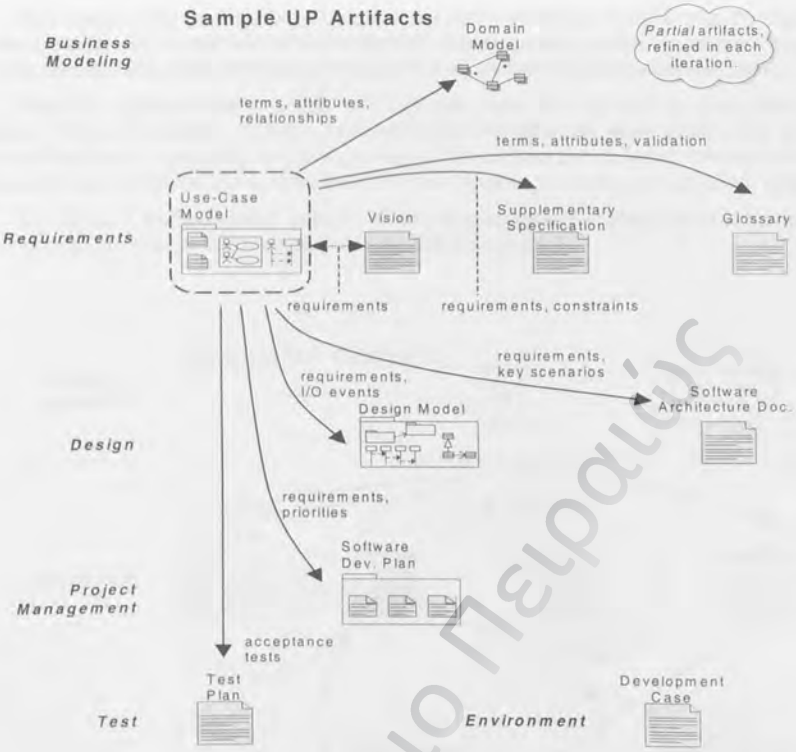
1.7. Άλλες Απαιτήσεις

Η πλήρης και αποτελεσματική γραφή των Use Cases δεν είναι αρκετή για την καταγραφή όλων των απαιτήσεων. Η Unified Process ορίζει τρία ακόμη παραδοτέα τα οποία συμπληρώνουν και ολοκληρώνουν την αποτύπωση των απαιτήσεων ενός πληροφοριακού συστήματος. Τα παραδοτέα αυτά περιγράφονται συνοπτικά στη συνέχεια.

- **Συμπληρωματικές Απαιτήσεις (Supplementary Specification)**

Τα Use Cases καταγράφουν όλη τη λειτουργικότητα ενός συστήματος, όπως επίσης και κάποιες μη Λειτουργικές (Non Functional) απαιτήσεις, οι οποίες σχετίζονται άμεσα με τα Use Cases. Μη Λειτουργικές απαιτήσεις που αναφέρονται σε ολόκληρο το σύστημα ή δεν μπορούν να προδιαγραφούν με τη βοήθεια των Use Cases, αποτελούν το περιεχόμενο των Συμπληρωματικών Απαιτήσεων.

Οι μη Λειτουργικές απαιτήσεις, αναφέρονται κυρίως στα ποιοτικά χαρακτηριστικά (Quality Attributes) ενός συστήματος, όπως είναι η απόδοση, η αξιοπιστία, η ακρίβεια, η ταχύτητα, η ευκολία στη χρήση, η συντηρησιμότητα κτλ. Η σημασία τους σε ένα έργο ανάπτυξης λογισμικού είναι πολύ μεγάλη δεδομένου ότι ο σχεδιασμός και η αρχιτεκτονική του συστήματος, είναι άρρηκτα δεμένα με τα απαιτούμενα ποιοτικά του χαρακτηριστικά.



Σχήμα 6. Η Θέση των Use Cases στη UP και η Σχέση τους με τα άλλα Παραδοτέα.

- **Σκοπός ή Όραμα του Συστήματος (Vision)**

Το παραδοτέο αυτό περιγράφει συνοπτικά τις κύριες έννοιες του υπό εξέταση προβλήματος, τις προσδοκίες των χρηστών/πελατών του συστήματος και τους βασικούς στόχους του όλου έργου. Επίσης, καθορίζει σε αδρές γραμμές τη λειτουργικότητα του συστήματος.

Τα Use Cases δεν είναι ο μοναδικός τρόπος καταγραφής των Λειτουργικών απαιτήσεων. Στο Vision συνήθως δημιουργούμε μια λίστα με δυνατότητες – χαρακτηριστικά του συστήματος (system features), στην οποία προσδιορίζονται επιγραμματικά οι λειτουργίες της εφαρμογής και αναλύονται περαιτέρω στα Use Cases. Οι λίστες χαρακτηριστικών αναφέρονται σε στόχους υψηλού επιπέδου και δεν έρχονται να αντικαταστήσουν τα Use Cases, αλλά να τα συμπληρώσουν.

Με παρόμοιο τρόπο, μπορούμε να καταγράψουμε συνοπτικά στο Vision και τις μη Λειτουργικές απαιτήσεις του συστήματος. Κάτι τέτοιο όμως ενδεχομένως να αποτελέσει πλεονασμό, εφόσον οι μη Λειτουργικές απαιτήσεις, περιγράφονται με την ίδια περίπου μορφή και στο παραδοτέο των Συμπληρωματικών Απαιτήσεων.

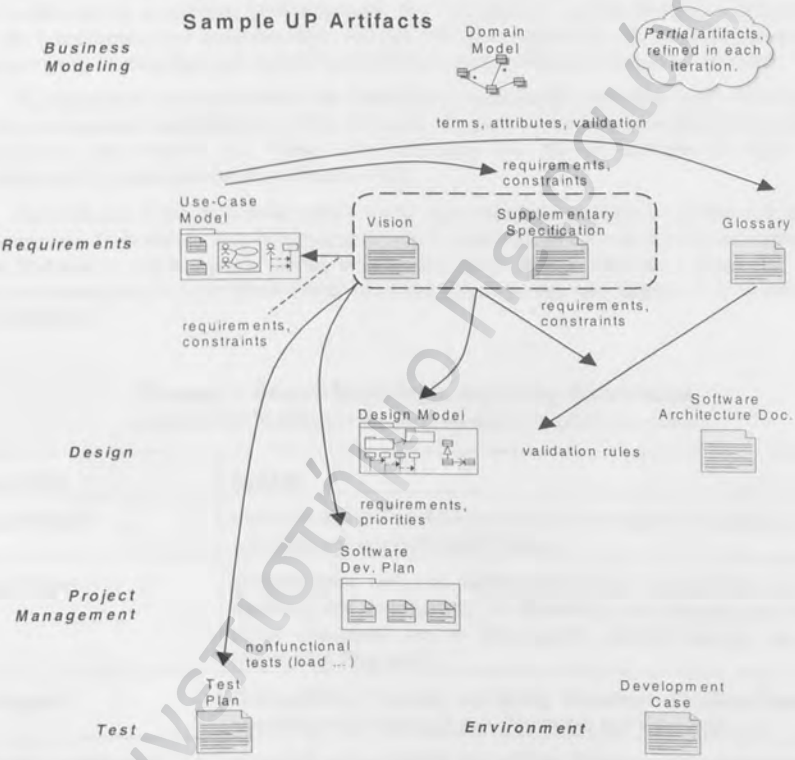
- **Γλωσσάρι Όρων (Glossary)**

Το Γλωσσάρι Όρων είναι μια λίστα, η οποία περιλαμβάνει τους βασικότερους όρους του συστήματος και την εννοιολογική τους σημασία (ορισμός). Η ύπαρξη του παραδοτέου αυτού είναι ιδιαίτερα χρήσιμη και συμβάλει στην ομαλή επικοινωνία μεταξύ όλων των εμπλεκομένων στο έργο ανάπτυξης (αναλυτές, προγραμματιστές, πελάτες, χρήστες κτλ), καθότι αποσαφηνίζει και τεκμηριώνει τις έννοιες των χρησιμοποιούμενων όρων.

Το Γλωσσάρι δεν περιλαμβάνει όλους τους όρους του συστήματος, αλλά τους πιο σημαντικούς, τους όρους που μπορεί να αποτελέσουν αιτία σύγχυσης ή παρερμηνείας, καθώς και εκείνους που χρήζουν ιδιαίτερης επεξεργασίας, όπως συγκεκριμένη μορφοποίηση (format) και κανόνες εγκυρότητας.

Μια άλλη χρησιμότητα που έχει το Γλωσσάρι, είναι ότι αποτελεί τη βάση ενός Λεξικού Δεδομένων (Data Dictionary), το οποίο κατασκευάζεται συνήθως στη φάση Elaboration και μεταξύ άλλων περιλαμβάνει : περιγραφή των όρων, συνώνυμα και εναλλακτικούς όρους, τύπο δεδομένων (data type), συσχετισμό με άλλους όρους, πεδίο δυνατών τιμών, κανόνες εγκυρότητας (validation rules) κτλ.

Στο Σχήμα 7 που ακολουθεί, φαίνεται η θέση προαναφερθέντων παραδοτέων μέσα στη Unified Process και η σχέση τους με τα άλλα παραδοτέα της μεθοδολογίας UP.



Σχήμα 7. Η Θέση των Παραδοτέων Supplementary Specification – Vision – Glossary στη UP και η Σχέση τους με τα άλλα Παραδοτέα.

1.8. Η Φάση Επεξεργασίας (Elaboration)

Η Επεξεργασία (Elaboration) είναι η δεύτερη φάση της UP και έπεται χρονικά της Αρχικής Διερεύνησης. Ο βασικός της σκοπός είναι η ολοκλήρωση (σε μεγάλο βαθμό) και η σταθεροποίηση των Απαιτήσεων, η μελέτη και η επίλυση των σημαντικότερων θεμάτων υψηλού κινδύνου, ο σχεδιασμός και η αξιολόγηση της βάσης της αρχιτεκτονικής του συστήματος και τέλος η εκτίμηση των απαιτούμενων πόρων και ο χρονοπρογραμματισμός του έργου ανάπτυξης.

Στη φάση της Επεξεργασίας ξεκινούν τα επαναληπτικά στάδια ανάπτυξης (iterations) της Unified Process, τα οποία συνεχίζονται αργότερα στις φάσεις Construction και Transition. Η φάση Elaboration περιλαμβάνει συνήθως 2 έως 4 iterations, διάρκειας 2 έως 6 εβδομάδων το καθένα.

Οι ενέργειες που περιλαμβάνονται σε κάθε επανάληψη, καθορίζονται κατά κύριο λόγο από την επιλογή κάποιων σεναρίων χρήσης ή ολόκληρων Use Cases προς επεξεργασία. Για το σκοπό αυτό τα Use Cases (καθώς και τα γενικότερα χαρακτηριστικά του συστήματος - system features) αξιολογούνται ως προς την κρισιμότητα, την επικινδυνότητα και την πολυπλοκότητά τους και κατατάσσονται με σειρά προτεραιότητας. Οι απαιτήσεις με υψηλή προτεραιότητα επεξεργάζονται στα πρώτα iterations.

Η αξιολόγηση των απαιτήσεων του συστήματος επαναλαμβάνεται στην αρχή κάθε iteration (ή στο τέλος του αμέσως προηγούμενου), ώστε να ληφθούν υπόψη τα νέα δεδομένα που προέκυψαν από τις προηγούμενες επαναλήψεις. Το πλάνο των ενεργειών που θα εκτελεστούν σε κάθε iteration καταγράφεται σε ένα κείμενο με τίτλο Iteration Plan.

Στον Πίνακα 3 που ακολουθεί, φαίνονται τα παραδοτέα που αρχίζουν να κατασκευάζονται κατά τη διάρκεια του Elaboration. Στις επόμενες ενότητες θα επικεντρωθούμε στα δύο πρώτα παραδοτέα της φάσης Elaboration, αφού προηγουμένως αναφερθούμε στα System Sequence Diagrams, τα οποία αποτελούν μέρος του Use Case Model (το μοντέλο των Use Cases έχει ήδη αρχίσει να υλοποιείται από τη φάση Inception).

Πίνακας 3. Βασικά Παραδοτέα της φάσης Elaboration.
(εξαιρούνται τα Παραδοτέα που ξεκίνησαν στη φάση Inception)

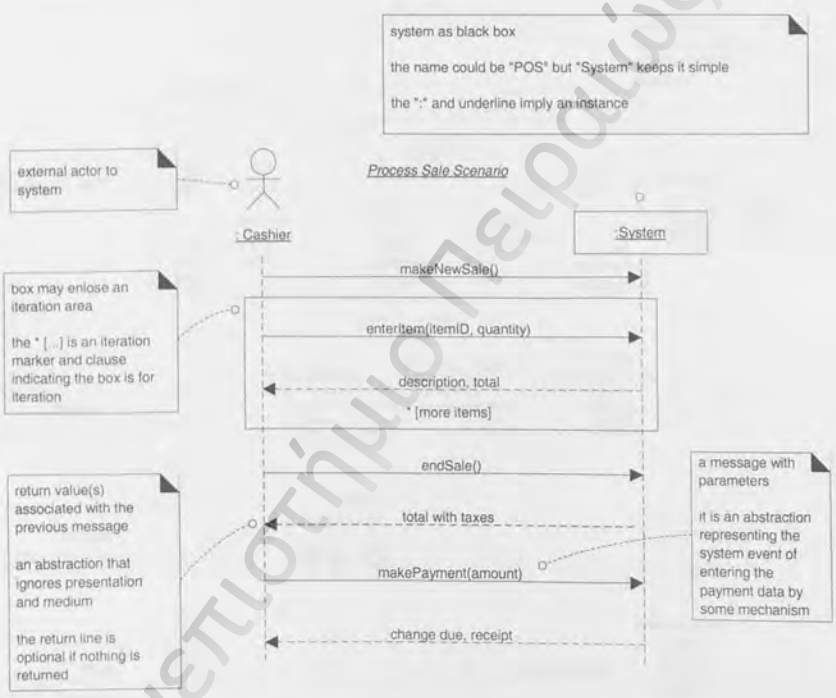
Παραδοτέο	Σχόλια
Domain Model	Είναι ένα στατικό μοντέλο στο οποίο απεικονίζονται οι βασικές έννοιες-οντότητες της φυσικού προβλήματος.
Design Model	Περιλαμβάνει ένα σετ διαγραμμάτων που περιγράφουν τη λογική σχεδίαση του συστήματος, με βασικότερα τα διαγράμματα κλάσεων (Class Diagrams) και τα διαγράμματα αλληλεπίδρασης των objects (Interaction Diagrams).
Data Model	Περιλαμβάνει τη μορφή της βάσης δεδομένων (Database Shcema) και την αντιστοίχηση μεταξύ των objects και των δεδομένων.
Software Architecture Document	Αποτελεί μια περίληψη των κύριων θεμάτων της αρχιτεκτονικής του συστήματος και τον τρόπο αντιμετώπισής τους στο σχεδιασμό.
Test Model	Ορίζει τα παραδοτέα που θα ελεγχθούν και τον τρόπο ελέγχου.
Implementation Model	Περιλαμβάνει τον πηγαίο κώδικα, τα εκτελέσιμα προγράμματα κτλ.
Use Case Storyboards, UI Prototypes	Περιγράφει το User Interface, την πλοήγηση μέσα στην εφαρμογή, τον τρόπο χρήσης της εφαρμογής κτλ.

1.9. Use Case Model - System Sequence Diagrams

Πριν προχωρήσουμε στη λογική σχεδίαση του τρόπου λειτουργίας μιας εφαρμογής λογισμικού, είναι απαραίτητο να μελετήσουμε και να καθορίσουμε τη συμπεριφορά του συστήματος (System

Behavior) θεωρώντας το σαν «Μαύρο Κουτί», δηλαδή να περιγράψουμε τι κάνει το σύστημα χωρίς να αναφερθούμε στο πως λειτουργεί εσωτερικά. Σε πολύ μεγάλο βαθμό, η συμπεριφορά ενός συστήματος αποτυπώνεται στα Use Cases και στα System Sequence Diagrams (SSD).

Τα Use Cases καταγράφουν την αλληλεπίδραση των Actors με το σύστημα, δηλαδή τα γεγονότα (system events) που δημιουργούν στο σύστημα και τις λειτουργίες (operations) που το σύστημα επιτελεί αποκρινόμενα σε αυτά. Οι πληροφορίες αυτές απεικονίζονται διαγραμματικά στα System Sequence Diagrams, τα οποία συνήθως κατασκευάζονται για τα βασικά σενάρια των Use Cases, καθώς και για τα εναλλακτικά σενάρια χρήσης που θεωρούνται πολύ σημαντικά ή παρουσιάζουν ιδιαίτερη πολυπλοκότητα. Παράδειγμα ενός απλού SSD που αναφέρεται στην αλληλεπίδραση ενός χρήστη/ταμία με ένα σημείο πώλησης (Point of Sale), παρουσιάζεται στο Σχήμα 8.

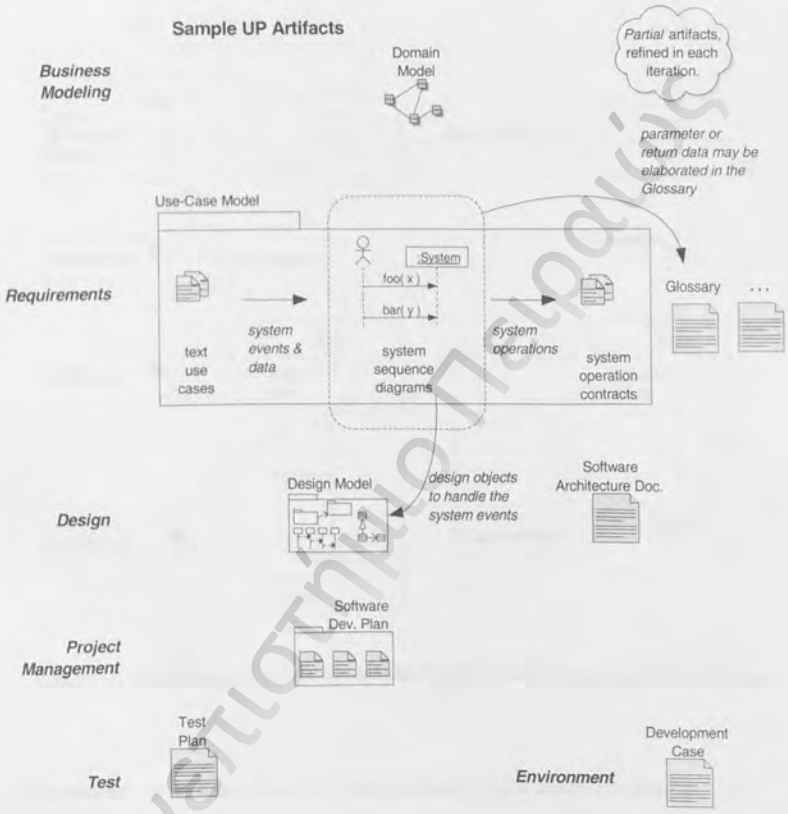


Σχήμα 8. Παράδειγμα ενός System Sequence Diagram (Διαδικασία Πώλησης σε ένα Point of Sale).

Εφόσον τα διαγράμματα αυτά προκύπτουν από τα Use Cases, πρέπει να υπάρχει στενή αντιστοιχία μεταξύ τους, δηλαδή τα γεγονότα που εμφανίζονται σε ένα SSD πρέπει να αντιστοιχούν (κατά το δυνατόν), στις σχετικές ενέργειες που καταγράφονται στα Use Cases. Τα διάφορα γεγονότα που πραγματοποιεί ο Actor/χρήστης, αναπαριστώνται στα διαγράμματα με τη μορφή συναρτήσεων περιέχοντας παραμέτρους και επιστρεφόμενες τιμές όπως στον προγραμματισμό. Όσον αφορά στην ονομασία τους, τα γεγονότα ξεκινούν με ρήμα, καθόσον αποδίδουν την πρόθεση των Actors και δεν πρέπει να είναι δεσμευτικά ως προς τη χρήση συγκεκριμένου User Interface.

Οι όροι που εμφανίζονται σε ένα SSD (λειτουργίες, παράμετροι, επιστρεφόμενες τιμές) πρέπει να επεξηγούνται επαρκώς μέσα στα αντίστοιχα Uses Cases, προς αποφυγή παρερμηνειών στη διαδικασία του σχεδιασμού. Για το σκοπό αυτό, σημαντικό ρόλο μπορεί να παίξει και το Γλωσσάρι όρων.

Τα System Sequence Diagrams αποτελούν μέρος του Use Case Model. Η θέση των System Sequence Diagrams μέσα στη Unified Process και η σχέση τους με τα άλλα παραδοτέα της μεθοδολογίας UP, φαίνεται στο Σχήμα 9.



Σχήμα 9. Η Θέση των System Sequence Diagrams στη UP και η Σχέση τους με τα άλλα Παραδοτέα.

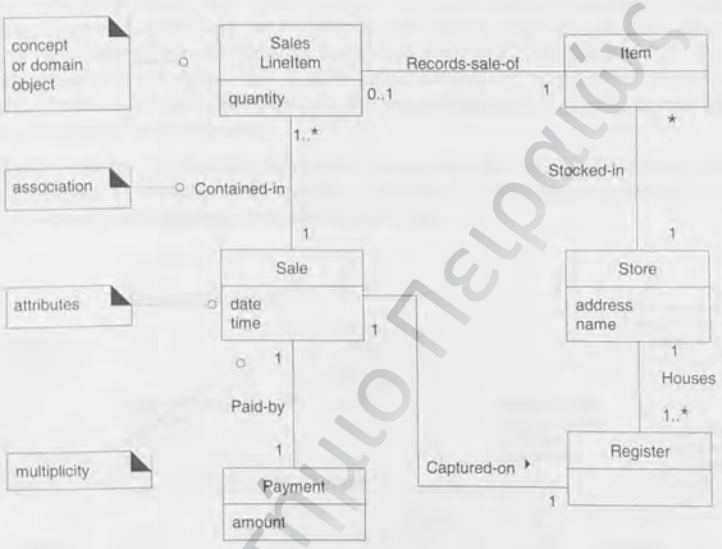
1.10. Domain Model

Το Domain Model είναι το σημαντικότερο παραδοτέο που κατασκευάζεται κατά τη διάρκεια της object oriented ανάλυσης. Αποτελεί ένα εξαιρετικό εποπτικό εργαλείο στο οποίο αναπαριστώνται γραφικά οι βασικές έννοιες – κλάσεις του φυσικού προβλήματος που εξετάζουμε και χρησιμοποιείται σαν βάση αναφοράς για το σχεδιασμό του συστήματος. Τα Domain Models εκφράζονται μέσω των διαγραμμάτων κλάσεων (class diagrams) της σημειογραφίας UML. Παράδειγμα ενός απλού Domain Model, φαίνεται στο Σχήμα 10 που ακολουθεί.

Οι βασικές πληροφορίες που απεικονίζονται σε ένα Domain Model είναι :

- Οι πραγματικές έννοιες – κλάσεις του προβλήματος (domain objects ή conceptual classes)
- Οι συσχετίσεις μεταξύ των κλάσεων (associations)
- Οι ιδιότητες των κλάσεων (attributes)

Οι πληροφορίες αυτές θα μπορούσαν φυσικά να περιγραφούν με τη μορφή απλού κειμένου. Τα διαγράμματα όμως είναι πάντοτε πλεονεκτικότερα έναντι του κειμένου, καθότι ο ανθρώπινος εγκέφαλος εξετάζει και κατανοεί τις έννοιες πιο αποτελεσματικά όταν παρουσιάζονται σχηματικά. Μπορούμε λοιπόν να πούμε ότι το Domain Model αποτελεί ένα είδος γραφικού – οπτικού λεξικού των βασικών εννοιών, ορισμών και των όρων που αναφέρονται στο φυσικό πρόβλημα που εξετάζουμε.



Σχήμα 10. Παράδειγμα Domain Model (Διαδικασία Πώλησης σε ένα Point of Sale).

Η κατασκευή του Domain Model, γίνεται συνήθως σε τρία διαδοχικά βήματα :

1. Εντοπισμός υποψήφιων κλάσεων και σχεδιασμός τους στο διάγραμμα
2. Προσθήκη συσχετίσεων μεταξύ των κλάσεων στο διάγραμμα
3. Προσθήκη ιδιοτήτων στις κλάσεις του διαγράμματος

Το σημαντικότερο πρόβλημα κατά τη δημιουργία του Domain Model, είναι ο προσδιορισμός των κύριων εννοιών - κλάσεων του φυσικού προβλήματος. Για το σκοπό αυτό υπάρχουν διάφορες τεχνικές, οι σημαντικότερες των οποίων είναι η χρήση της λίστας κατηγοριών κλάσεων, ο εντοπισμός ουσιαστικών ονομάτων και σχετικών φράσεων που αναφέρονται στα Use Cases και η χρήση των analysis patterns.

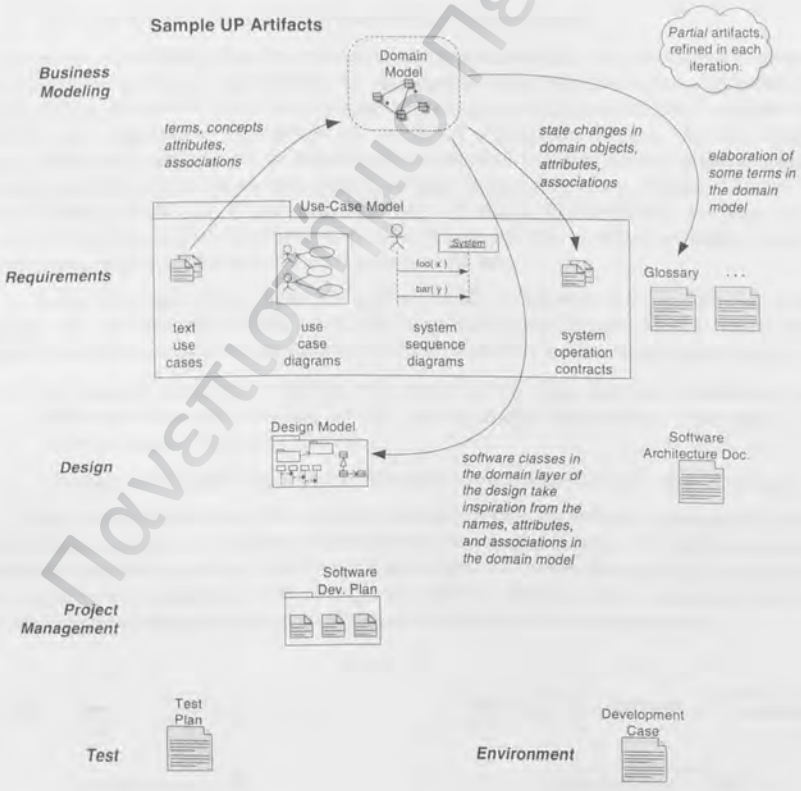
Ένας αρκετά απλός τρόπος για τον εντοπισμό των κλάσεων του Domain Model, είναι η χρήση πινάκων με κατηγορίες εννοιών – κλάσεων, οι οποίοι υπάρχουν σε διάφορα βιβλία που αναφέρονται σε object oriented ανάλυση και σχεδιασμό. Παραδείγματα κατηγοριών κλάσεων είναι φυσικά αντικείμενα, ρόλοι προσώπων, γεγονότα, τόποι, οργανισμοί κτλ. Οι πίνακες αυτοί μπορούν να χρησιμοποιηθούν σαν βάση αναφοράς και η ομάδα ανάπτυξης δεν έχει παρά να μελετήσει τις κατηγορίες μια προς μια και να τις συμπληρώσει με τις έννοιες που σχετίζονται με το φυσικό πρόβλημα.

Μια επίσης απλή τεχνική, είναι η επισήμανση ουσιαστικών (με τη γραμματική έννοια της λέξης) και σχετικών φράσεων μέσα στα κείμενα των αναλυτικών Use Cases. Τα ουσιαστικά αυτά είναι υποψήφιες έννοιες – κλάσεις για το Domain Model. Χρειάζεται προσοχή η χρήση της τεχνικής αυτής αφού δεν σημαίνει ότι κάθε ουσιαστικό είναι και κλάση του μοντέλου. Ο προφορικός και ο γραπτός λόγος χρησιμοποιούν συχνά διαφορετικές λέξεις για να περιγράψουν την ίδια έννοια και πολλές φορές είναι κάπως αόριστοι. Επίσης, πολλά από τα ουσιαστικά μέσα στα κείμενα των Use Cases θα είναι πιθανότατα ιδιότητες των κλάσεων και όχι αυτόνομες κλάσεις.

Τα analysis patterns είναι έτοιμα Domain Models τα οποία έχουν κατασκευαστεί από ειδικούς, υπάρχουν αυτούσια στη σχετική βιβλιογραφία και χρησιμοποιούνται σαν πρότυπα.

Το Domain Model, πέραν της εποπτείας που προσφέρει, αποτελεί μια πολύ καλή πηγή για την ονοματολογία των διαφόρων προγραμματιστικών κλάσεων (software/design classes) με ονόματα που προέρχονται από το φυσικό πρόβλημα. Ακολουθώντας την τακτική αυτή, μειώνουμε το λεγόμενο χάσμα αναπαράστασης (Representation Gap), με αποτέλεσμα την κοινή ονοματολογία όλων των εννοιών και των οντοτήτων που εμφανίζονται στα διάφορα παραδοτέα που ορίζει η μεθοδολογία UP. Αυτό αποτελεί ένα από τα σημαντικά πλεονεκτήματα της object oriented φιλοσοφίας, αφού διευκολύνει την κατανόηση του συστήματος από όλους τους ενδιαφερόμενους και τη μετάβαση από την ανάλυση στο σχεδιασμό και από εκεί στην κατασκευή του συστήματος.

Το Domain Model, κατά κανόνα αρχίζει να κατασκευάζεται και ολοκληρώνεται μέσα στη φάση Elaboration. Στο Σχήμα 11 που ακολουθεί, φαίνεται η θέση του Domain Model μέσα στη Unified Process και η σχέση του με τα άλλα παραδοτέα της μεθοδολογίας UP.



Σχήμα 11. Η Θέση του Domain Model στη UP και η Σχέση του με τα άλλα Παραδοτέα.

1.10.1. Domain Model – Προσθήκη Συσχετίσεων (Associations)

Ένα πολύ σημαντικό στοιχείο του Domain Model είναι οι συσχετίσεις μεταξύ των εννοιών-κλάσεων (Associations). Οι συσχετίσεις που εμφανίζονται στο Domain Model, υποδηλώνουν την ύπαρξη μιας λογικής σχέσης που συνδέει φυσικά τις κλάσεις και δεν αποσκοπούν στο να αποτυπώσουν τον τρόπο σύνδεσης των προγραμματιστικών αντικειμένων ή τη σχέση μεταξύ των διαφόρων πεδίων σε μια σχεσιακή βάση δεδομένων.

Στη UML, ένα association συμβολίζεται με μια γραμμή που ενώνει δύο κλάσεις. Η συσχέτιση μεταξύ των κλάσεων είναι αμφίδρομη (στη διαδικασία της ανάλυσης δεν ενδιαφέρει η φορά μιας σχέσης), προαιρετικά όμως μπορεί να συνοδεύεται από ένα βέλος για τη διευκόλυνση της ανάγνωσης του ονόματος της συσχέτισης. Η UML προτείνει τη χρήση ρημάτων ή σχετικών εκφράσεων έτσι ώστε η ονομασία της συσχέτισης, να είναι της μορφής : Κλάση Α – Ρήμα – Κλάση Β.

Στα δύο άκρα μιας συσχέτισης, συνήθως αναγράφεται η πολλαπλότητα (multiplicity) της σχέσης. Η πολλαπλότητα καθορίζει τον αριθμό των μελών-στιγμιότυπων (instances) της κλάσης Α που μπορούν να συσχετιστούν με ένα μέλος-στιγμιότυπο της κλάσης Β σε μια δεδομένη χρονική στιγμή.

Ένας αρκετά απλός και αποτελεσματικός τρόπος για τον εντοπισμό των συσχετίσεων σε ένα Domain Model, είναι η χρήση σχετικών πινάκων σαν βάση αναφοράς, όπου είναι καταγεγραμμένες οι συνήθεις κατηγορίες σχέσεων (τέτοιοι πίνακες υπάρχουν σε διάφορα βιβλία που αναφέρονται σε object oriented ανάλυση και σχεδιασμό). Οι σημαντικότερες από τις σχέσεις αυτές είναι :

1. Η κλάση Α αποτελεί φυσικό ή λογικό τμήμα-μέρος της κλάσης Β
2. Η κλάση Α περιέχεται φυσικά ή λογικά μέσα στην κλάση Β
3. Η κλάση Α καταγράφεται ή αποτυπώνεται μέσα στην κλάση Β

Η συσχέτιση που περιγράφεται στην πρώτη περίπτωση, αποτελεί ένα συγκεκριμένο τύπο σχέσης μεταξύ κλάσεων που καλείται Aggregation. Το Aggregation απεικονίζει μια σχέση whole-part, δηλαδή του ολόκληρου με τα απάρτιά του ή του συναρμολογήματος με τα εξαρτήματά του. Υπάρχουν δύο είδη aggregation : το composition aggregation και το shared aggregation (Σχήμα 12). Στο composition aggregation ένα μέλος (part), μπορεί να ανήκει σε ένα μόνο σύνθετο αντικείμενο (composite object) και η σχέση αυτή συναντάται πολύ συχνά τόσο μεταξύ φυσικών αντικειμένων (π.χ. Πίνακες Υλικών) όσο και μη φυσικών αντικειμένων (π.χ. Παραγγελία Πώλησης – Γραμμές Παραγγελίας). Αντίθετα στο shared aggregation, ένα μέλος μπορεί να ανήκει σε ένα, δύο ή περισσότερα σύνθετα αντικείμενα και βρίσκει εφαρμογή κυρίως σε λογικές έννοιες που σχετίζονται μεταξύ τους.

Η έννοια του aggregation είναι πιο χρήσιμη κατά τη διάρκεια του σχεδιασμού, παρά στην ανάλυση και για το λόγο αυτό πολλές φορές δεν εμφανίζεται στο Domain Model. Περιπτώσεις στις οποίες πρέπει να απεικονίσουμε το aggregation στο διάγραμμα του Domain Model, είναι οι εξής :

- Η διάρκεια ζωής και η ύπαρξη ενός αντικειμένου είναι άρρηκτα συνδεδεμένες με ένα σύνθετο αντικείμενο. Δηλαδή υπάρχει μια εξάρτηση δημιουργίας – διαγραφής (create – delete) ανάμεσα στα 2 αντικείμενα.
- Υπάρχει μια προφανής φυσική ή λογική σχέση συναρμολογήματος – εξαρτήματος.

Γενικά, στο Domain Model δεν είναι ούτε επιθυμητό ούτε απαραίτητο να συμπεριληφθούν όλες οι πιθανές σχέσεις μεταξύ των κλάσεων. Οι συσχετίσεις που αποτυπώνονται στο διάγραμμα επιλέγονται με κριτήριο τη σπουδαιότητά τους, δηλαδή αυτές που έχουν σημαντική διάρκεια ζωής και πρέπει να τις γνωρίζουμε βάσει των απαιτήσεων του συστήματος (Need to Know), καθώς και εκείνες που βοηθούν στην καλύτερη κατανόηση του φυσικού προβλήματος από όλους τους ενδιαφερόμενους.

Σχήμα 12. Composite και Shared Aggregation.

1.10.2. Domain Model – Προσθήκη Ιδιοτήτων (Attributes)

Ιδιότητα (Attribute) μιας κλάσης ονομάζεται ένα δεδομένο (datum), το οποίο αποτελεί ένα χαρακτηριστικό της κλάσης και μπορεί να πάρει συγκεκριμένη τιμή (στις σχεσιακές βάσεις δεδομένων οι ιδιότητες αντιστοιχούν στα πεδία-στήλες ενός πίνακα). Συνήθως μια κλάση έχει μια πληθώρα από ιδιότητες. Στο Domain Model, θα πρέπει να συμπεριληφθούν οι ιδιότητες εκείνες, οι οποίες εκφράζουν όλες τις απαραίτητες πληροφορίες για την υλοποίηση των Use Cases του συστήματος.

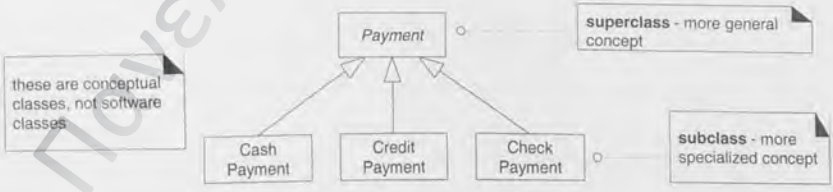
Ο προσδιορισμός των ιδιοτήτων των κλάσεων κατά την κατασκευή του Domain Model δεν είναι πάντα εύκολος, διότι θα πρέπει να ξεχωρίσουμε ανάμεσα στις διάφορες οντότητες του μοντέλου μας, ποιες θα γίνουν κλάσεις, ποιες θα γίνουν ιδιότητες και ποιες θα υλοποιηθούν σαν συσχετίσεις. Σε γενικές γραμμές θα λέγαμε ότι οι ιδιότητες αποτυπώνουν απλές έννοιες, οι οποίες μπορούν να εκφραστούν σαν πρωτογενείς τύποι δεδομένων (data types), όπως αριθμοί, κείμενο, ημερομηνίες κλπ. Πιο σύνθετα αντικείμενα και δεδομένα, πιθανότατα αποτελούν άλλες κλάσεις ή λειτουργούν σαν συσχετίσεις μεταξύ κλάσεων.

Η βασικότερη διάκριση μεταξύ κλάσεων και ιδιοτήτων, πέραν του ότι οι ιδιότητες αποτελούν συστατικά των κλάσεων, είναι ότι οι ιδιότητες δεν έχει νόημα να έχουν διαφορετικά στιγμιότυπα (instances), σε αντίθεση με τις κλάσεις, οι οποίες εξ' ορισμού περιλαμβάνουν πολλά ομοειδή αντικείμενα. Π.χ. ένα όνομα ή μια ημερομηνία γέννησης ορίζονται σαν απλοί τύποι δεδομένων (κείμενο και ημερομηνία αντίστοιχα) και δεν έχουν διαφορετικά instances, ενώ η κλάση «Υπάλληλος» μπορεί να έχει πολλά μέλη, καθένα από τα οποία θα έχει τη δική του μοναδική ταυτότητα (identity), που θα το ξεχωρίζει από τα υπόλοιπα μέλη.

Ένα συνηθισμένο λάθος κατά τη δημιουργία του Domain Model, είναι να εκφράζουμε μια σχέση μεταξύ δύο κλάσεων με την τεχνική του foreign key, συσχετίζοντας δηλαδή τις κλάσεις μέσω μιας κοινής ιδιότητας, όπως ακριβώς συνδέουμε τους πίνακες στις σχεσιακές βάσεις δεδομένων. Το foreign key είναι μια μέθοδος για την προγραμματιστική υλοποίηση μιας συσχέτισης. Η μέθοδος αυτή δεν είναι η μοναδική και η απόφαση για το αν θα χρησιμοποιηθεί θα ληφθεί στη διαδικασία του σχεδιασμού.

1.10.3. Domain Model – Γενίκευση Μοντέλου (Generalization)

Η Γενίκευση (Generalization) είναι ένας τύπος συσχέτισης μεταξύ μιας κλάσης η οποία εκφράζει μια γενικότερη έννοια και καλείται υπερ-κλάση (super class) και δύο ή περισσότερων υπο-κλάσεων (sub classes), οι οποίες αντιστοιχούν σε πιο ειδικές έννοιες, έχουν όμως κοινά χαρακτηριστικά με την υπερ-κλάση και αποτελούν υποσύνολα αυτής. Με τον τρόπο αυτό οργανώνουμε και ταξινομούμε τις διάφορες έννοιες-κλάσεις κατασκευάζοντας ιεραρχίες κλάσεων (Σχήμα 13).



Σχήμα 13. Απεικόνιση μιας Ιεραρχία Κλάσεων στη Σημειογραφία UML.

Μια κλάση, αποτελεί υπο-κλάση μιας υπερ-κλάσης όταν όλα τα μέλη-στιγμιότυπα της υπο-κλάσης είναι επίσης μέλη της υπερ-κλάσης και παράλληλα όλες οι ιδιότητες και οι συσχετίσεις της υπερ-κλάσης, εφαρμόζονται χωρίς καμία εξαίρεση και στην υπο-κλάση. Η υποδιαίρεση όμως μιας κλάσης σε

υπο-κλάσεις δεν είναι πάντα απαραίτητη. Τα κριτήρια που μας οδηγούν στη δημιουργία ιεραρχιών υπερ-κλάσεων και υπο-κλάσεων είναι τα εξής :

- αν η υπο-κλάση έχει επιπλέον ιδιότητες από την υπερ-κλάση που θεωρούνται σημαντικές
- αν η υπο-κλάση έχει επιπλέον συσχετίσεις από την υπερ-κλάση που θεωρούνται σημαντικές
- αν η υπο-κλάση λειτουργεί, αντιδρά ή εν γένει συμπεριφέρεται με διαφορετικό τρόπο σε σχέση με την υπερ-κλάση ή τις υπόλοιπες υπο-κλάσεις, και ο τρόπος αυτός θεωρείται σημαντικός για το πρόβλημα που εξετάζουμε

Η απεικόνιση των υπερ-κλάσεων και των υπο-κλάσεων στο Domain Model είναι πολύ χρήσιμη, καθότι συντελεί σε μεγάλο βαθμό στην καλύτερη κατανόηση των εννοιών, μειώνει την εμφάνιση επαναλαμβανόμενων πληροφοριών και επιτυγχάνει οικονομία έκφρασης. Επίσης αποτελεί πολύτιμη πηγή αναφοράς για τη δημιουργία ιεραρχιών προγραμματιστικών κλάσεων στη διαδικασία του σχεδιασμού και την εφαρμογή της κληρονομικότητας (inheritance) μεταξύ των κλάσεων.

Το inheritance είναι μια καθαρά προγραμματιστική έννοια και υποστηρίζεται από όλες τις object oriented γλώσσες προγραμματισμού. Είναι ο μηχανισμός αυτός, ο οποίος παρέχει τη δυνατότητα στις υπο-κλάσεις να κληρονομούν τις ιδιότητες και τις μεθόδους της υπερ-κλάσης στην οποία ανήκουν. Ο ρόλος του inheritance είναι ιδιαίτερα σημαντικός στη φάση του προγραμματισμού δεδομένου ότι με τη χρήση του αποφεύγεται η διπλο-συγγραφή κώδικα.

1.11. Από την Ανάλυση στο Σχεδιασμό

Έχοντας ολοκληρώσει, σε πρώτη τουλάχιστον φάση, την ανάλυση των απαιτήσεων του συστήματος και έχοντας κατασκευάσει τα βασικά μοντέλα της ανάλυσης όπως περιγράφηκαν στις προηγούμενες ενότητες (π.χ. Use Case Model, Domain Model κτλ), είμαστε έτοιμοι να περάσουμε στη διαδικασία του Σχεδιασμού. Εφόσον δηλαδή έχουμε καταγράψει το **ΤΙ** πρέπει να κάνει το σύστημα (What), στη συνέχεια πρέπει να σχεδιάσουμε τον τρόπο (How) με τον οποίον το σύστημα θα ικανοποιήσει τις διάφορες απαιτήσεις (Ανάλυση = Do the right thing!, Σχεδιασμός = Do the thing right!).

Υπενθυμίζεται ότι αυτή η αλληλουχία Ανάλυσης – Σχεδιασμού επαναλαμβάνεται σε κάθε iteration της Unified Process. Αυτό σημαίνει ότι δεν γίνεται εξαντλητική, λεπτομερής και παγιωμένη καταγραφή όλων των απαιτήσεων πριν περάσουμε στο σχεδιασμό αλλά μέσα από διαδοχικές επαναλήψεις ανάλυσης και σχεδιασμού (αλλά και κατασκευής τμημάτων του συστήματος) επιτυγχάνουμε συνεχές ραφινάρισμα και προσαρμογή των απαιτήσεων και των διαφόρων μοντέλων της ανάλυσης στις πραγματικές και συνεχώς μεταβαλλόμενες απαιτήσεις των χρηστών. Βέβαια καθώς προχωράμε από iteration σε iteration, οι απαιτούμενες αλλαγές μειώνονται διαρκώς και οι απαιτήσεις σταθεροποιούνται, ποτέ όμως απόλυτα. Στο τέλος της φάσης Elaboration, το 80% περίπου των απαιτήσεων έχει καταγραφεί και αναλυθεί σε λεπτομέρεια.

Κατά τη διάρκεια του object oriented σχεδιασμού, οι κύριες ενέργειες που πρέπει να υλοποιηθούν, είναι ο σχεδιασμός των προγραμματιστικών αντικειμένων (software objects), η ανάθεση αρμοδιοτήτων (responsibilities) σε αυτά και τέλος η διερεύνηση και ο καθορισμός των αλληλεπιδράσεων μεταξύ των αντικειμένων. Οι πληροφορίες αυτές αποτυπώνονται στα Διαγράμματα Αλληλεπίδρασης (Interaction Diagrams) και τα Διαγράμματα Κλάσεων (Design Class Diagrams).

Τα Interaction Diagrams παρέχουν μια δυναμική εικόνα των αλληλεπιδρώντων προγραμματιστικών αντικειμένων και απεικονίζουν τη ροή των μηνυμάτων (messages) μεταξύ τους, μέσω των οποίων γίνεται κλήση των μεθόδων. Σε αντίθεση με τα Interaction Diagrams, τα Design Class Diagrams αποτελούν μια στατική εικόνα των προγραμματιστικών κλάσεων (software ή design classes) και απεικονίζουν τις ιδιότητες (attributes) και τις μεθόδους (methods) των κλάσεων.

Τα ανωτέρω διαγράμματα αποτελούν τα βασικά παραδοτέα της διαδικασίας του Σχεδιασμού και στα πλαίσια της Μεθοδολογίας Unified Process, εντάσσονται στο παραδοτέο Design Model. Αρχικά κατασκευάζονται τα Interaction Diagrams και τα Class Diagrams έπονται, πολλές φορές όμως στην πράξη η κατασκευή τους γίνεται παράλληλα, και το ένα υποβοηθά στην κατασκευή του άλλου.

1.12. Interaction Diagrams – Σημειογραφία UML

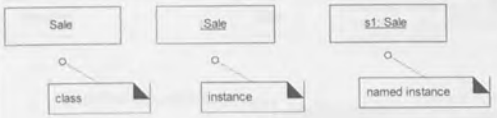
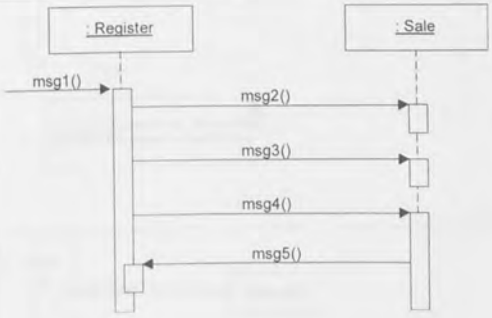
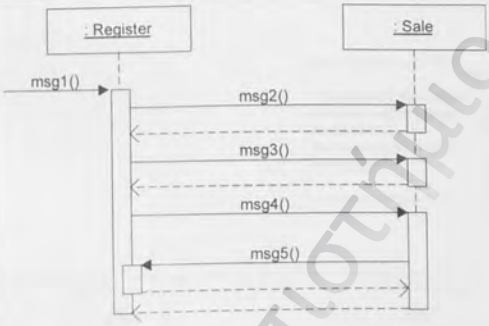
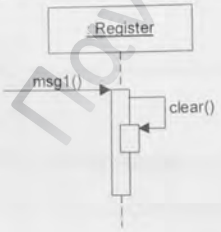
Τα Interaction Diagrams αποτελούν το σημαντικότερο παραδοτέο του object oriented σχεδιασμού, καθότι ανταγκλούν όλες τις αποφάσεις και τις λεπτομέρειες που σχετίζονται με το σχεδιασμό των αντικειμένων. Ο σχεδιασμός των διαγραμμάτων αυτών είναι εξαιρετικά δημιουργική αλλά και δύσκολη διαδικασία, διότι υπάρχουν πολλοί βαθμοί ελευθερίας και πολλοί διαφορετικοί τρόποι για να υλοποιηθούν. Κατά την κατασκευή τους είναι απαραίτητη η εφαρμογή των αρχών σχεδιασμού και η ορθολογιστική χρήση των design patterns, έννοιες που αναλύονται σε επόμενες ενότητες.

Ο όρος Interaction Diagrams, αποτελεί γενίκευση (generalization) δύο πιο εξειδικευμένων τύπων διαγραμμάτων που χρησιμοποιούνται στη UML, τα Collaboration Diagrams και τα Sequence Diagrams. Ο ρόλος τους είναι παρόμοιος, εφόσον και τα δύο απεικονίζουν την αλληλεπίδραση μεταξύ των αντικειμένων, μέσω των μηνυμάτων που ανταλλάσσουν τα διάφορα αντικείμενα μεταξύ τους.

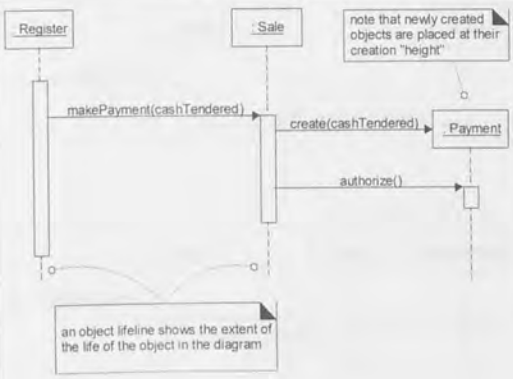
Τα Sequence Diagrams συναντώνται συχνότερα στην πράξη, διότι η σημειογραφία που ακολουθούν είναι αρκετά απλή και η χρονική αλληλουχία των μηνυμάτων που εμφανίζονται στο διάγραμμα, είναι πολύ ξεκάθαρη.

Στον Πίνακα 4 που ακολουθεί, παρουσιάζεται η βασική σημειογραφία που χρησιμοποιείται για το σχεδιασμό των Sequence Diagrams στη UML.

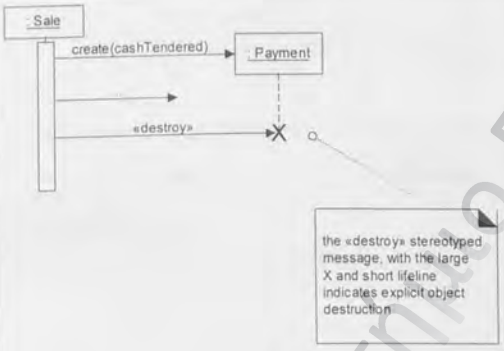
Πίνακας 4. Βασική Σημειογραφία των Sequence Diagrams στη UML.

	**Classes and Instances**: Τα instances μιας κλάσης εμφανίζονται μέσα σε ορθογώνια κουτιά με το όνομα της κλάσης υπογραμμισμένο και το σύμβολο « : » πριν από το όνομα. Ένα συγκεκριμένο instance χαρακτηρίζεται με τη χρήση ενός μοναδικού ονόματος.
	**Messages**: Κάθε μήνυμα αναγράφεται πάνω από το βέλος που συνδέει το object που στέλνει το μήνυμα με το object που το δέχεται. Τα μηνύματα απεικονίζονται στο διάγραμμα διαδοχικά από πάνω προς τα κάτω, σύμφωνα με τη χρονική αλληλουχία που στέλνονται. Τα μηνύματα μεταξύ των αντικειμένων εμφανίζονται με τον ακόλουθο τρόπο: return : = message(parameter : parameterType) : returnType Οι πληροφορίες που αναφέρονται στον τύπο των παραμέτρων, μπορούν να μην εμφανιστούν αν είναι προφανείς ή δεν ενδιαφέρουν.
	**Returns**: Πολλές φορές ένα object που δέχεται ένα μήνυμα επιστρέφει μια τιμή. Η επιστρεφόμενη αυτή τιμή συμβολίζεται με ένα διακεκομμένο βέλος. Πάνω στο βέλος αυτό, αν επιθυμούμε μπορούμε να προσθέσουμε σχόλια.
	**Messages to "self" or "this"**: Πολλές φορές ένα object χρειάζεται να στείλει ένα μήνυμα στον εαυτό του.

Πίνακας 4. Βασική Σημειογραφία των Sequence Diagrams στη UML (συνέχεια).

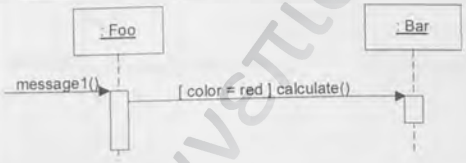


Creation of Instances: Ένας συχνός τύπος μηνύματος μεταξύ objects είναι η δημιουργία καινούργιου instance. Στην περίπτωση αυτή συνήθως χρησιμοποιείται το τυποποιημένο όνομα «create». Επίσης το μήνυμα μπορεί να περιέχει παραμέτρους που να συμβολίζουν τις απαιτούμενες αρχικές τιμές. Τέλος, μια σημαντική παρατήρηση είναι ότι το κουτί κάθε instance εμφανίζεται μέσα στο διάγραμμα στο ύψος στο οποίο δημιουργείται, δηλαδή εκεί που δέχεται το μήνυμα create.

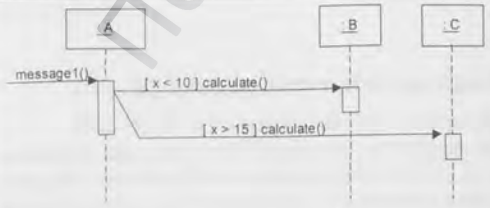


Object Lifelines and Destruction: Πολλές φορές είναι χρήσιμο να δείξουμε πάνω στο διάγραμμα την διάρκεια ζωής ενός object. Αυτή συμβολίζεται με μια κάθετη διακεκομμένη γραμμή που ξεκινάει από τη στιγμή δημιουργίας του object και ολοκληρώνεται τη στιγμή της καταστροφής του.

Η χρονική στιγμή της καταστροφής αποτυπώνεται με ένα βέλος/μήνυμα με το στερεότυπο «destroy» που καταλήγει με ένα X στο σημείο τερματισμού της ζωής του object. Αυτό έχει συνήθως αξία αν η γλώσσα προγραμματισμού που χρησιμοποιούμε δεν έχει garbage collection, που σημαίνει ότι ο προγραμματιστής πρέπει να δώσει ρητή εντολή για την καταστροφή του object.

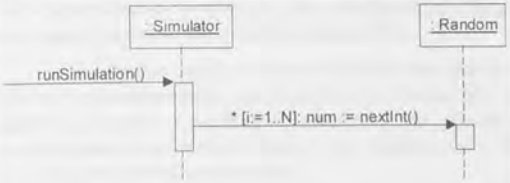
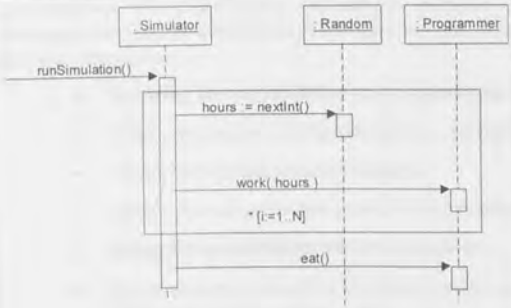
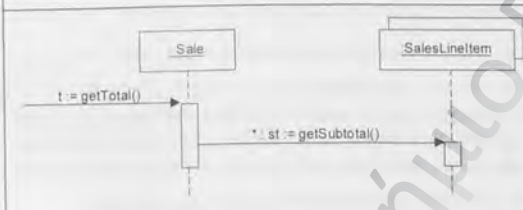
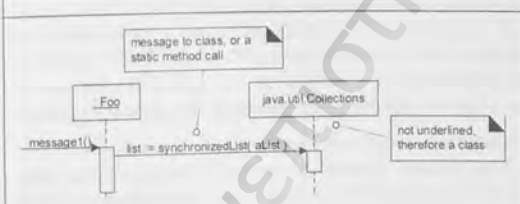


Conditional Message: Τα conditional μηνύματα εμφανίζονται με την προσθήκη της συνθήκης από την οποία εξαρτάται το μήνυμα, δοσμένη μέσα σε αγκύλες : [condition]. Το μήνυμα στέλνεται μόνο σε περίπτωση που η συνθήκη είναι αληθής.



Mutually Exclusive Conditional Messages: Πολλές φορές μπορεί ένα object να έχει τη δυνατότητα να στείλει δύο ή περισσότερα υποψήφια μηνύματα που το ένα αποκλείει το άλλο, ανάλογα με την ισχύ μιας συνθήκης. Τα αλληλοαποκλειόμενα αυτά μηνύματα συμβολίζονται με βέλη που έχουν το ίδιο σημείο εκκίνησης και που αναγράφουν τη συνθήκη από την οποία εξαρτάται η αποστολή τους ή όχι.

Πίνακας 4. Βασική Σημειογραφία των Sequence Diagrams στη UML (συνέχεια).

	**Iteration for a Single Message**: Αν ένα μήνυμα στέλνεται επαναληπτικά, τότε συμβολίζεται με ένα « * » και μια προαιρετική επεξήγηση μέσα σε αγκύλες πχ [i:=1…N].
	**Iteration of a Series of Messages**: Αν δύο ή περισσότερα μηνύματα στέλνονται επαναληπτικά μέσα σε ένα βρόγχο (loop), τότε ο βρόγχος συμβολίζεται με ένα ορθογώνιο που περικλείει όλα τα επαναλαμβανόμενα μηνύματα. Το σύμβολο « * » και μια προαιρετική επεξήγηση μέσα σε αγκύλες, π.χ. [i:=1…N] αναγράφεται στην κάτω πλευρά του ορθογωνίου του βρόγχου.
	**Iteration Over a Collection (Multiobject)**: Όταν έχουμε επανάληψη ενός μηνύματος πάνω στα μέλη μιας ομάδας αντικειμένων (collection of objects), τότε χρησιμοποιείται ένα διπλό κουτί για το συμβολισμό του collection (σύμβολο για το multiobject, όπως λέγεται εναλλακτικά στην UML το collection).
	**Messages to Class Object**: Πολλές φορές πρέπει να σταλεί μήνυμα σε μια κλάση και όχι σε ένα instance της κλάσης. Σε αυτήν την περίπτωση συμβολίζουμε την κλάση μέσα σε κουτί χωρίς να είναι υπογραμμισμένο το όνομά της. Αυτό σημαίνει ότι είναι σημαντικό να υπογραμμίζουμε σχολαστικά το όνομα των instances στις άλλες περιπτώσεις, ώστε να μην υπάρξει σύγχυση.

1.13. Σχεδιασμός Αντικειμένων με Αρμοδιότητες (Responsibilities)

Μετά την ολοκλήρωση της αρχικής καταγραφής των απαιτήσεων του συστήματος, η διαδικασία ανάπτυξης μιας εφαρμογής λογισμικού, συνεχίζεται με το σχεδιασμό των software objects. Στη φάση αυτή ορίζονται οι μέθοδοι των αντικειμένων και τα μηνύματα που ανταλλάσσουν μεταξύ τους, με σκοπό την ικανοποίηση των απαιτήσεων. Η διαδικασία αυτή είναι ιδιαίτερα κρίσιμη και αποτελεί την καρδιά του object oriented σχεδιασμού.

Η κατανομή των διαφόρων μεθόδων και γενικότερα των αρμοδιοτήτων (responsibilities) στις κλάσεις των αντικειμένων, αποτελεί καθοριστικό παράγοντα για την επιτυχία του συστήματος και εφόσον υλοποιηθεί σωστά, θα δώσει στο σύστημα όλα τα σημαντικά πλεονεκτήματα της object oriented

φιλοσοφίας, διαφορετικά τα αποτελέσματα μπορεί να απέχουν πολύ από τα αναμενόμενα. Γενικά, η ορθή κατανομή των responsibilities, κάνει το σύστημα πιο κατανοητό, εύκολο στη συντήρηση, επεκτάσιμο και δίνει περισσότερες δυνατότητες για επαναχρησιμοποίηση (reuse) των επιμέρους δομικών στοιχείων του συστήματος (components) σε άλλες εφαρμογές.

Στο σημείο αυτό έρχεται να βοηθήσει σημαντικά το έργο της ομάδας σχεδιασμού, η μεθοδολογία GRASP, η οποία αποτελεί ένα σύνολο από κανόνες και αρχές με τη μορφή design patterns. Η μελέτη και χρήση των patterns κάνει πιο κατανοητή και εύκολα εφαρμόσιμη την πολύπλοκη διαδικασία της αντιστοίχησης των responsibilities στα διάφορα αντικείμενα και κατ' επέκταση το σχεδιασμό των προγραμματιστικών αντικειμένων.

Στη UML, με τον όρο Αρμοδιότητα εννοούμε όλες τις δυνατότητες και τις υποχρεώσεις που περιλαμβάνει ένα αντικείμενο, δηλαδή τα στοιχεία αυτά που καθορίζουν τη συμπεριφορά του αντικειμένου μέσα στο σύστημα. Τα responsibilities που μπορεί να έχει ένα object, χωρίζονται σε δύο βασικές κατηγορίες:

1. **Knowing Responsibilities του αντικειμένου :**

- Γνώση σχετικά με εσωτερικά (private, encapsulated) δεδομένα
- Γνώση για σχετιζόμενα αντικείμενα
- Γνώση για δεδομένα που μπορούν να προκύψουν ή να υπολογιστούν

2. **Doing Responsibilities του αντικειμένου :**

- Ενέργειες που επιτελεί το ίδιο το αντικείμενο, όπως η δημιουργία ενός άλλου αντικειμένου ή η διενέργεια ενός υπολογισμού
- Έναρξη (triggering) μια ενέργειας σε κάποιο άλλο αντικείμενο
- Έλεγχος και συντονισμός ενεργειών σε άλλα αντικείμενα

Π.χ. ένα object ΠαραγγελίαΠώλησης μπορεί να είναι υπεύθυνο για την δημιουργία ενός άλλου object ΓραμμήΠαραγγελίαςΠώλησης (doing responsibility) ή να έχει γνώση σχετικά με το συνολικό ποσό της παραγγελίας (knowing responsibility). Πολλά knowing responsibilities συνήθως προκύπτουν από τις ιδιότητες και τις συσχετίσεις των κλάσεων που εμφανίζονται στο Domain Model.

Ανάλογα με το βαθμό λεπτομέρειας στον οποίον εκφράζουμε μια αρμοδιότητα, αυτή μπορεί να περιέχει από λίγες έως μερικές δεκάδες (ή και εκατοντάδες) μεθόδους. Για παράδειγμα αν εκφράσουμε ένα responsibility σαν «Πρόσβαση σε μια σχεσιακή βάση δεδομένων», τότε προφανώς χρειάζεται μια πληθώρα μεθόδων και συνεργασία αρκετών ίσως αντικειμένων για να επιτευχθεί. Συμπερασματικά, η έννοια responsibility δεν ταυτίζεται με την έννοια method. Ένα responsibility μπορεί να χρειάζεται πολλές μεθόδους και τη συνεργασία μεθόδων από άλλα αντικείμενα για να εκπληρωθεί.

Το βασικότερο διάγραμμα στο οποίο αποτυπώνονται οι διάφορες αρμοδιότητες των αντικειμένων είναι τα Interaction Diagrams, τη βασική σημειογραφία των οποίων είδαμε στην προηγούμενη ενότητα. Κατά την κατασκευή των Interaction Diagrams στη διαδικασία του σχεδιασμού, λαμβάνονται αποφάσεις σχετικά με την ανάθεση των αρμοδιοτήτων στα objects, οι οποίες αντανακλώνται στα μηνύματα που ανταλλάσσουν τα διάφορα αντικείμενα μεταξύ τους. Επίσης, πολλά από τα responsibilities, θα προκύψουν κατά τη φάση του προγραμματισμού και θα πρέπει μέσω feedback να συμπεριληφθούν στα αντίστοιχα Interaction Diagrams.

1.14. GRASP Design Patterns

Όπως αναφέρθηκε στην προηγούμενη ενότητα, η ανάθεση των αρμοδιοτήτων στα αντικείμενα της εφαρμογής, είναι μια διαδικασία εξαιρετικά σημαντική και λαμβάνει χώρα τόσο κατά το σχεδιασμό του συστήματος, όσο και κατά τη δημιουργία του κώδικα. Καθοριστικής σημασίας για την ορθή κατανομή των responsibilities στα διάφορα objects, είναι η χρήση των design patterns.

Ένα pattern είναι ένα αντιπροσωπευτικό πρόβλημα σχεδιασμού με την αντίστοιχη λύση του, καθώς και συμβουλές σχετικά με το πως μπορεί να χρησιμοποιηθεί και τα πλεονεκτήματα και

μειονεκτήματα της χρήσης του. Το μεγάλο τους πλεονέκτημα είναι ότι αποτελούν εφαρμογή δοκιμασμένων και αξιόπιστων λύσεων και αρχών και μπορούν να χρησιμοποιηθούν αυτούσια σε νέες παρόμοιες περιπτώσεις και προβλήματα σχεδιασμού των αντικειμένων. Έτσι η έννοια «νέο pattern» είναι οξύμωρη. Το pattern είναι κάτι που προϋπάρχει, είναι δοκιμασμένο και αναλύεται εκτενώς στη σχετική βιβλιογραφία.

Κάθε pattern έχει το δικό του χαρακτηριστικό όνομα, γεγονός που διευκολύνει την κατανόηση και απομνημόνευσή του αλλά και την επικοινωνία μεταξύ των μελών της ομάδας σχεδιασμού και των προγραμματιστών.

Τα GRASP design patterns, είναι πολύ καλοί οδηγοί κατά τη διαδικασία κατανομής αρμοδιοτήτων σε αντικείμενα, με δεδομένη μια συγκεκριμένη κατηγορία βασικών προβλημάτων σχεδιασμού. Ο όρος GRASP (grasp = κατανόηση) είναι ακρωνύμιο των λέξεων General Responsibility Assignment Software Patterns, και έχει επιλεγεί έτσι ώστε να υποδηλώνει άμεσα ότι η κατανόηση των patterns αυτών και φυσικά των αρχών και των κανόνων που εφαρμόζουν, είναι απαραίτητη προϋπόθεση για τον επιτυχημένο σχεδιασμό μιας object oriented εφαρμογής λογισμικού.

Στη συνέχεια αναφέρονται ορισμένα από τα σημαντικότερα και πιο ευρέως χρησιμοποιούμενα GRASP design patterns.

1.14.1. Information Expert ή Expert

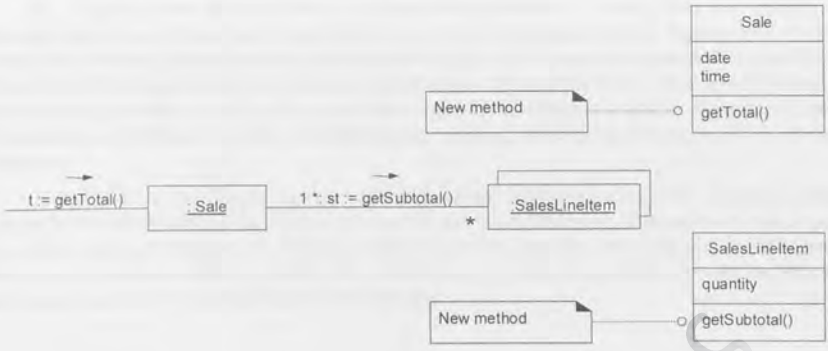
Η βασική διαδικασία κατανομής των responsibilities γίνεται θέτοντας το απλό ερώτημα : «σε ποιο object πρέπει να αναθέσουμε ένα συγκεκριμένο responsibility» ; Π.χ. θέτουμε το ερώτημα : «ποιο object είναι υπεύθυνο να γνωρίζει τη συνολική αξία μιας παραγγελίας πώλησης» ; Ένα βασικό κριτήριο για να επιλέξουμε το κατάλληλο object το οποίο θα αναλάβει την αρμοδιότητα αυτή, είναι ποιο object γνωρίζει τις πληροφορίες που είναι απαραίτητες για την εκπλήρωση της αρμοδιότητας. Αυτό το κριτήριο αποτελεί ουσιαστικά ένα pattern που ονομάζεται Information Expert ή απλά Expert.

Εφαρμόζοντας αυτό (ή κάποιο άλλο) pattern, πρέπει να αναζητήσουμε την κατάλληλη κλάση ανάμεσα στις προγραμματιστικές κλάσεις του Design Class Diagram η οποία θα αναλάβει την υλοποίηση της συγκεκριμένης αρμοδιότητας. Αν δεν υπάρχει εκεί η κατάλληλη κλάση ή δεν έχουν σχεδιαστεί ακόμη τα Class Diagrams, τότε την αναζητούμε στο Domain Model. Αν εκεί βρούμε μια κλάση η οποία είναι κατάλληλη για το δεδομένο responsibility, τότε δημιουργούμε μια αντίστοιχη design class στο Design Model. Στη νέα αυτή προγραμματιστική κλάση, προσθέτουμε μια μέθοδο η οποία θα αναλάβει να εκπληρώσει τη συγκεκριμένη αρμοδιότητα ή τουλάχιστον ένα μέρος αυτής.

Για παράδειγμα, έστω ότι θέλουμε να υπολογίσουμε τη συνολική αξία μιας παραγγελίας πώλησης. Οι απαραίτητες πληροφορίες για τον υπολογισμό αυτό, είναι η γνώση όλων των γραμμών της παραγγελίας και το μερικό σύνολο κάθε γραμμής (ποσότητα επί αξία). Με βάση το Information Expert design pattern, το αντικείμενο ΠαραγγελίαΠώλησης είναι κατάλληλο να αναλάβει την αρμοδιότητα αυτή. Για το σκοπό αυτό δημιουργούμε μια μέθοδο getTotal() στο object ΠαραγγελίαΠώλησης. Το αντικείμενο αυτό στη συνέχεια, πρέπει να στείλει ένα μήνυμα στο object ΓραμμήΠαραγγελίαςΠώλησης, το οποίο εκτελώντας τη μέθοδο getSubtotal(), την οποία πρέπει επίσης να κατασκευάσουμε, θα επιστρέψει τα μερικά σύνολα των γραμμών της παραγγελίας, ώστε αθροιζόμενα να δώσουν το σύνολο της παραγγελίας (Σχήμα 14).

Το Information Expert θα λέγαμε ότι είναι από τα πιο συχνά χρησιμοποιούμενα patterns, καθώς εκφράζει την απλή σκέψη ότι τα objects πρέπει να εκτελούν λειτουργίες σχετικές με τις πληροφορίες τις οποίες γνωρίζουν, δηλαδή τα δεδομένα τα οποία περιέχουν. Βέβαια πολλές φορές οι πληροφορίες που χρειάζονται για ένα responsibility υπάρχουν διασκορπισμένες σε περισσότερα objects τα οποία πρέπει να συνεργαστούν, ανταλλάσσοντας μηνύματα μεταξύ τους για να επιτελέσουν την επιθυμητή λειτουργία.

Μερικές φορές δεν είναι επιθυμητό να αναθέσουμε ένα responsibility σε ένα object, παρά το γεγονός ότι το object αυτό γνωρίζει τις απαιτούμενες πληροφορίες. Π.χ. το object ΠαραγγελίαΠώλησης δεν είναι κατάλληλο να αναλάβει την αρμοδιότητα της αποθήκευσης των παραγγελιών στην βάση δεδομένων, παρά το γεγονός ότι γνωρίζει το σύνολο σχεδόν των δεδομένων που απαιτούνται για το σκοπό αυτό.



Σχήμα 14. Υπολογισμός της Συνολικής Αξίας μιας Παραγγελίας Πώλησης.

Στην περίπτωση αυτή, θα καταστρατηγούσαμε μια βασική αρχή της αρχιτεκτονικής των object oriented πληροφοριακών συστημάτων, σύμφωνα με την οποία δεν πρέπει να αναμιγνύουμε τις βασικές έννοιες ενός συστήματος (π.χ. η λογική της εφαρμογής πρέπει να είναι ανεξάρτητη από τη βάση δεδομένων όπως και από το User Interface), οι οποίες συνήθως υλοποιούνται σε διαφορετικά και ανεξάρτητα μεταξύ τους «στρώματα» (layers).

Γενικά, το Information Expert design pattern αν χρησιμοποιηθεί σωστά παρέχει σημαντικά πλεονεκτήματα, όπως :

- Αποφεύγεται η άσκοπη διάχυση της πληροφορίας και μειώνεται η αλληλεξάρτηση μεταξύ των αντικειμένων (low coupling), εφόσον τα objects χρησιμοποιούν τις εσωτερικές τους πληροφορίες για να πραγματοποιήσουν τις αρμοδιότητες που τους έχουν ανατεθεί. Αυτό διευκολύνει τη συντήρηση και τη στιβαρότητα των συστημάτων.
- Ομοιογενής κατανομή των responsibilities σε περισσότερες μη υπερφορτωμένες κλάσεις, με μεγάλη συνοχή (high cohesion), εύκολα κατανοητές και συντηρήσιμες.

Ορισμένες χαρακτηριστικές εκφράσεις που τονίζουν τη βασική αρχή που εκφράζει το Information Expert είναι : "Place responsibilities with data", "That which knows, does" και "Do it Myself".

1.14.2. Creator

Μια κλάση Β είναι υποψήφια να αναλάβει την αρμοδιότητα για τη δημιουργία ενός στιγμιότυπου (instance) μιας άλλης κλάσης Α, σε μια από τις ακόλουθες περιπτώσεις :

- Η κλάση Α αποτελεί φυσικό ή λογικό τμήμα-μέρος της κλάσης Β (B aggregates A)
- Η κλάση Β περιέχει (contains) την Α
- Η κλάση Β καταγράφει (records) instances της Α
- Η κλάση Β χρησιμοποιεί στενά (closely uses) την Α
- Η κλάση Β έχει τα απαραίτητα δεδομένα για την αρχικοποίηση (initialization) της Α, όταν αυτή δημιουργείται.

Τότε λέμε ότι η κλάση Β αποτελεί έναν creator για τα αντικείμενα της κλάσης Α.

Το design pattern Creator δίνει απάντηση στο ερώτημα : «ποιος έχει την ευθύνη για τη δημιουργία ενός νέου στιγμιότυπου μιας κλάσης» ; Αξίζει να σημειωθεί ότι η δημιουργία αντικειμένων είναι από τις πιο κοινές δραστηριότητες ενός object oriented συστήματος και επομένως η σωστή ανάθεση των responsibilities δημιουργίας των objects, με τη βοήθεια του pattern αυτού, είναι ιδιαίτερα σημαντική. Είναι επίσης σημαντικό να επιλέξουμε σαν Creator την κλάση εκείνη που πρέπει να συνεργάζεται με το υπό δημιουργία αντικείμενο σε όλες τις πιθανές περιπτώσεις, δηλαδή σε όλα τα γεγονότα (events) του συστήματος.

Εφαρμόζοντας το Creator design pattern, η κλάση ΠαραγγελίαΠώλησης έχει την ευθύνη της δημιουργίας των στιγμιότυπων της κλάσης ΓραμμήΠαραγγελίαςΠώλησης, δεδομένου ότι μια παραγγελία περιέχει (στην πραγματικότητα μια γραμμή αποτελεί λογικό τμήμα μιας παραγγελίας) πολλές γραμμές. Η ανάθεση της αρμοδιότητας αυτής θα πραγματοποιηθεί με την κλήση της μεθόδου makeLineItem, η οποία πρέπει να προστεθεί στην κλάση ΠαραγγελίαΠώλησης.

1.14.3. Low Coupling

Το pattern αυτό εκφράζει την αρχή της χαμηλής αλληλεξάρτησης μεταξύ των αντικειμένων, έτσι ώστε τα αντικείμενα να είναι κατά το δυνατόν αυτόνομα και ανεξάρτητα. Η αρχή αυτή ισχύει όχι μόνο για τα αντικείμενα, αλλά και για τα επιμέρους δομικά στοιχεία του συστήματος (components), τα υποσυστήματα, ακόμα και ολόκληρα συστήματα. Είναι μια αρχή που πρέπει κατά το δυνατόν να τηρούμε σε όλα τα επίπεδα της σχεδίασης ενός πληροφοριακού συστήματος, γι' αυτό χαρακτηρίζεται συχνά σαν αρχή αξιολόγησης (Evaluative Principle). Αν μια κλάση βασίζεται ή εξαρτάται από πολλές άλλες κλάσεις, τότε εμφανίζονται τα ακόλουθα προβλήματα :

- Αλλαγές στις συσχετιζόμενες κλάσεις προκαλούν προβλήματα και ανάγκη για εσωτερικές αλλαγές στην κλάση
- Η συμπεριφορά της κλάσης είναι δυσνόητη και πρέπει να μελετηθεί σε συνδυασμό με τις υπόλοιπες κλάσεις
- Είναι δύσκολη η επαναχρησιμοποίησή της διότι απαιτεί την ύπαρξη και όλων των άλλων κλάσεων από τις οποίες εξαρτάται

Η εφαρμογή του pattern Low Coupling, υπαγορεύει την ανάθεση αρμοδιοτήτων στα αντικείμενα έτσι ώστε να παραμένει χαμηλός ο συνολικός βαθμός αλληλεξάρτησης μεταξύ των αντικειμένων. Το Low Coupling συνιστά το σχεδιασμό κλάσεων οι οποίες είναι αρκετά αυτόνομες, με αποτέλεσμα τη μείωση των αρνητικών επιδράσεων που προκαλούνται από ενδεχόμενες αλλαγές. Συνήθως δεν εφαρμόζεται ανεξάρτητα, αλλά χρησιμοποιείται σε συνδυασμό με τα υπόλοιπα patterns, όπως το Expert ή το High Cohesion.

Σε γενικές γραμμές θα λέγαμε ότι η αρχή του Low Coupling, πρέπει να λαμβάνεται πολύ σοβαρά υπόψη κυρίως κατά τη σχεδίαση των αντικειμένων αυτών, τα οποία έχουν γενική χρήση, πρόκειται να επαναχρησιμοποιηθούν σε διάφορες περιπτώσεις και παρουσιάζουν υψηλές πιθανότητες εξέλιξης και τροποποιήσεων. Αντίθετα τα πιο σταθερά objects (όπως π.χ. objects σε μια standard βιβλιοθήκη της java) δεν αποτελούν συνήθως πρόβλημα στην περίπτωση που δεν συμβαδίζουν με την αρχή του Low Coupling.

1.14.4. High Cohesion

Η έννοια High Cohesion (συνεκτικότητα) αντικατοπτρίζει το πόσο συνδεδεμένες και συναφείς μεταξύ τους είναι οι διάφορες αρμοδιότητες που περιλαμβάνει ένα αντικείμενο. Έτσι μια κλάση με High Cohesion έχει συγκεκριμένο και ξεκάθαρο σκοπό. Αντίθετα, μια κλάση με Low Cohesion ή εκτελεί άσχετες μεταξύ τους ενέργειες ή είναι επιφορτισμένη με πάρα πολλές αρμοδιότητες (ή και τα δύο ταυτόχρονα). Μια τέτοια κλάση παρουσιάζει σημαντικά μειονεκτήματα, όπως :

- Είναι πολύπλοκη και δυσνόητη
- Δεν μπορεί να επαναχρησιμοποιηθεί εύκολα

- Είναι δύσκολα συντηρήσιμη
- Επηρεάζεται σημαντικά από τις αλλαγές

Όπως και το Low Coupling, έτσι και το High Cohesion ισχύει όχι μόνο για κλάσεις αλλά και για τα components μιας εφαρμογής, υποσυστήματα, συστήματα κτλ. Αποτελεί μια πολύ σημαντική αρχή που πρέπει πάντα να λαμβάνεται υπόψη κατά τη λήψη αποφάσεων που σχετίζονται με τη διαδικασία του σχεδιασμού ενός πληροφοριακού συστήματος (Evaluative Principle). Επίσης, το High Cohesion θα πρέπει να εφαρμόζεται σε συνδυασμό με τα υπόλοιπα patterns, όπως το Expert και το Low Coupling, για την ορθή ανάθεση των αρμοδιοτήτων στα objects.

Μια κλάση με High Cohesion έχει λίγες μεθόδους με σχετική μεταξύ τους λειτουργικότητα. Αν η λειτουργικότητα είναι μεγάλη και πολύπλοκη, η κλάση θα πρέπει να συνεργάζεται με άλλες κλάσεις για να την καλύψει. Μια κλάση που παρουσιάζει High Cohesion, είναι κατανοητή, εύκολα συντηρήσιμη και μπορεί να αναβαθμιστεί και να επαναχρησιμοποιηθεί σε μεγάλο βαθμό.

Μια αναλογία σχετική με το High Cohesion υπάρχει και στον πραγματικό κόσμο. Αν σε ένα άτομο αναθέσει κανείς πολλές αρμοδιότητες και μάλιστα όχι σχετικές μεταξύ τους, τότε προφανώς το άτομο αυτό δεν μπορεί να είναι ούτε αποτελεσματικό, ούτε αποδοτικό.

Βέβαια υπάρχουν και κάποιες περιπτώσεις όπου το low cohesion είναι δικαιολογημένο και ανεκτό. Για παράδειγμα σε μια ομάδα όπου υπάρχει ένας μόνο ειδικός στην SQL, πιθανόν να πρέπει να δημιουργηθεί μια κλάση που να περιλαμβάνει πολλές από τις λειτουργίες που χρειάζονται SQL κώδικα και ας μην είναι όλες και τόσο σχετικές μεταξύ τους ή είναι πολλές στον αριθμό. Μια άλλη περίπτωση είναι στα κατανεμημένα (distributed) συστήματα. Τα αντικείμενα που βρίσκονται στο server πρέπει να είναι λιγότερα και με περισσότερες αρμοδιότητες, ώστε να αποφευχθούν προβλήματα απόδοσης και επικοινωνίας στο δίκτυο, λόγω της απομακρυσμένης (remote) πρόσβασης στα objects αυτά.

Γενικά, το High Cohesion εξαρτάται άμεσα από το Low Coupling. Συνήθως μια κλάση με κακό Coupling έχει και κακό Cohesion. Λόγω αυτού του στενού συσχετισμού, στη βιβλιογραφία αναφέρονται σαν το «Yin and Yang» του software engineering.

1.14.5. Controller

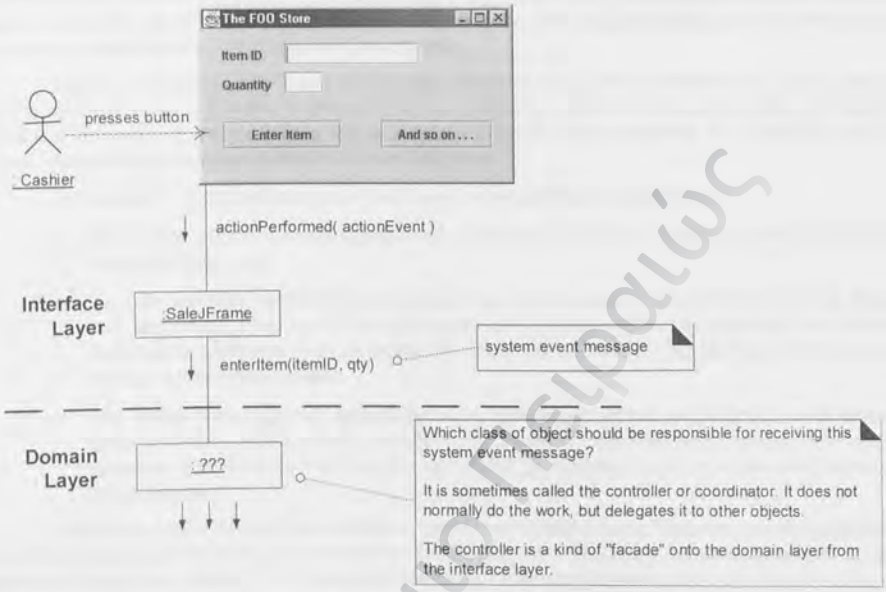
Το pattern αυτό χρησιμοποιείται σε μια πληθώρα περιπτώσεων, όταν θέλουμε να αναθέσουμε την αρμοδιότητα της παραλαβής ή του χειρισμού ενός μηνύματος που αντιστοιχεί σε ένα system event, σε μια κλάση που αντιπροσωπεύει :

- Το ίδιο το σύστημα, ένα υποσύστημα ή μια συσκευή (facade controller)
- Ένα σενάριο ενός Use Case ή ένα ολόκληρο Use Case μέσα στο οποίο συμβαίνει το system event. Αυτές οι κλάσεις συνήθως ονομάζονται <UseCaseName>Handler, <UseCaseName> Coordinator ή <UseCaseName>Session (use case ή session controller)

Με τον όρο system event, εννοούμε μια ενέργεια που προκαλεί ένας εξωτερικός actor στο σύστημα. Τα system events σχετίζονται με λειτουργίες του συστήματος (system operations) οι οποίες εκτελούνται σε απόκριση/απάντηση του συστήματος σε αυτά, κατ' αντιστοιχία με τη σχέση μεταξύ μηνυμάτων και μεθόδων. Ο Controller είναι ένα αντικείμενο το οποίο δέχεται ένα system event και καθορίζει την κατάλληλη μέθοδο που θα εκτελεστεί για την εκπλήρωση του αντίστοιχου system operation.

Πρέπει να σημειωθεί ότι με το design pattern Controller δεν εννοούμε τις οθόνες ή τα παράθυρα μιας εφαρμογής. Σε ένα object oriented σύστημα, το User Interface δεν εκτελεί καθήκοντα (tasks) που σχετίζονται με ένα system event, απλά καταγράφει τα γεγονότα κατά την αλληλεπίδρασή του με το χρήστη και στη συνέχεια τα μεταβιβάζει σε κάποιον Controller. Αυτό σχετίζεται με την γενική αρχή ότι τα στοιχεία του User Interface δεν πρέπει να έχουν καμία ευθύνη για χειρισμό και επεξεργασία των γεγονότων του συστήματος. Η ευθύνη αυτή ανήκει στο application logic ή domain layer (δηλαδή στο κομμάτι του συστήματος το οποίο είναι αρμόδιο για τη διαχείριση όλων των επιχειρηματικών διαδικασιών της εφαρμογής) και όχι στο User Interface ή presentation layer.

Όπως φαίνεται διαγραμματικά στο Σχήμα 15, ο Controller αποτελεί ένα είδος πρόσοψης – βιτρίνας (facade) μεταξύ του interface layer και του domain layer. Συνήθως, ο Controller εξουσιοδοτεί άλλα αντικείμενα για την εκπλήρωση μιας λειτουργίας του συστήματος και ο ρόλος του είναι κυρίως να συντονίζει και να ελέγχει όλες τις απαραίτητες ενέργειες για το σκοπό αυτό.



Σχήμα 15. Ο Ρόλος του Controller σε ένα Object Oriented Σύστημα.

Η πρώτη κατηγορία του Controller, είναι το facade controller, το οποίο αναπαριστά το σύστημα στο σύνολό του, ένα υποσύστημα ή μια συσκευή. Στην περίπτωση αυτή το ίδιο Controller αναλαμβάνει τη διαχείριση όλων των system events που δέχεται από το User Interface. Η χρήση του ενδείκνυται όταν δεν υπάρχουν πολλά system events, ή στην περίπτωση κατά την οποία το User Interface δεν είναι σε θέση να μεταβιβάσει τα γεγονότα του συστήματος σε εναλλακτικά Controllers.

Εάν επιλεγεί η χρήση του use case controller, τότε ένα διαφορετικό Controller διαχειρίζεται τα system events που εμφανίζονται κατά την εκτέλεση ενός Use Case. Το use case controller δεν αποτελεί κλάση του πραγματικού προβλήματος (domain problem), γι' αυτό άλλωστε και δεν εμφανίζεται ποτέ στο Domain Model. Είναι μια τεχνητή (artificial), υποστηρικτική για το σύστημα κλάση. Η περίπτωση του use case controller μας δίνει τη δυνατότητα της καταγραφής της κατάστασης του Use Case, οπότε μπορούμε εύκολα να ελέγξουμε εάν τα system events εκτελούνται με τη σωστή σειρά. Χρησιμοποιείται συνήθως όταν το σύνολο των system events είναι μεγάλο και η χρήση του facade controller, έχει σαν αποτέλεσμα τη δημιουργία low cohesion ή/και high coupling.

Το μεγάλο πλεονέκτημα της χρήσης ενός ή περισσότερων Controllers, είναι η ανεξαρτησία μεταξύ του User Interface και του domain layer, με αποτέλεσμα είτε την επαναχρησιμοποίηση των δομικών συστατικών του συστήματος σε άλλες εφαρμογές, είτε την αλλαγή του User Interface της ίδιας της υπό ανάπτυξη εφαρμογής.

1.15.1. Design Model - Υλοποίηση των Use Cases (Use Case Realization)

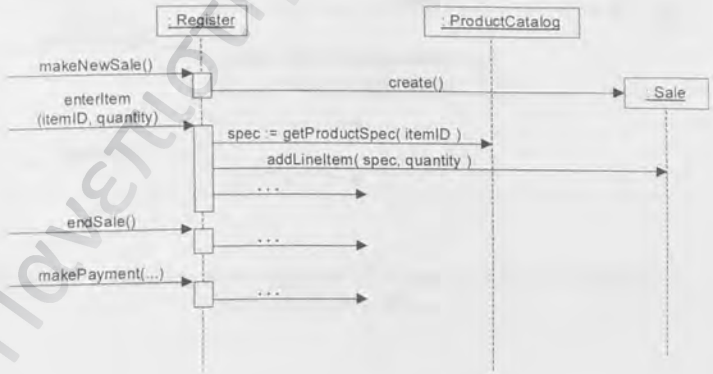
Ο όρος «Υλοποίηση των Use Cases», χρησιμοποιείται στη μεθοδολογία Unified Process, για να αποδώσει τη σχέση μεταξύ των απαιτήσεων που καταγράφονται στα Use Cases και το σχεδιασμό των αντικειμένων για την ικανοποίηση των απαιτήσεων αυτών. Για την Υλοποίηση των Use Cases, η οποία εντάσσεται στα πλαίσια της κατασκευής του Design Model, οι βασικές ενέργειες που διεξάγονται είναι η ανάθεση αρμοδιοτήτων στα αντικείμενα και ο καθορισμός των αλληλεπιδράσεων των αντικειμένων, μέσω των μηνυμάτων που ανταλλάσσουν μεταξύ τους.

Για την Υλοποίηση των Use Cases εφαρμόζονται οι αρχές και οι κανόνες του object oriented σχεδιασμού των πληροφοριακών, οι βασικότερες των οποίων αναφέρθηκαν στις προηγούμενες ενότητες, μέσα από την ανάλυση και τη μελέτη των κυριότερων GRASP Design Patterns. Η Υλοποίηση των Use Cases, αποτυπώνεται σχηματικά στα Interaction Diagrams.

Η διαδικασία της Υλοποίησης των Use Cases, περιλαμβάνει τα εξής βήματα :

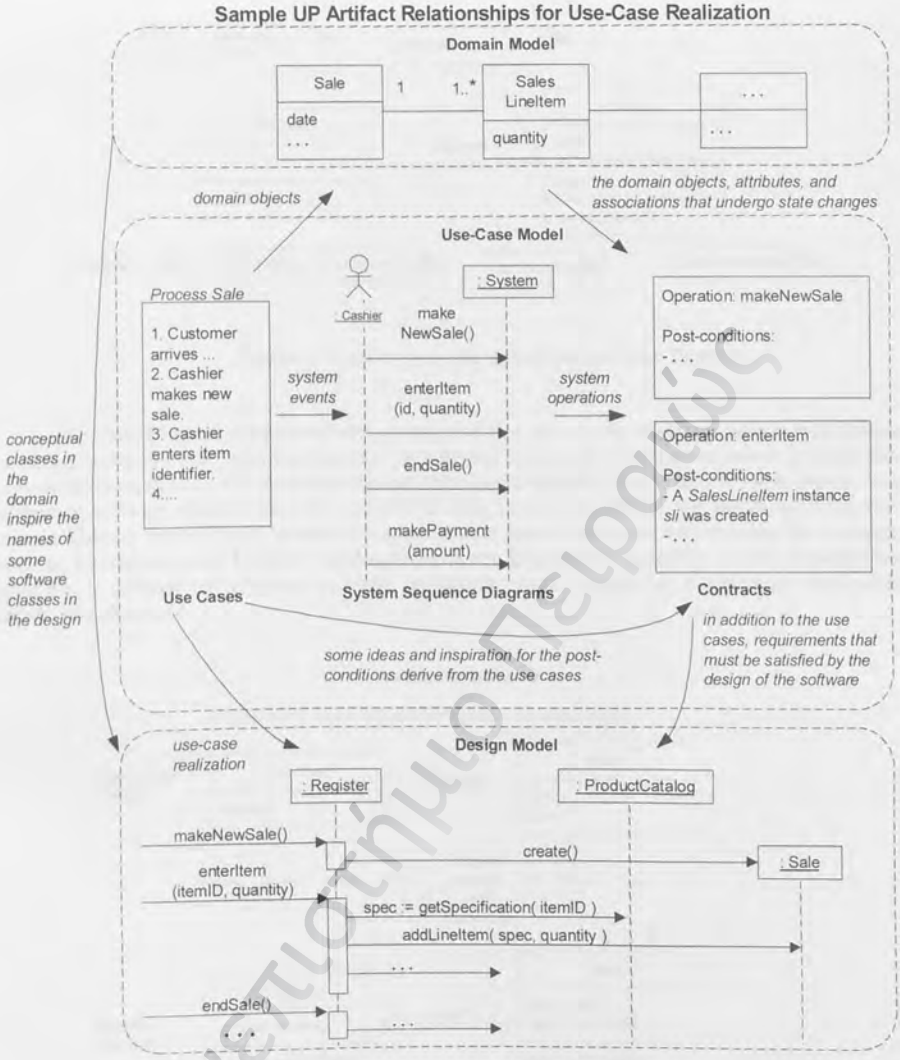
- Τα system events που καταγράφονται μέσα στα Use Cases, αποτυπώνονται στα System Sequence Diagrams.
- Τα system events ουσιαστικά μεταφράζονται σε μηνύματα, τα οποία αποτελούν την έναρξη των Interaction Diagrams. Τα διαγράμματα αυτά, απεικονίζουν τον τρόπο με τον οποίο τα αντικείμενα αλληλεπιδρούν με σκοπό την ικανοποίηση των λειτουργιών του συστήματος, σε απόκριση στα system events.
- Στα Interaction Diagrams, εμφανίζονται τα προγραμματιστικά αντικείμενα και τα διάφορα μηνύματα που ανταλλάσσουν μεταξύ τους. Συνήθως, τόσο για την επιλογή, όσο και την ονομασία των αντικειμένων, το Domain Model χρησιμοποιείται ως η κύρια πηγή έμπνευσης και αναφοράς.

Εφόσον η ομάδα ανάπτυξης επιλέξει τη χρήση των Collaboration Diagrams για την αποτύπωση της Υλοποίησης των Use Cases, τότε απαιτείται ένα διαφορετικό διάγραμμα για κάθε system event. Στην περίπτωση όμως που επιλεγούν τα Sequence Diagrams, τη βασική σημειογραφία των οποίων είδαμε σε προηγούμενη ενότητα, μας δίνεται η δυνατότητα να συμπεριλάβουμε όλα τα system events σε ένα διάγραμμα, όπως φαίνεται στο Σχήμα 16 που ακολουθεί.



Σχήμα 16. Απεικόνιση όλων των System Events σε ένα Sequence Diagram.

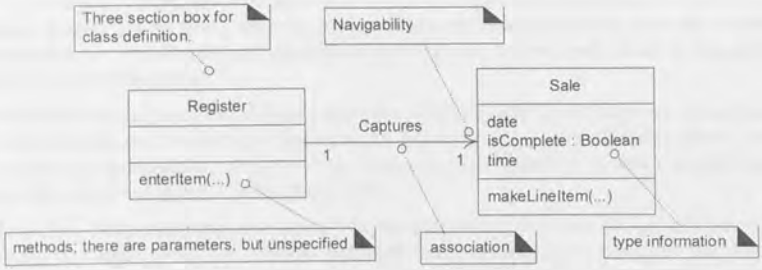
Η Υλοποίηση των Use Cases διενεργείται στα πλαίσια της κατασκευής του Design Model. Στη φάση Elaboration υλοποιούνται τα σημαντικότερα από πλευράς σχεδιασμού και ενεχόμενου ρίσκου Use Cases και συμπληρώνονται στη φάση Construction. Στο Σχήμα 17 που ακολουθεί, φαίνεται η θέση της Υλοποίησης των Use Cases μέσα στη Unified Process και η σχέση της με τα άλλα παραδοτέα της μεθοδολογίας UP.



Σχήμα 17. Η Θέση της Υλοποίησης των Use Cases στη UP και η Σχέση της με τα άλλα Παραδοτέα.

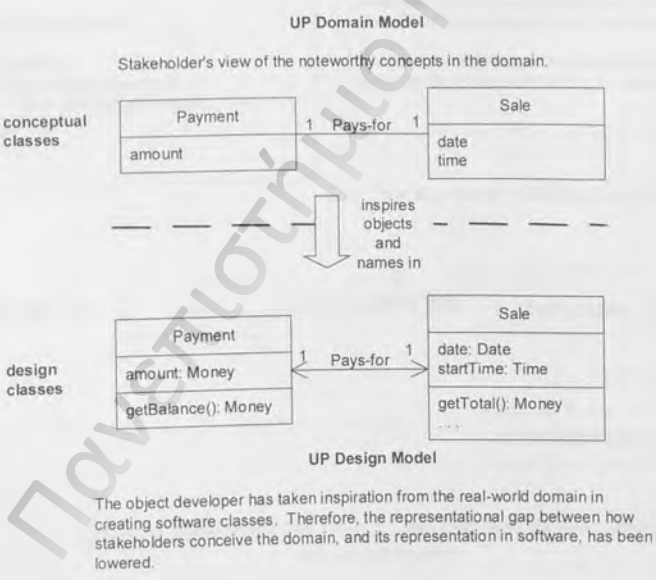
1.15.2. Design Model – Design Class Diagrams

Μετά την ολοκλήρωση των Interaction Diagrams για την υλοποίηση των Use Cases, ακολουθεί η σχεδίαση των Διαγραμμάτων Κλάσεων (στην πράξη πολλές φορές τα Interaction και τα Design Class Diagrams κατασκευάζονται παράλληλα). Στην κατασκευή των Class Diagrams περιγράφονται αναλυτικά και με όλα τα απαραίτητα στοιχεία (attributes, methods, associations κτλ) όλες οι software κλάσεις του συστήματός μας, αυτές δηλαδή που θα υλοποιηθούν από τους προγραμματιστές στη φάση του construction, με τη χρήση κάποιας object oriented γλώσσας προγραμματισμού (Σχήμα 18).



Σχήμα 18. Παράδειγμα ενός απλού Design Class Diagram.

Το Class Diagram συγγενεύει σαν μορφή με το Domain Model, αλλά έχει ουσιαστικές διαφορές με αυτό (Σχήμα 19). Κατ' αρχήν απεικονίζει τις software classes του συστήματος ενώ το Domain Model περιλαμβάνει τις κλάσεις του προβλήματος που εξετάζουμε, δηλαδή έννοιες από τον πραγματικό κόσμο. Βέβαια οι software κλάσεις συνήθως εμπνέονται από το Domain Model και συχνά αντιστοιχούν σε domain κλάσεις (γεγονός που μειώνει το representation gap), αλλά αυτό δεν σημαίνει ότι συμπίπτουν πάντοτε. Επίσης στο Class Diagram περιλαμβάνονται στοιχεία που δεν εμφανίζονται στο Domain Model, όπως π.χ. οι μέθοδοι των κλάσεων, οι τύποι των attributes και η φορά των συσχετίσεων (navigability) μεταξύ των κλάσεων.



Σχήμα 19. Ομοιότητες – Διαφορές μεταξύ Domain Model και Design Class Diagram.

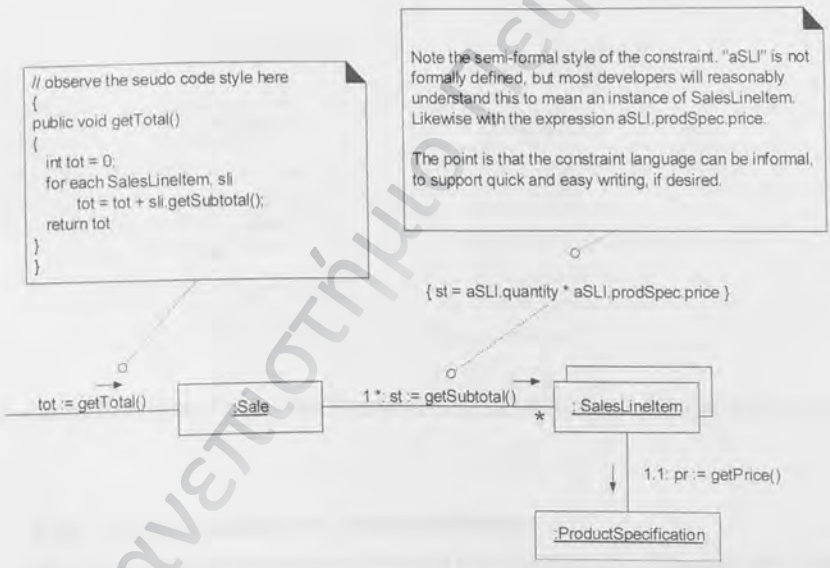
Το πρώτο βήμα για την κατασκευή των Design Class Diagrams είναι να αναλύσουμε όλα τα Interaction Diagrams που έχουμε ήδη κατασκευάσει και να εντοπίσουμε όλες τις κλάσεις που εμφανίζονται σε αυτά. Αφού σχεδιάσουμε τις κλάσεις αυτές στο διάγραμμα, μεταφέρουμε στις κλάσεις τις ιδιότητες από τις αντίστοιχες κλάσεις του Domain Model.

Στη συνέχεια, μελετώντας και πάλι τα Interaction Diagrams, καθορίζουμε τις μεθόδους που θα περιλαμβάνει η κάθε κλάση. Ένας κανόνας που υποβοηθά τη διαδικασία αυτή, είναι ότι οι μέθοδοι μιας κλάσης αντιστοιχούν στο σύνολο των μηνυμάτων που δέχεται η κλάση αυτή, μέσα σε όλα τα Interaction Diagrams στα οποία συμμετέχει.

Οι συσχετίσεις μεταξύ των κλάσεων που απεικονίζονται στα Class Diagrams, προκύπτουν επίσης από την ανασκόπηση των Interaction Diagrams. Σε αντίθεση όμως με το Domain Model, στα Design Class Diagrams μας ενδιαφέρει και η φορά της συσχέτισης (navigability), η οποία συμβολίζεται με ένα βέλος στο άκρο της γραμμής που συνδέει τις κλάσεις.

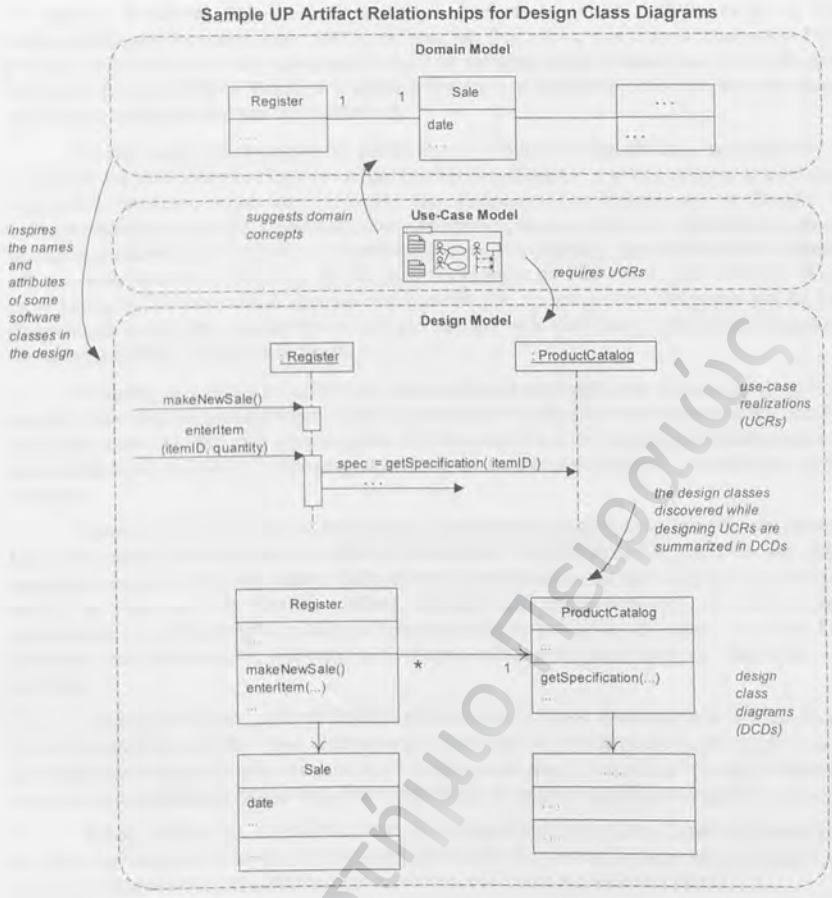
Στα Class Diagrams μπορούμε επίσης να αποτυπώσουμε τον τύπο δεδομένων για τις ιδιότητες των κλάσεων, τις παραμέτρους των μεθόδων καθώς και τις επιστρεφόμενες τιμές των μεθόδων. Συνήθως στο διάγραμμα συμπεριλαμβάνουμε επιλεκτικά τις σπουδαιότερες από τις παραπάνω πληροφορίες, δεδομένου ότι η εμφάνιση όλων των τύπων δεδομένων, επιβαρύνει υπερβολικά το διάγραμμα και το κάνει δυσνόητο.

Τέλος, πολλές φορές είναι ιδιαίτερα χρήσιμο να εμφανίσουμε στα Class Diagrams (αλλά και στα Interaction Diagrams) επιπλέον σχόλια για την επεξήγηση των μεθόδων των κλάσεων, όπως περιορισμούς, αλγορίθμους κτλ. Η σημειογραφία που χρησιμοποιείται στη UML για το σκοπό αυτό, φαίνεται στο Σχήμα 20 που ακολουθεί.



Σχήμα 20. Σχολιασμός των Design Class Diagrams με Αλγορίθμους και Περιορισμούς.

Τα Design Class Diagrams αποτελούν μέρος του Design Model. Αν και το μεγαλύτερο μέρος τους κατασκευάζεται στη φάση Elaboration, συνεχίζουν να σχεδιάζονται και στη φάση Construction, κυρίως υποβοηθητικά για την αποτύπωση της στατικής δομής του συστήματος. Στο Σχήμα 21 που ακολουθεί, φαίνεται η θέση των Design Class Diagrams μέσα στη Unified Process και η σχέση τους με τα άλλα παραδοτέα της μεθοδολογίας UP.



Σχήμα 21. Η Θέση των Design Class Diagrams στη UP και η Σχέση τους με τα άλλα Παραδοτέα.

1.16. Από το Σχεδιασμό στη Συγγραφή Κώδικα

Με την ολοκλήρωση των Interaction Diagrams και των Design Class Diagrams για ένα δεδομένο iteration, έχουμε στη διάθεσή μας αρκετό υλικό ώστε να προχωρήσουμε στη συγγραφή του κώδικα, δηλαδή στον προγραμματισμό. Ο πηγαίος κώδικας, η δημιουργία της βάσης δεδομένων του συστήματος, αλλά και κάποια άλλα παραδοτέα όπως π.χ. οι HTML ή XML σελίδες που πιθανόν απαιτούνται, ανήκουν στο μοντέλο το οποίο στη μεθοδολογία Unified Process καλείται Implementation Model.

Η συγγραφή του κώδικα σε μία object oriented γλώσσα προγραμματισμού, δεν αποτελεί μέρος της object oriented ανάλυσης και σχεδιασμού, αποτελεί τον τελικό στόχο για τη δημιουργία μιας εφαρμογής. Υλοποιούμενες οι δραστηριότητες αυτές μέσα στο πλαίσιο που θέτει η μεθοδολογία Unified Process, συνεργάζονται αρμονικά μεταξύ τους και παρέχουν το πλάνο για την ανάπτυξη ενός πληροφοριακού συστήματος, από την αρχή μέχρι το τέλος, δηλαδή από τις απαιτήσεις μέχρι τον κώδικα.

Τα παραδοτέα του Design Model που κατασκευάστηκαν κατά τη διαδικασία του σχεδιασμού, παρέχουν σε μεγάλο βαθμό τις απαραίτητες πληροφορίες για τη συγγραφή του πηγαίου κώδικα. Βέβαια, η ομάδα των προγραμματιστών δεν έχει απλά να μεταφέρει μηχανικά τα αποτελέσματα του σχεδιασμού

σε κώδικα. Αντίθετα, έχει πολύ δημιουργική δουλειά να κάνει, καθώς κατά τη διάρκεια του προγραμματισμού θα προκύψουν πολλές αλλαγές και βελτιώσεις στα αποτελέσματα του Design Model. Επίσης, κατά τη διάρκεια του προγραμματισμού θα προκύψουν νέα θέματα και νέοι προβληματισμοί που θα πρέπει να μελετηθούν, ώστε να συμπληρωθούν και να ραφιναριστούν τα διάφορα παραδοτέα του σχεδιασμού (ενδεχομένως και της ανάλυσης).

Τη διαδικασία αυτή έρχεται να βοηθήσει και η ίδια η Unified Process, που είναι εξ' ορισμού μια επαναληπτική μεθοδολογία. Π.χ. όταν είμαστε στην επανάληψη **v**, ο κώδικας που παράγεται συνήθως δεν συμβαδίζει απόλυτα, αλλά έχει αλλαγές και βελτιώσεις σε σχέση με το Design Model που κατασκευάστηκε στην επανάληψη αυτή. Στην περίπτωση αυτή, μπορούμε στην επόμενη επανάληψη **v+1**, να λάβουμε υπόψη μας τον κώδικα της προηγούμενης επανάληψης, χρησιμοποιώντας κάποιο Case Tool και κάνοντας reverse engineering (δημιουργία διαγραμμάτων από τον πηγαίο κώδικα) τον κώδικα της επανάληψης **v**. Με τον τρόπο, έχουμε στη διάθεσή μας τα Interaction Diagrams και τα Design Class Diagrams τα οποία θα τροφοδοτήσουν την επανάληψη **v+1**, έτσι όπως αυτά αναπροσαρμόστηκαν κατά τον προγραμματισμό της επανάληψης **v**.

Η πρώτη ενέργεια που πρέπει να γίνει κατά τη δημιουργία του κώδικα με τη χρήση μιας objet oriented γλώσσας προγραμματισμού, είναι ο ορισμός των κλάσεων που αποτυπώνονται στα Design Class Diagrams. Όπως είδαμε, στα διαγράμματα αυτά αναφέρονται τα ονόματα των κλάσεων, οι μέθοδοι με τον ορισμό τους, δηλαδή τις παραμέτρους και την επιστρεφόμενη τιμή τους και βέβαια οι ιδιότητες των κλάσεων.

Επίσης, στη φάση αυτή, οι συσχετίσεις (associations) μεταξύ των κλάσεων που εμφανίζονται στα Class Diagrams μετατρέπονται σε reference attributes, δηλαδή ιδιότητες μέσα σε μια κλάση που δεν εκφράζονται με πρωτογενείς τύπους δεδομένων, αλλά αναφέρονται σε άλλες κλάσεις του μοντέλου. Σε αυτήν την περίπτωση το ίδιο το attribute αποτελεί μια κλάση και όχι ένα απλό κείμενο, αριθμό, ημερομηνία κτλ. Εδώ βέβαια χρειάζεται προσοχή διότι τα reference attributes, στα Class Diagrams δεν φαίνονται σαν πραγματικές ιδιότητες μια κλάσης, αλλά απορρέουν από τις συσχετίσεις μεταξύ των κλάσεων.

Το επόμενο βήμα, μετά τη δημιουργία του σκελετού των κλάσεων από τα Class Diagrams, είναι η κατασκευή των μεθόδων των κλάσεων, χρησιμοποιώντας τα Interaction Diagrams που δείχνουν τα μηνύματα που ανταλλάσσουν οι διάφορες κλάσεις μεταξύ τους. Η σειρά με την οποία στέλνονται αυτά τα μηνύματα, μεταφράζεται σε μια σειρά από εντολές μέσα σε μια συγκεκριμένη μέθοδο μιας κλάσης.

Τέλος, σχετικά με τη σειρά με τη οποία υλοποιούνται οι κλάσεις και οι μέθοδοι των κλάσεων, συνήθως κατασκευάζονται πρώτα οι κλάσεις που έχουν τις λιγότερες συσχετίσεις, δηλαδή το μικρότερο coupling, προχωρώντας σταδιακά μέχρι την κατασκευή του συνόλου των κλάσεων.

1.17. Γλωσσάρι Όρων Πληροφορικής

Με το τέλος του 1ου μέρους της διπλωματικής εργασίας, δίνεται ένας συνοπτικός ορισμός των βασικότερων όρων πληροφορικής που εμφανίστηκαν μέχρι τώρα στο κείμενο.

- **Object = Αντικείμενο**

Οντότητα με σαφή όρια και μοναδική ταυτότητα. Τα objects έχουν κατάσταση (state) και συμπεριφορά (behavior). Η κατάσταση ενός object προσδιορίζεται από τις ιδιότητες και τις σχέσεις του, ενώ η συμπεριφορά του καθορίζεται από τις μεθόδους του αντικειμένου. Ένα object αποτελεί ένα μέλος-στιγμιότυπο μιας κλάσης (class instance).

- **Systems Analysis = Ανάλυση Πληροφοριακών Συστημάτων**

Το μέρος εκείνο της διαδικασίας ανάπτυξης πληροφοριακών συστημάτων, πρωταρχικός σκοπός του οποίου είναι η δημιουργία ενός μοντέλου για την περιγραφή και την αποτύπωση του υπό εξέτασης προβλήματος του πραγματικού κόσμου καθώς και η ανάλυση των απαιτήσεων του συστήματος. Η Ανάλυση εστιάζεται στο **ΤΙ** πρέπει να κάνει ένα πληροφοριακό σύστημα..

- **Systems Design = Σχεδιασμός Πληροφοριακών Συστημάτων**

Το μέρος εκείνο της διαδικασίας ανάπτυξης πληροφοριακών συστημάτων, πρωταρχικός σκοπός του οποίου είναι η σχεδίαση μιας λύσης για την υλοποίηση του συστήματος. Ο Σχεδιασμός εστιάζεται στον τρόπο (**ΠΩΣ**) με τον οποίο θα ικανοποιηθούν οι απαιτήσεις του συστήματος.

- **CASE Tool (Computer Assisted Software Engineering Tool) = Εργαλείο Ανάπτυξης**

Εμπορικό λογισμικό το οποίο χρησιμοποιείται από Αναλυτές συστημάτων, Σχεδιαστές και Προγραμματιστές, για τη δημιουργία πολλών από τα παραδοτέα που απαιτούνται κατά τη διάρκεια της ανάπτυξης ενός πληροφοριακού συστήματος. Σχεδόν όλα τα σύγχρονα CASE Tools παρέχουν τη δυνατότητα για Reverse Engineering, δηλαδή δημιουργία διαγραμμάτων και ανάκτηση πληροφοριών από τον πηγαίο κώδικα της εφαρμογής ή από σχεσιακές βάσεις δεδομένων.

- **Functional Requirements = Λειτουργικές Απαιτήσεις**

Οι απαιτήσεις που αναφέρονται στη λειτουργικότητα του συστήματος, δηλαδή στις δυνατότητες και τα χαρακτηριστικά του. Οι λειτουργικές απαιτήσεις καθορίζουν τις λειτουργίες και τις ενέργειες εκείνες, τις οποίες το σύστημα είναι (ή θα πρέπει να είναι) σε θέση να υλοποιήσει, προδιαγράφοντας τη συμπεριφορά εισόδου και εξόδου του συστήματος.

- **Non Functional Requirements = Μη Λειτουργικές Απαιτήσεις**

Όλες οι υπόλοιπες απαιτήσεις του συστήματος πέραν των λειτουργικών. Συνήθως περιλαμβάνουν απαιτήσεις αξιοπιστίας, απόδοσης, ευχρηστίας, συντηρησιμότητας, ευελιξίας, παραμετροποίησης κλπ.

- **Actor**

Εξωτερική οντότητα (δεν ανήκει στο υπό εξέταση σύστημα) η οποία αλληλεπιδρά με το σύστημα. Μπορεί να είναι πρόσωπο, οργανισμός, άλλο σύστημα ή μηχανή. Διακρίνονται τρεις κατηγορίες : οι Primary Actors χρησιμοποιούν το σύστημα για να εκπληρώσουν κάποιους στόχους τους, οι Supporting Actors παρέχουν υπηρεσίες (π.χ. πληροφορίες) στο σύστημα και τέλος οι Offstage Actors ενδιαφέρονται για τη συμπεριφορά του συστήματος, χωρίς να ανήκουν σε μία από τις δύο παραπάνω

κατηγορίες (π.χ. ένας κρατικός φορέας ο οποίος επιθυμεί να ενημερώνεται για τις κινήσεις – Transactions του συστήματος).

- **System Events - Operations = Γεγονότα - Λειτουργίες Συστήματος**

Με τον όρο system event, νοείται μια ενέργεια που προκαλεί ένας εξωτερικός Actor στο σύστημα. Τα system events σχετίζονται με λειτουργίες του συστήματος (system operations) οι οποίες εκτελούνται σε απόκριση/απάντηση του συστήματος σε αυτά.

- **Class = Κλάση**

Ο όρος αυτός χρησιμοποιείται για την περιγραφή μιας ομάδας αντικειμένων με κοινές ιδιότητες, κοινές μεθόδους και κοινές συσχετίσεις. Τα μέλη-στιγμιότυπα μιας κλάσης (class instances) αποτελούν τα αντικείμενα της κλάσης. Οι κλάσεις διακρίνονται σε : software ή design classes οι οποίες αναπαριστούν προγραμματιστικά αντικείμενα (software objects) που υλοποιούνται με τη βοήθεια μιας γλώσσας προγραμματισμού και τις conceptual ή domain classes, οι οποίες αναπαριστούν έννοιες του πραγματικού κόσμου (domain objects).

- **Association = Συσχέτιση**

Λογική σχέση η οποία υποδηλώνει την ύπαρξη διασύνδεσης μεταξύ των μελών-στιγμιοτύπων των κλάσεων.

- **Attribute = Ιδιότητα**

Ιδιότητα μιας κλάσης ονομάζεται ένα δεδομένο (datum), το οποίο αποτελεί ένα χαρακτηριστικό της κλάσης και μπορεί να πάρει συγκεκριμένη τιμή.

- **Multiplicity = Πολλαπλότητα**

Η πολλαπλότητα αναφέρεται στη συσχέτιση μεταξύ δύο κλάσεων, καθορίζοντας τον αριθμό των μελών-στιγμιότυπων της κλάσης Α που μπορούν να συσχετιστούν με ένα μέλος-στιγμιότυπο της κλάσης Β σε μια δεδομένη χρονική στιγμή.

- **Aggregation**

Ειδικός τύπος συσχέτισης μεταξύ κλάσεων που απεικονίζει μια σχέση whole-part, δηλαδή του ολόκληρου με τα απάρτιά του ή του συναρμολογήματος με τα εξαρτήματά του. Στην περίπτωση του Aggregation η κλάση Α (part) αποτελεί φυσικό ή λογικό τμήμα-μέρος της κλάσης Β (whole).

- **Generalization = Γενίκευση**

Ειδικός τύπος συσχέτισης μεταξύ μιας κλάσης η οποία εκφράζει μια γενικότερη έννοια και καλείται υπερ-κλάση (super class) και δύο ή περισσότερων υπο-κλάσεων (sub class), οι οποίες αντιστοιχούν σε πιο ειδικές έννοιες, έχουν όμως κοινά χαρακτηριστικά με την υπερ-κλάση και αποτελούν υποσύνολα αυτής. Οι υποκλάσεις είναι πλήρως συμβατές με την υπερ-κλάση, περιέχουν όμως επιπρόσθετες πληροφορίες. Μια κλάση, αποτελεί υπο-κλάση μιας υπερ-κλάσης όταν όλα τα μέλη-στιγμιότυπα της υπο-κλάσης είναι επίσης μέλη της υπερ-κλάσης και παράλληλα όλες οι ιδιότητες και οι συσχετίσεις της υπερ-κλάσης, εφαρμόζονται χωρίς καμία εξαίρεση και στην υπο-κλάση. Με τη χρήση του Generalization, οι διάφορες κλάσεις οργανώνονται και ταξινομούνται σε ιεραρχίες κλάσεων.

- **Inheritance = Κληρονομικότητα**

Μηχανισμός που παρέχει τη δυνατότητα στις υπο-κλάσεις να χρησιμοποιούν τις ιδιότητες, τις συσχετίσεις και τις μεθόδους της ή των υπερ-κλάσεων στις οποίες ανήκουν. Η Κληρονομικότητα αποτελεί επίσης μια object oriented τεχνική, που επιτρέπει τη χρήση υφιστάμενων κλάσεων για τη δημιουργία άλλων κλάσεων.

- **Method = Μέθοδος**

Η μέθοδος καθορίζει έναν συστηματικό τρόπο (αλγόριθμος ή διαδικασία) για την εκπλήρωση μιας λειτουργίας ενός αντικειμένου. Οι μέθοδοι μιας κλάσης προσδιορίζουν τη συμπεριφορά της κλάσης.

- **Responsibility = Αρμοδιότητα**

Οι αρμοδιότητες προσδιορίζουν τις δυνατότητες και τις υποχρεώσεις μιας κλάσης. Για την εκπλήρωση μιας αρμοδιότητας, απαιτείται η εκτέλεση μιας ή περισσότερων μεθόδων της κλάσης καθώς και η συνεργασία με μεθόδους άλλων κλάσεων του συστήματος.

- **Design Pattern**

Ένα design pattern περιγράφει ένα αντιπροσωπευτικό και συχνά εμφανιζόμενο στην καθημερινή πρακτική της πληροφορικής πρόβλημα σχεδιασμού, μαζί με την αντίστοιχη λύση του. Τα design patterns (τυπικά τουλάχιστον) δεν εξαρτώνται από κάποια συγκεκριμένη γλώσσα προγραμματισμού.

ΜΕΡΟΣ 2ο

2.1. Πληροφοριακά Συστήματα Διοίκησης Παραγωγής

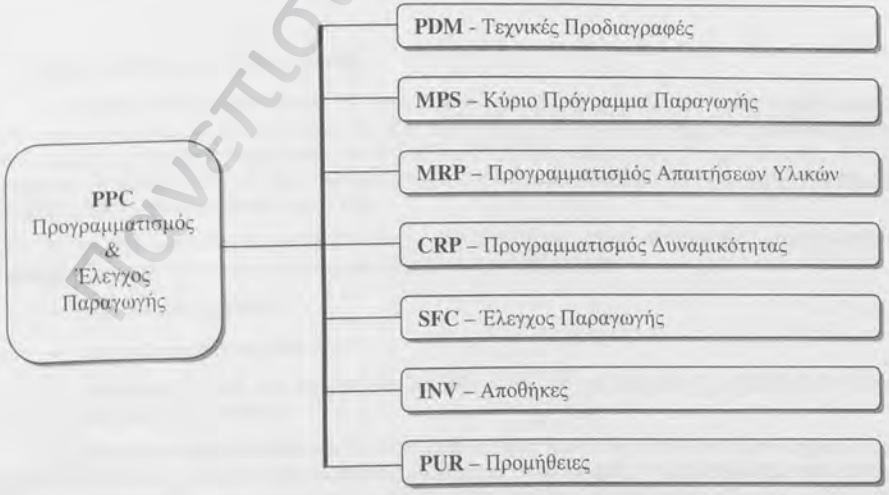
Τα Πληροφοριακά Συστήματα Διοίκησης Παραγωγής αναπτύσσονται και εφαρμόζονται για αρκετές ήδη δεκαετίες, εξελισσόμενα ταυτόχρονα με τις επιστήμες της Επιχειρησιακής Έρευνας και της Πληροφορικής.

Στις αρχές της δεκαετίας του '60 έκαναν την εμφάνισή τους τα συστήματα Ελέγχου Αποθεμάτων (Inventory Control – IC), τα οποία έδιναν έμφαση στη βελτίωση της διαχείρισης των αποθεμάτων των επιχειρήσεων. Στη συνέχεια, στις αρχές της δεκαετίας του '70 παρουσιάστηκαν τα συστήματα MRP (Material Requirements Planning), εισάγοντας τις βασικές αρχές της βιομηχανικής μοντελοποίησης με τη δημιουργία των δομών BOM (Bill of Materials). Ο στόχος των συστημάτων αυτών είναι ο προγραμματισμός των απαιτούμενων πρώτων υλών και εξαρτημάτων για την παραγωγή συγκεκριμένου μίγματος προϊόντων.

Εξέλιξη των συστημάτων MRP στα τέλη της δεκαετίας του '70, αποτέλεσαν τα συστήματα MRP II (Manufacturing Resources Planning). Σκοπός των συστημάτων αυτών είναι η βελτιστοποίηση της διαχείρισης όλων των πόρων της παραγωγής, διαχειριζόμενα το σύνολο της εμπλεκόμενης πληροφορίας στο βιομηχανικό περιβάλλον. Τέλος, στα τέλη της δεκαετίας του '80, η εισαγωγή των συστημάτων ERP (Enterprise Resources Planning), είχε ως αποτέλεσμα τη διαχείριση του συνόλου των πληροφοριών μιας επιχείρησης, καλύπτοντας όλες τις ανάγκες πληροφοριακής υποστήριξης.

Στο Σχήμα 22 που ακολουθεί, παρατίθενται τα βασικά υποσυστήματα ενός συστήματος MRP II (το οποίο αποτελεί υποσύνολο ενός σύγχρονου ολοκληρωμένου πληροφοριακού συστήματος ERP). Από το σχήμα αυτό καθίσταται προφανές ότι το MRP, αποτελεί ένα μέρος ενός συστήματος MRP II, στο οποίο τα υπάρχοντα αποθέματα, οι προγνώσεις ζήτησης του τμήματος πωλήσεων, οι παραγγελίες των πελατών και οι προτεραιότητες της Διοίκησης, λαμβάνονται υπόψη για την κατάστρωση του Κύριου Προγράμματος Παραγωγής (MPS – Master Production Schedule) των τελικών προϊόντων.

Στη συνέχεια, το MRP με τη χρήση των Πινάκων Υλικών (τμήμα των Τεχνικών Προδιαγραφών), δημιουργεί τα αναλυτικά προγράμματα αγορών για τις πρώτες ύλες και τα προγράμματα παραγωγής για τα ενδιάμεσα εξαρτήματα των προϊόντων. Με τον στενά συνδεδεμένο με τα MPS και MRP προγραμματισμό δυναμικότητας (CRP - Capacity Requirements Planning), επιτυγχάνεται ο έλεγχος και η εξισορρόπηση του φόρτου απασχόλησης των παραγωγικών πόρων, έτσι ώστε τόσο το πρόγραμμα παραγωγής των τελικών προϊόντων όσο και τα προγράμματα παραγωγής των εξαρτημάτων να είναι πραγματοποιήσιμα, εξασφαλίζοντας συγχρόνως έναν ομαλό ρυθμό εργασίας.



Σχήμα 22. Βασικά Υποσυστήματα (Modules) ενός Συστήματος MRP II.

Η επιτυχία ενός συστήματος MRP II εξαρτάται από την ταχύτητα και την ακρίβεια ανάδρασης του συστήματος παραγωγής με πληροφορίες για την ημερήσια παραγωγή, την κίνηση των αποθεμάτων, την παρουσία των εργαζομένων, την απασχόληση των μηχανών, την παραγωγή σκάρτων, και γενικά για την τρέχουσα κατάσταση όλων των βιομηχανικών πόρων που υπεισέρχονται στην εκτέλεση των προγραμμάτων παραγωγής. Η συλλογή και επεξεργασία των ανωτέρω πληροφοριών, γίνεται (είτε αυτόματα είτε χειρονακτικά) μέσω του υποσυστήματος του ελέγχου παραγωγής (SFC – Shop Floor Control).

Στις ενότητες που ακολουθούν, αφού πρώτα αναφερθούμε στους Πίνακες Υλικών οι οποίοι αποτελούν αναπόσπαστο τμήμα των συστημάτων MRP II, θα αναπτύξουμε αναλυτικότερα τα υποσυστήματα MPS και MRP, η ανάλυση και ο σχεδιασμός των οποίων στα πλαίσια της ανάπτυξης ενός object oriented πληροφοριακού συστήματος, αποτελούν το αντικείμενο της εργασίας αυτής.

2.2. Τεχνικές Προδιαγραφές (PDM)

Το θεμέλιο πάνω στο οποίο στηρίζονται όλες οι διαδικασίες και οι μέθοδοι της οργάνωσης παραγωγής, είναι οι τεχνικές προδιαγραφές (PDM – Production Data Management), δηλαδή η τεκμηρίωση των προϊόντων, των υλικών από τα οποία συγκροτούνται, των κατεργασιών τις οποίες υφίστανται και των αντίστοιχων μέσων παραγωγής. Οι απαραίτητες αυτές πληροφορίες, που μεσοπρόθεσμα έως μακροπρόθεσμα έχουν μόνιμο χαρακτήρα, οργανώνονται σε τρεις κύριες ομάδες :

- Πίνακες Υλικών – Συνταγολόγια
- Φασεολόγια ή Διαδρομές Παραγωγής (διαδοχή φάσεων κατεργασίας, χρόνοι παραγωγής)
- Μέσα Παραγωγής (κέντρα εργασίας, μηχανήματα, εργαλεία)

Είναι ιδιαίτερα μεγάλη η σημασία της ποιότητας των ανωτέρω βασικών πληροφοριών της παραγωγής (ακρίβεια και βαθμός ενημερότητας), ώστε να έχουν πλήρη αντιστοιχία με την πραγματικότητα μέσα στο χώρο του εργοστασίου. Η σπουδαιότητα αυτή γίνεται ακόμη μεγαλύτερη όταν ο προγραμματισμός και ο έλεγχος της παραγωγής υποστηρίζεται από βιομηχανικό λογισμικό. Όταν οι διαδικασίες αυτοματοποιούνται και ως ένα βαθμό απουσιάζει η ευελιξία του ανθρώπινου παράγοντα στην αναγνώριση των σφαλμάτων, είναι απαραίτητη η αναδιοργάνωση και η ουσιαστική αναβάθμιση του τρόπου με τον οποίο αντιμετωπίζεται η τεκμηρίωση των τεχνικών προδιαγραφών της παραγωγής.

2.2.1. Πίνακες Υλικών (BOM)

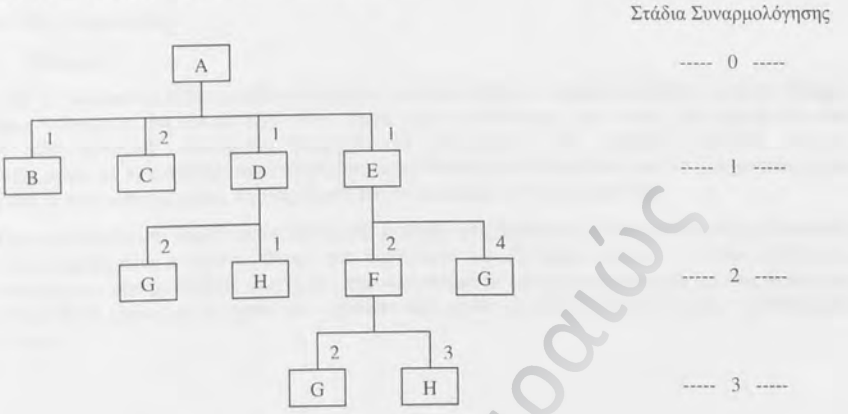
Ο Πίνακας Υλικών απεικονίζει τις πρώτες ύλες, τα εξαρτήματα και τα συγκροτήματα από τα οποία αποτελείται ένα προϊόν, τις ποσότητες που απαιτούνται για την παραγωγή μιας μονάδας (ή μιας τυπικής παρτίδας) του προϊόντος, καθώς και τα διαδοχικά στάδια κατασκευής/συναρμολόγησης. Ο όρος πίνακες υλικών (BOM – Bill of Materials) προέρχεται από τη μηχανολογία, ενώ αντί αυτού στις χημικές βιομηχανίες προτιμάται ο όρος συνταγολόγια.

Η τεκμηρίωση των πινάκων υλικών – συνταγολογίων είναι απαραίτητη προϋπόθεση σε οποιαδήποτε βιομηχανία για τη εκτέλεση των εξής βασικών διαδικασιών :

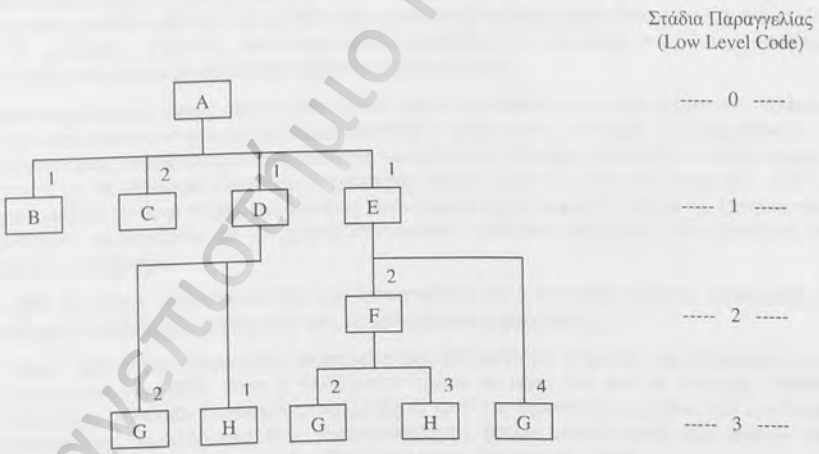
- Μελέτη ενός προϊόντος
- Κοστολόγηση των προϊόντων
- Προγραμματισμός των προμηθειών πρώτων υλών και της παραγωγής εξαρτημάτων (μέσω της τεχνικής του MRP)

Οι πίνακες υλικών αποτυπώνουν την δεντρική δομή/σύνθεση των προϊόντων. Στο Σχήμα 23 που ακολουθεί δίνεται το δεντρικό διάγραμμα ενός υποθετικού προϊόντος Α. Οι διακλαδώσεις του δέντρου αντιστοιχούν στα στάδια κατασκευής/συναρμολόγησης, τα οποία αριθμούνται αρχίζοντας με 0 (μηδέν) στο στάδιο του τελικού προϊόντος. Εύκολα διακρίνει κανείς, ότι υπάρχουν κοινά υλικά σε διαφορετικά στάδια. Το γεγονός αυτό δημιουργεί προβλήματα στον προγραμματισμό των παραγγελιών των υλικών

και για το λόγο αυτό, είναι σκόπιμο αντί για το πραγματικό στάδιο συναρμολόγησης να χρησιμοποιείται το λεγόμενο χαμηλότερο σημείο εμφάνισης του υλικού (LLC – Low Level Code), όπως φαίνεται παραστατικά στο Σχήμα 24. Έτσι, είναι δυνατή η παραγγελία των υλικών (π.χ τα υλικά G και H) σε μεγάλες και χρονικά συντονισμένες παρτίδες. Τα επίπεδα του δέντρου του προϊόντος ονομάζονται πλέον στάδια παραγγελίας, αντί για στάδια συναρμολόγησης.



Σχήμα 23. Δεντρικό Διάγραμμα της Δομής ενός Προϊόντος.



Σχήμα 24. Δεντρικό Διάγραμμα της Δομής ενός Προϊόντος με Στάδια Παραγγελίας (LLC).

2.2.2. Είδη Πινάκων Υλικών

Οι πίνακες υλικών ταξινομούνται σε δύο μεγάλες κατηγορίες, τους αναλυτικούς (explosive lists) και τους συνθετικούς (implosive lists) ή πίνακες χρήσης υλικών (where-used lists). Οι αναλυτικοί πίνακες απαντούν στο ερώτημα «από ποια εξαρτήματα και πρώτες ύλες αποτελείται ένα προϊόν», αναπαριστώντας την δεντρική δομή του προϊόντος. Οι συνθετικοί πίνακες έχουν την ανάστροφη δεντρική μορφή από τους αναλυτικούς και δίνουν απάντηση στο ερώτημα «σε ποια προϊόντα χρησιμοποιείται ένα

εξάρτημα ή μια πρώτη ύλη». Τόσο οι αναλυτικοί όσο και οι συνθετικοί πίνακες υλικών ταξινομούνται σε τέσσερις υποκατηγορίες :

- Ποσοτικοί
- Δομικοί
- Με Παραλλαγές
- Βαθμικοί

Από τα παραπάνω είδη, μεγάλο ενδιαφέρον παρουσιάζουν οι βαθμικοί πίνακες υλικών (Single-Level Bills), δεδομένου ότι αποτελούν τον de facto τρόπο αποτύπωσης της δομής των προϊόντων στα σύγχρονα πληροφοριακά συστήματα βιομηχανικού λογισμικού. Οι βαθμικοί πίνακες υλικών, αναφέρονται μόνο σε ένα στάδιο κατασκευής/συναρμολόγησης του προϊόντος και ως εκ τούτου, είναι ιδιαίτερα απλοί στην καταχώρηση, τη διαχείριση και τη συντήρησή τους γενικότερα.

Για την απεικόνιση της πλήρους δεντρικής μορφής ενός προϊόντος (ή της ανάστροφης δεντρικής μορφής ενός εξαρτήματος ή πρώτης ύλης), στη περίπτωση των βαθμικών πινάκων υλικών, εκτελείται σύνθετη αναζήτηση που βασίζεται στις αρχές της «αναδρομής» (recursion), δηλαδή δημιουργείται μια αλυσιδωτή σύνδεση (join) του αρχείου των σχέσεων των ειδών με τον εαυτό του, σε όλο το βάθος του πίνακα υλικών.

2.3. Κύριο Πρόγραμμα Παραγωγής (MPS)

Το Master Production Schedule ή Κύριο Πρόγραμμα Παραγωγής αφορά κατά κύριο λόγο παραγωγικές επιχειρήσεις. Η χρησιμοποίησή του συνίσταται στην εύρεση του τι θα παράξει μια επιχείρηση σε ένα χρονικό ορίζοντα που ποικίλει από μερικές εβδομάδες μέχρι τρεις μήνες η ακόμα και έξι μήνες. Ο χρονικός ορίζοντας προγραμματισμού χωρίζεται σε χρονικές περιόδους, συνήθως εβδομάδες, αν και μπορούν να οριστούν σαν περίοδοι ημέρες ή μήνες.

Ο προγραμματισμός αυτός αφορά μόνο τελικά προϊόντα (έκδοση εντολών παραγωγής τελικών προϊόντων) και κατά κανόνα δε μελετά ημιέτοιμα προϊόντα ή πρώτες ύλες. Το κύριο χαρακτηριστικό των τελικών προϊόντων μιας επιχείρησης είναι η λεγόμενη «ανεξάρτητη ζήτηση», δηλαδή δεν υπάρχει μεταξύ τους εξάρτηση ή με τα υπόλοιπα υλικά της επιχείρησης, αλλά η ζήτησή τους διαμορφώνεται από τις συνθήκες που επικρατούν στην αγορά και συνήθως είναι τυχαία και συνεχής. Το ύψος της ζήτησης των τελικών προϊόντων υπολογίζεται με τη χρήση στατιστικών μεθόδων πρόβλεψης και βασίζεται σε ιστορικά δεδομένα πωλήσεων.

Το MPS δε μελετά τη δυναμικότητα του εργοστασίου, αλλά προτείνει εντολές παραγωγής με βάση την επιθυμητή στάθμη των αποθεμάτων και τις υπάρχουσες παραγγελίες.

Το Κύριο Πρόγραμμα Παραγωγής αποτελεί μέρος της μεσαίας κλίμακας της διοίκησης και ο υπεύθυνος δημιουργίας του MPS, είναι ή τουλάχιστον πρέπει να είναι ένα μεσαίο στέλεχος (Middle Manager). Γενικότερα οι Middle Managers επωμίζονται το έργο της δημιουργίας σχεδίων και εκτέλεσης ενεργειών μεσαίου έως χαμηλού επιπέδου, γεφυρώνοντας το χάσμα μεταξύ χονδρικών πλάνων και γενικότερης στρατηγικής με τα καθημερινά προβλήματα λειτουργίας της επιχείρησης.

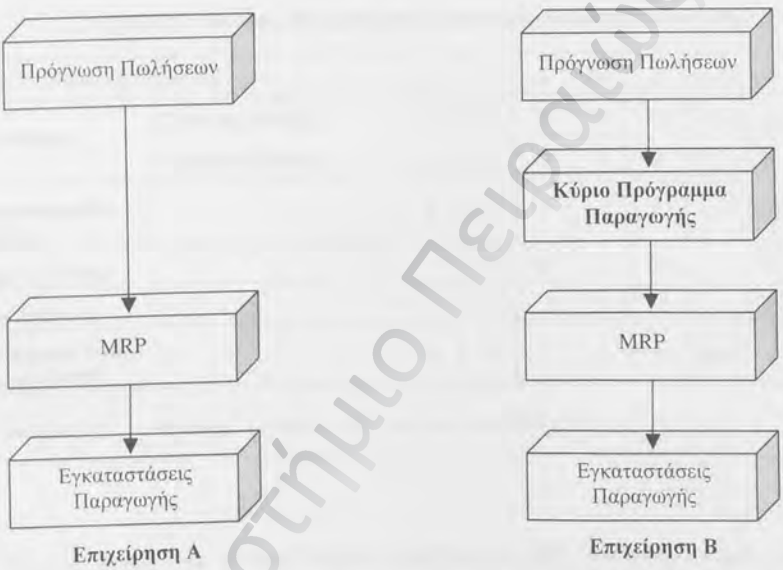
Ο υπεύθυνος δημιουργίας του MPS είναι ο συνδετικός κρίκος μεταξύ του τμήματος πωλήσεων και μάρκετινγκ και του τμήματος κατασκευών. Έτσι, έχει την αρμοδιότητα να αποσβένει μέσω του MPS τις διακυμάνσεις της ζήτησης των τελικών προϊόντων (με την τήρηση αποθέματος), τις οποίες το τμήμα κατασκευών από μόνο του δε μπορεί να ικανοποιήσει. Αλλά ακόμα και αν το τμήμα κατασκευών έχει την ικανότητα να ακολουθεί τις διακυμάνσεις της ζήτησης, το κόστος που αυτό συνεπάγεται συνήθως είναι τόσο μεγάλο που καθιστά αυτή την τακτική ασύμφορη.

Επεκτείνοντας την ανωτέρω χρησιμότητα του MPS, είναι προφανές ότι το Κύριο Πρόγραμμα Παραγωγής δρα ως ενδιάμεσο στρώμα (Buffer) μεταξύ της πρόγνωσης της ζήτησης των τελικών προϊόντων και του Προγραμματισμού Απαιτήσεων Υλικών (MRP). Με άλλα λόγια, το MPS είναι ουσιαστικά μια ζώνη απορρόφησης των διακυμάνσεων της πρόγνωσης και της αναπόφευκτης διαφοράς της από την πραγματική ζήτηση, όπως φαίνεται παραστατικά στο Σχήμα 25.

Στην περίπτωση της επιχείρησης Α, δεν παρεμβάλλεται τίποτα μεταξύ πρόγνωσης ζήτησης και MRP, με αποτέλεσμα κάθε λάθος εκτίμηση να μεταφέρεται αυτούσια στην παραγωγική αλυσίδα. Και βέβαια, ενώ η πρόγνωση από μόνη της δεν απαιτεί μεγάλο κόστος εφαρμογής, εντούτοις μπορεί να αυξήσει κατακόρυφα το κόστος παραγωγής αν ακολουθηθεί κατά γράμμα και αποδειχθεί λανθασμένη.

Αντίθετα, στην επιχείρηση Β έχουμε σαν αποσβεστήρα το Κύριο Πρόγραμμα Παραγωγής, το οποίο αποσβένει τις διακυμάνσεις της πρόγνωσης και οδηγεί σε ρεαλιστικότερο MRP. Για αυτό ακριβώς το λόγο επιχειρήσεις που ακολουθούν το πρότυπο της Α τείνουν να εκλείψουν.

Οι διακυμάνσεις αυτές, τις περισσότερες φορές προκαλούν πολλαπλάσιες διακυμάνσεις στην παραγωγή, με αποτέλεσμα την πολύ μεγάλη αύξηση κόστους και την αποσταθεροποίηση της όλης παραγωγικής διαδικασίας. Στην πραγματικότητα, τα συστήματα MRP άρχισαν να αποδίδουν πραγματικά μετά τη δημιουργία κατάλληλου λογισμικού κύριου προγραμματισμού παραγωγής και της υιοθέτησης της διαδικασίας του MPS στα μέσα της δεκαετίας του '70.



Σχήμα 25. Το MPS ως Buffer μεταξύ Πρόγνωσης Ζήτησης και MRP.

2.3.1. Κατάστρωση Κύριου Προγράμματος Παραγωγής

Η κατάστρωση του Κύριου Προγράμματος Παραγωγής είναι η διαδικασία κατά την οποία, ο προϊστάμενος παραγωγής καλείται να καθορίσει ποια τελικά προϊόντα και σε τι ποσότητα θα παραχθούν σε καθορισμένες χρονικές περιόδους, οι οποίες αποτελούν τον ορίζοντα προγραμματισμού. Για τη δημιουργία του MPS, οι κύριες πληροφορίες εισόδου που χρησιμοποιούνται είναι οι εξής :

- Οι υπάρχουσες παραγγελίες των πελατών
- Η πρόγνωση της μελλοντικής ζήτησης
- Το Αρχείο Ειδών, το οποίο κρατείται ενήμερο για τα υπόλοιπα των αποθεμάτων (τελικά προϊόντα), τα αποθέματα ασφαλείας, τους χρόνους αναμονής, την πολιτική αναπαραγγελίας και το μέγεθος των παρτίδων παραγωγής

- Οι εντολές παραγωγής των τελικών προϊόντων που βρίσκονται σε εξέλιξη, απ' όπου συνάγονται οι ποσότητες των υλικών που αναμένονται για μελλοντική εισαγωγή στην αποθήκη

Σημαντικό εργαλείο για τον καθορισμό του Κύριου Προγράμματος Παραγωγής, αποτελεί ο Πίνακας του MPS που ακολουθεί (Πίνακας 5). Η κατασκευή του πίνακα αυτού χαρακτηρίζεται από τέσσερα βασικά βήματα. Τα βήματα αυτά είναι τα εξής :

- Υπολογισμός μικτών αναγκών σε υλικά
- Υπολογισμός καθαρών αναγκών σε υλικά
- Καθορισμός παρτίδων/μερίδων παραγωγής
- Υπολογισμός ημερομηνιών έναρξης των εντολών παραγωγής

Πίνακας 5. Πίνακας Υπολογισμού Κύριου Προγράμματος Παραγωγής.

Χρονική Περίοδος			1	2	3	4	5	6	7	8
Μικτές Ανάγκες	Πρόγνωση Ζήτησης		30	30	30	30	35	35	35	35
	Παραγγελίες Πελατών		38	27	24	8	0	0	0	0
Προγραμματισμένες Παραλαβές			0	150	0	0	0	0	**150***	0
Διαθέσιμο Απόθεμα		55	17	137	107	77	42	7	122	87
Καθαρές Ανάγκες									28	
Προτεινόμενες Εντολές Παραγωγής (MPS)			0	0	0	0	0	**150***	0	0
Αρχικό Απόθεμα = 55, Μέγεθος Παρτίδας = 150, Χρόνος Παράδοσης = 1										

Η πρώτη γραμμή στον Πίνακα 5 δείχνει τις χρονικές περιόδους προγραμματισμού, η διάρκεια των οποίων κυμαίνεται από μία μέρα έως τρίμηνο ή και περισσότερο. Η συνηθέστερα χρησιμοποιούμενη περίοδος είναι η εβδομάδα. Ως πρώτη περίοδος θεωρείται πάντα η αμέσως επόμενη από την τρέχουσα, στην οποία και αντιστοιχεί το τρέχον διαθέσιμο απόθεμα. Το πλήθος των περιόδων αποτελεί τον ορίζοντα προγραμματισμού, δηλαδή πόσο μακριά στο μέλλον εκτείνεται ο υπολογισμός των αναγκών σε προϊόντα.

Η δεύτερη γραμμή, οι Μικτές Ανάγκες, αποτελούν τη μελλοντική ζήτηση ενός υλικού. Οι Μικτές Ανάγκες είναι διαχρονικές, δηλαδή αντιστοιχούν μια προς μια στις χρονικές περιόδους και δεν είναι συγκεντρωτικά νούμερα. Αν σε μια χρονική περίοδο εμφανίζονται τόσο πραγματικές παραγγελίες πελατών όσο και απαιτήσεις σε προϊόντα από πρόγνωση της ζήτησης, σαν μικτή ανάγκη για την περίοδο αυτή, ορίζεται (κατά κανόνα) το μέγιστο των δύο ανωτέρω τιμών. Μικτή Ανάγκη σε μία συγκεκριμένη χρονική περίοδο σημαίνει ότι η ζήτηση αυτή θα μείνει ανικανοποίητη εκτός εάν υπάρχουν διαθέσιμα υλικά σ' αυτή την ίδια περίοδο. Η διαθεσιμότητα των υλικών επιτυγχάνεται είτε έχοντας απόθεμα του υλικού στην αποθήκη, είτε έχοντας ανοιχτή εντολή παραγωγής με ημερομηνία παράδοσης μέχρι την ημερομηνία της μικτής ζήτησης.

Η γραμμή των προγραμματισμένων παραλαβών, περιγράφει την τρέχουσα κατάσταση των εντολών παραγωγής που βρίσκονται σε εξέλιξη για το συγκεκριμένο προϊόν. Αναφέρονται οι ποσότητες που έχουν παραγγελθεί και πότε αναμένεται η παραλαβή τους.

Η επόμενη γραμμή αναφέρεται στα αποθέματα και απεικονίζει τη διαχρονική εξέλιξη των αποθεμάτων ως αποτέλεσμα της αφαίρεσης των μικτών αναγκών και της προσθήκης των

προγραμματισμένων παραλαβών. Η λέξη διαθέσιμο έχει ιδιαίτερη έννοια διότι δεν ταυτίζεται με τα πραγματικά υπάρχοντα και δυνάμενα να μετρηθούν αποθέματα μέσα στην αποθήκη. Ο τύπος που δίνει τον υπολογισμό των διαθέσιμων αποθεμάτων, είναι :

Διαθέσιμο Απόθεμα	=	Υπάρχοντα στην Αποθήκη (Stock On Hand)	Για κάθε Χρονική περίοδο
	+	Προγραμματισμένες Παραλαβές (Stock On Order)	
	-	Απόθεμα Ασφαλείας (Safety Stock)	
	-	Δεσμευμένα (Allocated Stock)	

Το απόθεμα ασφαλείας είναι σχεδόν προφανές ότι δεν πρέπει να θεωρείται μέρος του διαθέσιμου αποθέματος, αλλά μια «ξεχασμένη» ποσότητα που θα χρησιμοποιηθεί μόνο σε περίπτωση απρόβλεπτης ή έκτακτης ανάγκης. Τα δεσμευμένα αποθέματα αποτελούν τις ποσότητες αυτές που για κάποιο λόγο (π.χ. για ποιοτικό έλεγχο, σκάρτα υλικά κλπ) δεν μπορούν (ή δεν πρέπει) να χρησιμοποιηθούν για την κάλυψη της μελλοντικής ζήτησης που εκφράζουν οι Μικτές Ανάγκες.

Η γραμμή των καθαρών αναγκών, είναι πλέον εύκολο να υπολογιστεί και αποτελεί για κάθε χρονική περίοδο τη μικτή ζήτηση που δεν είναι δυνατόν να καλυφθεί από τα διαθέσιμα αποθέματα.

Καθαρές Ανάγκες (Net Requirements)	=	Μικτές Ανάγκες (Gross Requirements)	Για κάθε Χρονική περίοδο
	-	Διαθέσιμο Απόθεμα	

Η γραμμή των εντολών παραγωγής είναι το αποτέλεσμα της προς τα πίσω χρονικής μετατόπισης των καθαρών αναγκών, για να ληφθεί υπόψη ο χρόνος αναμονής (Lead Time), δηλαδή ο χρόνος εκτέλεσης των εντολών παραγωγής των προϊόντων. Έτσι υπολογίζονται οι ημερομηνίες ή οι χρονικές περίοδοι στις οποίες πρέπει να εκδοθούν οι εντολές, ώστε τα παραγγελθέντα υλικά να παραληφθούν στις χρονικές περιόδους που προβλέπονται από τις Καθαρές Ανάγκες.

Εκτός από την προς τα πίσω χρονική μετατόπιση, ο υπολογισμός των εντολών παραγωγής, είναι δυνατόν να περιλαμβάνει και σχηματισμό μεγαλύτερων παρτίδων από αυτές που αντιστοιχούν στις Καθαρές Ανάγκες. Αυτό γίνεται για να υπάρξουν τα πλεονεκτήματα των μεγάλων παρτίδων στην παραγωγική διαδικασία (ακριβό/χρονοβόρο setup μηχανημάτων για αλλαγή από το ένα είδος παραγωγής σε ένα άλλο κλπ).

Βάσει της ανωτέρω διαδικασίας, το MPS προτείνει την παραγωγή τελικών προϊόντων για ένα χρονικό ορίζοντα που συχνά υπερβαίνει τον ένα μήνα. Αυτό όμως δε σημαίνει ότι μια επιχείρηση «τρέχει» MPS ανά τόσο μεγάλο χρονικό διάστημα. Ο υπεύθυνος παραγωγής, συνήθως τρέχει το MPS μία ή ακόμα και δύο φορές την εβδομάδα, προκειμένου να αντιμετωπίζονται τυχόν αιφνίδιες αλλαγές.

Είναι προφανές ότι κάθε φορά που «τρέχει» ένα σύστημα MPS, πρέπει να λαμβάνει υπόψη του το προηγούμενο έτσι ώστε να υπάρχει μια συνέχεια. Έτσι, για τις μεν εντολές που έχουν ήδη εκδοθεί (εντολές παραγωγής σε εξέλιξη), το MPS τις αφήνει ως έχουν ενώ για αυτές που δεν έχουν ξεκινήσει, υπάρχει η δυνατότητα είτε να διαγραφούν και να προγραμματιστούν νέες, είτε οι ήδη υπάρχουσες να επαναπρογραμματιστούν (όσον αφορά στην προθεσμία τους) ή να ακυρωθούν. Υπάρχουν βέβαια και άλλες τεχνικές που εξαρτώνται από τις ανάγκες και την πολιτική της επιχείρησης, όπως η δήλωση στο MPS να μην διαγράψει ή τροποποιήσει κάποιες εντολές που αφορούν κάποιες συγκεκριμένες παραγγελίες (π.χ. για ειδικό πελάτη) ακόμα κι αν δεν έχουν ξεκινήσει.

Τελικά ανά πάσα στιγμή σε μια επιχείρηση υπάρχει ένα τρέχον MPS, το οποίο εφαρμόζεται μέχρι οι υπεύθυνοι να «τρέξουν» το επόμενο. Πάντως, συχνά ο υπεύθυνος παραγωγής προβαίνει σε αλλαγές του προγράμματος που προτείνει ο υπολογιστής, ειδικά αν το τελευταίο δεν είναι ρεαλιστικό για την υλοποίηση του πλάνου της παραγωγής (πολλές εντολές παραγωγής, καθυστέρηση παραγγελιών σημαντικών πελατών, ανεπάρκεια δυναμικότητας κλπ), υπεισέρχεται δηλαδή η ανθρώπινη κρίση σε μια υπολογιστική διαδικασία (judgment).

2.4. Προγραμματισμός Απαιτήσεων Υλικών (MRP)

Το επόμενο στάδιο μετά τη διαμόρφωση του Κύριου Προγράμματος Παραγωγής είναι η υλοποίησή του. Απαραίτητη προϋπόθεση, είναι η λειτουργία ενός συστήματος μέσω του οποίου η παραγωγή θα τροφοδοτείται (στον κατάλληλο χρόνο) με όλα εκείνα τα υλικά που απαιτούνται (στις κατάλληλες ποσότητες) για την κατασκευή των τελικών προϊόντων.

Το MRP είναι μια τεχνική για την κατάρτιση του προγράμματος παραγωγής των ημιετοίμων (εξαρτημάτων, συγκροτημάτων και υποσυγκροτημάτων) και του προγράμματος προμηθειών των πρώτων και των βοηθητικών υλών, που απαιτούνται για την υλοποίηση του MPS. Σκοπός του είναι η εξασφάλιση των ανωτέρω εισερχόμενων στην παραγωγή υλικών, στις ποσότητες και στους χρόνους που χρειάζονται ώστε να αποφεύγονται (α) ελλείψεις και διακοπές της παραγωγικής διαδικασίας και (β) υπεραποθεματοποίηση και αύξηση του κόστους. Η μέθοδος εφαρμόζεται σχεδόν αποκλειστικά με τη βοήθεια ηλεκτρονικών υπολογιστών.

Τα ημιέτοιμα υλικά και οι πρώτες ύλες χαρακτηρίζονται από τη λεγόμενη «εξαρτημένη ζήτηση», δηλαδή ζήτηση που εξαρτάται από το αποφασισμένο Κύριο Πρόγραμμα Παραγωγής (MPS), σε αντίθεση με τα τελικά προϊόντα ή όσα ενδιάμεσα συγκροτήματα των προϊόντων διατίθενται στο εμπόριο (π.χ. τα ανταλλακτικά), τα οποία χαρακτηρίζονται από «ανεξάρτητη ζήτηση». Η τεχνική του MRP είναι προσανατολισμένη στην αντιμετώπιση των ιδιαίτερων χαρακτηριστικών της εξαρτημένης ζήτησης, που είναι :

- Δυνατότητα ακριβούς υπολογισμού με βάση τους Πίνακες Υλικών και το Κύριο Πρόγραμμα Παραγωγής των ετοίμων, ανεξάρτητα από τεχνικές πρόγνωσης της ζήτησης και στατιστικές μεθόδους και (θεωρητικά) χωρίς την ανάγκη ύπαρξης αποθεμάτων ασφαλείας
- Ασυνέχεια και ανομοιομορφία μέσα στο χρόνο

Το MRP δεν είναι εκτελεστικό εργαλείο. Προτείνει κάποιες ενέργειες που είτε εκτελούνται, είτε αγνοούνται από τους υπεύθυνους παραγωγής. Όπως το MPS, έτσι και το MRP κατά τους υπολογισμούς, δεν λαμβάνει υπόψη τους περιορισμούς δυναμικότητας των παραγωγικών πόρων.

2.4.1. Οι Πληροφορίες Εισόδου και Εξόδου του MRP

Ως τελικά αποτελέσματα (έξοδοι) του MRP καταρτίζονται το Πρόγραμμα Παραγγελιών με τις ημερομηνίες έναρξης και παράδοσης των παραγγελιών πρώτων υλών και το Πρόγραμμα Παραγωγής των εξαρτημάτων με τις ημερομηνίες έναρξης και τέλους της παραγωγής κάθε εξαρτήματος ή ενδιάμεσου συγκροτήματος (Σχήμα 26).

Σχήμα 26. Πληροφορίες Εισόδου και Εξόδου ενός Συστήματος MRP.

Οι βασικές πληροφορίες εισόδου για τη λειτουργία ενός συστήματος MRP, είναι οι εξής :

- Το Κύριο Πρόγραμμα Παραγωγής (MPS)
- Το Αρχείο Ειδών, το οποίο κρατείται ενήμερο για τα υπόλοιπα των αποθεμάτων (ημιέτοιμα και πρώτες ύλες), τα αποθέματα ασφαλείας και τις δεσμεύσεις των υλικών σε υπό εκτέλεση εντολές παραγωγής, τους χρόνους αναμονής, την πολιτική αναπαραγγελίας και το μέγεθος των παρτίδων παραγωγής/προμήθειας
- Οι Πίνακες Υλικών/Συνταγολόγια με τη σύνθεση και τη δομή των προϊόντων από πρώτες ύλες, εξαρτήματα και συγκροτήματα
- Οι παραγγελίες πρώτων υλών και οι εντολές παραγωγής που βρίσκονται σε εξέλιξη, απ' όπου συνάγονται οι ποσότητες των υλικών που αναμένονται για μελλοντική εισαγωγή στην αποθήκη

Χρησιμοποιώντας τις πληροφορίες αυτές, το σύστημα MRP αναγνωρίζει τις κινήσεις που πρέπει να εκτελεστούν, ώστε να πραγματοποιηθεί το Κύριο Πρόγραμμα Παραγωγής, όπως έχει ήδη καταστρωθεί. Τέτοιες κινήσεις είναι η έκδοση παραγγελιών πρώτων υλών, η έκδοση εντολών παραγωγής ημιετοίμων, ο καθορισμός παρτίδων παραγωγής/παραγγελίας και η επιτάχυνση/επιβράδυνση ή και η ακύρωση ήδη υπαρχόντων εντολών (επαναπρογραμματισμός εντολών).

2.4.2. Η Μέθοδος Επεξεργασίας του MRP

Η λογική με την οποία το MRP επεξεργάζεται τις πληροφορίες εισόδου, είναι παρόμοια με αυτή της κατάστρωσης του Κύριου Προγράμματος Παραγωγής, με το MPS (εντολές παραγωγής των τελικών προϊόντων) να αποτελεί τις μικτές ανάγκες του MRP.

Οι τέσσερις φάσεις υπολογισμών που αναλύθηκαν κατά την κατασκευή του Πίνακα 5, δηλαδή (α) Μικτές Ανάγκες, (β) Καθαρές Ανάγκες, (γ) Μέγεθος παρτίδων και (δ) Ημερομηνίες έκδοσης εντολών, αφορούν πάντοτε ένα επίπεδο κατασκευής/συναρμολόγησης ή προμήθειας. Μετά την κατάστρωση του MPS, οι εντολές παραγωγής ενός τελικού προϊόντος, αποτελούν τις μικτές ανάγκες των υλικών του αμέσως επόμενου επιπέδου στον πίνακα υλικών του προϊόντος.

Οι ίδιοι υπολογισμοί επαναλαμβάνονται για κάθε επίπεδο του δέντρου δομής των προϊόντων, μέχρι και το τελευταίο επίπεδο των πρώτων υλών. Ο τρόπος/κανόνας που γίνεται η μετάβαση από ένα υψηλότερο σε ένα χαμηλότερο επίπεδο, είναι ο εξής :

Εντολές Παραγωγής Προϊόντος/Συγκροτήματος Γονέα	=	Για κάθε Χρονική περίοδο
Μικτές Ανάγκες Εξαρτήματος/Πρώτης Ύλης Παιδιού		

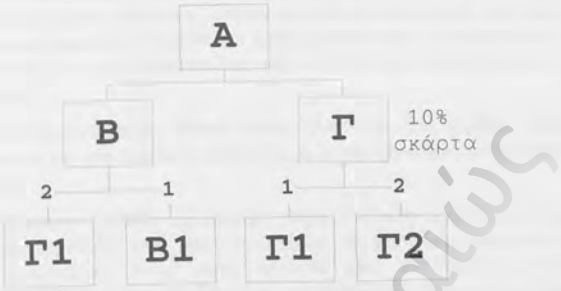
Η διαδικασία αυτή της διαδοχικής ανάλυσης των προγραμματισμένων αναγκών σε υλικά όλων των επιπέδων κατασκευής/συναρμολόγησης βάσει των πινάκων υλικών, είναι γνωστή στη βιβλιογραφία σαν «Εκρηξη Υλικών» (Bill of Materials Explosion).

Προκειμένου να αποφευχθούν πολλαπλοί υπολογισμοί για το ίδιο υλικό, οι υπολογισμοί γίνονται επίπεδο προς επίπεδο, βασιζόμενοι στα στάδια παραγγελίας και όχι στα πραγματικά στάδια συναρμολόγησης των πινάκων υλικών (Σχήματα 23, 24). Έτσι, η επεξεργασία όλων των ειδών γίνεται σε διαδοχικές οριζόντιες διαδρομές και καθυστερώντας τους υπολογισμούς για ένα υλικό, μέχρι το χαμηλότερο σημείο στο οποίο το υλικό αυτό συναντάται στα δέντρα κατασκευής όλων των προϊόντων (Low Level Code).

Μια παρατήρηση που πρέπει να κάνουμε στο σημείο αυτό, είναι η εξής : στην ενότητα που αναλύσαμε τον τρόπο υπολογισμού του διαθέσιμου αποθέματος, αναφερθήκαμε στη σημασία της δεσμευμένης ποσότητας. Πέραν των ήδη αναφερθέντων, στην περίπτωση των ημιετοίμων ή των πρώτων υλών, το δεσμευμένο απόθεμα περιλαμβάνει και τις ποσότητες που προορίζονται για ήδη εκδοθείσες ή και υπό εκτέλεση εντολές παραγωγής του αμέσως ανώτερου επιπέδου.

2.4.3. Παράδειγμα Υπολογισμών του MRP

Έστω το προϊόν Α που αποτελείται από τα συστατικά Β και Γ, εκ των οποίων το Γ αποτελείται από τα $Γ_1$ και $Γ_2$ σε αναλογία 1 και 2 , ενώ το Β αποτελείται από τα $Γ_1$ και $Β_1$ σε αναλογία 2 και 1 (Σχήμα 27). Επιπλέον κατά την παραγωγή του Γ προκύπτει συνήθως ποσότητα 10% σκάρτων. Το Κύριο Πρόγραμμα Παραγωγής του προϊόντος Α και οι υπολογισμοί του MRP για το είδος Γ, δίνονται στον Πίνακα 6.



Σχήμα 27. Πίνακας Υλικών Προϊόντος Α.

Πίνακας 6. Κύριο Πρόγραμμα Παραγωγής του Προϊόντος Α, Υπολογισμοί του MRP για το Είδος Γ.

Είδος: Α	Απρίλιος				Μάιος			
Περίοδοι	1	2	3	4	5	6	7	8
MPS		20	15		5	10		25
Μικτές ανάγκες σε Γ λόγω 10% σκάρτων	0	23	17	0	6	12		28

(Α)

Μερίδα παραγωγής :10 τεμ.

Αρχικό απόθεμα : 20

Χρόνος παράδοσης : 1 εβδομάδα

Είδος: Γ	Απρίλιος				Μάιος			
Περίοδοι	1	2	3	4	5	6	7	8
Μικτές ανάγκες		23	17		6	12		28
Διαθέσιμο απόθεμα	20							
Προγραμματισμένες παραλαβές								
Καθαρές ανάγκες		3	17		6	12		28

(Β)

Καθαρές ανάγκες		3	17		6	12		28
Μερίδες		10	10		10	10		3*10
Εντολές παραγωγής	10	10	10	10	10	10	10	

(Γ)

Είδος: Γ	Απρίλιος				Μάιος			
Περίοδοι	1	2	3	4	5	6	7	8
Μικτές ανάγκες		23	17		6	12		28
Διαθέσιμο απόθεμα	20	7	0	10	14	12	22	4
Προγραμματισμένες παραλαβές		10	10	10	10	10	10	10
Εντολές παραγωγής	10	10	10	10	10	10	10	

2.4.4. Πολιτικές Καθορισμού Παρτίδων

Ένα από τα βασικά βήματα κατά τη διάρκεια των υπολογισμών του MRP, είναι ο καθορισμός των παρτίδων παραγγελίας των πρώτων υλών και των παρτίδων παραγωγής των εξαρτημάτων. Υπάρχουν πολλές εναλλακτικές τεχνικές καθορισμού των παρτίδων (Order Policy), από πολύπλοκα μαθηματικά μοντέλα που βασίζονται στην εξισορρόπηση του κόστους Setup (ή του σταθερού κόστους παραγγελίας) και του κόστους διατήρησης αποθέματος, μέχρι πρακτικές μέθοδοι όπως η παρτίδα σταθερής ποσότητας.

Οι τελευταίες, δεν τεκμηριώνονται αναλυτικά και δεν δίνουν βέλτιστες λύσεις, ενώ οδηγούν σε διαφορετικά συνήθως αποτελέσματα. Εντούτοις, δίνουν αρκετά ικανοποιητική απάντηση στο πρόβλημα του Προγραμματισμού Απαιτήσεων των Υλικών και χρησιμοποιούνται ευρύτατα στην πράξη λόγω της απλότητας που τις διακρίνει. Οι πιο συνηθισμένες και ευρέως χρησιμοποιούμενες απλές πολιτικές καθορισμού παρτίδων, είναι :

- Παρτίδα Σταθερής Ποσότητας (Fixed Order Quantity). Το μέγεθος κάθε προτεινόμενης παρτίδας, ισούται με μια σταθερή ποσότητα, η οποία συνήθως εκφράζει έναν περιορισμό δυναμικότητας.
- Σταθερή Οικονομική Παρτίδα (Economic Order Quantity). Αποτελεί υποπερίπτωση της Παρτίδας Σταθερής Ποσότητας, όπου το μέγεθος της παρτίδας ισούται με την Οικονομική Ποσότητα Παραγγελίας, όπως αυτή ορίζεται από το βασικό μοντέλο διαχείρισης αποθεμάτων. Βέβαια, η χρήση της Οικονομικής Ποσότητας Παραγγελίας δεν τεκμηριώνεται θεωρητικά, καθότι για τον υπολογισμό της γίνονται αρκετές παραδοχές όπως σταθερή μέση ζήτηση, τυχαία και συνεχής, οι οποίες προφανώς δεν ισχύουν όταν αναφερόμαστε σε ενδιάμεσα υλικά με εξαρτημένη ζήτηση.
- Παρτίδα προς Παρτίδα (Lot for Lot). Στην περίπτωση αυτή, για κάθε χρονική περίοδο, το σύστημα προτείνει μια εντολή, το μέγεθος της οποίας είναι ίσο με τις καθαρές ανάγκες της περιόδου αυτής.
- Παρτίδα Σταθερού Αριθμού Χρονικών Περιόδων (Periodic Order Quantity). Το ύψος των προτεινόμενων εντολών προμήθειας/παραγωγής, υπολογίζεται έτσι ώστε να καλύπτει το άθροισμα των καθαρών αναγκών ενός προκαθορισμένου αριθμού συνεχών χρονικών περιόδων.

Οι ανωτέρω απλές μέθοδοι καθορισμού παρτίδων, εφαρμόζονται συνήθως και στους υπολογισμούς που εκτελούνται κατά τη κατάστρωση του Κύριου Προγράμματος Παραγωγής. Στην περίπτωση του MPS πάντως, ο πιο συνηθισμένος τρόπος για τον καθορισμό των παρτίδων των τελικών προϊόντων, είναι ο συνδυασμός του επιπέδου ή σημείου αναπαραγγελίας (ROL–Reorder Level ή ROP–Reorder Point) και της ποσότητας αναπαραγγελίας (ROQ–Reorder Quantity).

Ουσιαστικά πρόκειται για τη μέθοδο της «συνεχούς παρακολούθησης αποθέματος» της κλασικής θεωρίας διαχείρισης αποθεμάτων. Σύμφωνα με τη μέθοδο αυτή, όταν η στάθμη του αποθέματος πέσει κάτω από ένα προκαθορισμένο επίπεδο (ROL ή ROP), τοποθετείται παραγγελία σταθερής ποσότητας (ROQ) για την αναπλήρωση του αποθέματος. Η ποσότητα αυτή κατά κανόνα τίθεται ίση με την Οικονομική Ποσότητα Παραγγελίας. Η διαφορά με την κλασική μέθοδο, έγκειται στο γεγονός, ότι κατά τους υπολογισμούς του MPS, το διαθέσιμο απόθεμα συγκρίνεται σε κάθε χρονική περίοδο του ορίζοντα προγραμματισμού με το σημείο αναπαραγγελίας, και εφόσον κρίνεται απαραίτητο, το σύστημα προτείνει μια εντολή παραγωγής (Time-Phased Reorder Point).

2.4.5. Αδυναμίες ενός Συστήματος MRP

Το MRP αποτελεί ένα ιδιαίτερα ισχυρό και ευέλικτο εργαλείο στη διαχείριση της παραγωγής. Από την αρχή λειτουργίας του, είναι φανερό ότι επιχειρήσεις που παράγουν είδη με μεγάλους και πολύπλοκους πίνακες υλικών ή η ζήτηση των προϊόντων τους παρουσιάζει έντονη εποχικότητα και μεγάλες διακυμάνσεις, μπορούν να ωφεληθούν πάρα πολύ με τη χρήση ενός συστήματος MRP. Επιτυχείς εφαρμογές του συστήματος MRP έχουν λάβει χώρα σε αναρίθμητες επιχειρήσεις τα τελευταία 25-30 χρόνια. Παρόλα αυτά όμως, το σύστημα MRP έχει κάποιες σημαντικές αδυναμίες, όπως :

- Για τον προσδιορισμό των χρόνων κατά τους οποίους θα εκδοθούν οι εντολές παραγωγής, το MRP λαμβάνει υπόψη τους χρόνους αναμονής (Lead Times), όπως αυτοί ορίζονται από το χρήστη. Τους χρόνους αυτούς το σύστημα τους θεωρεί ντετερμινιστικούς. Αυτό όμως έχει ως αποτέλεσμα, ότι δεν λαμβάνεται υπόψη το γεγονός πως η παραγωγή μίας μεγάλης παρτίδας, θα διαρκέσει περισσότερο από τον προβλεπόμενο χρόνο παραγωγής. Ανάλογα, αν το μέγεθος της παρτίδας είναι σημαντικά μικρότερο από τη μέση ποσότητα, τότε η παραγωγή της θα διαρκέσει λιγότερο από τον προβλεπόμενο χρόνο. Συνήθως, οι επιχειρήσεις αντιμετωπίζουν το μειονέκτημα αυτό του MRP, θέτοντας μεγαλύτερους χρόνους παραγωγής (ή προμήθειας) για λόγους ασφάλειας. Έτσι, παρατηρείται συχνά το φαινόμενο οι εργασίες να ολοκληρώνονται νωρίτερα από ότι αναμένεται, και επομένως τήρηση αποθέματος για μεγαλύτερο χρόνο από ότι χρειάζεται, με όλα τα κόστη που αυτό συνεπάγεται.

- Τα συστήματα MRP τις περισσότερες φορές δεν παρέχουν την δυνατότητα αυτόματου υπολογισμού των αποθεμάτων ασφαλείας (Safety Stock). Αντίθετα, απαιτούν από το χρήστη την εισαγωγή μίας τιμής, χωρίς να προτείνουν ή να βοηθούν στον τρόπο υπολογισμού της. Πολλές φορές αυτό οδηγεί στην τήρηση υψηλότερων αποθεμάτων ασφαλείας από αυτά που είναι απαραίτητα, καθώς λόγω ανεπαρκούς γνώσης επί του θέματος, οι χρήστες αναγκάζονται να «μαντεύουν» πόση ποσότητα θα είναι αρκετή ώστε να μην υπάρξει περίπτωση ελλείψεων. Στο σημείο αυτό πρέπει να αναφέρουμε ότι η θεωρία του MRP, υποστηρίζει ότι δεν υφίσταται η ανάγκη ύπαρξης αποθεμάτων ασφαλείας για τα ενδιάμεσα υλικά, καθότι η τεχνική του MRP στοχεύει στη εξασφάλιση των ανωτέρω εισερχόμενων στην παραγωγή υλικών, στις ποσότητες και στους χρόνους που χρειάζονται. Στην πράξη όμως, διάφοροι λόγοι τυχαιότητας (αιφνίδιες παραγγελίες, βλάβες μηχανών, παραγωγή σκάρτων, εκπρόθεσμες παραδόσεις υλικών κλπ) καθιστούν αναγκαία τα αποθέματα ασφαλείας, για την αντιμετώπιση των λόγων αυτών.

- Τα συστήματα MRP, πέρα από συνήθεις πολιτικές καθορισμού παρτίδων που αναφέρθηκαν στην προηγούμενη ενότητα, συνήθως δεν ενσωματώνουν μεθόδους οι οποίες οδηγούν σε βέλτιστες λύσεις υπολογισμού των παρτίδων παραγωγής. Ακόμη όμως και στις περιπτώσεις που υπάρχει η δυνατότητα επιλογής σύνθετων ευρετικών μεθόδων (heuristics) ή μεθόδων βελτιστοποίησης, έχει παρατηρηθεί ότι οι επιχειρήσεις δεν τις υιοθετούν, λόγω της αδυναμίας κατανόησης αυτών και της πολυπλοκότητας που παρουσιάζουν στην εφαρμογή τους.

- Τα συστήματα MRP είναι ιδιαίτερα ευαίσθητα σε μικρά λάθη και ανακρίβειες που μπορεί να εμπεριέχονται στις πληροφορίες εισόδου τους. Συγκεκριμένα αναφέρεται ενδεικτικά στη βιβλιογραφία, ότι το ποσοστό ακρίβειας που συνήθως ζητείται από ένα τέτοιο σύστημα για τα αρχεία του αποθέματος είναι 95% ενώ για τους πίνακες υλικών 99%. Στην περίπτωση που τα σφάλματα σε αυτές τις πηγές πληροφοριών είναι μεγαλύτερα από τα προβλεπόμενα, το σύστημα θα δίνει αποτελέσματα, τα οποία δεν θα ανταποκρίνονται στην πραγματικότητα.

2.4.6. Πότε Εφαρμόζεται η Μέθοδος MRP

Το MRP εφαρμόζεται μόνο μέσα στο εργοστάσιο και σε άμεση σχέση με τον προγραμματισμό της παραγωγής. Δεν έχει καμία σχέση με τη διαχείριση των αποθεμάτων των τελικών προϊόντων, το κύκλωμα διανομής ή τα αποθέματα ανταλλακτικών για service.

Το MRP εφαρμόζεται σχεδόν αποκλειστικά με τη χρήση ηλεκτρονικών υπολογιστών, γιατί απαιτεί συνεχή ενημέρωση και ικανότητα αναπροσαρμογής ποσοτήτων και ημερομηνιών, πολλές φορές για εκατοντάδες ή χιλιάδες εξαρτήματα και πρώτες ύλες. Η ανάγκη για τέτοιες αναπροσαρμογές προκύπτει από αλλαγές στο Κύριο Πρόγραμμα Παραγωγής (π.χ. λάθη στην πρόγνωση της ζήτησης) ή στις προγραμματισμένες παραλαβές (απρόβλεπτα μεγάλοι χρόνοι παράδοσης πρώτων υλών ή παραγωγής εξαρτημάτων).

Το MRP δεν είναι κατάλληλο για βιομηχανίες με ροϊκή/συνεχή παραγωγή. Αντίθετα έχει μεγάλες δυνατότητες για οικονομικά οφέλη στην περίπτωση παραγωγής διακριτών προϊόντων με βαθμίδες κατασκευής/συναρμολόγησης.

Το MRP προϋποθέτει την ύπαρξη τυποποιημένων πινάκων υλικών, δηλαδή τα προϊόντα (τουλάχιστον τα κυριότερα συναρμολογήματα αυτών) είναι τυποποιημένα και παρέχουν στους πελάτες τη δυνατότητα μόνο παραλλαγών ή μικρών μετατροπών.

Εάν έχει αποφασιστεί η εφαρμογή του MRP για το σύνολο των πρώτων υλών και των εξαρτημάτων μέσα στο εργοστάσιο, πρέπει να εξεταστεί αν μερικές κατηγορίες υλικών συμφέρει να εξαιρεθούν και να διαχειρίζονται με διαφορετικό τρόπο. Τέτοιες κατηγορίες είναι :

- Υλικά με πολύ μικρό κόστος (μικροϋλικά)
- Υλικά με χρόνο παράδοσης ή διάρκεια παραγωγής εξαιρετικά μεγάλο σε σχέση με τα περισσότερα υπόλοιπα υλικά/εξαρτήματα, όταν αυτά είναι λίγα.

2.5. Καθορισμός Ύψους Αποθέματος Ασφαλείας

Τα αποθέματα ασφαλείας εξασφαλίζουν ότι η ζήτηση θα ικανοποιείται έγκαιρα (όταν αφορούν έτοιμα προϊόντα) ή ότι δεν θα διακοπεί η παραγωγική διαδικασία (όταν αφορούν πρώτες ύλες ή ενδιάμεσα υλικά), αν η ζήτηση/ανάλωση του αποθέματος υπερβεί το μέσο ρυθμό που αναμένεται κατά το χρόνο αναμονής μιας παραγγελίας για την αναπλήρωση του αποθέματος. Η ύπαρξη αποθεμάτων ασφαλείας συνεπάγεται ένα αντίστοιχο κόστος αποθεματοποίησης, που οφείλεται στη δέσμευση κεφαλαίων, τη δαπάνη αποθήκευσης και τον κίνδυνο φθοράς ή απαξίωσης των αποθεμάτων, αλλά και ένα όφελος από την εξουδετέρωση του κινδύνου μιας κατάστασης έλλειψης αποθέματος.

Ο καθορισμός του ύψους του αποθέματος ασφαλείας είναι δυνατόν να βασιστεί στην εξισορρόπηση των δύο ανωτέρω αντίρροπων συντελεστών κόστους - οφέλους. Επειδή όμως το όφελος που προκύπτει από την πρόληψη της εμφάνισης έλλειψης αποθέματος είναι γενικά δύσκολο να ποσοτικοποιηθεί, στην πράξη συνήθως ο καθορισμός του ύψους των αποθεμάτων ασφαλείας, βασίζεται στην έννοια του βαθμού ή επιπέδου εξυπηρέτησης του πελάτη («εσωτερικού» ή «εξωτερικού»). Σύμφωνα με την προσέγγιση αυτή, καθορίζεται ένα επιθυμητό επίπεδο εξυπηρέτησης με βάση μια κατανομή πιθανότητας για τη δημιουργία ζήτησης κατά τη διάρκεια του χρόνου αναμονής.

Για παράδειγμα, θα εξετάσουμε τη μέθοδο της «συνεχούς παρακολούθησης αποθέματος» ή σταθερής ποσότητας παραγγελίας της κλασικής θεωρίας διαχείρισης αποθεμάτων, θεωρώντας μεταβλητή ζήτηση η οποία ακολουθεί μια από τις γνωστές κατανομές πιθανότητας, ενώ ο χρόνος αναμονής είναι δεδομένος και σταθερός. Στην περίπτωση αυτή, το σημείο αναπαραγγελίας και το απόθεμα ασφαλείας, δίνονται από τους τύπους :

$$ROP = d \times LT + z \times (s_d') \qquad SS = z \times (s_d') = z \times (s_d \times \sqrt{LT})$$

d = μέση ημερήσια ζήτηση

s_d = τυπική απόκλιση της ημερήσιας ζήτησης

s_d' = τυπική απόκλιση της ζήτησης στη διάρκεια του χρόνου αναμονής

LT = χρόνος αναμονής (ημέρες)

Ο αριθμός z είναι ένας συντελεστής ασφαλείας (προκύπτει από στατιστικούς πίνακες για τον επιθυμητό βαθμό εξυπηρέτησης) : όσο μεγαλύτερος, τόσο μεγαλύτερη είναι η βεβαιότητα ότι η ζήτηση θα ικανοποιηθεί, αν αυτή ξεπεράσει τη μέση. Αν για παράδειγμα, η ζήτηση ακολουθεί την κανονική κατανομή (κάτι που συμβαίνει συχνά στην πράξη), τότε z = 1 σημαίνει ότι η ζήτηση θα καλυφθεί στο 84,13% των περιπτώσεων, ενώ το ποσοστό αυτό για z = 2 γίνεται 97,72% και για z = 3 γίνεται 99,87%.

Από τα παραπάνω προκύπτει ότι, αν είναι γνωστή η κατανομή της πιθανότητας της ζήτησης στη διάρκεια του (σταθερού) χρόνου αναμονής, τότε για τον προσδιορισμό του σημείου αναπαραγγελίας απαιτείται μόνο να καθοριστεί η στάθμη εξυπηρέτησης της ζήτησης ή το αντίστοιχο επίπεδο ασφάλειας που επιδιώκεται. Ο καθορισμός του επιπέδου αυτού είναι θέμα πολιτικής της επιχείρησης.

Μέχρι τώρα, έγινε η παραδοχή ότι ενώ η ζήτηση μεταβάλλεται, ο χρόνος αναμονής παραμένει σταθερός. Όταν και οι δύο παράμετροι είναι μεταβλητές, τότε πρέπει να ληφθεί υπόψη η μέση τιμή και η τυπική απόκλιση και του χρόνου αναμονής. Θεωρώντας ότι οι δύο παράμετροι ακολουθούν την κανονική κατανομή και ότι είναι ανεξάρτητες μεταξύ τους, αν η τυπική απόκλιση του χρόνου αναμονής είναι s_{LT}, έχουμε :

$$ROP = d \times LT + z \times (s_d') \qquad SS = z \times (s_d') = z \times (\sqrt{LT \times s_d^2 + d^2 \times s_{LT}^2})$$

Πρέπει να σημειωθεί ότι για τους ανωτέρω υπολογισμούς, ορίζεται ενιαίος βαθμός εξυπηρέτησης και για τις δύο παραμέτρους. Επίσης, στην περίπτωση κατά την οποία οι δύο παράμετροι ακολουθούν διαφορετικές κατανομές πιθανοτήτων, μπορούν να χρησιμοποιηθούν ειδικοί πίνακες συνδυασμένων κατανομών.

Η ύπαρξη των αποθεμάτων ασφαλείας θεωρείται ως επί το πλείστον κατάλληλη για την αντιμετώπιση της διακύμανσης της ζήτησης των υλικών που χαρακτηρίζονται από ανεξάρτητη ζήτηση. Στην περίπτωση των υλικών που παρουσιάζουν εξαρτημένη ζήτηση και διαχειρίζονται με τη τεχνική του MRP, η αποφυγή ελλείψεων αποθέματος και υπεραποθεματοποίησης των ειδών, αντιμετωπίζονται συνήθως με κατάλληλη προσαρμογή των χρόνων αναμονής, επίσπευση εντολών και γενικότερα με συχνές και έγκαιρες παρεμβάσεις στην προτεραιότητα εκτέλεσης τόσο των εντολών παραγωγής των ημιετοίμων υλικών, όσο και των παραγγελιών των προμηθευόμενων ειδών.

Συμπερασματικά, θα λέγαμε ότι η πιο συχνή προσέγγιση για τη μείωση των φαινομένων που απορρέουν από την αβεβαιότητα της ζήτησης/ανάλωσης των ειδών, είναι η χρήση αποθεμάτων ασφαλείας για τα τελικά προϊόντα. Πολύ συχνά πάντως στην καθημερινή πρακτική των επιχειρήσεων, θεωρείται σκόπιμη η διατήρηση αποθεμάτων ασφαλείας και για τα ενδιάμεσα και τα αγοραζόμενα είδη, ώστε να αποφευχθούν συνεχείς παρεμβάσεις στην παραγωγική διαδικασία. Και στις δύο περιπτώσεις, ο καθορισμός του ύψους των αποθεμάτων ασφαλείας, διαμορφώνεται με βάση τη λογική που αναπτύχθηκε στις προηγούμενες παραγράφους.

2.6. Έλεγχος Δυναμικότητας

Το MRP είναι μια τεχνική για τον προγραμματισμό των υλικών, αλλά δεν ελέγχει αν το MPS είναι υλοποιήσιμο. Είναι απλώς μια διαδικασία που υπολογίζει τις απαιτήσεις σε υλικά ενός εξωτερικά δοσμένου σχεδίου. Κατά τους υπολογισμούς δεν λαμβάνει υπόψη τους περιορισμούς δυναμικότητας των πόρων της παραγωγής, θεωρώντας ότι η επιχείρηση διαθέτει άπειρη δυναμικότητα. Το κενό αυτό ήρθαν να το καλύψουν τα συστήματα MRP II, ενσωματώνοντας στους υπολογισμούς και τον προγραμματισμό των απαιτήσεων σε δυναμικότητα, ο οποίος διενεργείται σε δύο στάδια.

Α. Χονδρικός Προγραμματισμός Δυναμικότητας (RCCP)

Αρχικά, πριν το τρέξιμο του MRP, γίνεται ένας χονδρικός έλεγχος της δυναμικότητας (RCCP – Rough Cut Capacity Planning), ο οποίος εξετάζει τη ρεαλιστικότητα του MPS. Ο χονδρικός προγραμματισμός δυναμικότητας εφαρμόζεται πάντοτε τουλάχιστον σε εκείνους τους πόρους που είτε είναι ιδιαίτερα κρίσιμοι, είτε παρουσιάζουν συνήθως «λαιμό» δυναμικότητας (Bottlenecks), χρησιμοποιώντας συγκεντρωτικά στοιχεία (ομαδοποίηση προϊόντων σε οικογένειες, ομαδοποίηση παραγωγικών τμημάτων, αναφορά σε ολόκληρο το προϊόν και όχι σε επιμέρους εξαρτήματα κλπ).

Για την εκτέλεση του χονδρικού προγραμματισμού δυναμικότητας, συνήθως χρησιμοποιούνται τα Προφίλ Δυναμικότητας. Στα Προφίλ Δυναμικότητας καταγράφονται οι απαιτούμενοι πόροι και ο βαθμός χρησιμοποίησής τους για την παραγωγή ενός προϊόντος. Για τη δημιουργία των Προφίλ Δυναμικότητας, απαιτείται συνεργασία μεταξύ του μηχανικού παραγωγής και των τμημάτων μελέτης, σχεδιασμού και κατασκευής προϊόντων. Όσον αφορά στο βαθμό χρησιμοποίησης ενός πόρου, η συνηθέστερη μονάδα μέτρησης είναι η ώρα, ενώ ο υπολογισμός του χρόνου παραγωγής μιας μονάδας προϊόντος, γίνεται ως εξής :

1. Επιλέγουμε οικογένεια προϊόντων
2. Αναλύουμε την οικογένεια προϊόντων
3. Καθορισμός χρόνου. Διαλέγουμε ένα αντιπροσωπευτικό είδος από κάθε οικογένεια και υπολογίζουμε το μέσο χρόνο παραγωγής μιας μονάδας του προϊόντος, με βάση το χρόνο εκτέλεσης μιας τυπικής παρτίδας παραγωγής. Κατόπιν, σταθμίζουμε τον ανωτέρω χρόνο πολλαπλασιάζοντάς τον με ένα συντελεστή βάρους (συνήθως είναι διαφορετικός για κάθε προϊόν της οικογένειας). Ο τύπος υπολογισμού του χρόνου Προφίλ Δυναμικότητας είναι:

   [(Χρόνος Επεξεργασίας) + (Χρόνος Setup)] / Μέγεθος Παραγωγής * Συντελεστής Βάρους

Εναλλακτικά, η εκτέλεση του χονδρικού προγραμματισμού δυναμικότητας, μπορεί να γίνει με τη χρήση των Πινάκων Πόρων (Bill of Resources). Οι Πίνακες Πόρων είναι παρόμοιοι με τους Πίνακες Υλικών, με τη διαφορά ότι αποτυπώνουν την ποσότητα χρησιμοποίησης/ανάλωσης των κρίσιμων πόρων, οι οποίοι απαιτούνται για την παραγωγή μιας μονάδας του προϊόντος, και κατά κανόνα αναφέρονται σε τελικά προϊόντα. Με τους Πίνακες Πόρων καθορίζονται οι ανάγκες σε εργαζόμενους και μηχανές, αλλά πολλές φορές και συγκεντρωτικά προγράμματα αποθεμάτων.

Ο χονδρικός προγραμματισμός δυναμικότητας είναι ακατάλληλος για βραχυπρόθεσμη χρήση, αλλά ιδανικός για παραγγελίες αρκετά μπροστά στο χρονικό ορίζοντα καθώς δε λαμβάνει υπόψη του τις διεργασίες που συμβαίνουν εκείνη ακριβώς τη στιγμή που εκτελείται, ή διεργασίες που έχουν ήδη ολοκληρωθεί. Επίσης, ένα άλλο μειονέκτημα του RCCP, είναι ότι δε λαμβάνει υπόψη του όλους τους απαιτούμενους πόρους για την κατασκευή ενός προϊόντος, αλλά τους πιο βασικούς.

Το RCCP, επιχειρεί και συνήθως καταφέρνει να εντοπίσει κατά μέσο όρο το 80% (σύμφωνα με το νόμο του Pareto) των προβλημάτων δυναμικότητας που μπορεί να εμφανιστούν στην παραγωγική διαδικασία, πριν καν ακόμα αναπτυχθούν τα αναλυτικά προγράμματα παραγωγής υλικών (MRP) και δυναμικότητας (CRP), τα οποία προλαμβάνουν συνήθως το υπόλοιπο 20% των προβλημάτων.

Β. Προγραμματισμός Απαιτήσεων Δυναμικότητας (CRP)

Αν δεν προκύψει κάποια υπέρβαση της διαθέσιμης δυναμικότητας των πόρων, ακολουθεί η εκτέλεση του MRP και στη συνέχεια ο αναλυτικός Προγραμματισμός Απαιτήσεων Δυναμικότητας ή Μεσοπρόθεσμος Προγραμματισμός Δυναμικότητας (CRP – Capacity Requirements Planning). Το CRP μεταφράζει το Πρόγραμμα Απαιτήσεων Υλικών (όσον αφορά στα είδη που κατασκευάζονται μέσα στην ίδια τη βιομηχανία) σε απαιτήσεις δυναμικότητας για την υλοποίησή του, χρησιμοποιώντας αναλυτικά στοιχεία από το υποσύστημα των Τεχνικών Προδιαγραφών (φασεολόγια και μέσα παραγωγής).

Η διαδικασία του CRP γίνεται σε τρία στάδια :

- Υπολογισμός Χρόνων Διέλευσης στην Παραγωγή (Throughput Times)

Στο στάδιο αυτό υπολογίζονται οι χρονικές στιγμές έναρξης και λήξης κάθε φάσης κατεργασίας, που προβλέπεται στη διαδρομή παραγωγής (φασεολόγιο) κάθε εντολής παραγωγής.

Ο συνολικός χρόνος διέλευσης μιας εντολής παραγωγής συντίθεται από τους επί μέρους χρόνους διέλευσης των φάσεων κατεργασίας με βάση τους πρότυπους χρόνους του φασεολογίου και τους χρόνους αναμονής και μεταφοράς (transit times). Οι τελευταίοι αυτοί χρόνοι είναι ιστορικοί μέσοι όροι, οι οποίοι συνήθως αποθηκεύονται σε πίνακες της μορφής «Από – Σε» (από κέντρο εργασίας σε κέντρο εργασίας).

Οι χρόνοι διέλευσης χρησιμοποιούνται για τον χρονικό προγραμματισμό των εντολών είτε «προς τα εμπρός» (forward scheduling) από την ημερομηνία δυνατής έναρξης της εντολής, είτε «προς τα πίσω» (backward scheduling) από την προθεσμία παράδοσης. Με τον τρόπο αυτό υπολογίζονται ημερομηνίες έναρξης και λήξης των φάσεων κατεργασίας «το ενωρίτερο» και «το αργότερο». Η χρονική τους απόσταση αποτελεί το περιθώριο ασφαλείας (buffer) που χρησιμοποιείται για την εξισορρόπηση του φόρτου εργασίας των μηχανών. Η παραβίαση του περιθωρίου ασφαλείας, σημαίνει αυτόματα και παραβίαση της τελικής προθεσμίας παράδοσης ολόκληρης της εντολής παραγωγής.

- Φόρτιση Κέντρων Εργασίας

Το επόμενο βήμα του Μεσοπρόθεσμου Προγραμματισμού Δυναμικότητας, είναι ο υπολογισμός του φόρτου εργασίας που προκύπτει από τις προγραμματισμένες εντολές παραγωγής και τις απαιτήσεις

σε δυναμικότητα που αυτές συνεπάγονται, για κάθε μηχανή και χρονική περίοδο του ορίζοντα προγραμματισμού. Πολλαπλασιάζοντας την ποσότητα της εντολής παραγωγής με τους πρότυπους χρόνους ανά τεμάχιο και προσθέτοντας τους χρόνους προετοιμασίας (setup), υπολογίζονται οι χρόνοι φόρτισης των μηχανών. Οι χρόνοι φόρτισης επιρρίπτονται στις χρονικές περιόδους έναρξης των φάσεων κατεργασίας, όπως έχουν προκύψει από το προηγούμενο στάδιο του υπολογισμού των χρόνων διέλευσης. Έτσι, καταστρώνονται γραφικά διαγράμματα φόρτισης ανά μηχανή (ή ομάδα μηχανών).

- Εξισορρόπηση Δυναμικότητας

Στην περίπτωση κατά την οποία, η απαιτούμενη δυναμικότητα για την υλοποίηση των εντολών υπερβαίνει τη διαθέσιμη σε μια χρονική περίοδο, είναι προφανές ότι πρέπει να υπάρξουν τροποποιήσεις. Οι κύριες ενέργειες που λαμβάνουν χώρα για την εξισορρόπηση του φόρτου, είναι : (α) Απαλλαγή του κέντρου εργασίας από την εντολή παραγωγής που προκαλεί υπερφόρτιση, (β) Χρονική μετατόπιση μιας φάσης κατεργασίας στην ίδια μηχανή (γ) Χρησιμοποίηση εναλλακτικής μηχανής και (δ) Συνδυασμός των ενεργειών (β) και (γ). Η χρονική μετατόπιση θα πρέπει να είναι εφικτή, δηλαδή να μην επηρεάζει τις άλλες φάσεις κατεργασίας. Αυτό μπορεί να συμβεί αν βρίσκεται μέσα στα όρια του χρονικού περιθωρίου ασφαλείας μεταξύ της «ενωρίτερης» και της «αργότερης» ημερομηνίας έναρξης της φάσης κατεργασίας.

Η εξισορρόπηση της δυναμικότητας, παρά την απλή της διατύπωση, αποτελεί ένα από τα δυσκολότερα προβλήματα της Οργάνωσης Παραγωγής, λόγω του εξαιρετικά μεγάλου πλήθους των εναλλακτικών λύσεων. Για την επίλυσή του, συνήθως χρησιμοποιούνται δύο προσεγγίσεις.

Το πρόβλημα αντιμετωπίζεται με διάλογο ανθρώπου – υπολογιστή (Interactive). Η επιλογή δοκιμαστικών λύσεων γίνεται από το χειριστή του προγράμματος CRP, ενώ ο Η/Υ διενεργεί με ταχύτητα τους απαραίτητους υπολογισμούς σε κατάσταση προσομοίωσης (what if simulation), για την παρουσίαση και την αξιολόγηση των αποτελεσμάτων.

Το πρόβλημα μπορεί επίσης να επιλυθεί αλγοριθμικά με μεθόδους Μαθηματικού Προγραμματισμού ή/και άλλες τεχνικές της Επιχειρησιακής Έρευνας. Τα συνηθέστερα χρησιμοποιούμενα κριτήρια βελτιστοποίησης είναι : (α) Η ελαχιστοποίηση του αθροίσματος των απόλυτων διαφορών μεταξύ πραγματικής και επιθυμητής φόρτισης για όλες τις μηχανές και όλες τις χρονικές περιόδους και (β) Η ελαχιστοποίηση του αθροίσματος των αποκλίσεων της συσσωρευτικής πραγματικής φόρτισης από την επιθυμητή.

Η εφικτότητα του προγράμματος παραγωγής, το οποίο ταυτόχρονα δεν θα παραβιάζει τις προθεσμίες παράδοσης των εντολών, μπορεί τέλος να διευθετηθεί με εξωτερικές παρεμβάσεις. Τέτοιου είδους ενέργειες περιλαμβάνουν κατάλληλες ρυθμίσεις στο MPS, υπερωριακή απασχόληση, ανάθεση υπεργολαβιών, χρησιμοποίηση εφεδρικής δυναμικότητας, παράλληλη κατεργασία μικρότερων παρτίδων (splitting), ακύρωση ή αλλαγή ποσοτήτων εντολών παραγωγής κλπ.

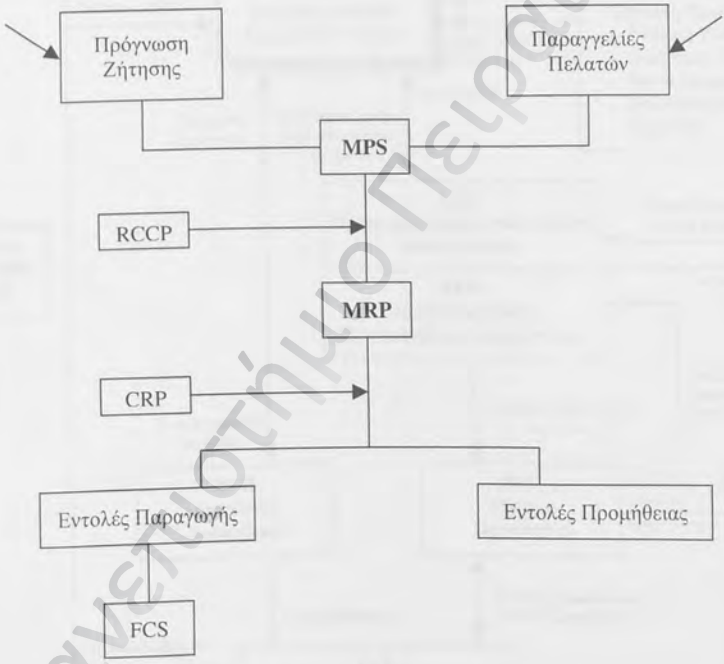
Ο Πίνακας 7 που ακολουθεί, είναι ενδεικτικός των διαφορών μεταξύ RCCP και CRP.

Πίνακας 7. Διαφορές μεταξύ RCCP και CRP.

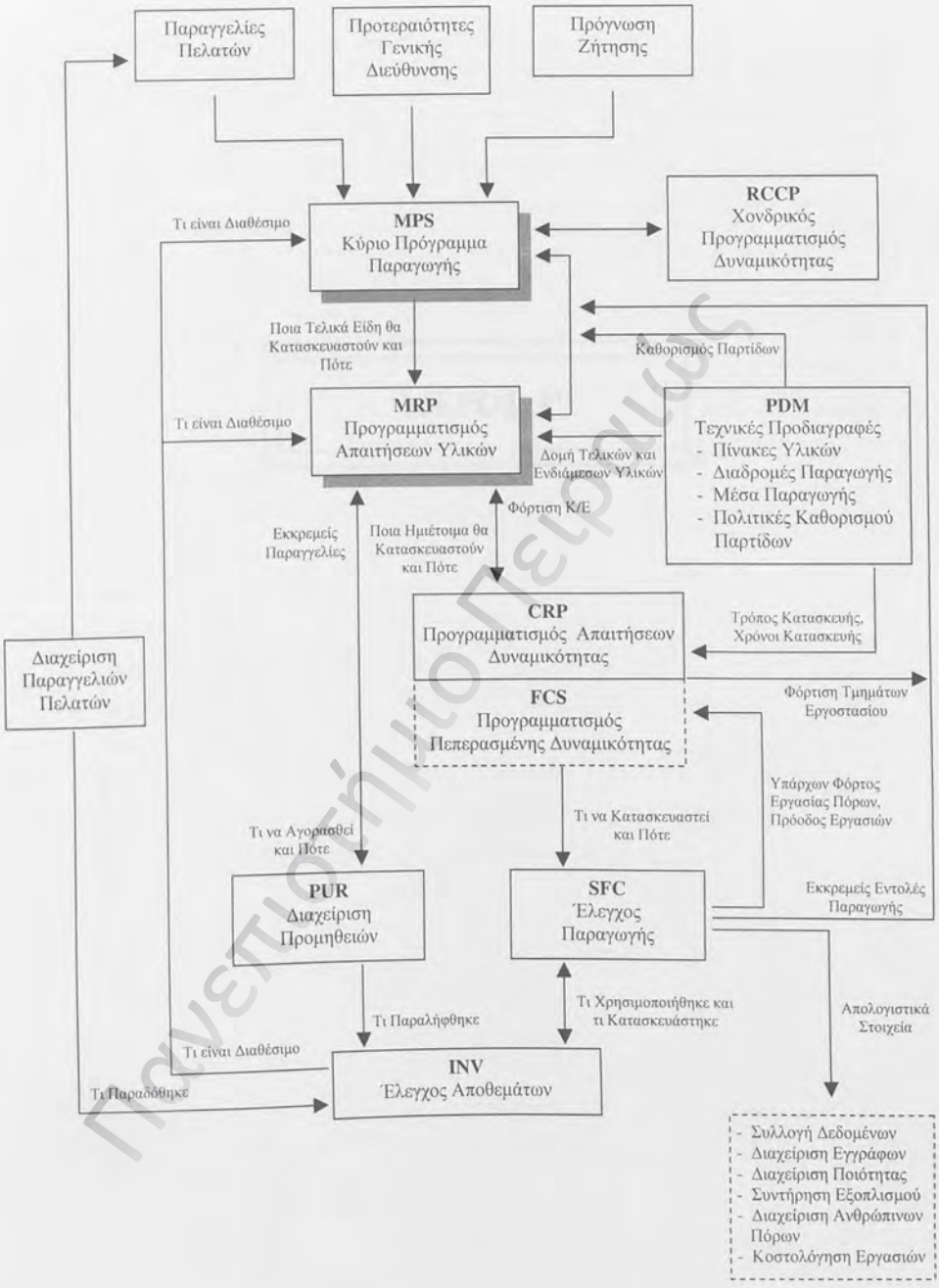
	RCCP	CRP
Τι	Πρόβλεψη δυναμικότητας για πόρους-κλειδιά	Πρόβλεψη δυναμικότητας για όλους τους πόρους
Πως	Σάρωση εντολών παραγωγής MPS μέσω των Προφίλ Δυναμικότητας ή των Πινάκων Πόρων	Σάρωση MPS & MRP μέσω αναλυτικών οδών (φασεολόγια, μέσα παραγωγής)
Πότε	Όποτε απαιτείται για Προσομοίωση	Για Ετήσιο και Τριμηνιαίο προϋπολογισμό καθώς και εβδομαδιαίως και μηνιαίως
Γιατί	Προ-MRP εκτίμηση πλάνου Παραγωγής	Μετά-MRP λεπτομερής Ανάλυση
Ακρίβεια	Μικρή	Μεγάλη
Πολυπλοκότητα	Πολύ μικρότερη από του CRP	Συνήθως ξεπερνά αυτή του MRP
Ορίζοντας Σχεδιασμού	Περιορισμένος από τα όρια του πλάνου παραγωγής	Ίδιος με του MRP μείον τους Χρόνους αναμονής
Υλοποίηση	Βραχεία (χειρόγραφη ή με PC)	Απαιτεί πολλούς υπολογιστικούς πόρους

Η όλη διαδικασία του προγραμματισμού της παραγωγής (Σχήμα 28), συνήθως ολοκληρώνεται με την εκτέλεση του προγραμματισμού πεπερασμένης δυναμικότητας (FCS – Finite Capacity Scheduling). Στη φάση αυτή, επιτυγχάνεται λεπτομερής χρονικός προγραμματισμός των εργασιών, όπου το υποσύστημα FCS, με τη βοήθεια διαφόρων κριτηρίων και τη χρήση ειδικών αλγορίθμων, «φορτώνει» όλες τις ανοικτές εντολές παραγωγής στις μηχανές και τα κέντρα εργασίας, λαμβάνοντας υπόψη όλους τους περιορισμούς δυναμικότητας των πόρων αλλά και τις εναλλακτικές δυνατότητες δρομολόγησης των εντολών στις διάφορες μηχανές / κέντρα εργασίας.

Είναι φυσικά προφανές, ότι για την επιτυχή εφαρμογή ενός συστήματος MRP II, τα ανωτέρω υποσυστήματα πρέπει να έχουν τη δυνατότητα αλληλοενημέρωσης μέσω ανατροφοδότησης των πληροφοριών (feedback), ώστε ανά πάσα στιγμή, όλοι οι πόροι της παραγωγής να συνεργάζονται αρμονικά και να δουλεύουν για την επίτευξη ενός κοινού στόχου. Η ροή των βασικών πληροφοριών καθώς και η δομή ενός συστήματος MRP II, φαίνεται διαγραμματικά στο Σχήμα 29.



Σχήμα 28. Διαδικασία Προγραμματισμού Παραγωγής.



Σχήμα 29. Ροή Πληροφοριών και Δομή ενός Συστήματος MRP II.

ΜΕΡΟΣ 3ο

3.1. Εισαγωγή

Το 3ο αυτό μέρος, αποτελεί το πρακτικό μέρος της παρούσας διπλωματικής εργασίας. Στις επόμενες ενότητες αναλύεται και σχεδιάζεται το υπό ανάπτυξη πληροφοριακό σύστημα για την Κατάστρωση του Κύριου Προγράμματος Παραγωγής (MPS), τον Προγραμματισμό Απαιτήσεων Υλικών (MRP) και τη διαχείριση των Πινάκων Υλικών, οι οποίοι αποτελούν τη βάση για την εκτέλεση του MRP. Οι έννοιες αυτές, παρουσιάστηκαν συνοπτικά στο προηγούμενο κεφάλαιο και αποτελούν το Επιχειρηματικό Μοντέλο (Business Model), πάνω στο οποίο βασίστηκε η ανάπτυξη του συστήματος.

Η ανάλυση και ο σχεδιασμός του συστήματος, υλοποιήθηκαν μέσα στα πλαίσια που θέτει η μεθοδολογία ανάπτυξης πληροφοριακών συστημάτων Unified Process, οι βασικές αρχές της οποίας αναφέρθηκαν στο 1ο κεφάλαιο, ενώ για την αποτύπωση των διαγραμμάτων και των μοντέλων του συστήματος, χρησιμοποιήθηκε η σημειογραφία UML.

Για τη δημιουργία των παραδοτέων που ακολουθούν (με εξαίρεση αυτά που έχουν τη μορφή απλού κειμένου), χρησιμοποιήθηκε σαν Εργαλείο Ανάπτυξης (CASE Tool) ο Power Designer version 9 της Sybase. Το συγκεκριμένο εργαλείο χρησιμοποιήθηκε για την κατασκευή, τόσο του object oriented μοντέλου όσο και του φυσικού μοντέλου (Data Model), ενώ τα πρωτότυπα των οθονών σχεδιάστηκαν στον Power Builder version 7.03 της Sybase. Επιγραμματικά, τα παραδοτέα που παρατίθενται στη συνέχεια, είναι κατά σειρά :

- Σκοπός του Συστήματος (Vision)
- Use Cases – Λειτουργικές Απαιτήσεις της Εφαρμογής
- Use Case Diagram
- Πρωτότυπα Οθονών (GUI Prototypes)
- Γλωσσάρι Όρων
- Domain Model
- Sequence Diagrams
- Design Class Diagram
- Data Model
- Script Δημιουργίας της Βάσης Δεδομένων για SQL Server 2000 (Παράρτημα Α)
- Σκελετός Κώδικα της Εφαρμογής σε VB Net (Παράρτημα Β)

3.2. Σκοπός του Συστήματος (Vision)

Ο στόχος του παρόντος έργου είναι η ανάπτυξη τριών αλληλένδετων μεταξύ τους υποσυστημάτων για την υποστήριξη του κυκλώματος της παραγωγής μιας βιομηχανικής επιχείρησης. Τα υποσυστήματα αυτά, καθώς και τα βασικά τους χαρακτηριστικά είναι :

Κύριος Προγραμματισμός Παραγωγής (MPS)

Ο σκοπός του Κύριου Προγραμματισμού Παραγωγής (MPS) είναι να εξασφαλίσει τον καλίτερο δυνατό συντονισμό μεταξύ του τμήματος πωλήσεων/μάρκετινγκ και της παραγωγής. Το υποσύστημα Κύριου Προγραμματισμού Παραγωγής, λαμβάνοντας υπόψη του την πραγματική ζήτηση των πελατών, την εκτίμηση ή πρόβλεψη των πωλήσεων για το διάστημα προγραμματισμού, τα τρέχοντα αποθέματα και τις διάφορες πολιτικές και παραμέτρους που έχουν καθοριστεί για τα τελικά προϊόντα (είδη με ανεξάρτητη ζήτηση) όπως ύψος αποθεμάτων ασφαλείας, ελάχιστες ποσότητες εντολών παραγωγής κτλ, επιτρέπει την παρακολούθηση του διαθέσιμου αποθέματος συναρτήσει του χρόνου ανά κωδικό είδος, προτείνοντας νέες εντολές παραγωγής για κάθε είδος που εμφανίζει έλλειψη σε σχέση με την πραγματική ζήτηση ή την πρόβλεψη μέσα στην περίοδο (χρονικό ορίζοντα) προγραμματισμού.

Πρόγραμμα Απαιτήσεων Υλικών (MRP)

Απαραίτητη προϋπόθεση για την εκτέλεση του Κύριου Προγράμματος Παραγωγής, είναι η τροφοδότηση της παραγωγής (στον κατάλληλο χρόνο) με όλα εκείνα τα υλικά που απαιτούνται (στις κατάλληλες ποσότητες) για την κατασκευή των τελικών προϊόντων.

Το υποσύστημα Προγραμματισμού Απαιτήσεων Υλικών (MRP) έχει ως κύριο σκοπό την αυτόματη δημιουργία εντολών παραγωγής ημιετοίμων και εντολών προμήθειας αγοραζομένων ειδών (είδη με εξαρτημένη ζήτηση) έτσι ώστε να εξασφαλίζεται με τον καλύτερο δυνατό τρόπο η διαθεσιμότητα των υλικών για το τρέχον Κύριο Πρόγραμμα Παραγωγής. Για να πετύχει τον στόχο αυτό, το MRP λαμβάνει υπόψη του το ισχύον Κύριο Πρόγραμμα Παραγωγής, τους Πίνακες Υλικών, τα διαθέσιμα και δεσμευμένα αποθέματα, τις εντολές παραγωγής και προμήθειας που βρίσκονται σε εξέλιξη και όλες τις παραμέτρους προγραμματισμού που έχουν οριστεί για τα είδη (αποθέματα ασφαλείας, χρόνοι αναμονής – lead times, ελάχιστες, μέγιστες και πολλαπλάσιες ποσότητες παραγγελίας, πολιτικές αναπαραγγελίας κτλ) στο βασικό αρχείο ειδών.

Πίνακες Υλικών (BOM)

Το υποσύτημα των Πινάκων Υλικών, έχει σαν στόχο την αποτύπωση της δεντρικής δομής/σύνθεσης των προϊόντων μιας επιχείρησης. Οι Πίνακες Υλικών απεικονίζουν τις πρώτες ύλες, τα εξαρτήματα και τα συγκροτήματα από τα οποία αποτελείται ένα προϊόν, τις ποσότητες που απαιτούνται για την παραγωγή μιας μονάδας (ή μιας τυπικής παρτίδας) του προϊόντος, καθώς και τα διαδοχικά στάδια κατασκευής/συναρμολόγησης. Η τεκμηρίωση των πινάκων υλικών – συνταγολογίων είναι απαραίτητη προϋπόθεση σε οποιαδήποτε βιομηχανία για τη εκτέλεση των εξής βασικών διαδικασιών :

- Μελέτη ενός προϊόντος
- Κοστολόγηση των προϊόντων
- Προγραμματισμός των προμηθειών πρώτων υλών και της παραγωγής εξαρτημάτων (μέσω της τεχνικής του MRP).

3.3.1. Use Case : Διαχείριση Βαθμικών Πινάκων Υλικών

Primary Actor : Υπεύθυνος Παραγωγής

Ενδιαφερόμενοι	Στόχοι - Οφέλη
Υπεύθυνος Παραγωγής	Αποτύπωση της δεντρικής δομής/σύνθεσης των προϊόντων Εκτέλεση Προγράμματος Παραγωγής Έλεγχος αλληλεξάρτησης Εντολών Παραγωγής/Προμήθειας
Υπεύθυνος Προγραμματισμού Αποθεμάτων	Κατάστρωση Κύριου Προγράμματος Παραγωγής (MPS) Προγραμματισμός Απαιτήσεων Υλικών (MRP)
Υπεύθυνος Προμηθειών	Εξασφάλιση διαθεσιμότητας προμηθευόμενων ειδών για την απρόσκοπτη λειτουργία της παραγωγικής διαδικασίας
Υπεύθυνος Κοστολόγησης	Κοστολόγηση προϊόντων
Υπεύθυνος Αποθηκών	Έλεγχος διαθεσιμότητας υλικών για την εκτέλεση Εντολών Παραγωγής Χορήγηση υλικών έναντι Εντολής Παραγωγής Δέσμευση υλικών έναντι Εντολής Παραγωγής

Προϋποθέσεις Χρήσης (Preconditions) :

Ο Actor πρέπει να έχει κάνει login και να έχει αντίστοιχα δικαιώματα.

Πρέπει να έχουν καταχωρηθεί τα είδη που συμμετέχουν σε έναν Πίνακα Υλικών.

Αποτελέσματα Ολοκλήρωσης του Use Case (Postconditions) :

Ένας Βαθμικός Πίνακας Υλικών έχει εισαχθεί, ενημερωθεί ή διαγραφεί από τη βάση δεδομένων.

Βασικό Σενάριο (Basic Flow) :

1. Ο Actor επιλέγει τη διαδικασία διαχείρισης Πινάκων Υλικών.
2. Ο Actor εισάγει τον κωδικό του συναρμολογήματος.
3. Το Σύστημα ελέγχει την εγκυρότητα του εισαχθέντος κωδικού και εμφανίζει την περιγραφή, τον τύπο και την κύρια μονάδα μέτρησης του είδους.
4. Ο Actor επιλέγει εισαγωγή συστατικού στον Πίνακα Υλικών.
5. Ο Actor εισάγει τον κωδικό του συστατικού.
6. Το Σύστημα ελέγχει την εγκυρότητα του εισαχθέντος κωδικού και εμφανίζει την περιγραφή, τον τύπο και την κύρια μονάδα μέτρησης του συστατικού.
7. Ο Actor εισάγει την ποσότητα του συστατικού που απαιτείται για την κατασκευή/συναρμολόγηση μιας μονάδας του γονέα και το ποσοστό παραγωγής σκάρτων για τη συγκεκριμένη σχέση συναρμολογήματος – συστατικού.

   Τα βήματα 4 έως 7 επαναλαμβάνονται μέχρι να εισαχθούν όλα τα συστατικά του Πίνακα Υλικών.
8. Ο Actor επιλέγει αποθήκευση του Πίνακα Υλικών.
9. Το Σύστημα αποθηκεύει τα δεδομένα.
10. Ο Actor επιλέγει έξοδο από τη διαδικασία.
11. Το Use Case ολοκληρώνεται.

Εναλλακτικά Σενάρια (Alternate Flows) :

3a. Ο κωδικός είδους δεν υπάρχει.
1. Το Σύστημα εμφανίζει κατάλληλο μήνυμα και προτρέπει τον Actor να εισάγει νέο έγκυρο κωδικό είδους.
2. Το Use Case επιστρέφει στο βήμα 2 του Βασικού Σεναρίου.

3b. Ο κωδικός είδους δεν μπορεί να δεχθεί συστατικά.
1. Το Σύστημα εμφανίζει κατάλληλο μήνυμα και προτρέπει τον Actor να εισάγει νέο έγκυρο κωδικό είδους.
2. Το Use Case επιστρέφει στο βήμα 2 του Βασικού Σεναρίου.

3c. Ο κωδικός είδους υπάρχει, έχει όμως ήδη μια δομή Πίνακα Υλικών.
1. Το Σύστημα επιστρέφει όλα τα συστατικά του συναρμολογήματος και εισέρχεται σε κατάσταση επεξεργασίας του Πίνακα Υλικών.
   1a. Ο Actor επιλέγει εισαγωγή νέου συστατικού στον υπάρχοντα Πίνακα Υλικών.
      1. Το Use Case επιστρέφει στο βήμα 5 του Βασικού Σεναρίου.
   1b. Ο Actor επιλέγει διαγραφή ενός υφιστάμενου συστατικού από τον Πίνακα Υλικών.
      1. Το Σύστημα μαρκάρει το συστατικό προς διαγραφή.
         Το βήμα αυτό επαναλαμβάνεται μέχρι να επιλεχθούν όλα τα προς διαγραφή συστατικά.
      2. Το Use Case επιστρέφει στο βήμα 8 του Βασικού Σεναρίου
   1c. Ο Actor επιλέγει τροποποίηση των δεδομένων ενός συστατικού του Πίνακα Υλικών.
      1. Ο Actor καταχωρεί τις επιθυμητές αλλαγές (ποσότητα ή/και ποσοστό σκάρτων).
         Το βήμα αυτό επαναλαμβάνεται μέχρι να καταχωρηθούν όλες οι αλλαγές στα συστατικά.
      2. Το Use Case επιστρέφει στο βήμα 8 του Βασικού Σεναρίου
   1d. Ο Actor επιλέγει διαγραφή όλου του υπάρχοντα Πίνακα Υλικών.
      1. Το Use Case επιστρέφει στο βήμα 8 του Βασικού Σεναρίου.

6a. Ο κωδικός είδους δεν υπάρχει.
1. Το Σύστημα εμφανίζει κατάλληλο μήνυμα και προτρέπει τον Actor να εισάγει νέο έγκυρο κωδικό είδους.
2. Το Use Case επιστρέφει στο βήμα 5 του Βασικού Σεναρίου.

6b. Ο κωδικός είδους δεν μπορεί να χρησιμοποιηθεί σαν συστατικό σε έναν Πίνακα Υλικών.
1. Το Σύστημα εμφανίζει κατάλληλο μήνυμα και προτρέπει τον Actor να εισάγει νέο έγκυρο κωδικό είδους.
2. Το Use Case επιστρέφει στο βήμα 5 του Βασικού Σεναρίου.

Ειδικές Απαιτήσεις (Special Requirements) - Business Rules

- Ένα προμηθευόμενο είδος δεν επιτρέπεται να εμφανίζεται σαν συναρμολόγημα σε έναν Πίνακα Υλικών.
- Ένα τελικό προϊόν (είδος MPS) δεν επιτρέπεται να εμφανίζεται σαν συστατικό σε έναν Πίνακα Υλικών.
- Ένα είδος δεν μπορεί να εμφανίζεται περισσότερο από μια φορά σαν συστατικό σε έναν Πίνακα Υλικών.
- Σε έναν Πίνακα Υλικών, δεν επιτρέπεται ο κωδικός του συναρμολογήματος να εμφανίζεται και σαν συστατικό στον Πίνακα Υλικών.
- Κατά τη διαχείριση των Βαθμικών Πινάκων Υλικών, το Σύστημα πρέπει να ελέγχει και να αποτρέπει την ενδεχόμενη δημιουργία κυκλικών αναφορών στις σχέσεις μεταξύ των ειδών.

3.3.2. Use Case : Εμφάνιση Δομής Προϊόντων/Σημείων Χρήσης Υλικών

Primary Actor : Υπεύθυνος Παραγωγής

Ενδιαφερόμενοι	Στόχοι - Οφέλη
Υπεύθυνος Παραγωγής	Έλεγχος της δεντρικής δομής/σύνθεσης των προϊόντων Απεικόνιση των Σημείων Χρήσης εξαρτημάτων/υλικών Εντοπισμός των Εντολών Παραγωγής που επηρεάζονται από ενδεχόμενες ελλείψεις συστατικών/εξαρτημάτων
Υπεύθυνος Αποθηκών	Έλεγχος διαθεσιμότητας υλικών για την εκτέλεση Εντολών Παραγωγής

Προϋποθέσεις Χρήσης (Preconditions) :

Ο Actor πρέπει να έχει κάνει login και να έχει αντίστοιχα δικαιώματα.

Αποτελέσματα Ολοκλήρωσης του Use Case (Postconditions) :

Εμφάνιση της ιεραρχικής δομής ενός τελικού προϊόντος ή ημιέτοιμου υλικού.

Εμφάνιση των Σημείων Χρήσης ενός συστατικού.

Βασικό Σενάριο (Basic Flow) :

1. Ο Actor επιλέγει τη διαδικασία εμφάνισης της Δομής Προϊόντων/Σημείων Χρήσης Υλικών.
2. Ο Actor εισάγει τον κωδικό του είδους.
3. Το Σύστημα ελέγχει την εγκυρότητα του εισαχθέντος κωδικού και εμφανίζει την περιγραφή, τον τύπο και την κύρια μονάδα μέτρησης του είδους.
4. Ο Actor επιλέγει την εμφάνιση του Συνθετικού Πίνακα Υλικών του είδους.
5. Ο Actor επιλέγει αν θέλει να εμφανιστούν μόνο τα συστατικά του αμέσως κατώτερου επιπέδου της δομής του είδους (Single Level Bill) ή όλα τα επίπεδα της ιεραρχικής δομής του (Multi Level Bill).
6. Το Σύστημα ανακτά τις απαραίτητες πληροφορίες και τις εμφανίζει στην οθόνη.
7. Ο Actor επιλέγει έξοδο από τη διαδικασία.
8. Το Use Case ολοκληρώνεται.

Εναλλακτικά Σενάρια (Alternate Flows) :

3a. Ο κωδικός είδους δεν υπάρχει.
   1. Το Σύστημα εμφανίζει κατάλληλο μήνυμα και προτρέπει τον Actor να εισάγει νέο έγκυρο κωδικό είδους.
   2. Το Use Case επιστρέφει στο βήμα 2 του Βασικού Σεναρίου.

4a. Ο Actor επιλέγει την εμφάνιση των Σημείων Χρήσης του είδους.
   1. Ο Actor επιλέγει αν θέλει να εμφανιστούν μόνο οι γονείς του αμέσως ανώτερου επιπέδου του είδους στη δομή των οποίων συμμετέχει ή όλοι οι γονείς του είδους, μέχρι την κορυφή της ιεραρχίας.
   2. Το Use Case επιστρέφει στο βήμα 6 του Βασικού Σεναρίου.

6a. Το Σύστημα δεν εντόπισε σχετικές πληροφορίες προς εμφάνιση.
   1. Το Σύστημα εμφανίζει κατάλληλο μήνυμα και προτρέπει τον Actor να εισάγει νέο κωδικό είδους.
   2. Το Use Case επιστρέφει στο βήμα 2 του Βασικού Σεναρίου.

3.3.3. Use Case : Κατάστρωση Κύριου Προγράμματος Παραγωγής (MPS)

Primary Actor : Υπεύθυνος Προγραμματισμού Αποθεμάτων

Ενδιαφερόμενοι	Στόχοι – Οφέλη
Υπεύθυνος Προγραμματισμού Αποθεμάτων	Κατάστρωση του Κύριου Προγράμματος Παραγωγής (MPS) για την αναπλήρωση του αποθέματος των τελικών προϊόντων
Υπεύθυνος Παραγωγής	Εκτέλεση του Κύριου Προγράμματος Παραγωγής (MPS) Εξασφάλιση ομοιόμορφου και ομαλού ρυθμού παραγωγής Προγραμματισμός δυναμικότητας των παραγωγικών πόρων
Υπεύθυνος Πωλήσεων – Marketing	Εξασφάλιση της διαθεσιμότητας αποθεμάτων τελικών προϊόντων για την ικανοποίηση των Παραγγελιών των πελατών και της πρόγνωσης της ζήτησης αυτών
Υπεύθυνος Αποθηκών	Χορήγηση υλικών έναντι Εντολών Παραγωγής τελικών προϊόντων Δέσμευση υλικών έναντι Εντολών Παραγωγής τελικών προϊόντων

Προϋποθέσεις Χρήσης (Preconditions) :

Ο Actor πρέπει να έχει κάνει login και να έχει αντίστοιχα δικαιώματα.

Αποτελέσματα Ολοκλήρωσης του Use Case (Postconditions) :

Το Σύστημα έχει διαγράψει τις Προγραμματισμένες (ή και τις Εγκεκριμένες) Εντολές του προηγούμενου MPS και έχει δημιουργήσει μια λίστα νέων Προγραμματισμένων Εντολών Παραγωγής για τα τελικά προϊόντα.

Το Σύστημα έχει επαναπρογραμματίσει (όσον αφορά στην προθεσμία) ήδη υπάρχουσες Εγκεκριμένες Εντολές Παραγωγής τελικών προϊόντων.

Βασικό Σενάριο (Basic Flow) :

1. Ο Actor επιλέγει τη διαδικασία υπολογισμού του Κύριου Προγράμματος Παραγωγής (MPS).
2. Ο Actor εισάγει την καταληκτική ημερομηνία του χρονικού ορίζοντα Προγραμματισμού.
3. Ο Actor επιλέγει υπολογισμό του Κύριου Προγράμματος Παραγωγής (MPS) χωρίς Επαναπρογραμματισμό των υφιστάμενων Εγκεκριμένων Εντολών Παραγωγής των τελικών προϊόντων.
4. Ο Actor σηματοδοτεί την έναρξη των υπολογισμών.
5. Το Σύστημα, μετά το τέλος των υπολογισμών εμφανίζει μήνυμα ότι η διαδικασία εκτελέστηκε με επιτυχία.
6. Το Use Case ολοκληρώνεται.

Εναλλακτικά Σενάρια (Alternate Flows) :

3a. Ο Actor επιλέγει Επαναπρογραμματισμό των υφιστάμενων Εγκεκριμένων Εντολών Παραγωγής των τελικών προϊόντων.

1. Το Use Case επιστρέφει στο βήμα 4 του Βασικού Σεναρίου.

Ειδικές Απαιτήσεις (Special Requirements) - Business Rules

- Εφόσον δεν εισαχθεί καταληκτική ημερομηνία η οποία καθορίζει σαφώς το χρονικό ορίζοντα Προγραμματισμού, το σύστημα συνεχίζει τους υπολογισμούς μέχρι την ικανοποίηση και της τελευταίας χρονικά ζήτησης όλων των τελικών προϊόντων.

3.3.4. Use Case : Προγραμματισμός Απαιτήσεων Υλικών (MRP)

Primary Actor : Υπεύθυνος Προγραμματισμού Αποθεμάτων

Ενδιαφερόμενοι	Στόχοι – Οφέλη
Υπεύθυνος Προγραμματισμού Αποθεμάτων	Προγραμματισμός των Απαιτήσεων σε ημιέτοιμα και αγοραζόμενα υλικά για την υλοποίηση του Κύριου Προγράμματος Παραγωγής (MPS)
Υπεύθυνος Παραγωγής	Εκτέλεση των Εντολών Παραγωγής των ενδιαμέσων υλικών για την υλοποίηση του Κύριου Προγράμματος Παραγωγής (MPS) Εξασφάλιση ομοιόμορφου και ομαλού ρυθμού παραγωγής Προγραμματισμός δυναμικότητας των παραγωγικών πόρων
Υπεύθυνος Προμηθειών	Εξασφάλιση διαθεσιμότητας προμηθευόμενων ειδών για την απρόσκοπτη λειτουργία της παραγωγικής διαδικασίας και την υλοποίηση του Κύριου Προγράμματος Παραγωγής (MPS)
Υπεύθυνος Αποθηκών	Χορήγηση υλικών έναντι Εντολών Παραγωγής ενδιάμεσων υλικών Δέσμευση υλικών έναντι Εντολών Παραγωγής ενδιάμεσων υλικών

Προϋποθέσεις Χρήσης (Preconditions) :

Ο Actor πρέπει να έχει κάνει login και να έχει αντίστοιχα δικαιώματα.

Αποτελέσματα Ολοκλήρωσης του Use Case (Postconditions) :

Το Σύστημα έχει επανα-υπολογίσει το χαμηλότερο σημείο εμφάνισης (LLC) όλων των ειδών.

Το Σύστημα έχει διαγράψει τις Προγραμματισμένες (ή και τις Εγκεκριμένες) Εντολές του προηγούμενου MRP και έχει δημιουργήσει μια λίστα νέων Προγραμματισμένων Εντολών Παραγωγής για τα ημιέτοιμα υλικά και Εντολών Προμήθειας για τα αγοραζόμενα είδη.

Το Σύστημα έχει επαναπρογραμματίσει (όσον αφορά στην προθεσμία) ήδη υπάρχουσες Εγκεκριμένες Εντολές Παραγωγής ημιετοίμων υλικών και Εντολές Προμήθειας αγοραζόμενων ειδών.

Βασικό Σενάριο (Basic Flow) :

1. Ο Actor επιλέγει τη διαδικασία Προγραμματισμού Απαιτήσεων Υλικών (MRP).
2. Ο Actor ζητά επανα-υπολογισμό του χαμηλότερου σημείου εμφάνισης όλων των ειδών.
3. Ο Actor εισάγει την καταληκτική ημερομηνία του χρονικού ορίζοντα Προγραμματισμού.
4. Ο Actor επιλέγει Προγραμματισμό Απαιτήσεων Υλικών (MRP) χωρίς Επαναπρογραμματισμό των υφιστάμενων Εγκεκριμένων Εντολών Παραγωγής των ημιετοίμων υλικών και των Εντολών Προμήθειας των αγοραζόμενων ειδών.
5. Ο Actor σηματοδοτεί την έναρξη των υπολογισμών.
6. Το Σύστημα, μετά το τέλος των υπολογισμών εμφανίζει μήνυμα ότι η διαδικασία εκτελέστηκε με επιτυχία.
7. Το Use Case ολοκληρώνεται.

Εναλλακτικά Σενάρια (Alternate Flows) :

4a. Ο Actor επιλέγει Επαναπρογραμματισμό των υφιστάμενων Εγκεκριμένων Εντολών Παραγωγής των ημιετοίμων υλικών και των Εντολών Προμήθειας των αγοραζόμενων ειδών.

1. Το Use Case επιστρέφει στο βήμα 5 του Βασικού Σεναρίου.

Ειδικές Απαιτήσεις (Special Requirements) - Business Rules

- Εφόσον δεν εισαχθεί καταληκτική ημερομηνία η οποία καθορίζει σαφώς το χρονικό ορίζοντα Προγραμματισμού, το σύστημα συνεχίζει τους υπολογισμούς μέχρι την ικανοποίηση και της τελευταίας χρονικά ζήτησης όλων των ενδιάμεσων και αγοραζόμενων ειδών.

3.3.5. Use Case : Διαχείριση Προτεινόμενων Εντολών MPS

Primary Actor : Υπεύθυνος Παραγωγής

Ενδιαφερόμενοι	Στόχοι – Οφέλη
Υπεύθυνος Παραγωγής	Έγκριση Προγραμματισμένων Εντολών Παραγωγής τελικών προϊόντων, οι οποίες στη συνέχεια είτε θα εκδοθούν προς εκτέλεση στην παραγωγή, είτε θα ληφθούν υπόψη κατά τον επαναπρογραμματισμό του επόμενου MPS Τροποποίηση των Προτεινόμενων από το MPS Εντολών Παραγωγής τελικών προϊόντων, όσον αφορά στην ποσότητα και την προθεσμία τους

Προϋποθέσεις Χρήσης (Preconditions) :

Ο Actor πρέπει να έχει κάνει login και να έχει αντίστοιχα δικαιώματα.

Αποτελέσματα Ολοκλήρωσης του Use Case (Postconditions) :

Επιλεκτικά, Προτεινόμενες (Προγραμματισμένες και Εγκεκριμένες) από το σύστημα Εντολές του Κύριου Προγράμματος Παραγωγής (MPS), έχουν διαγραφεί ή έχουν αλλάξει κατάσταση (status) ή/και ποσότητα ή/και προθεσμία. Ανάλογη μεταβολή έχει επέλθει και στις αντίστοιχες Εντολές Εξαρτημένης Ζήτησης των Εντολών MPS.

Βασικό Σενάριο (Basic Flow) :

1. Ο Actor επιλέγει τη διαδικασία διαχείρισης των Προτεινόμενων Εντολών του Κύριου Προγράμματος Παραγωγής (MPS).
2. Ο Actor επιλέγει εάν στην εμφανιζόμενη λίστα θα συμπεριληφθούν μόνο Προγραμματισμένες Εντολές, μόνο Εγκεκριμένες Εντολές ή και οι δύο κατηγορίες εντολών.
3. Ο Actor ζητά την εμφάνιση των Εντολών με βάση τα κριτήρια που καταχώρησε.
4. Το Σύστημα ανακτά τις απαραίτητες πληροφορίες και τις εμφανίζει στην οθόνη.
5. Ο Actor επιλέγει μια εντολή από τη λίστα και καταχωρεί τις επιθυμητές αλλαγές (κατάσταση εντολής, ποσότητα εντολής, προθεσμία εντολής).

   Το βήμα αυτό επαναλαμβάνεται μέχρι να τροποποιηθούν όλες οι εντολές που επιθυμεί ο Actor.
6. Ο Actor επιλέγει αποθήκευση των δεδομένων.
7. Το Σύστημα αποθηκεύει τα δεδομένα.
8. Ο Actor επιλέγει έξοδο από τη διαδικασία.
9. Το Use Case ολοκληρώνεται.

Εναλλακτικά Σενάρια (Alternate Flows) :

4a. Το Σύστημα δεν εντόπισε σχετικές πληροφορίες προς εμφάνιση.
   1. Το Σύστημα εμφανίζει κατάλληλο μήνυμα και προτρέπει τον Actor για νέα αναζήτηση.
   2. Το Use Case επιστρέφει στο βήμα 2 του Βασικού Σεναρίου.

5a. Ο Actor επιλέγει διαγραφή μιας εντολής από τη λίστα.
   1. Το Σύστημα μαρκάρει την εντολή που πρόκειται να διαγραφεί.

      Το βήμα αυτό επαναλαμβάνεται μέχρι να επιλεχθούν όλες οι εντολές προς διαγραφή.
   2. Το Use Case επιστρέφει στο βήμα 6 του Βασικού Σεναρίου.

3.3.6. Use Case : Διαχείριση Προτεινόμενων Εντολών MRP

Primary Actor : Υπεύθυνος Παραγωγής

Ενδιαφερόμενοι	Στόχοι – Οφέλη
Υπεύθυνος Παραγωγής	Έγκριση Προγραμματισμένων Εντολών παραγόμενων ενδιάμεσων υλικών, οι οποίες στη συνέχεια είτε θα εκδοθούν προς εκτέλεση στην παραγωγή, είτε θα ληφθούν υπόψη κατά τον επαναπρογραμματισμό του επόμενου MRP Τροποποίηση των Προτεινόμενων από το MRP Εντολών παραγόμενων ενδιάμεσων υλικών, όσον αφορά στην ποσότητα και την προθεσμία τους
Υπεύθυνος Προμηθειών	Έγκριση Προγραμματισμένων Εντολών προμηθευόμενων ειδών, οι οποίες στη συνέχεια είτε θα εκδοθούν προς διεκπεραίωση, είτε θα ληφθούν υπόψη κατά τον επαναπρογραμματισμό του επόμενου MRP Τροποποίηση των Προτεινόμενων από το MRP Εντολών προμηθευόμενων ειδών, όσον αφορά στην ποσότητα και την προθεσμία τους

Προϋποθέσεις Χρήσης (Preconditions) :

Ο Actor πρέπει να έχει κάνει login και να έχει αντίστοιχα δικαιώματα.

Αποτελέσματα Ολοκλήρωσης του Use Case (Postconditions) :

Επιλεκτικά, Προτεινόμενες (Προγραμματισμένες και Εγκεκριμένες) από το σύστημα Εντολές MRP, έχουν διαγραφεί ή έχουν αλλάξει κατάσταση (status) ή/και ποσότητα ή/και προθεσμία. Ανάλογη μεταβολή έχει επέλθει και στις αντίστοιχες Εντολές Εξαρτημένης Ζήτησης των Εντολών MRP.

Βασικό Σενάριο (Basic Flow) :

1. Ο Actor επιλέγει τη διαδικασία διαχείρισης των Προτεινόμενων Εντολών του Προγραμματισμού Απαιτήσεων Υλικών (MRP).
2. Ο Actor επιλέγει εάν στην εμφανιζόμενη λίστα θα συμπεριληφθούν μόνο παραγόμενα υλικά, μόνο προμηθευόμενα υλικά ή και τα δύο.
3. Ο Actor επιλέγει εάν στην εμφανιζόμενη λίστα θα συμπεριληφθούν μόνο Προγραμματισμένες Εντολές, μόνο Εγκεκριμένες Εντολές ή και οι δύο κατηγορίες εντολών.
4. Ο Actor ζητά την εμφάνιση των Εντολών με βάση τα κριτήρια που καταχώρησε.
5. Το Σύστημα ανακτά τις απαραίτητες πληροφορίες και τις εμφανίζει στην οθόνη.
6. Ο Actor επιλέγει μια εντολή από τη λίστα και καταχωρεί τις επιθυμητές αλλαγές (κατάσταση εντολής, ποσότητα εντολής, προθεσμία εντολής).

   Το βήμα αυτό επαναλαμβάνεται μέχρι να τροποποιηθούν όλες οι εντολές που επιθυμεί ο Actor.
7. Ο Actor επιλέγει αποθήκευση των δεδομένων.
8. Το Σύστημα αποθηκεύει τα δεδομένα.
9. Ο Actor επιλέγει έξοδο από τη διαδικασία.
10. Το Use Case ολοκληρώνεται.

Εναλλακτικά Σενάρια (Alternate Flows) :

5a. Το Σύστημα δεν εντόπισε σχετικές πληροφορίες προς εμφάνιση.

1. Το Σύστημα εμφανίζει κατάλληλο μήνυμα και προτρέπει τον Actor για νέα αναζήτηση.
2. Το Use Case επιστρέφει στο βήμα 2 του Βασικού Σεναρίου.

6a. Ο Actor επιλέγει διαγραφή μιας εντολής από τη λίστα.

1. Το Σύστημα μαρκάρει την εντολή που πρόκειται να διαγραφεί.

   Το βήμα αυτό επαναλαμβάνεται μέχρι να επιλεχθούν όλες οι εντολές προς διαγραφή.
2. Το Use Case επιστρέφει στο βήμα 7 του Βασικού Σεναρίου.

3.3.7. Use Case : Εμφάνιση Ειδών απλού ROP για Αναπλήρωση Αποθέματος

Primary Actor : Υπεύθυνος Προγραμματισμού Αποθεμάτων

Ενδιαφερόμενοι	Στόχοι – Οφέλη
Υπεύθυνος Προγραμματισμού Αποθεμάτων	Αναπλήρωση αποθέματος ημιετοίμων και προμηθευόμενων υλικών που δεν λαμβάνονται υπόψη κατά τον Προγραμματισμό Απαιτήσεων Υλικών (υλικά τα οποία διαχειρίζονται με τη λογική του απλού Reorder Point – ROP)
Υπεύθυνος Παραγωγής	Έκδοση και εκτέλεση Εντολών για την παραγωγή ημιετοίμων υλικών που εμπίπτουν στην ανωτέρω κατηγορία
Υπεύθυνος Προμηθειών	Εξασφάλιση διαθεσιμότητας προμηθευόμενων ειδών που εμπίπτουν στην ανωτέρω κατηγορία

Προϋποθέσεις Χρήσης (Preconditions) :

Ο Actor πρέπει να έχει κάνει login και να έχει αντίστοιχα δικαιώματα.

Αποτελέσματα Ολοκλήρωσης του Use Case (Postconditions) :

Εμφάνιση ημιετοίμων και προμηθευόμενων υλικών που διαχειρίζονται με τη λογική του απλού Reorder Point (ROP) και χρήζουν αναπαραγγελίας (το ύψος του αποθέματος των υλικών είναι χαμηλότερο από τη στάθμη που ορίζει το επίπεδο αναπαραγγελίας).

Βασικό Σενάριο (Basic Flow) :

1. Ο Actor επιλέγει τη διαδικασία εμφάνισης των υλικών απλού ROP για αναπλήρωση αποθέματος.
2. Ο Actor επιλέγει εάν στην εμφανιζόμενη λίστα θα συμπεριληφθούν μόνο παραγόμενα υλικά, μόνο προμηθευόμενα υλικά ή και τα δύο.
3. Ο Actor ζητά την εμφάνιση των υλικών που χρήζουν αναπαραγγελίας.
4. Το Σύστημα ανακτά τις απαραίτητες πληροφορίες και τις εμφανίζει στην οθόνη.
5. Ο Actor επιλέγει έξοδο από τη διαδικασία.
6. Το Use Case ολοκληρώνεται.

Εναλλακτικά Σενάρια (Alternate Flows) :

4a. Το Σύστημα δεν εντόπισε σχετικές πληροφορίες προς εμφάνιση.

1. Το Σύστημα ενημερώνει τον Actor εμφανίζοντας κατάλληλο μήνυμα.
2. Το Use Case ολοκληρώνεται.

3.3.8. Use Case : Εμφάνιση Προβλεπόμενου Αποθέματος Ειδών

Primary Actor : Υπεύθυνος Προγραμματισμού Αποθεμάτων

Ενδιαφερόμενοι	Στόχοι – Οφέλη
Υπεύθυνος Προγραμματισμού Αποθεμάτων	Έλεγχος της διαθεσιμότητας των αποθεμάτων όλων των ειδών για τη δημιουργία νέων εντολών αναπλήρωσης του αποθέματος και την κάλυψη έκτακτων αναγκών
Υπεύθυνος Παραγωγής	Έλεγχος της διαθεσιμότητας των αποθεμάτων των ειδών για την έκδοση και εκτέλεση Εντολών Παραγωγής ημιετοίμων και τελικών προϊόντων
Υπεύθυνος Προμηθειών	Έλεγχος της διαθεσιμότητας προμηθευόμενων ειδών για την έκδοση και διεκπεραίωση Εντολών Προμήθειας
Υπεύθυνος Πωλήσεων – Marketing	Έλεγχος της διαθεσιμότητας των αποθεμάτων των πωλούμενων ειδών για την ικανοποίηση της ζήτησης Καθορισμός ρεαλιστικής ημερομηνίας παράδοσης των Παραγγελιών

Προϋποθέσεις Χρήσης (Preconditions) :

Ο Actor πρέπει να έχει κάνει login και να έχει αντίστοιχα δικαιώματα.

Αποτελέσματα Ολοκλήρωσης του Use Case (Postconditions) :

Εμφάνιση του Προβλεπόμενου αποθέματος ενός είδους, λαμβάνοντας υπόψη όλες τις εκδοθείσες (ανοιχτές) εντολές και επιλεκτικά τις Προγραμματισμένες ή/και τις Εγκεκριμένες εντολές του είδους.

Βασικό Σενάριο (Basic Flow) :

1. Ο Actor επιλέγει τη διαδικασία εμφάνισης του Προβλεπόμενου Αποθέματος Ειδών.
2. Ο Actor εισάγει τον κωδικό του είδους.
3. Το Σύστημα ελέγχει την εγκυρότητα του εισαχθέντος κωδικού και εμφανίζει την περιγραφή, τον τύπο και όλες τις άλλες βασικές ιδιότητες του είδους.
4. Ο Actor επιλέγει αν στον υπολογισμό του Προβλεπόμενου αποθέματος του είδους θέλει να συμπεριληφθούν και οι Προγραμματισμένες ή/και οι Εγκεκριμένες εντολές του είδους.
5. Ο Actor ζητά την εμφάνιση του Προβλεπόμενου αποθέματος του είδους.
6. Το Σύστημα ανακτά τις απαραίτητες πληροφορίες και τις εμφανίζει στην οθόνη.
7. Ο Actor επιλέγει έξοδο από τη διαδικασία.
8. Το Use Case ολοκληρώνεται.

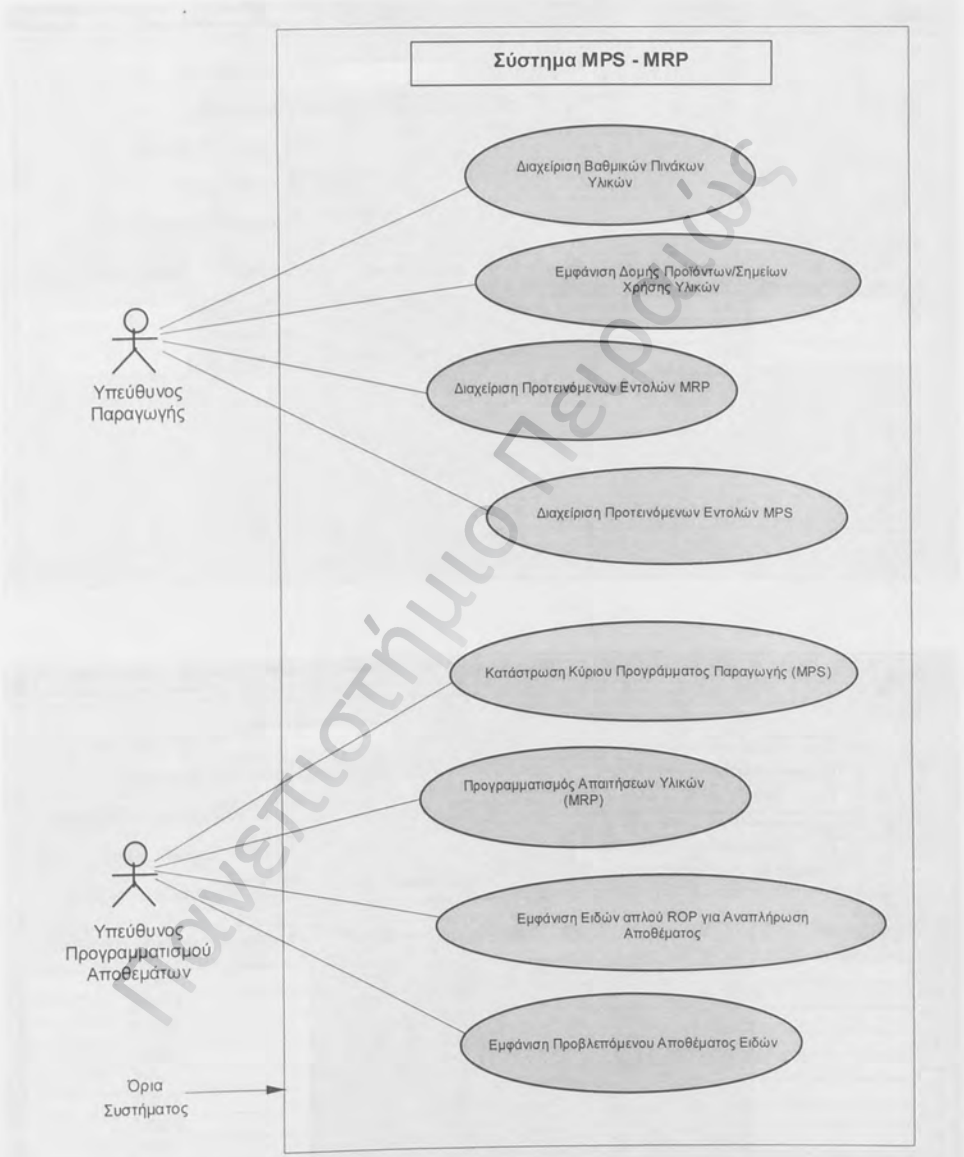
Εναλλακτικά Σενάρια (Alternate Flows) :

3a. Ο κωδικός είδους δεν υπάρχει.
   1. Το Σύστημα εμφανίζει κατάλληλο μήνυμα και προτρέπει τον Actor να εισάγει νέο έγκυρο κωδικό είδους.
   2. Το Use Case επιστρέφει στο βήμα 2 του Βασικού Σεναρίου.

6a. Το Σύστημα δεν εντόπισε σχετικές πληροφορίες προς εμφάνιση.
   1. Το Σύστημα εμφανίζει κατάλληλο μήνυμα και προτρέπει τον Actor να εισάγει νέο κωδικό είδους.
   2. Το Use Case επιστρέφει στο βήμα 2 του Βασικού Σεναρίου.

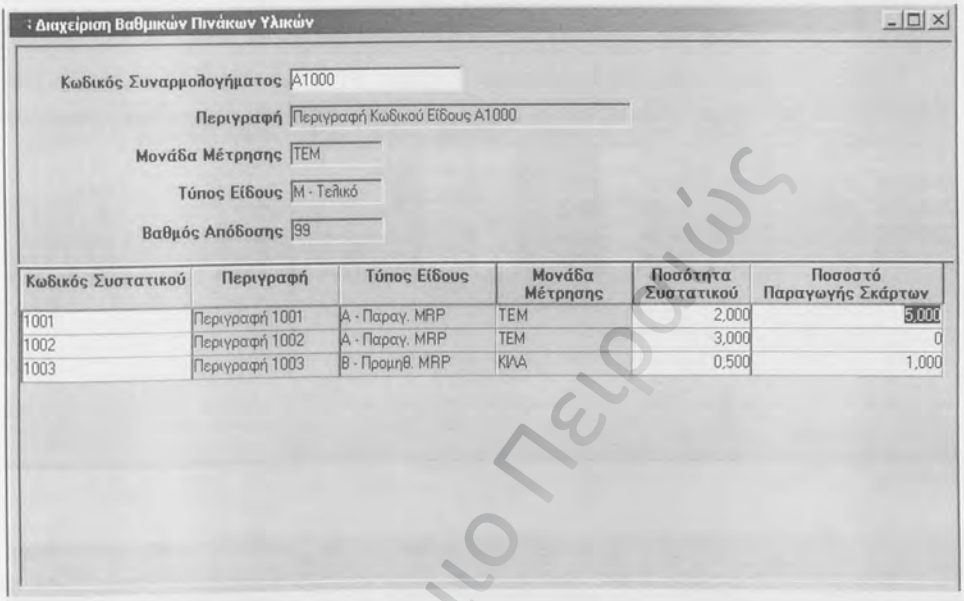
3.3.9. Use Case Diagram

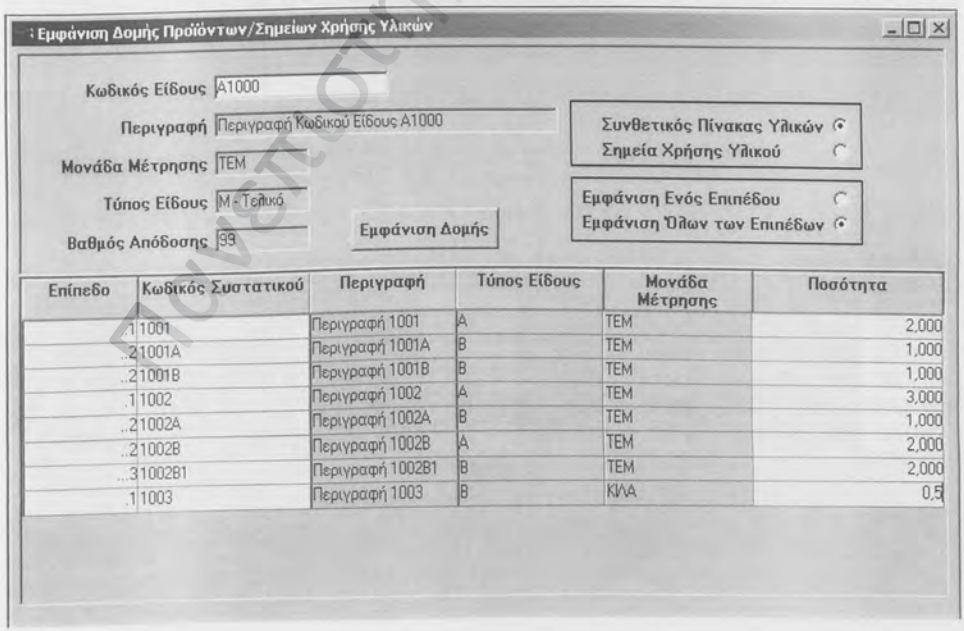
Στο Use Case Diagram που ακολουθεί, απεικονίζονται οι Primary Actiors του συστήματος, τα Use Cases και οι μεταξύ τους σχέσεις. Το Use Case Diagram αποτελεί ένα πολύ χρήσιμο εργαλείο επικοινωνίας, στο οποίο αποτυπώνεται συνοπτικά η συμπεριφορά του συστήματος και καθορίζονται τα όριά του.

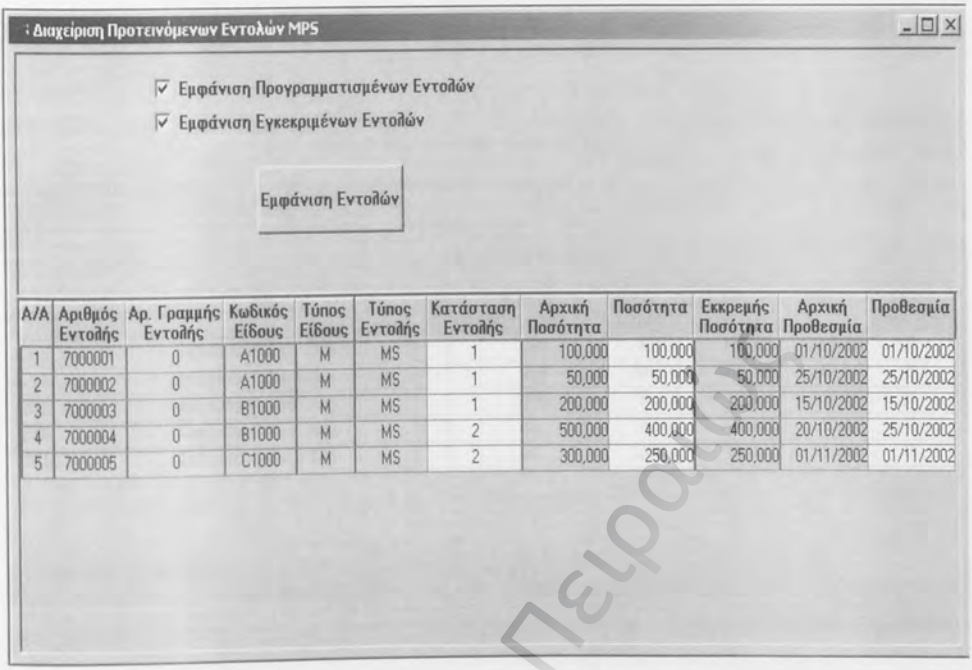


3.4. Πρωτότυπα Οθονών (GUI Prototypes)

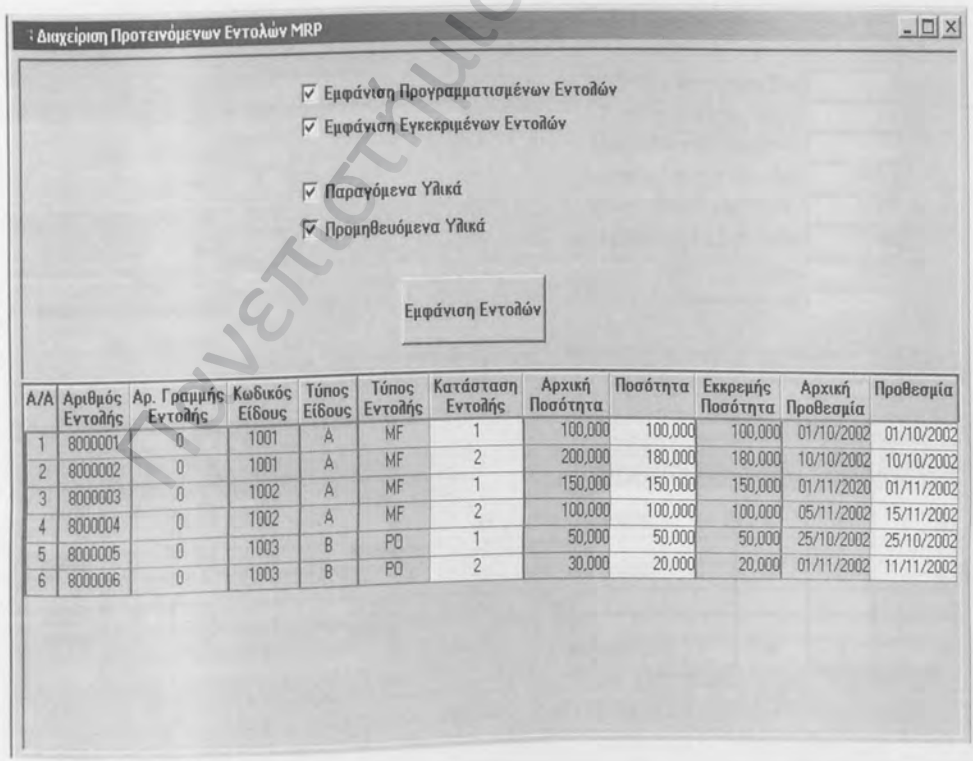
Στην ενότητα αυτή, παρατίθενται τα Πρωτότυπα των Οθονών του συστήματος. Δεν έχουν σχεδιαστεί μόνον οι οθόνες που σχετίζονται με την εκτέλεση του Κύριου Προγραμματισμού Παραγωγής και του Προγραμματισμού Απαιτήσεων Υλικών, δεδομένου ότι δεν παρουσιάζουν κάποιο ιδιαίτερο ενδιαφέρον από πλευράς εμφάνισης.



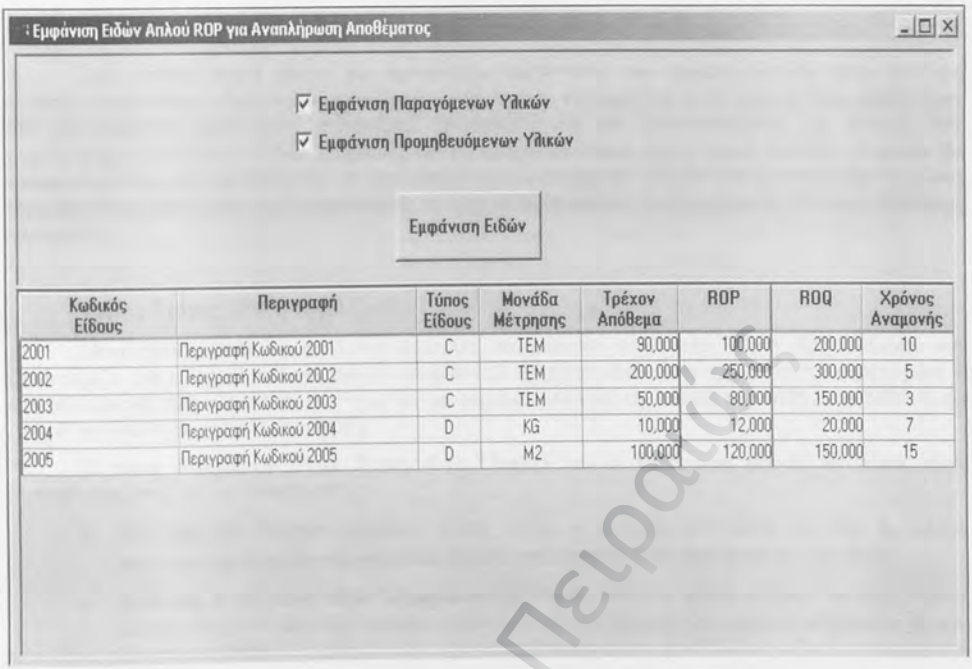




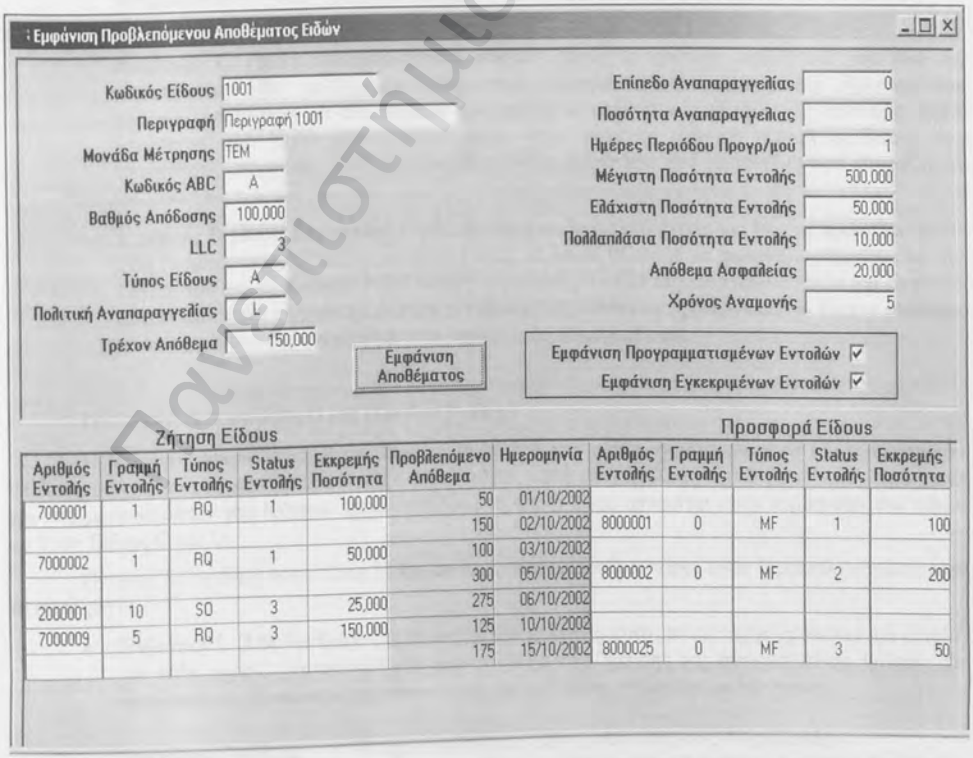
Α/Α	Αριθμός Εντολής	Αρ. Γραμμής Εντολής	Κωδικός Είδους	Τύπος Είδους	Τύπος Εντολής	Κατάσταση Εντολής	Αρχική Ποσότητα	Ποσότητα	Εκκρεμής Ποσότητα	Αρχική Προθεσμία	Προθεσμία
1	7000001	0	A1000	M	MS	1	100,000	100,000	100,000	01/10/2002	01/10/2002
2	7000002	0	A1000	M	MS	1	50,000	50,000	50,000	25/10/2002	25/10/2002
3	7000003	0	B1000	M	MS	1	200,000	200,000	200,000	15/10/2002	15/10/2002
4	7000004	0	B1000	M	MS	2	500,000	400,000	400,000	20/10/2002	25/10/2002
5	7000005	0	C1000	M	MS	2	300,000	250,000	250,000	01/11/2002	01/11/2002



Α/Α	Αριθμός Εντολής	Αρ. Γραμμής Εντολής	Κωδικός Είδους	Τύπος Είδους	Τύπος Εντολής	Κατάσταση Εντολής	Αρχική Ποσότητα	Ποσότητα	Εκκρεμής Ποσότητα	Αρχική Προθεσμία	Προθεσμία
1	8000001	0	1001	A	MF	1	100,000	100,000	100,000	01/10/2002	01/10/2002
2	8000002	0	1001	A	MF	2	200,000	180,000	180,000	10/10/2002	10/10/2002
3	8000003	0	1002	A	MF	1	150,000	150,000	150,000	01/11/2020	01/11/2002
4	8000004	0	1002	A	MF	2	100,000	100,000	100,000	05/11/2002	15/11/2002
5	8000005	0	1003	B	PO	1	50,000	50,000	50,000	25/10/2002	25/10/2002
6	8000006	0	1003	B	PO	2	30,000	20,000	20,000	01/11/2002	11/11/2002



Κωδικός Είδους	Περιγραφή	Τύπος Είδους	Μονάδα Μέτρησης	Τρέχον Απόθεμα	ROP	ROQ	Χρόνος Αναμονής
2001	Περιγραφή Κωδικού 2001	C	TEM	90,000	100,000	200,000	10
2002	Περιγραφή Κωδικού 2002	C	TEM	200,000	250,000	300,000	5
2003	Περιγραφή Κωδικού 2003	C	TEM	50,000	80,000	150,000	3
2004	Περιγραφή Κωδικού 2004	D	KG	10,000	12,000	20,000	7
2005	Περιγραφή Κωδικού 2005	D	M2	100,000	120,000	150,000	15



Αριθμός Εντολής	Γραμμή Εντολής	Τύπος Εντολής	Status Εντολής	Εκκρεμής Ποσότητα	Προβλεπόμενο Απόθεμα	Ημερομηνία	Αριθμός Εντολής	Γραμμή Εντολής	Τύπος Εντολής	Status Εντολής	Εκκρεμής Ποσότητα
7000001	1	RQ	1	100,000	50	01/10/2002					
					150	02/10/2002	8000001	0	MF	1	100
7000002	1	RQ	1	50,000	100	03/10/2002					
					300	05/10/2002	8000002	0	MF	2	200
2000001	10	SO	3	25,000	275	06/10/2002					
7000009	5	RQ	3	150,000	125	10/10/2002					
					175	15/10/2002	8000025	0	MF	3	50

3.5. Γλωσσάρι Όρων (Glossary)

Στην ενότητα αυτή γίνεται μια προσπάθεια επεξήγησης των σημαντικότερων όρων του υπό ανάπτυξη συστήματος. Σκοπός του παραδοτέου αυτού είναι να συμβάλει στην ομαλή επικοινωνία όλων των εμπλεκομένων στο έργο ανάπτυξης, τεκμηριώνοντας και διευκρινίζοντας τις έννοιες των χρησιμοποιούμενων όρων. Είναι προφανές ότι θα επικεντρωθούμε στους όρους αυτούς, οι οποίοι θα χρησιμοποιηθούν στη συνέχεια για το σχεδιασμό των αντικειμένων και δεν θα επεκταθούμε σε όλους τους βασικούς όρους που περιλαμβάνονται σε ένα ολοκληρωμένο πληροφοριακό σύστημα διοίκησης παραγωγής.

Τύπος Είδους (Part Type Code)

Μονοψήφιος κωδικός, ο οποίος αποτελεί υποχρεωτικό πεδίο στο Κύριο Αρχείο Ειδών και χαρακτηρίζει ένα είδος. Η επιλογή του κωδικού αυτού εξαρτάται από το αν πρόκειται για παραγόμενα ή προμηθευόμενα είδη και αν το σύστημα θα τα παρακολουθεί με τη λογική του MPS, του MRP ή τη λογική του απλού Reorder Point (ROP).

Ο τύπος δεδομένων (Data Type) είναι Char(1) και οι τιμές που μπορεί να πάρει είναι (προκαθορισμένες από το σύστημα) :

- **Κωδικός M (Master Schedule Part):** Αυτός ο κωδικός επιλέγεται για όλα τα τελικά προϊόντα της επιχείρησης, τα οποία λαμβάνονται υπόψη κατά την εκτέλεση του MPS.
- **Κωδικός A (Normal MRP Manufacturing Part):** Αυτός ο κωδικός αφορά σε παραγόμενα είδη (ενδιάμεσα είδη στον πίνακα υλικών), τα οποία θέλουμε να παρακολουθήσουμε με την λογική του MRP.
- **Κωδικός B (Normal MRP Purchased Part):** Αυτός ο κωδικός χαρακτηρίζει προμηθευόμενα είδη (πρώτες ύλες, υλικά συσκευασίας κλπ), τα οποία θέλουμε να παρακολουθήσουμε με την λογική του MRP.
- **Κωδικός C (ROP Manufacturing Part):** Αυτός ο κωδικός επιλέγεται για όλα τα παραγόμενα είδη (ενδιάμεσα είδη στον πίνακα υλικών), τα οποία θέλουμε να παρακολουθήσουμε με την λογική του απλού Reorder Point (ROP). Το σύστημα απλά δημιουργεί μια λίστα για τα υλικά αυτά, όταν κατά τη χρονική στιγμή εκτέλεσης της αντίστοιχης ενέργειας, το τρέχον απόθεμα είναι μικρότερο από τη στάθμη που εκφράζει το Reorder Point.
- **Κωδικός D (ROP Purchased Part):** Αυτός ο κωδικός επιλέγεται για όλα τα προμηθευόμενα είδη (πρώτες ύλες, υλικά συσκευασίας κλπ), τα οποία θέλουμε να παρακολουθήσουμε με την λογική του απλού Reorder Point (ROP). Το σύστημα απλά δημιουργεί μια λίστα για τα υλικά αυτά, όταν κατά τη χρονική στιγμή εκτέλεσης της αντίστοιχης ενέργειας, το τρέχον απόθεμα είναι μικρότερο από τη στάθμη που εκφράζει το Reorder Point.

Πολιτική Αναπαραγγελίας (Order Policy)

Η Πολιτική Αναπαραγγελίας καθορίζει τη λογική την οποία χρησιμοποιεί το σύστημα για τον υπολογισμό της ποσότητας των προτεινόμενων εντολών, κατά την εκτέλεση του MPS ή MRP. Πρόκειται για υποχρεωτικό πεδίο του Κύριου Αρχείου Ειδών, το οποίο όμως αγνοείται στην περίπτωση των ειδών με Τύπο Είδους C και D.

Ο τύπος δεδομένων είναι Char(1) και οι τιμές που μπορεί να πάρει είναι (προκαθορισμένες από το σύστημα) :

- **Πολιτική L (Lot for Lot):** Σύμφωνα με την πολιτική αυτή, το σύστημα προτείνει μία εντολή για κάθε ημέρα, για την κάλυψη του συνόλου της ζήτησης της συγκεκριμένης ημέρας. Το διαθέσιμο απόθεμα υπολογίζεται σε καθημερινή βάση, σύμφωνα με τον τύπο :

  Διαθέσιμο Απόθεμα = Τρέχον Απόθεμα + Προγραμματισμένες Παραλαβές – Ανεξάρτητη Ζήτηση – Εξαρτημένη Ζήτηση – Απόθεμα Ασφαλείας

Εξαρτημένη ζήτηση θεωρείται η ζήτηση που προκύπτει από τις εντολές των ειδών του αμέσως ανώτερου επιπέδου στους πίνακες υλικών, ενώ η ανεξάρτητη ζήτηση αποτελεί το μέγιστο μεταξύ της ζήτησης των πελατών (πραγματικές παραγγελίες) και της πρόβλεψης της ζήτησης για μια συγκεκριμένη ημέρα. Για την κάλυψη της ζήτησης μιας ημέρας που δεν μπορεί να καλυφθεί από το διαθέσιμο απόθεμα, το σύστημα προτείνει μία ή περισσότερες εντολές (προμήθειας ή παραγωγής, ανάλογα με το είδος).

Για τον υπολογισμό της ποσότητας των προτεινόμενων εντολών, λαμβάνονται υπόψη η Μέγιστη και η Ελάχιστη Ποσότητα καθώς και η Πολλαπλάσια Ποσότητα Εντολής, ενώ αγνοούνται το Σημείο και η Ποσότητας Αναπαραγγελίας.

- **Πολιτική F (Fixed Order Quantity):** Η πολιτική αυτή λειτουργεί όπως και η πολιτική Lot for Lot, με τη διαφορά ότι με την πολιτική αυτή δημιουργείται μια εντολή με συγκεκριμένη και σταθερή ποσότητα παραγγελίας, ανεξάρτητα από την ποσότητα της ζήτησης. Αν π.χ. η σταθερή ποσότητα έχει οριστεί ίση με 100, όλες οι εντολές που δημιουργούνται είναι για 100 τεμάχια, ακόμα και αν η ζήτηση είναι μικρότερη. Αν η ζήτηση για μια ημέρα είναι μεγαλύτερη από την σταθερή ποσότητα παραγγελίας, τότε δημιουργούνται δύο ή περισσότερες εντολές για την ίδια ημέρα.

  Για τον υπολογισμό της ποσότητας των προτεινόμενων εντολών, λαμβάνεται υπόψη μόνο η Ποσότητα Αναπαραγγελίας (Reorder Quantity), ενώ αγνοούνται όλες οι υπόλοιπες παράμετροι προγραμματισμού.

- **Πολιτική P (Periodic Order Quantity):** Με την πολιτική αυτή το σύστημα αθροίζει τη ζήτηση για μια περίοδο και δημιουργεί μια ή περισσότερες εντολές για την κάλυψη της ζήτησης ολόκληρης της περιόδου. Η περίοδος ισούται με την τιμή του πεδίου Ημέρες Περιόδου (Periodic Days). Οι προτεινόμενες εντολές έχουν σαν προθεσμία την πρώτη ημέρα της εκάστοτε περιόδου.

  Για τον υπολογισμό της ποσότητας των προτεινόμενων εντολών, λαμβάνονται υπόψη η Μέγιστη και η Ελάχιστη Ποσότητα καθώς και η Πολλαπλάσια Ποσότητα Εντολής, ενώ αγνοούνται το Σημείο και η Ποσότητα Αναπαραγγελίας.

- **Πολιτική R (Time Phased Reorder Point):** Η πολιτική αυτή είναι παρόμοια με τη λειτουργία του απλού Reorder Point (όπως δηλαδή διαχειρίζονται τα υλικά με Τύπο Είδους C και D), με τη διαφορά ότι κατά την εκτέλεση του MRP ή του MPS, το σύστημα υπολογίζει το διαθέσιμο απόθεμα σε καθημερινή βάση (όπως ακριβώς και η πολιτική Lot for Lot), προτείνοντας εντολές προμήθειας ή παραγωγής, όταν το διαθέσιμο απόθεμα γίνει μικρότερο από το Σημείο Αναπαραγγελίας (Reorder Point). Η ποσότητα των προτεινόμενων εντολών ισούται με την Ποσότητα Αναπαραγγελίας (Reorder Quantity).

  Στην περίπτωση αυτή, η Μέγιστη και η Ελάχιστη Ποσότητα καθώς και η Πολλαπλάσια Ποσότητα Εντολής, αγνοούνται κατά τους υπολογισμούς.

Επίπεδο Αναπαραγγελίας (Reorder Point – ROP)

Το πεδίο αυτό συμπληρώνεται υποχρεωτικά για υλικά που παρακολουθούνται με τη λογική του απλού ROP, δηλαδή για υλικά με Τύπο Είδους C και D. Το σύστημα δημιουργεί μια λίστα για τα υλικά αυτά, όταν κατά τη χρονική στιγμή εκτέλεσης της αντίστοιχης ενέργειας, το τρέχον απόθεμα είναι μικρότερο από τη στάθμη που εκφράζει το Reorder Point

Το πεδίο αυτό είναι επίσης υποχρεωτικό για υλικά MPS ή MRP, με Πολιτική Αναπαραγγελίας R (Time Phased Reorder Point). Όταν το διαθέσιμο απόθεμα πέσει κάτω από το Σημείο Αναπαραγγελίας, το σύστημα προτείνει νέα εντολή (προμήθειας ή παραγωγής) για την αναπλήρωση του αποθέματος.

Ο τύπος δεδομένων είναι Numeric(16,3) με τιμές ≥ 0.

Ποσότητα Αναπαραγγελίας (Reorder Quantity – ROQ)

Το πεδίο αυτό συμπληρώνεται υποχρεωτικά για υλικά που παρακολουθούνται με τη λογική του απλού ROP (υλικά με Τύπο Είδους C και D) καθώς και για υλικά MPS ή MRP, με Πολιτική Αναπαραγγελίας R (Time Phased Reorder Point) ή F (Fixed Order Quantity).

Για τα υλικά τύπου απλού ROP, η ποσότητα αυτή παρατίθεται στη σχετική λίστα των υλικών, για τα οποία το τρέχον απόθεμα είναι μικρότερο από το ROP κατά τη χρονική στιγμή της εκτέλεσης της σχετικής ενέργειας του συστήματος. Για τα υλικά MPS ή MRP με Πολιτική Αναπαραγγελίας R ή F, η ποσότητα κάθε προτεινόμενης από το σύστημα εντολής, ισούται με την τιμή του πεδίου αυτού.

Ο τύπος δεδομένων είναι Numeric(16,3) με τιμές ≥ 0.

Απόθεμα Ασφαλείας (Safety Stock)

Εκφράζει την ποσότητα εκείνη του αποθέματος η οποία παραμένει ανέπαφη στην αποθήκη και δεν αναλώνεται υπό φυσιολογικές συνθήκες, παρά μόνο για την αντιμετώπιση έκτακτων αναγκών ή σε απρόβλεπτες περιπτώσεις, όπως αιφνίδιες παραγγελίες, βλάβες μηχανών, εκπρόθεσμες παραδόσεις υλικών, παραγωγή σκάρτων υλικών κλπ. Χρησιμοποιείται για τον υπολογισμό του διαθέσιμου αποθέματος σε όλες τις προαναφερθείσες Πολιτικές Αναπαραγγελίας με τρόπο ώστε το ύψος του αποθέματος ενός είδους να μην πέσει ποτέ κάτω από το Απόθεμα Ασφαλείας. Αγνοείται μόνο στην περίπτωση των υλικών που διαχειρίζονται με τη λογική του απλού ROP.

Ο τύπος δεδομένων είναι Numeric(16,3) με τιμές ≥ 0.

Ελάχιστη Ποσότητα Εντολής (Minimum Order Quantity)

Η Ελάχιστη Ποσότητα Εντολής εξασφαλίζει ότι η ποσότητα όλων των εντολών που προτείνονται από το σύστημα κατά την εκτέλεση του MPS ή του MRP, θα είναι μεγαλύτερη ή τουλάχιστον ίση με την τιμή του πεδίου αυτού. Αγνοείται στην περίπτωση των υλικών που παρακολουθούνται με τη λογική του απλού ROP, ενώ επίσης δεν λαμβάνεται υπόψη από τις Πολιτικές Αναπαραγγελίας R (Time Phased Reorder Point) και F (Fixed Order Quantity).

Ο τύπος δεδομένων είναι Numeric(16,3) με τιμές ≥ 0.

Μέγιστη Ποσότητα Εντολής (Maximum Order Quantity)

Η Μέγιστη Ποσότητα Εντολής εξασφαλίζει ότι η ποσότητα όλων των εντολών που προτείνονται από το σύστημα κατά την εκτέλεση του MPS ή του MRP, θα είναι μικρότερη ή τουλάχιστον ίση με την τιμή του πεδίου αυτού. Αγνοείται στην περίπτωση των υλικών που παρακολουθούνται με τη λογική του απλού ROP, ενώ επίσης δεν λαμβάνεται υπόψη από τις Πολιτικές Αναπαραγγελίας R (Time Phased Reorder Point) και F (Fixed Order Quantity).

Στην περίπτωση κατά την οποία, η ζήτηση είναι μεγαλύτερη και δεν μπορεί να καλυφθεί με μία εντολή η ποσότητα της οποίας ισούται με τη Μέγιστη Ποσότητα Εντολής, τότε το σύστημα θα δημιουργήσει αναγκαστικά δύο ή περισσότερες εντολές για την ικανοποίηση της ζήτησης.

Ο τύπος δεδομένων είναι Numeric(16,3) με τιμές ≥ 0.

Πολλαπλάσια Ποσότητα Εντολής (Multiple Order Quantity)

Η Πολλαπλάσια Ποσότητα Εντολής εξασφαλίζει ότι η ποσότητα όλων των εντολών που προτείνονται από το σύστημα κατά την εκτέλεση του MPS ή του MRP, θα είναι πολλαπλάσια της τιμής του πεδίου αυτού. Αγνοείται στην περίπτωση των υλικών που παρακολουθούνται με τη λογική του απλού ROP, ενώ επίσης δεν λαμβάνεται υπόψη από τις Πολιτικές Αναπαραγγελίας R (Time Phased Reorder Point) και F (Fixed Order Quantity).

Ο τύπος δεδομένων είναι Numeric(16,3) με τιμές ≥ 0.

Ημέρες Περιόδου (Periodic Days)

Ακέραιος θετικός αριθμός ο οποίος εκφράζει τις ημέρες (χρονικές περιόδους) για τις οποίες το σύστημα αθροίζει την ζήτηση ενός είδους με Πολιτική Αναπαραγγελίας P (Periodic Order Quantity) κατά την εκτέλεση του MPS ή του MRP. Οι εντολές που προτείνει το σύστημα στη συνέχεια για την ικανοποίηση της ζήτησης, καλύπτουν το άθροισμα της ζήτησης για την περίοδο αυτή.

Ο τύπος δεδομένων είναι Numeric(3,0) με τιμές ≥ 1.

Χαμηλότερο Σημείο Εμφάνισης Υλικού (Low Level Code – LLC)

Ακέραιος θετικός αριθμός ο οποίος εκφράζει το χαμηλότερο σημείο (στάδιο παραγγελίας) στο οποίο συναντάται ένα υλικό, στα δέντρα κατασκευής όλων των προϊόντων στα οποία συμμετέχει. Υπολογίζεται από το σύστημα και αποθηκεύεται στο Κύριο Αρχείο Ειδών, κάθε φορά πριν την εκτέλεση του MRP, από το οποίο χρησιμοποιείται στη συνέχεια για την αποφυγή πολλαπλών υπολογισμών για το ίδιο υλικό.

Ο τύπος δεδομένων είναι Numeric(3,0) με τιμές ≥ 0. Η τιμή 0 αναφέρεται στο υψηλότερο επίπεδο, δηλαδή υποδηλώνει ότι πρόκειται για τελικά προϊόντα.

Βαθμός Απόδοσης (Quantity Yield)

Συντελεστής ο οποίος εκφράζει τη μέση ποσοστιαία απόδοση κατά την παραγωγή ή την προμήθεια ενός είδους. Χρησιμοποιείται για τον υπολογισμό της ποσότητας όλων των προτεινόμενων από το σύστημα εντολών, διαιρώντας τη ζήτηση (απαίτηση) του είδους με το βαθμό απόδοσής του,

Για παράδειγμα, αν για ένα είδος ο μέσος όρος εμφάνισης σκάρτων κατά την παραγωγή του είναι 5%, τότε ο βαθμός απόδοσης του είδους είναι 95%. Αυτό σημαίνει ότι σε μια παρτίδα παραγωγής 100 τεμαχίων, αναμένεται τα 95 να είναι αποδεκτά και τα 5 σκάρτα.

Ο τύπος δεδομένων είναι Numeric(6,3) με τιμές > 0 και ≤ 100.

Χρόνος Αναμονής (Lead Time)

Ο Χρόνος Αναμονής είναι ο χρόνος σε ημέρες που απαιτείται για την ολοκλήρωση της παραγωγής ή της προμήθειας μιας μέσης παρτίδας για το συγκεκριμένο είδος. Ο χρόνος αυτός χρησιμοποιείται από το σύστημα για τον υπολογισμό της ημερομηνίας έναρξης μιας εντολής, δηλαδή το πότε θα εκδοθεί μια εντολή, ώστε να ολοκληρωθεί έγκαιρα σύμφωνα με την ημερομηνία παράδοσης (προθεσμία) της εντολής. Για τον υπολογισμό αυτό είναι προφανές ότι από την επιθυμητή ημερομηνία παράδοσης της εντολής, αφαιρείται ο Χρόνος Αναμονής.

Ο τύπος δεδομένων είναι Numeric(3,0) με τιμές ≥ 0.

Ποσοστό Παραγωγής Σκάρτων (Scrap Factor)

Συντελεστής ο οποίος αναφέρεται σε μια συγκεκριμένη σχέση Συναρμολογήματος – Συστατικού ενός Πίνακα Υλικών και εκφράζει το μέσο ποσοστό δημιουργίας σκάρτων του Συστατικού κατά την συναρμολόγηση/κατασκευή του Συναρμολογήματος. Λαμβάνεται υπόψη κατά τον υπολογισμό της εξαρτημένης ζήτησης του Συστατικού που προκύπτει από τις απαιτήσεις του συγκεκριμένου Συναρμολογήματος, για την κάλυψη των σκάρτων τεμαχίων που εμφανίζονται κατά την παραγωγική διαδικασία.

Για παράδειγμα, έστω ότι ένα είδος Β αποτελεί συστατικό του είδους Α και η μεταξύ τους σχέση στον Πίνακα Υλικών έχει Ποσοστό Παραγωγής Σκάρτων 5% (το ποσοστό αυτό ισχύει μόνο για τη σχέση ανάμεσα στον κωδικό Β και τον κωδικό Α). Εάν το είδος Α έχει μια απαίτηση για 100 τεμάχια, τότε η απαίτηση για το είδος Β υπολογίζεται σε 105 τεμάχια. Στην περίπτωση κατά την οποία το είδος Β έχει επίσης Βαθμό Απόδοσης 90%, τότε η συνολική απαίτηση του είδους αυτού ανέρχεται σε 105/0.9, δηλαδή 117 τεμάχια.

Ο τύπος δεδομένων είναι Numeric(6,3) με τιμές ≥ 0 και < 100.

Ποσότητα Συστατικού (Component Per Assembly)

Αναφέρεται σε μια συγκεκριμένη δομή ενός Πίνακα Υλικών και αποτελεί την απαιτούμενη ποσότητα ενός Συστατικού για τη δημιουργία μιας μονάδας του Συναρμολογήματος. Η ποσότητα αυτή εκφράζεται πάντα στην κύρια μονάδα μέτρησης του Συστατικού και αναφέρεται επίσης στην κύρια μονάδα μέτρησης του Συναρμολογήματος.

Ο τύπος δεδομένων είναι Numeric(16,3) με τιμές ≥ 0.

Επαναπρογραμματισμός Εντολών (Rescheduling)

Η εκτέλεση τόσο του MPS όσο και του MRP μπορεί να γίνει με δύο τρόπους :

(α) Χωρίς επαναπρογραμματισμό των εντολών. Στην περίπτωση αυτή, πριν την έναρξη των υπολογισμών, το σύστημα διαγράφει όλες τις υπάρχουσες Προτεινόμενες εντολές (Προγραμματισμένες και Εγκεκριμένες) καθώς και τις αντίστοιχες εντολές Εξαρτημένης Ζήτησης, ενώ οι ήδη Εκδοθείσες (σε εξέλιξη) εντολές παραμένουν ανέπαφες. Για την κάλυψη της ζήτησης που δεν ικανοποιείται από το διαθέσιμο απόθεμα, προτείνονται νέες εντολές.

(β) Με επαναπρογραμματισμό των εντολών. Στην περίπτωση αυτή, πριν την έναρξη των υπολογισμών, το σύστημα διαγράφει όλες τις Προγραμματισμένες εντολές καθώς και τις αντίστοιχες εντολές Εξαρτημένης Ζήτησης, ενώ οι ήδη Εκδοθείσες (σε εξέλιξη) εντολές παραμένουν ανέπαφες). Για την κάλυψη της ζήτησης που δεν ικανοποιείται από το διαθέσιμο απόθεμα, το σύστημα ελέγχει αν μπορεί να καλυφθεί με επιτάχυνση ή επιβράδυνση των υφιστάμενων Εγκεκριμένων εντολών. Αν κάτι τέτοιο δεν είναι εφικτό, προτείνονται νέες εντολές. Εγκεκριμένες εντολές οι οποίες δεν είναι πλέον απαραίτητες, διαγράφονται (μαζί με τις αντίστοιχες εντολές Εξαρτημένης Ζήτησης) .

Τύπος Εντολής (Order Type)

Διψήφιος κωδικός, ο οποίος χαρακτηρίζει το είδος μιας εντολής.

Ο τύπος δεδομένων (Data Type) είναι Char(2) και οι τιμές που μπορεί να πάρει είναι (δίνονται από το σύστημα) :

SO : Εντολές Πώλησης

PO : Εντολές Προμήθειας

FO : Εντολές Πρόγνωσης Ζήτησης

MS : Εντολές Παραγωγής Τελικών Προϊόντων

MF : Εντολές Παραγωγής Ημιέτοιμων Υλικών

RQ : Εντολές Εξαρτημένης Ζήτησης

Κατάσταση Εντολής (Order Status)

Μονοψήφιος κωδικός, ο οποίος χαρακτηρίζει την κατάσταση μιας εντολής.

Ο τύπος δεδομένων (Data Type) είναι Char(1) και οι τιμές που μπορεί να πάρει είναι (δίνονται από το σύστημα) :

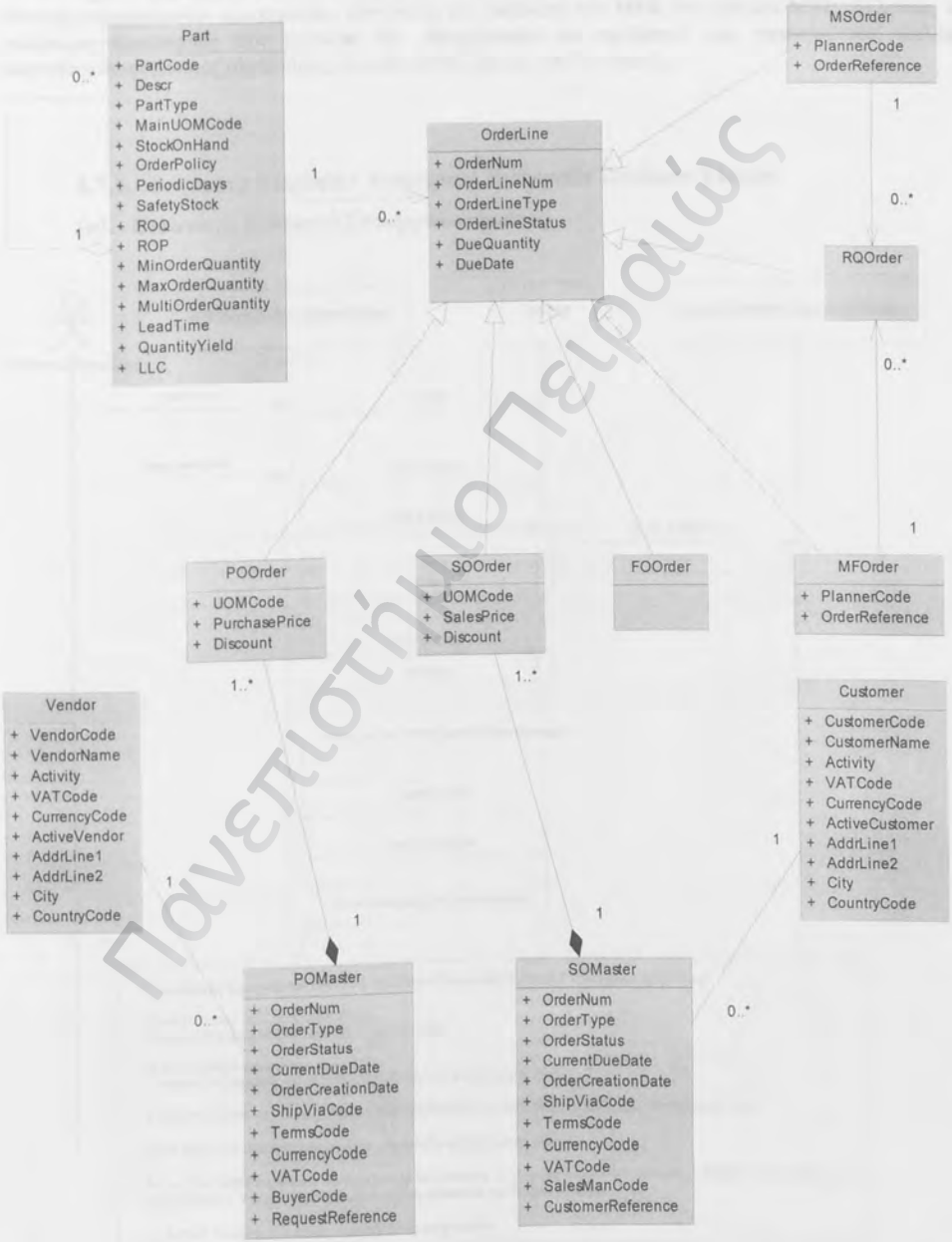
1 : Προγραμματισμένη Εντολή
2 : Εγκεκριμένη Εντολή
} Προτεινόμενες Εντολές

3 : Εκδοθείσα Εντολή (σε εξέλιξη)

4 : Κλειστή Εντολή (ολοκληρωμένη ή ακυρωθείσα)

3.6. Domain Model

Στο διάγραμμα που ακολουθεί εμφανίζεται το Domain Model, το σημαντικότερο παραδοτέο που κατασκευάζεται κατά τη διάρκεια της ανάλυσης, όπου αναπαριστώνται γραφικά όλες οι βασικές έννοιες – κλάσεις του φυσικού προβλήματος, οι κύριες ιδιότητές τους καθώς και οι σχέσεις μεταξύ των κλάσεων. Οι πληροφορίες που απεικονίζονται προκύπτουν τόσο από τη μελέτη των Use Cases, όσο και από το θεωρητικό μέρος της εφαρμογής, το οποίο παρουσιάστηκε στο προηγούμενο κεφάλαιο.



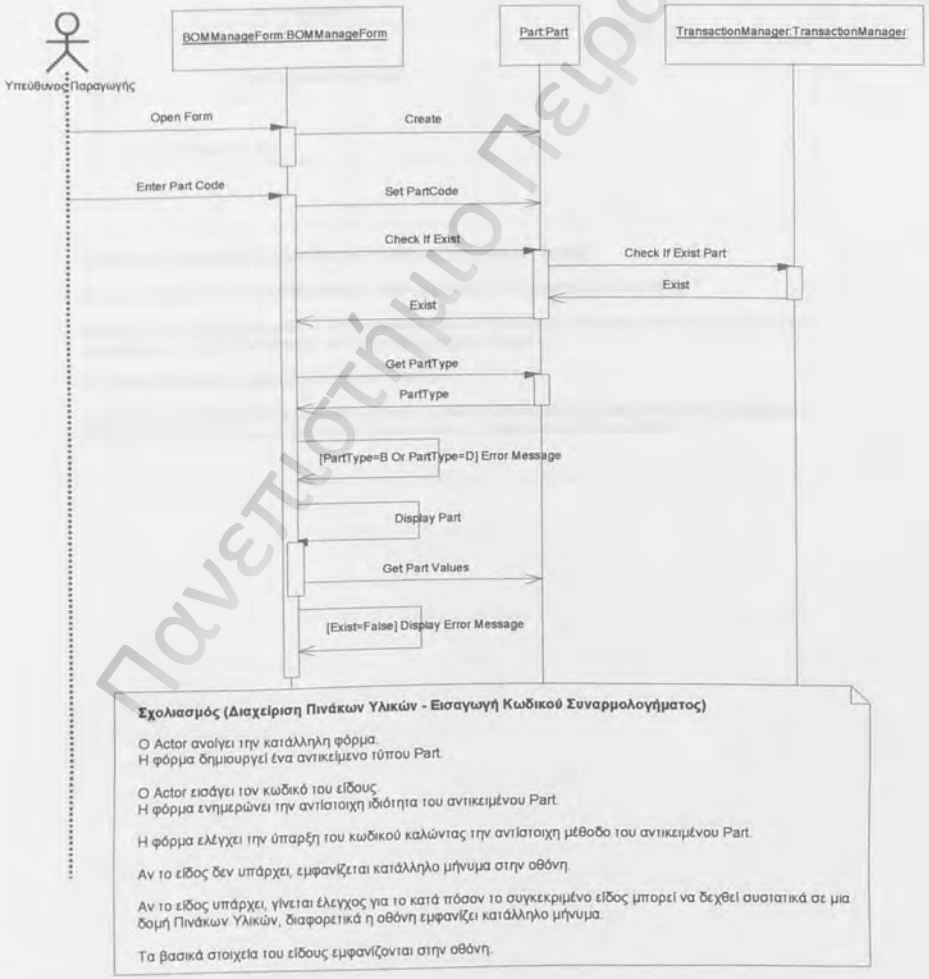
3.7. Design Model – Υλοποίηση των Use Cases (Use Cases Realization)

Στην ενότητα αυτή παρουσιάζονται τα Sequence Diagrams, στα οποία εμφανίζονται τα προγραμματιστικά αντικείμενα της εφαρμογής και τα διάφορα μηνύματα που ανταλλάσσουν μεταξύ τους. Τα Sequence Diagrams απεικονίζουν τον τρόπο σχεδιασμού των αντικειμένων, με σκοπό την ικανοποίηση των απαιτήσεων που καταγράφηκαν στα Use Cases.

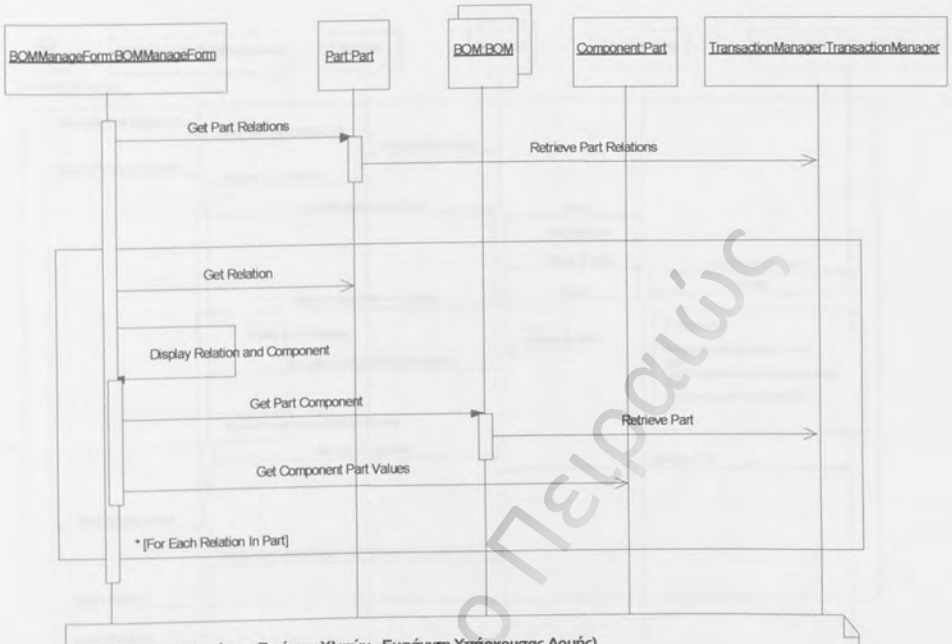
Έχουν υλοποιηθεί όλα τα σενάρια των Use Cases του συστήματος, με μόνη εξαίρεση τον επαναπρογραμματισμό των εντολών, τόσο κατά την εκτέλεση του MPS, όσο και του MRP. Άλλωστε, ο επαναπρογραμματισμός των εντολών δεν διαφοροποιεί το σχεδιασμό των αντικειμένων, απλώς απαιτείται διαφορετικός αλγόριθμος για τους κατάλληλους υπολογισμούς.

3.7.1. Sequence Diagram : Διαχείριση Βαθμικών Πινάκων Υλικών

(α). Εισαγωγή Κωδικού Συναρμολογήματος



(β). Εμφάνιση Υπάρχουσας Δομής



Σχολιασμός (Διαχείριση Πινάκων Υλικών - Εμφάνιση Υπάρχουσας Δομής)

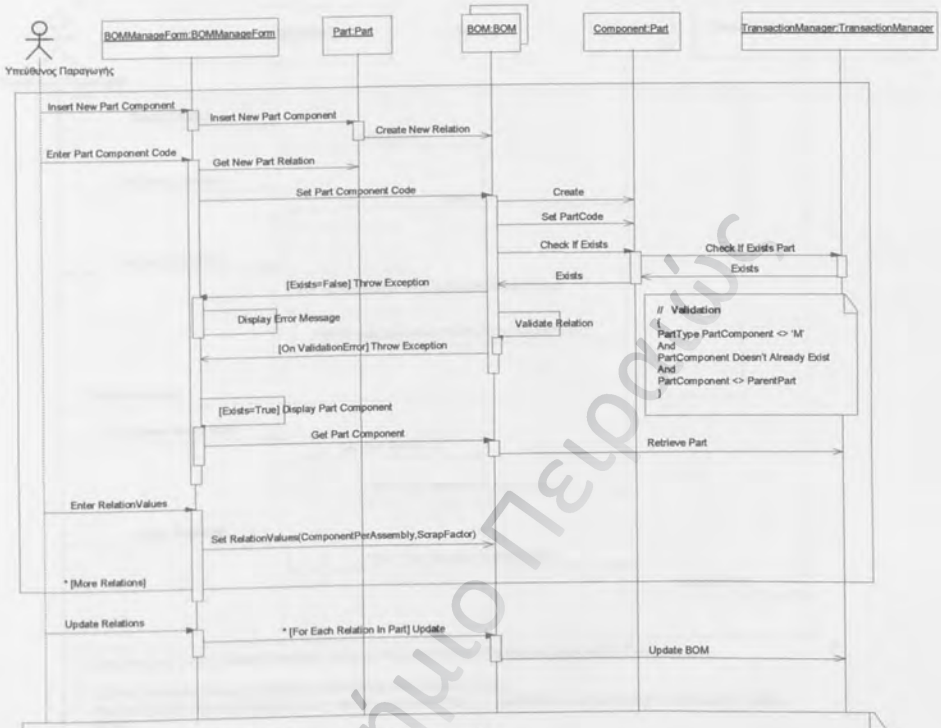
Μετά την εισαγωγή του συναρμολογήματος, η οθόνη εμφανίζει την υπάρχουσα δομή του είδους.

Η φόρμα καλεί την αντίστοιχη μέθοδο του αντικειμένου Part, με την οποία ανακτώνται τα άμεσα συστατικά του είδους (ενημέρωση του multi-object BOM με τα στοιχεία της δομής του είδους).

Για κάθε συστατικό που συμμετέχει στη δομή του είδους :

Εμφανίζονται στην οθόνη τόσο τα στοιχεία της δομής (γραμμές multi-object BOM) όσο και τα βασικά στοιχεία κάθε συστατικού (αντικείμενο Component τύπου Part που αποτελεί ιδιότητα του αντικειμένου BOM).

(γ). Εισαγωγή Συστατικών



Σχολιασμός (Διαχείριση Πινάκων Υλικών - Εισαγωγή Συστατικών)

Ο Actor επιλέγει εισαγωγή συστατικού στον Πίνακα Υλικών.
Η φόρμα καλεί την αντίστοιχη μέθοδο του αντικειμένου Part, με την οποία εισάγεται μια νέα γραμμή στο multi-object BOM.

Ο Actor εισάγει τον κωδικό του συστατικού.
Πριν την ενημέρωση της αντίστοιχης ιδιότητας της νέας γραμμής του multi-object BOM, ελέγχεται μέσω του αντικείμενου Component τύπου Part, η ύπαρξη του κωδικού είδους του συστατικού καθώς και εάν το είδος αυτό μπορεί να συμμετέχει σαν συστατικό στη δομή του Πίνακα Υλικών.

Σε διαφορετική περίπτωση το αντικείμενο multi-object BOM δημιουργεί εξαίρεση μέσω της οποίας εμφανίζεται κατάλληλο μήνυμα στην οθόνη.

Εμφανίζονται στην οθόνη τα βασικά στοιχεία του συστατικού.

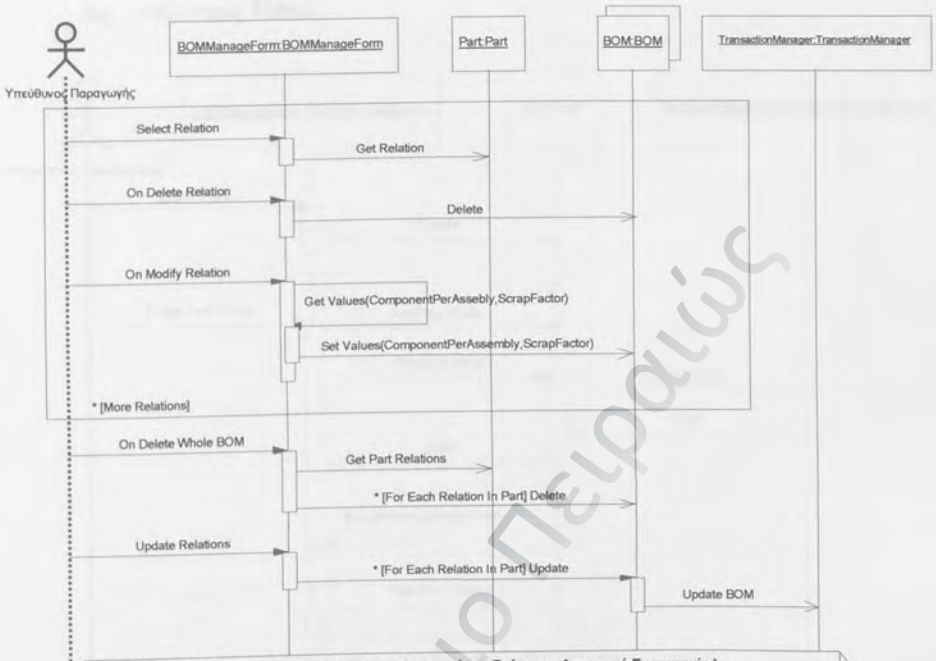
Ο Actor εισάγει τις απαραίτητες πληροφορίες της σχέσης και η οθόνη ενημερώνει τις αντίστοιχες ιδιότητες του αντικειμένου multi-object BOM.

Η ανωτέρω διαδικασία επαναλαμβάνεται μέχρι να καταχωρηθούν όλα τα συστατικά του συναρμολογήματος.

Ο Actor επιλέγει την αποθήκευση του Πίνακα Υλικών.

Η οθόνη καλεί την αντίστοιχη μέθοδο του αντικειμένου multi-object BOM.

(δ). Τροποποίηση Σχέσεων, Διαγραφή Συστατικών



Σχολιασμός (Διαχείριση Πινάκων Υλικών - Τροποποίηση Σχέσεων, Διαγραφή Συστατικών)

Ο Actor επιλέγει ένα συγκεκριμένο συστατικού του Πίνακα Υλικών.
Η φόρμα καλεί την αντίστοιχη μέθοδο του αντικειμένου Part, με την οποία επιστρέφεται ένα αντικείμενο τύπου BOM.

Ο Actor επιλέγει τη διαγραφή του συστατικού.
Η οθόνη καλεί την αντίστοιχη μέθοδο του αντικειμένου BOM μέσω της οποίας μαρκάρεται προς διαγραφή το συστατικό.

Ο Actor τροποποιεί τα δεδομένα του συστατικού.
Η οθόνη ενημερώνει τις αντίστοιχες ιδιότητες του αντικειμένου BOM.

Η ανωτέρω διαδικασία επαναλαμβάνεται μέχρι να καταχωρηθούν όλα οι επιθυμητές αλλαγές στον Πίνακα Υλικών.

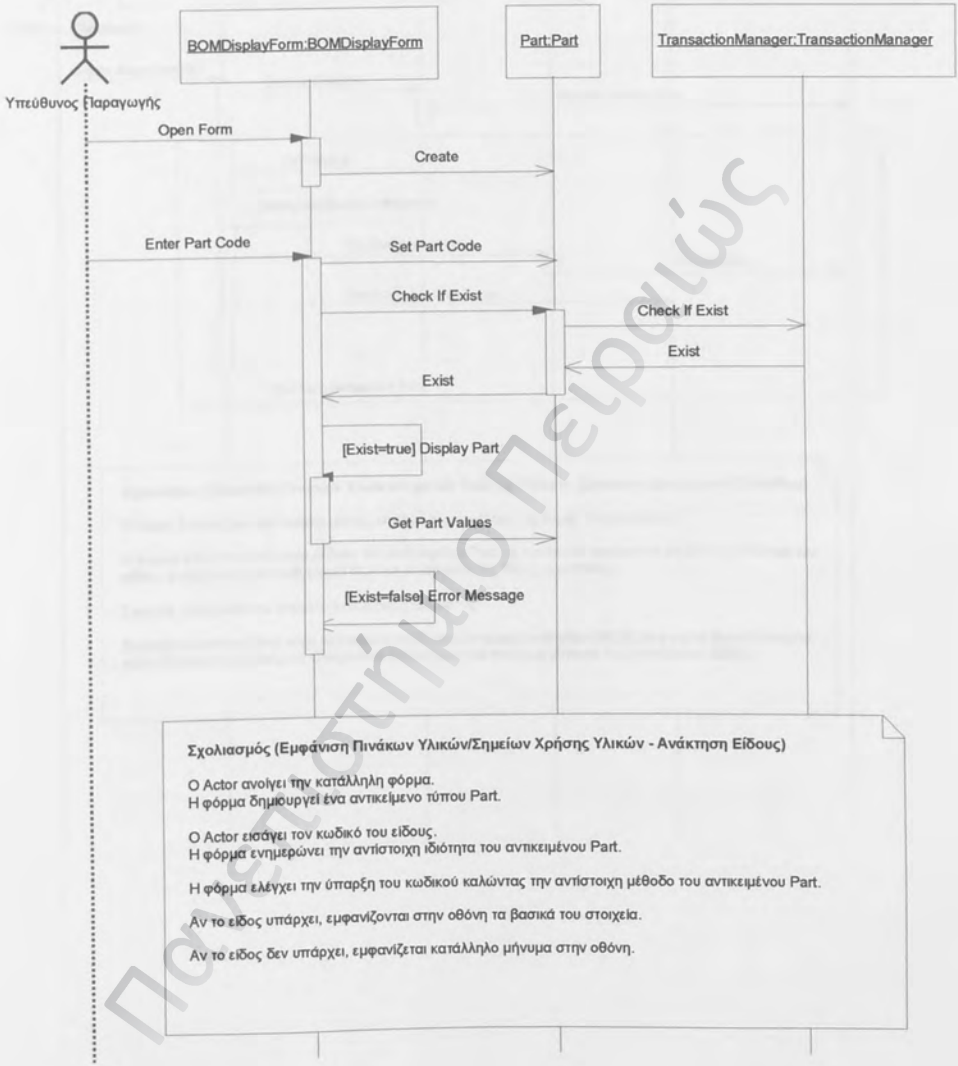
Ο Actor επιλέγει τη διαγραφή όλου του υπάρχοντα Πίνακα Υλικών.
Η οθόνη μαρκάρει προς διαγραφή όλα τα συστατικά του Πίνακα Υλικών, καλώντας την αντίστοιχη μέθοδο του αντικειμένου BOM.

Ο Actor επιλέγει την αποθήκευση των δεδομένων.

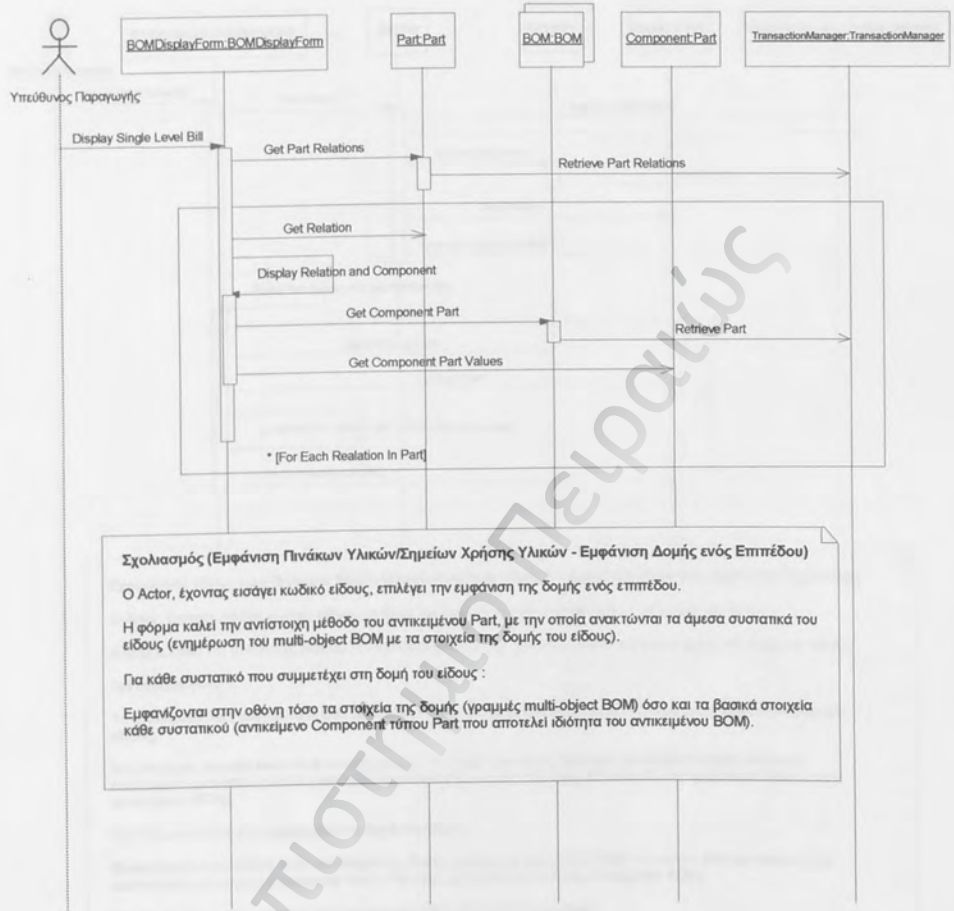
Η οθόνη καλεί την αντίστοιχη μέθοδο του αντικειμένου multi-object BOM.

3.7.2. Sequence Diagram : Εμφάνιση Δομής Προϊόντων/Σημείων Χρήσης Υλικών

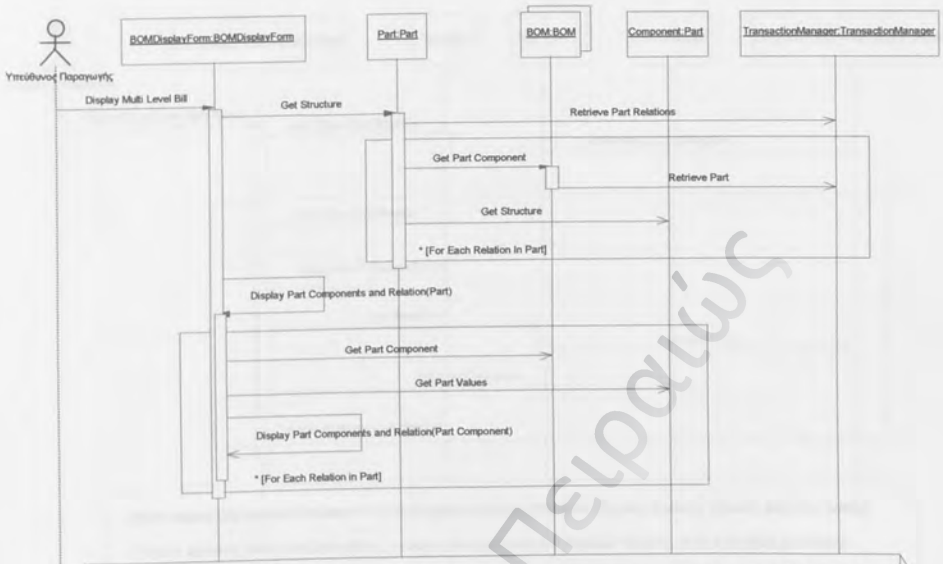
(α). Ανάκτηση Είδους



(β). Εμφάνιση Δομής Ενός Επιπέδου



(γ). Εμφάνιση Όλων των Επιπέδων της Δομής



Σχολιασμός (Εμφάνιση Πινάκων Υλικών/Σημείων Χρήσης Υλικών - Εμφάνιση όλων των Επιπέδων της Δομής)

Ο Actor, έχοντας εισάγει κωδικό είδους, επιλέγει την εμφάνιση όλων των επιπέδων της δομής του είδους.

Η φόρμα καλεί την αντίστοιχη μέθοδο του αντικειμένου Part, με την οποία ανακτάται όλη η ιεραρχική δομή του είδους.

Πιο συγκεκριμένα :

Το αντικείμενο Part, ανακτά τα άμεσα συστατικά του (ενημέρωση του multi-object BOM με τα στοιχεία της δομής του είδους).

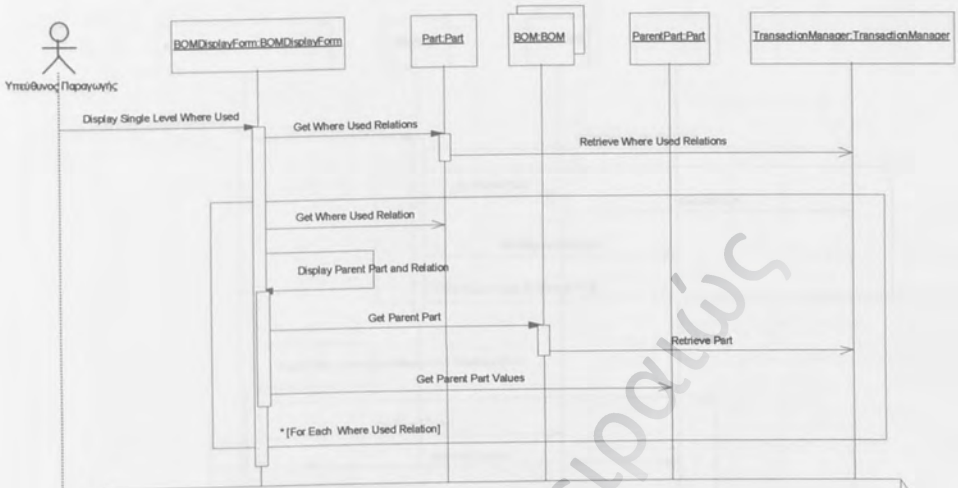
Στη συνέχεια, για κάθε συστατικό που συμμετέχει στη δομή του είδους (γραμμές multi-object BOM), εκτελείται αναδρομικά η ανωτέρω μέθοδος για κάθε συστατικό (αντικείμενο Component τύπου Part που αποτελεί ιδιότητα του αντικειμένου BOM).

Για κάθε συστατικό που συμμετέχει στη δομή του είδους :

Εμφανίζονται στην οθόνη τόσο τα στοιχεία της δομής (γραμμές multi-object BOM) όσο και τα βασικά στοιχεία κάθε συστατικού (αντικείμενο Component τύπου Part που αποτελεί ιδιότητα του αντικειμένου BOM).

Αναδρομικά εμφανίζονται τα αντίστοιχα στοιχεία κάθε συστατικού του είδους.

(δ). Σημεία Χρήσης Υλικού, Άμεσοι Γονείς



Σχολιασμός (Εμφάνιση Πινάκων Υλικών/Σημείων Χρήσης Υλικών - Σημεία Χρήσης Υλικού, Άμεσοι Γονείς)

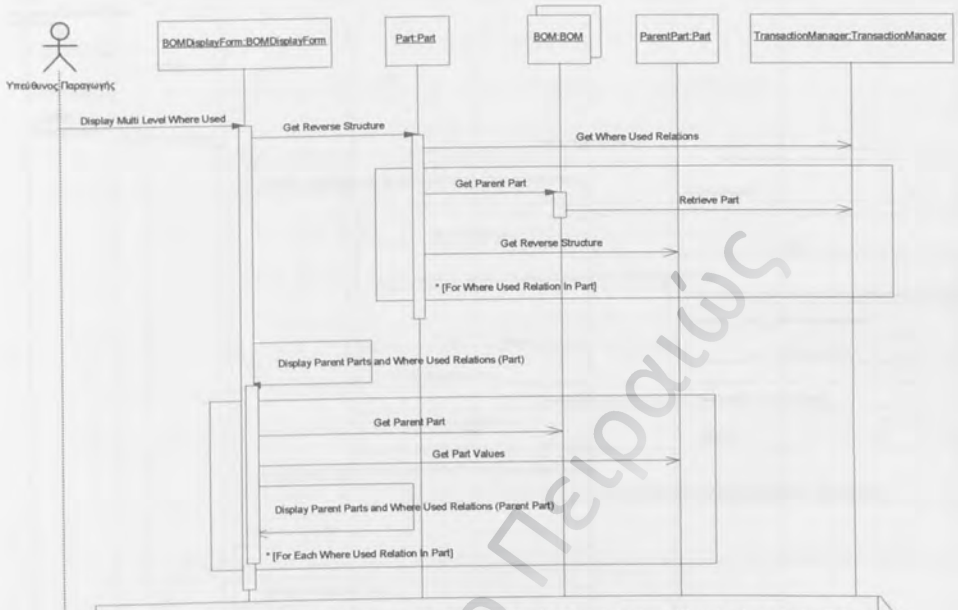
Ο Actor, έχοντας εισάγει κωδικό είδους, επιλέγει την εμφάνιση των Σημείων Χρήσης ενός επιπέδου του είδους.

Η φόρμα καλεί την αντίστοιχη μέθοδο του αντικειμένου Part, με την οποία ανακτώνται οι άμεσοι γονείς του είδους (ενημέρωση του multi-object BOM με τα στοιχεία της αντίστροφης δομής του είδους).

Για κάθε Γονέα του είδους :

Εμφανίζονται στην οθόνη τόσο τα στοιχεία της δομής (γραμμές multi-object BOM) όσο και τα βασικά στοιχεία κάθε Γονέα (αντικείμενο ParentPart τύπου Part που αποτελεί ιδιότητα του αντικειμένου BOM).

(ε). Σημεία Χρήσης Υλικού, Όλα τα Επίπεδα



Σχολιασμός (Εμφάνιση Πινάκων Υλικών/Σημείων Χρήσης Υλικών - Σημεία Χρήσης Υλικού, Όλα τα Επίπεδα)

Ο Actor, έχοντας εισάγει κωδικό είδους, επιλέγει την εμφάνιση των Σημείων Χρήσης όλων των επιπέδων του είδους.

Η φόρμα καλεί την αντίστοιχη μέθοδο του αντικειμένου Part, με την οποία ανακτώνται όλοι οι γονείς του είδους, μέχρι την κορυφή της κάθε ιεραρχικής δομής στις οποίες το είδος αυτό συμμετέχει.

Πιο συγκεκριμένα :

Το αντικείμενο Part, ανακτά τους άμεσους γονείς του (ενημέρωση του multi-object BOM με τα στοιχεία της αντίστροφης δομής του είδους).

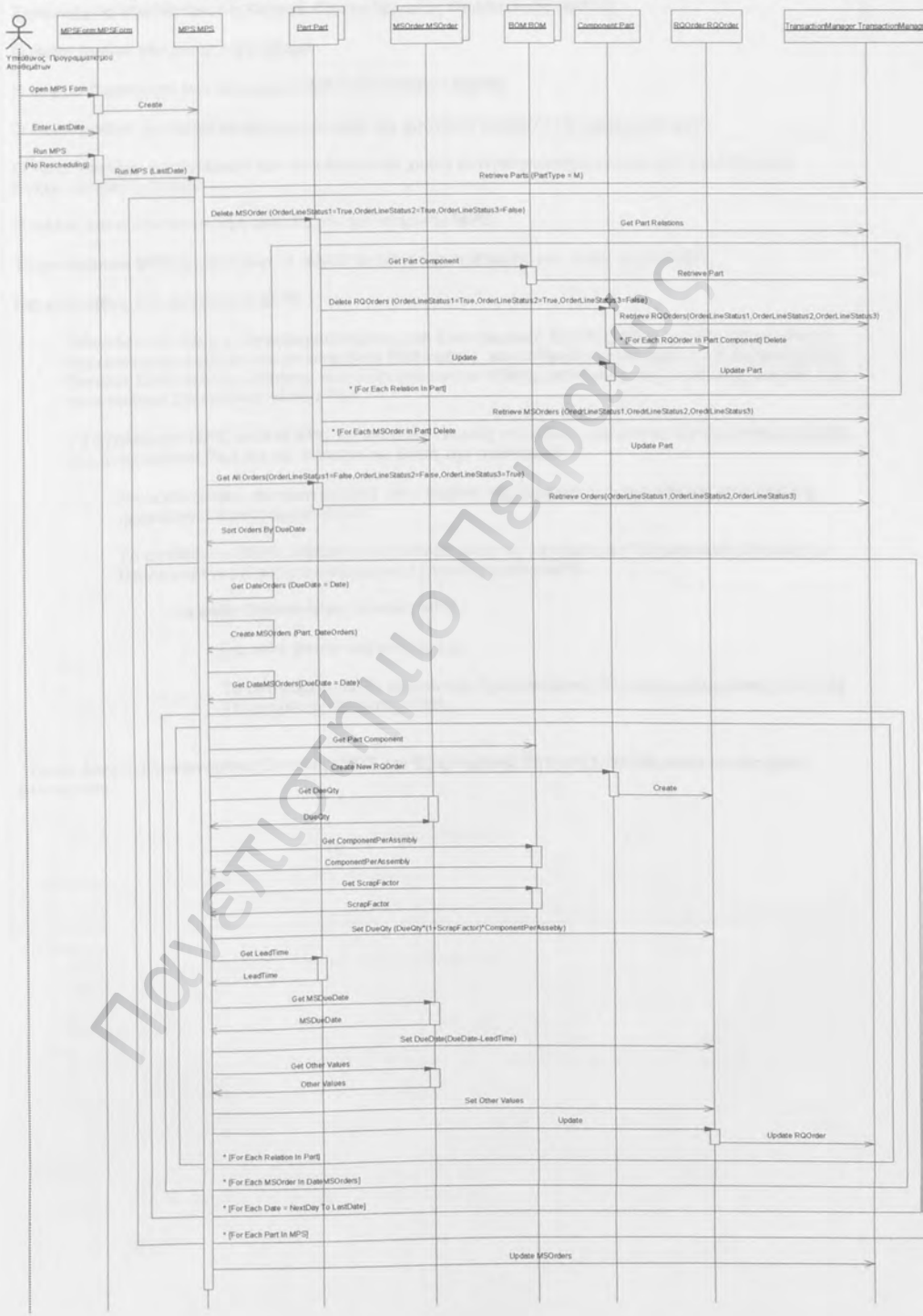
Στη συνέχεια, για κάθε γονέα που συμμετέχει στην αντίστροφη δομή του είδους (γραμμές multi-object BOM), εκτελείται αναδρομικά η ανωτέρω μέθοδος για κάθε γονέα (αντικείμενο ParentPart τύπου Part που αποτελεί ιδιότητα του αντικειμένου BOM).

Για κάθε γονέα που συμμετέχει στην αντίστροφη δομή του είδους :

Εμφανίζονται στην οθόνη τόσο τα στοιχεία της δομής (γραμμές multi-object BOM) όσο και τα βασικά στοιχεία κάθε γονέα (αντικείμενο ParentPart τύπου Part που αποτελεί ιδιότητα του αντικειμένου BOM).

Αναδρομικά εμφανίζονται τα αντίστοιχα στοιχεία κάθε γονέα του είδους.

3.7.3. Sequence Diagram : Κατάστρωση Κύριου Προγράμματος Παραγωγής (MPS)



Σχολιασμός (Κατάστρωση Κύριου Προγράμματος Παραγωγής (MPS))

Ο Actor ανοίγει την αντίστοιχη φόρμα.

Η φόρμα δημιουργεί ένα αντικείμενο MPS (Controller Object).

Ο Actor εισάγει την καταληκτική ημερομηνία του χρονικού ορίζοντα Προγραμματισμού.

Ο Actor επιλέγει την εκτέλεση των υπολογισμών χωρίς επαναπρογραμματισμό των υφιστάμενων Εγκεκριμένων Εντολών.

Η οθόνη καλεί την αντίστοιχη μέθοδο του αντικειμένου MPS.

Το αντικείμενο MPS ανακτά όλα τα τελικά προϊόντα (ενημέρωση του multi-object Part).

Για κάθε είδος στο αντικείμενο MPS :

Διαγράφονται όλες οι Προγραμματισμένες και Εγκεκριμένες Εντολές MPS του είδους καλώντας την αντίστοιχη μέθοδο του αντικειμένου Part καθώς, και οι Προγραμματισμένες και Εγκεκριμένες Εντολές Εξαρτημένης Ζήτησης των συστατικών του είδους, καλώντας την αντίστοιχη μέθοδο του αντικειμένου Component τύπου Part.

Το αντικείμενο MPS ανακτά όλες τις ανοιχτές εντολές του είδους καλώντας την αντίστοιχη μέθοδο του αντικειμένου Part και τις ταξινομεί με βάση την προθεσμία.

Για κάθε ημέρα, ξεκινώντας από την επομένη της τρέχουσας μέχρι και την καταληκτική ημερομηνία προγραμματισμού :

Το αντικείμενο MPS, λαμβάνοντας υπόψη όλες τις απαραίτητες πληροφορίες δημιουργεί Προτεινόμενες (Προγραμματισμένες) Εντολές τύπου MPS.

Για κάθε Προτεινόμενη Εντολή MPS :

Για κάθε συστατικό του είδους :

Το αντικείμενο MPS δημιουργεί Προτεινόμενες (Προγραμματισμένες) Εντολές Εξαρτημένης Ζήτησης (RQ).

Τελικά, όλες οι Προτεινόμενες Εντολές (MPS και Εξαρτημένης Ζήτησης), αποθηκεύονται στη βάση δεδομένων.

// Υπολογισμός Ποσότητας Προτεινόμενων Εντολών (σε μορφή ψευδοκώδικα)

// 1. Part.OrderPolicy = "L" – Lot for Lot

```
{
AvailableStock = Part.StockOnHand - Part.SafetyStock
For Each Date = NextDate To LastDate
AvailableStock = AvailableStock + Σ(MSOrders.DueQty) - MAX ( Σ(SOOrderLines.DueQty), Σ(FOOrders.DueQty) )

If AvailableStock < 0 Then
   SuggestedOrderQty = ABS(AvailableStock) / Part.QtyYield * 100
   If SuggestedOrderQty <  Part.MinOrderQty Then
      DueQty = CEILING(Part.MinOrderQty / Part.MultiOrderQty) * Part.MultiOrderQty
      CreateNewMSOrder()
      MSOrder.DueQty = DueQty
   ElseIf SuggestedOrderQty >=  Part.MinOrderQty And SuggestedOrderQty <=  Part.MaxOrderQty Then
      DueQty = CEILING(SuggestedOrderQty / Part.MultiOrderQty) * Part.MultiOrderQty
      CreateNewMSOrder()
      MSOrder.DueQty = DueQty
   Else
      NumOfOrders = (SuggestedOrderQty \ Part.MaxOrderQty)        ' INTEGER DIVISION
      BalanceQty = SuggestedOrderQty
      For i = 1 To NumOfOrders
         DueQty = CEILING(Part.MaxOrderQty / Part.MultiOrderQty) * Part.MultiOrderQty
         BalanceQty = BalanceQty - DueQty
         CreateNewMSOrder()
         MSOrder.DueQty = DueQty
      End For
      If BalanceQty > 0 Then
         If BalanceQty <  Part.MinOrderQty Then
            DueQty = CEILING(Part.MinOrderQty / Part.MultiOrderQty) * Part.MultiOrderQty
            CreateNewMSOrder()
            MSOrder.DueQty = DueQty
         ElseIf BalanceQty >=  Part.MinOrderQty And BalanceQty <  Part.MaxOrderQty Then
            DueQty = CEILING(BalanceQty / Part.MultiOrderQty) * Part.MultiOrderQty
            CreateNewMSOrder()
            MSOrder.DueQty = DueQty
         End If
      End If
   End If
   For Each NewMSOrder In DateMSOders(Date)
      AvailableStock = AvailableStock + NewMSOrder.DueQty
   End For
End If
End For
}
```

// 2. Part.OrderPolicy = "F" – Fixed Order Quantity

```
{
AvailableStock = Part.StockOnHand - Part.SafetyStock
For Each Date = NextDate To LastDate
AvailableStock = AvailableStock + Σ(MSOrders.DueQty) - MAX ( Σ(SOOrderLines.DueQty), Σ(FOOrders.DueQty) )

If AvailableStock < 0 Then
   SuggestedOrderQty = ABS(AvailableStock) / Part.QtyYield * 100
   If SuggestedOrderQty <=  Part.ROQ Then
      DueQty = Part.ROQ
      CreateNewMSOrder()
      MSOrder.DueQty = DueQty
   Else
      NumOfOrders = (SuggestedOrderQty \ Part.ROQ) + 1        ' INTEGER DIVISION
      For i = 1 To NumOfOrders
         DueQty = Part.ROQ
         CreateNewMSOrder()
         MSOrder.DueQty = DueQty
      End For
   End If
   For Each NewMSOrder In DateMSOders(Date)
      AvailableStock = AvailableStock + NewMSOrder.DueQty
   End For
End If
End For
}
```

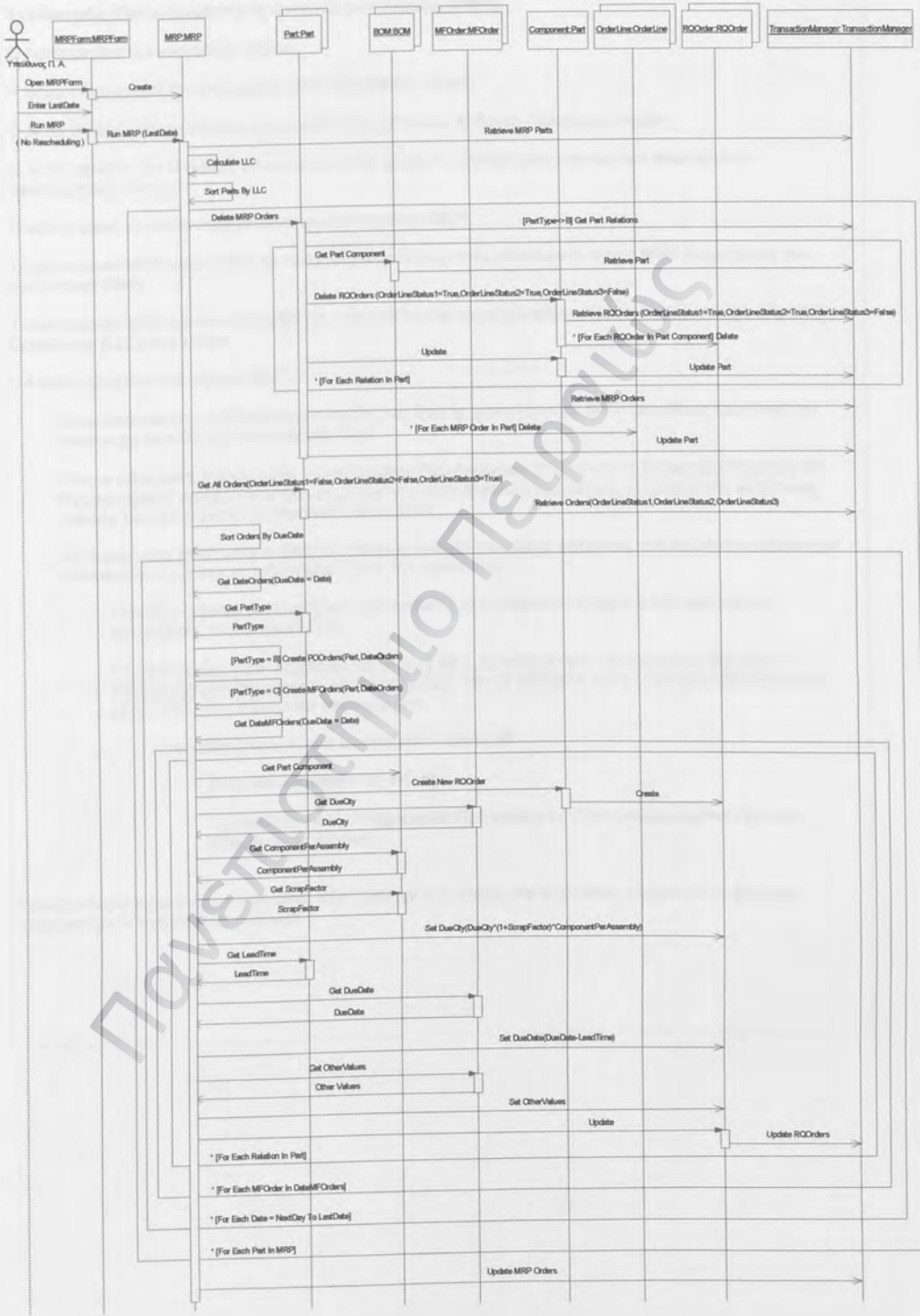
```
// 3. Part.OrderPolicy = "R" – Time Phased Reorder Point
{
AvailableStock = Part.StockOnHand - Part.SafetyStock
For Each Date = NextDate To LastDate
AvailableStock = AvailableStock + Σ(MSOrders.DueQty) - MAX ( Σ(SOOrderLines.DueQty), Σ(FOOrders.DueQty) )

If AvailableStock < Part.ROP Then
   SuggestedOrderQty = ABS(AvailableStock) / Part.QtyYield * 100
   If SuggestedOrderQty <= Part.ROQ Then
      DueQty = Part.ROQ
      CreateNewMSOrder()
      MSOrder.DueQty = DueQty
   Else
      NumOfOrders = (SuggestedOrderQty \ Part. ROQ) + 1        ' INTEGER DIVISION
      For i = 1 To NumOfOrders
         DueQty = Part.ROQ
         CreateNewMSOrder()
         MSOrder.DueQty = DueQty
      End For
   End If
   For Each NewMSOrder In DateMSOders(Date)
      AvailableStock = AvailableStock + NewMSOrder.DueQty
   End For
End If
End For
}

// 4. Part.OrderPolicy = "P" – Periodic Order Quantity
{
AvailableStock = Part.StockOnHand - Part.SafetyStock
For Each Date = NextDate To LastDate
   For Each PeriodDay = Date To (Date + Part.PeriodicDays)
      AvailableStock=AvailableStock + Σ(MSOrders.DueQty) - MAX (Σ(SOOrderLines.DueQty),Σ(FOOrders.DueQty))
   End For

If AvailableStock < 0 Then
   SuggestedOrderQty = ABS(AvailableStock) / Part.QtyYield * 100
   If SuggestedOrderQty < Part.MinOrderQty Then
      DueQty = CEILING(Part.MinOrderQty / Part.MultiOrderQty) * Part.MultiOrderQty
      CreateNewMSOrder()
      MSOrder.DueQty = DueQty, MSOrder.DueDate = Date
   ElseIf SuggestedOrderQty >= Part.MinOrderQty And SuggestedOrderQty <= Part.MaxOrderQty Then
      DueQty = CEILING(SuggestedOrderQty / Part.MultiOrderQty) * Part.MultiOrderQty
      CreateNewMSOrder()
      MSOrder.DueQty = DueQty, MSOrder.DueDate = Date
   Else
      NumOfOrders = (SuggestedOrderQty \ Part.MaxOrderQty)        ' INTEGER DIVISION
      BalanceQty = SuggestedOrderQty
      For i = 1 To NumOfOrders
         DueQty = CEILING(Part.MaxOrderQty / Part.MultiOrderQty) * Part.MultiOrderQty
         BalanceQty = BalanceQty - DueQty
         CreateNewMSOrder()
         MSOrder.DueQty = DueQty, MSOrder.DueDate = Date
      End For
      If BalanceQty > 0 Then
         If BalanceQty < Part.MinOrderQty Then
            DueQty = CEILING(Part.MinOrderQty / Part.MultiOrderQty) * Part.MultiOrderQty
            CreateNewMSOrder()
            MSOrder.DueQty = DueQty, MSOrder.DueDate = Date
         ElseIf BalanceQty >= Part.MinOrderQty And BalanceQty < Part.MaxOrderQty Then
            DueQty = CEILING(BalanceQty / Part.MultiOrderQty) * Part.MultiOrderQty
            CreateNewMSOrder()
            MSOrder.DueQty = DueQty, MSOrder.DueDate = Date
         End If
      End If
   End If
   For Each NewMSOrder In DateMSOders(Date)
      AvailableStock = AvailableStock + NewMSOrder.DueQty
   End For
End If
Date = Date + Part.PeriodicDays
End For
}
```

3.7.4. Sequence Diagram : Προγραμματισμός Απαιτήσεων Υλικών (MRP)



Σχολιασμός (Προγραμματισμός Απαιτήσεων Υλικών (MRP))

Ο Actor ανοίγει την αντίστοιχη φόρμα.

Η φόρμα δημιουργεί ένα αντικείμενο MRP (Controller Object).

Ο Actor εισάγει την καταληκτική ημερομηνία του χρονικού ορίζοντα Προγραμματισμού.

Ο Actor επιλέγει την εκτέλεση των υπολογισμών χωρίς επαναπρογραμματισμό των υφιστάμενων Εγκεκριμένων Εντολών.

Η οθόνη καλεί την αντίστοιχη μέθοδο του αντικειμένου MRP .

Το αντικείμενο MRP ανακτά όλα τα είδη (παραγόμενα και προμηθευόμενα) τύπου MRP (ενημέρωση του multi-object Part).

Το αντικείμενο MRP επαναϋπολογίζει και ταξινομεί όλα τα ανωτέρω είδη με βάση το Χαμηλότερο Σημείο Εμφάνισης (LLC) των ειδών.

Για κάθε είδος στο αντικείμενο MRP :

Διαγράφονται όλες οι Προγραμματισμένες και Εγκεκριμένες Εντολές MRP του είδους καλώντας την αντίστοιχη μέθοδο του αντικειμένου Part.

Εάν το είδος είναι παραγόμενο, το αντικείμενο Part διαγράφει επίσης και τις Προγραμματισμένες και Εγκεκριμένες Εντολές Εξαρτημένης Ζήτησης των συστατικών του είδους, καλώντας την αντίστοιχη μέθοδο του αντικειμένου Component τύπου Part.

Το αντικείμενο MRP ανακτά όλες τις ανοιχτές εντολές του είδους καλώντας την αντίστοιχη μέθοδο του αντικειμένου Part και τις ταξινομεί με βάση την προθεσμία.

Για κάθε ημέρα, ξεκινώντας από την επομένη της τρέχουσας μέχρι και την καταληκτική ημερομηνία προγραμματισμού :

Το αντικείμενο MRP, λαμβάνοντας υπόψη όλες τις απαραίτητες πληροφορίες δημιουργεί Προτεινόμενες (Προγραμματισμένες) Εντολές τύπου PO αν το είδος είναι προμηθευόμενο και τύπου MF αν το είδος είναι παραγόμενο.

Για κάθε Προτεινόμενη Εντολή MRP τύπου MF :

Για κάθε συστατικό του είδους :

Το αντικείμενο MRP δημιουργεί Προτεινόμενες (Προγραμματισμένες) Εντολές Εξαρτημένης Ζήτησης.

Τελικά, όλες οι Προτεινόμενες Εντολές MRP (MF και PO) καθώς και οι Εντολές Εξαρτημένης Ζήτησης, αποθηκεύονται στη βάση δεδομένων.

// Υπολογισμός Ποσότητας Προτεινόμενων Εντολών

Οι αλγόριθμοι υπολογισμού της ποσότητας των προτεινόμενων εντολών, ανάλογα με την Πολιτική Αναπαραγγελίας που αναφέρθηκαν στην προηγούμενη ενότητα του MPS, χρησιμοποιούνται και για τον προσδιορισμό της ποσότητας των προτεινόμενων εντολών του MRP. Η διαφορά είναι ότι στην περίπτωση των ειδών που διαχειρίζονται με τη λογική του MRP, υπεισέρχεται πλέον και η Εξαρτημένη Ζήτηση, οπότε ο τύπος υπολογισμού του Διαθέσιμου Αποθέματος έχει ως εξής :

Α) Παραγόμενα Ημιέτοιμα Είδη

```
AvailableStock = AvailableStock + Σ(MFOrders.DueQty) - MAX ( Σ(SOOrderLines.DueQty), Σ(FOOrders.DueQty) )
                                - Σ(RQOrders.DueQty)
```

Β) Προμηθευόμενα Είδη

```
AvailableStock = AvailableStock + Σ(POOrders.DueQty) - MAX ( Σ(SOOrderLines.DueQty), Σ(FOOrders.DueQty) )
                                - Σ(RQOrders.DueQty)
```

όπου οι RQOrders, αποτελούν τις εκδοθείσες (σε εξέλιξη) Εντολές Εξαρτημένης Ζήτησης των ειδών.

// Υπολογισμός Χαμηλότερου Σημείου Εμφάνισης (LLC) των Ειδών

// Μεθοδος CalculateLLC του Αντικειμένου MRP

```
For Each Part In MRP
   Part.CalculatePartLLC(Part,0)
End For
```

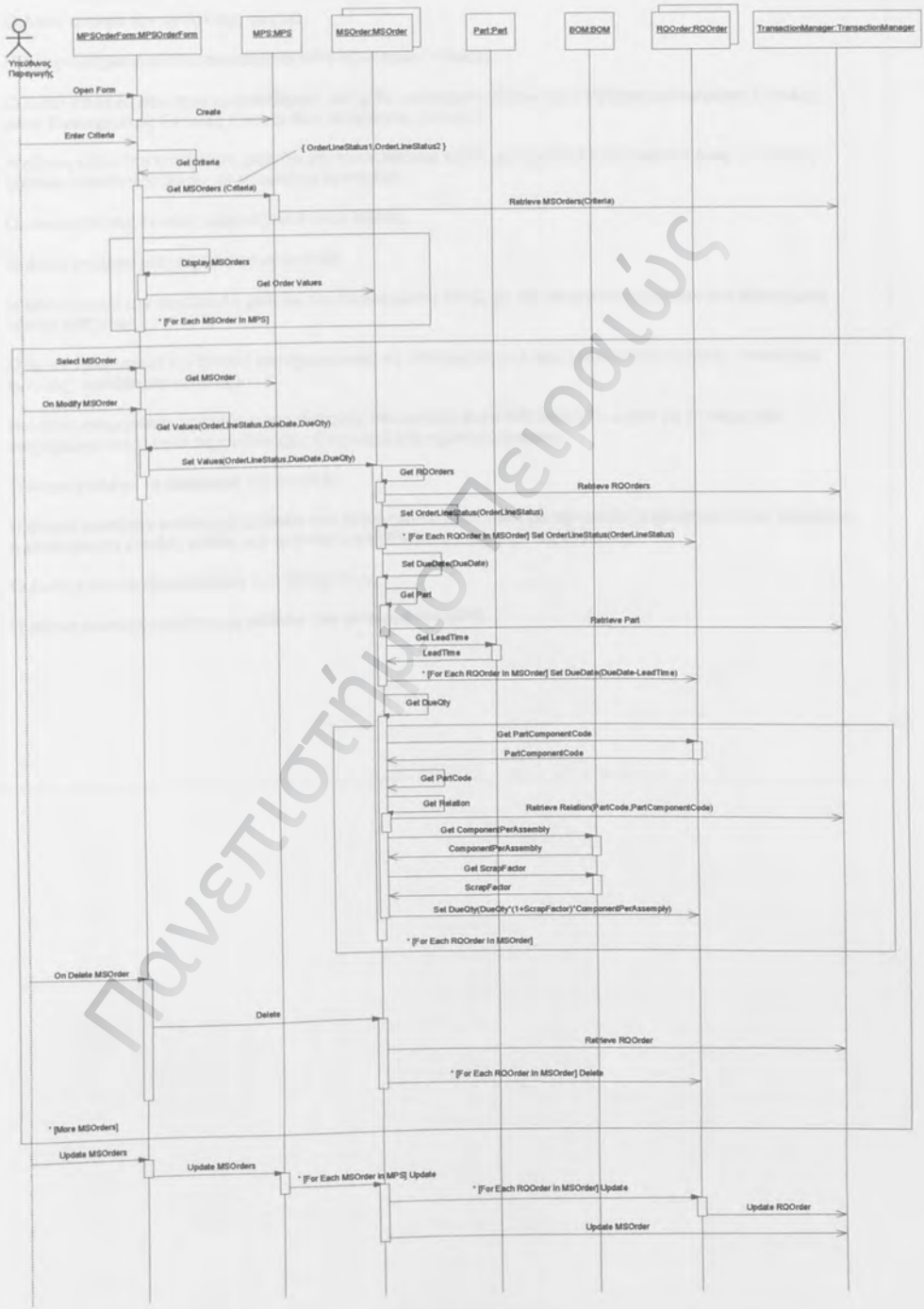
// Μέθοδος CalculatePartLLC (Part As Part, ByRef LLC As Integer) του Αντικειμένου Part

```
Max = 0

For Each WhereUsedRelation In Part
   LLC = LLC + 1
   CalculatePartLLC (WhereUsedRelation.ParentPart, LLC)
   If Max < LLC Then
      Max = LLC
   End If
   LLC = 0
End For

Part.LLC = Max
```

3.7.5. Sequence Diagram : Διαχείριση Προτεινόμενων Εντολών MPS



Σχολιασμός (Διαχείριση Προτεινόμενων Εντολών MPS)

Ο Actor ανοίγει την αντίστοιχη φόρμα.

Η φόρμα δημιουργεί ένα αντικείμενο MPS (Controller Object).

Ο Actor επιλέγει εάν στην εμφανιζόμενη λίστα θα συμπεριληφθούν μόνο Προγραμματισμένες Εντολές, μόνο Εγκεκριμένες Εντολές ή και οι δύο κατηγορίες εντολών.

Η οθόνη καλεί την κατάλληλη μέθοδο του αντικειμένου MPS, με την οποία ανακτώνται όλες οι εντολές τελικών προϊόντων βάσει των ανωτέρω κριτηρίων.

Οι ανακτηθείσες εντολές εμφανίζονται στην οθόνη.

Ο Actor επιλέγει μια συγκεκριμένη εντολή.

Η οθόνη καλεί την κατάλληλη μέθοδο του αντικειμένου MPS, με την οποία επιστρέφεται ένα αντικείμενο τύπου MSOrder.

Ο Actor τροποποιεί την εντολή καταχωρώντας τις επιθυμητές αλλαγές (κατάσταση εντολής, ποσότητα εντολής, προθεσμία εντολής).

Η οθόνη ενημερώνει τις αντίστοιχες ιδιότητες του αντικειμένου MSOrder, το οποίο με τη σειρά του ενημερώνει κατάλληλα τις αντίστοιχες Εντολές Εξαρτημένης Ζήτησης.

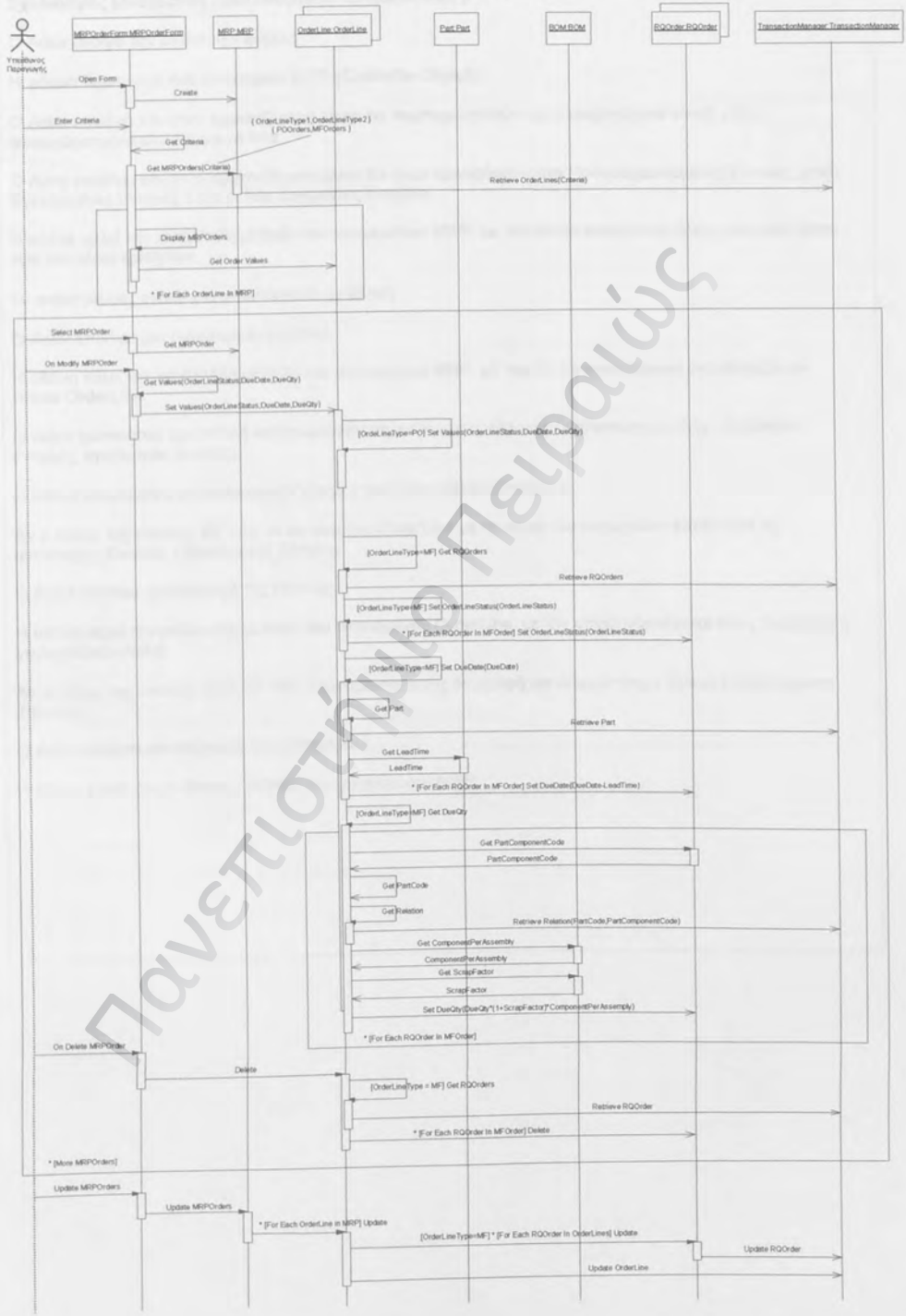
Ο Actor επιλέγει τη διαγραφή της εντολής.

Η οθόνη καλεί την αντίστοιχη μέθοδο του αντικειμένου MSOrder, με την οποία μαρκάρεται προς διαγραφή η επιλεχθείσα εντολή, καθώς και οι αντίστοιχες Εντολές Εξαρτημένης Ζήτησης.

Ο Actor επιλέγει αποθήκευση των δεδομένων.

Η οθόνη καλεί την αντίστοιχη μέθοδο του αντικειμένου MPS.

3.7.6. Sequence Diagram : Διαχείριση Προτεινόμενων Εντολών MRP



Σχολιασμός (Διαχείριση Προτεινόμενων Εντολών MRP)

Ο Actor ανοίγει την αντίστοιχη φόρμα.

Η φόρμα δημιουργεί ένα αντικείμενο MRP (Controller Object).

Ο Actor επιλέγει εάν στην εμφανιζόμενη λίστα θα συμπεριληφθούν μόνο παραγόμενα υλικά, μόνο προμηθευόμενα υλικά ή και τα δύο.

Ο Actor επιλέγει εάν στην εμφανιζόμενη λίστα θα συμπεριληφθούν μόνο Προγραμματισμένες Εντολές, μόνο Εγκεκριμένες Εντολές ή και οι δύο κατηγορίες εντολών.

Η οθόνη καλεί την κατάλληλη μέθοδο του αντικειμένου MRP, με την οποία ανακτώνται όλες οι εντολές βάσει των ανωτέρω κριτηρίων.

Οι ανακτηθείσες εντολές εμφανίζονται στην οθόνη.

Ο Actor επιλέγει μια συγκεκριμένη εντολή.

Η οθόνη καλεί την κατάλληλη μέθοδο του αντικειμένου MRP, με την οποία επιστρέφεται ένα αντικείμενο τύπου OrderLine.

Ο Actor τροποποιεί την εντολή καταχωρώντας τις επιθυμητές αλλαγές (κατάσταση εντολής, ποσότητα εντολής, προθεσμία εντολής).

Η οθόνη ενημερώνει τις αντίστοιχες ιδιότητες του αντικειμένου OrderLine.

Αν ο τύπος της εντολής MF τότε το αντικείμενο OrderLine με τη σειρά του ενημερώνει κατάλληλα τις αντίστοιχες Εντολές Εξαρτημένης Ζήτησης.

Ο Actor επιλέγει τη διαγραφή της εντολής.

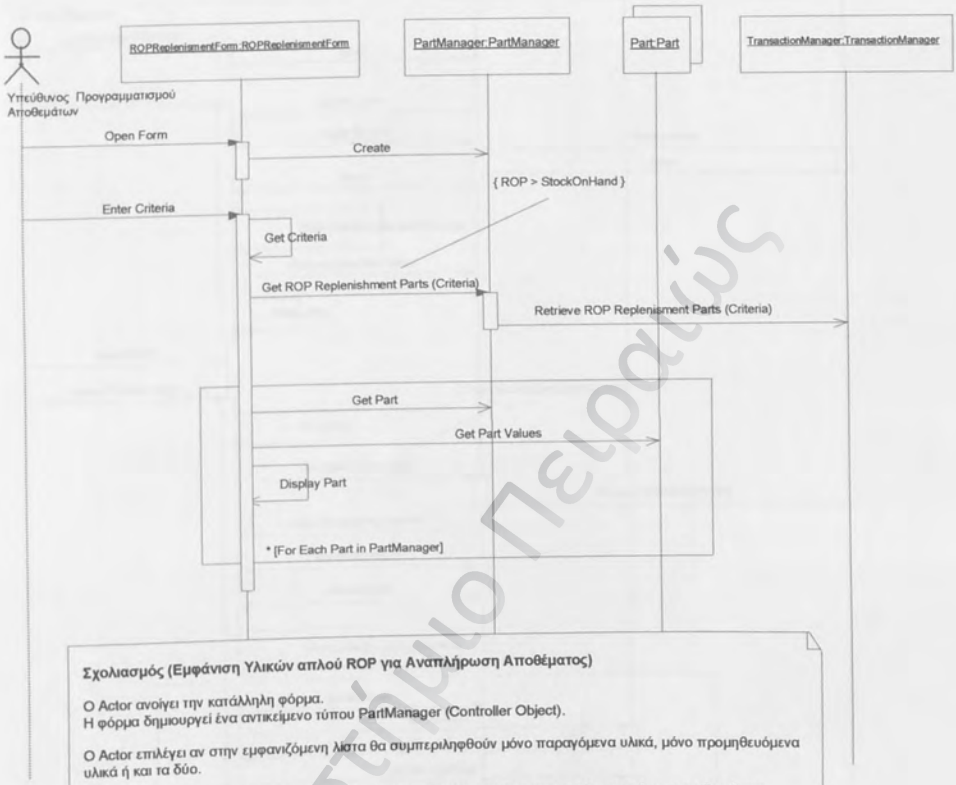
Η οθόνη καλεί την αντίστοιχη μέθοδο του αντικειμένου OrderLine, με την οποία μαρκάρεται προς διαγραφή η επιλεχθείσα εντολή.

Αν ο τύπος της εντολής είναι MF τότε μαρκάρονται προς διαγραφή και οι αντίστοιχες Εντολές Εξαρτημένης Ζήτησης.

Ο Actor επιλέγει αποθήκευση των δεδομένων.

Η οθόνη καλεί την αντίστοιχη μέθοδο του αντικειμένου MRP.

3.7.7. Sequence Diagram : Εμφάνιση Ειδών απλού ROP για Αναπλήρωση Αποθέματος



Σχολιασμός (Εμφάνιση Υλικών απλού ROP για Αναπλήρωση Αποθέματος)

Ο Actor ανοίγει την κατάλληλη φόρμα.
Η φόρμα δημιουργεί ένα αντικείμενο τύπου PartManager (Controller Object).

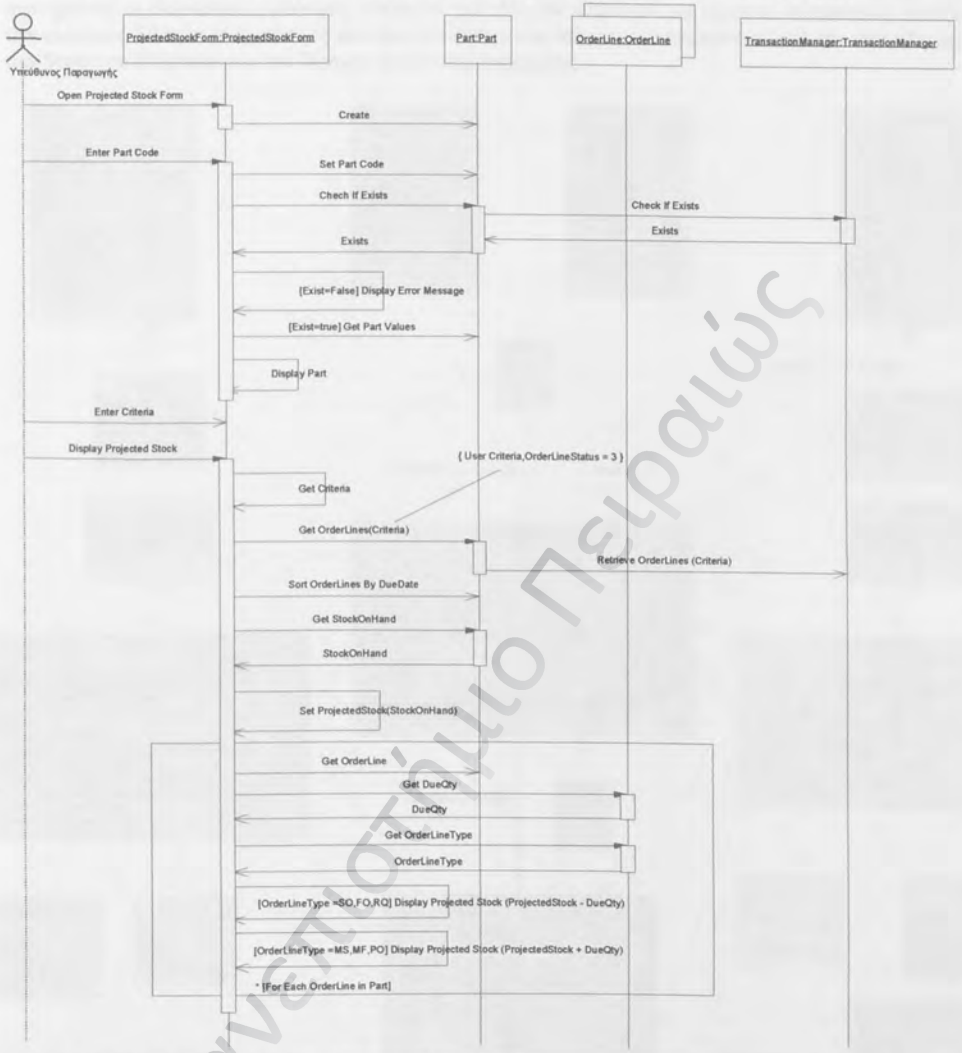
Ο Actor επιλέγει αν στην εμφανιζόμενη λίστα θα συμπεριληφθούν μόνο παραγόμενα υλικά, μόνο προμηθευόμενα υλικά ή και τα δύο.

Η οθόνη καλεί την αντίστοιχη μέθοδο του αντικειμένου PartManager με την οποία ανακτώνται τα είδη τύπου απλού ROP, τα οποία χρήζουν αναπαραγγελίας (ROP > StockOnHand).

Για κάθε είδος του PartManager :

Εμφανίζονται στην οθόνη τα βασικά του στοιχεία.

3.7.8. Sequence Diagram : Εμφάνιση Προβλεπόμενου Αποθέματος Ειδών



Σχολιασμός (Εμφάνιση Προβλεπόμενου Αποθέματος Ειδών)

Ο Actor ανοίγει την κατάλληλη φόρμα. Η φόρμα δημιουργεί ένα αντικείμενο τύπου Part.
Ο Actor εισάγει τον κωδικό του είδους. Η φόρμα ενημερώνει την αντίστοιχη ιδιότητα του αντικειμένου Part.
Η φόρμα ελέγχει την ύπαρξη του κωδικού καλώντας την αντίστοιχη μέθοδο του αντικειμένου Part.

Αν το είδος υπάρχει, εμφανίζονται στην οθόνη τα βασικά του στοιχεία.
Αν το είδος δεν υπάρχει, εμφανίζεται κατάλληλο μήνυμα στην οθόνη.

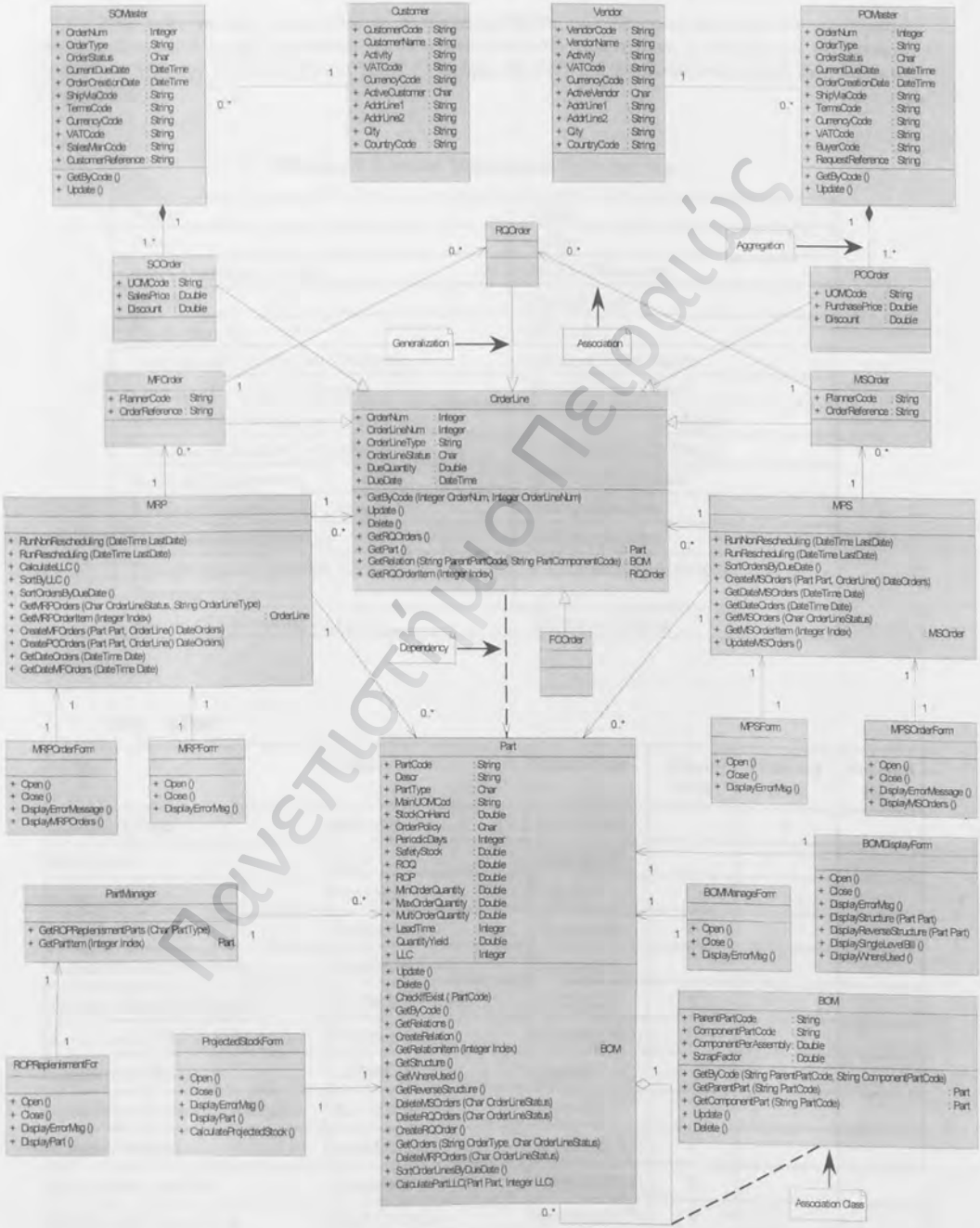
Ο Actor επιλέγει αν στον υπολογισμό του Προβλεπόμενου αποθέματος του είδους θέλει να συμπεριληφθούν και οι Προγραμματισμένες ή/και οι Εγκεκριμένες εντολές του είδους.

Η οθόνη καλεί την κατάλληλη μέθοδο του αντικειμένου Part, ώστε να ανακτηθούν όλες οι εντολές βάσει των ανωτέρω κριτηρίων, οι οποίες και ταξινομούνται ανά προθεσμία.

Για κάθε εντολή του αντικειμένου Part, υπολογίζεται το Προβλεπόμενο Απόθεμα του είδους, λαμβάνοντας υπόψη τον τύπο και την ποσότητα της εντολής και οι πληροφορίες εμφανίζονται στην οθόνη.

3.8. Design Model – Design Class Diagram

Στο Class Diagram που ακολουθεί, αποτυπώνονται όλες οι προγραμματιστικές κλάσεις του συστήματος, οι βασικότερες ιδιότητες αυτών, οι μέθοδοι των κλάσεων και τέλος οι συσχετίσεις μεταξύ των κλάσεων. Όλες οι πληροφορίες που απεικονίζονται στο διάγραμμα, αντλούνται από την επισκόπηση των Sequence Diagrams και του Domain Model της εφαρμογής.



3.9. Data Model – Γραμμογράφηση Βάσης Δεδομένων (Data Base Schema)

Μέσα στα πλαίσια της μεθοδολογία ανάπτυξης πληροφοριακών συστημάτων Unified Process, ο σχεδιασμός της βάσης δεδομένων ενός συστήματος εντάσσεται στη διαδικασία του Design και αποτελεί μέρος του παραδοτέου Data Model. Η μορφή της βάσης αρχίζει να κατασκευάζεται κατά τη διεξαγωγή της φάσης Elaboration και συνεχίζει να επεξεργάζεται καθ' όλη τη διάρκεια του Construction.

Στην ενότητα αυτή παρατίθενται η γραμμογράφηση των βασικών πινάκων του συστήματος (Πίνακας 8) συνοδευόμενη όπου θεωρείται απαραίτητο από σύντομα σχόλια, οι συσχετίσεις (References) μεταξύ των πινάκων (Πίνακας 9) και τέλος το σχήμα της βάσης δεδομένων υπό μορφή διαγράμματος (Σχήμα 30).

Πίνακας 8. Βασικοί Πίνακες του Συστήματος.

Name	Code
Είδη	tbPart
Δομή Πινάκων Υλικών	tbBOMStructure
Πελάτες	tbCustomer
Προμηθευτές	tbVendor
Επικεφαλίδες Εντολών Πώλησης	tbSalesOrderMaster
Γραμμές Εντολών Πώλησης	tbSalesOrderDetail
Επικεφαλίδες Εντολών Προμήθειας	tbPurchaseOrderMaster
Γραμμές Εντολών Προμήθειας	tbPurchaseOrderDetail
Εντολές Παραγωγής	tbProductionOrder
Εντολές Εξαρτημένης Ζήτηση	tbRequirements
Προτεινόμενες Εντολές	tbPlannedOrder
Προτεινόμενες Εντολές Εξαρτημένης Ζήτησης	tbPlannedRequirements
Εντολές Πρόγνωσης Ζήτησης	tbForecastOrder

Είδη - tbPart

Name	Code	Data Type	Allow Null	Primary Key	Foreign Key
Κωδικός Είδους	PartCode	varchar(20)		Y	
Περιγραφή	Descr	varchar(50)			
Τύπος Είδους	PartType	char(1)			
Κατηγορία Είδους	PartCategory	varchar(20)	Y		
Κύρια Μονάδα Μέτρησης	MainUOMCode	varchar(3)			
Δευτερ. Μονάδα Μέτρησης	SecondUOMCode	varchar(3)	Y		
Συντελεστής Μετατροπής	ConvertFactor	numeric(16,3)	Y		
Ιχνηλάτηση Παρτίδων	LotTrackFlag	char(1)			
Ιχνηλάτηση Σειριακών Αριθμών	SerialTrackFlag	char(1)			
Τρέχον Απόθεμα	StockOnHand	numeric(16,3)	Y		
Δεσμευμένο Απόθεμα	AllocatedStock	numeric(16,3)	Y		
Απόθεμα σε Παραγγελία	StockOnOrder	numeric(16,3)	Y		

Κατηγορία ABC	AbcCode	char(1)	Y		
Απογραφές ανά Έτος	CycleCountPeriod	numeric(3)	Y		
Όριο Απόκλισης Απογραφής	CycleCountTolerance	numeric(16,3)	Y		
Ημ/νία Τελευταίας Απογραφής	LastCycleCountDate	datetime	Y		
Παραλαβές Τρέχοντος Έτους	YearToDateReceipt	numeric(16,3)	Y		
Παραλαβές Προηγούμενου Έτους	LastYearReceipt	numeric(16,3)	Y		
Χορηγήσεις Τρέχοντος Έτους	YearToDateIssue	numeric(16,3)	Y		
Χορηγήσεις Προηγούμενου Έτους	LastYearIssue	numeric(16,3)	Y		
Κόστος Μονάδας	UnitCost	numeric(16,3)	Y		
Τιμή Πώλησης	UnitPrice	numeric(16,3)	Y		
Πολιτική Αναπαραγγελίας	OrderPolicy	char(1)			
Ημέρες Περιόδου Προγραμματισμού	PeriodicDays	numeric(3)			
Ποσότητα Αναπαραγγελίας	ROQ	numeric(16,3)			
Σημείο Αναπαραγγελίας	ROP	numeric(16,3)			
Απόθεμα Ασφαλείας	SafetyStock	numeric(16,3)			
Ελάχιστη Ποσότητα Εντολής	MinOrderQuantity	numeric(16,3)			
Μέγιστη Ποσότητα Εντολής	MaxOrderQuantity	numeric(16,3)			
Πολλαπλάσια Ποσότητα Εντολής	MultiOrderQuantity	numeric(16,3)			
Χρόνος Αναμονής Εντολής	LeadTime	numeric(3)			
Βαθμός Απόδοσης	QuantityYield	numeric(6,3)			
Χαμηλότερο Σημείο Εμφάνισης	LLC	numeric(3)			
Γραμμωτός Κώδικας	BarCode	varchar(50)	Y		
Σχόλια	Comments	varchar(1000)	Y		
Χρήστης Εισαγωγής Εγγραφής	AddUser	varchar(10)	Y		
Ημ/νία Εισαγωγής Εγγραφής	AddDate	datetime	Y		

Δομή Πινάκων Υλικών - tbBOMStructure

Name	Code	Data Type	Allow Null	Primary Key	Foreign Key
Κωδικός Συναρμολογήματος	ParentPartCode	varchar(20)		Y	Y
Κωδικός Συστατικού	ComponentPartCode	varchar(20)		Y	Y
Ποσότητα Συστατικού	ComponentPerAssembly	numeric(16,3)			
Ποσοστό Παραγωγής Σκάρτων	ScrapFactor	numeric(6,3)			
Σχόλια	Comments	varchar(1000)	Y		
Χρήστης Εισαγωγής Εγγραφής	AddUser	varchar(10)	Y		
Ημ/νία Εισαγωγής Εγγραφής	AddDate	datetime	Y		

Πελάτες - tbCustomer

Name	Code	Data Type	Allow Null	Primary Key	Foreign Key
Κωδικός Πελάτη	CustomerCode	varchar(20)		Y	
Επωνυμία	CustomerName	varchar(50)			
Δραστηριότητα	Activity	varchar(50)	Y		
Τρόπος Αποστολής	ShipViaCode	varchar(2)	Y		
Τρόπος Πληρωμής	TermsCode	varchar(2)	Y		
Αρ. Φορολογικού Μητρώου	AFM	varchar(10)	Y		
Δημόσια Οικονομική Υπηρεσία	DOY	varchar(20)	Y		
Κωδικός ΦΠΑ	VATCode	varchar(2)			
Κωδικός Νομίσματος	CurrencyCode	char(3)			
Αρμόδιος Επικοινωνίας	Contact	varchar(50)	Y		
Ενεργός Πελάτης	ActiveCustomer	char(1)			
Κωδικός Πωλητή	SalesManCode	varchar(10)	Y		
1η Γραμμή Διεύθυνσης	AddrLine1	varchar(50)	Y		
2η Γραμμή Διεύθυνσης	AddrLine2	varchar(50)	Y		
Ταχυδρομικός Κώδικας	ZipCode	varchar(20)	Y		
Πόλη	City	varchar(20)	Y		
Νομός	State	varchar(20)	Y		
Κωδικός Χώρας	CountryCode	char(2)	Y		
Τηλέφωνο 1	Phone1	varchar(20)	Y		
Τηλέφωνο 2	Phone2	varchar(20)	Y		
Κινητό Τηλέφωνο	MobilePhone	varchar(20)	Y		
ΦΑΞ	FAX	varchar(20)	Y		
Ηλεκτρονική Διεύθυνση	Email	varchar(50)	Y		
Σχόλια	Comments	varchar(1000)	Y		
Χρήστης Εισαγωγής Εγγραφής	AddUser	varchar(10)	Y		
Ημ/νία Εισαγωγής Εγγραφής	AddDate	datetime	Y		

Προμηθευτές - tbVendor

Name	Code	Data Type	Allow Null	Primary Key	Foreign Key
Κωδικός Προμηθευτή	VendorCode	varchar(20)		Y	
Επωνυμία	VendorName	varchar(50)			
Δραστηριότητα	Activity	varchar(50)	Y		
Τρόπος Παραλαβής	ShipViaCode	varchar(2)	Y		
Τρόπος Πληρωμής	TermsCode	varchar(2)	Y		

Αρ. Φορολογικού Μητρώου	AFM	varchar(10)	Y		
Δημόσια Οικονομική Υπηρεσία	DOY	varchar(20)	Y		
Κωδικός ΦΠΑ	VATCode	varchar(2)			
Κωδικός Νομίσματος	CurrencyCode	char(3)			
Αρμόδιος Επικοινωνίας	Contact	varchar(50)	Y		
Ενεργός Προμηθευτής	ActiveVendor	char(1)			
Κωδικός Αγοραστή	BuyerCode	varchar(10)	Y		
1η Γραμμή Διεύθυνσης	AddrLine1	varchar(50)	Y		
2η Γραμμή Διεύθυνσης	AddrLine2	varchar(50)	Y		
Ταχυδρομικός Κώδικας	ZipCode	varchar(20)	Y		
Πόλη	City	varchar(20)	Y		
Νομός	State	varchar(20)	Y		
Κωδικός Χώρας	CountryCode	char(2)	Y		
Τηλέφωνο 1	Phone1	varchar(20)	Y		
Τηλέφωνο 2	Phone2	varchar(20)	Y		
Κινητό Τηλέφωνο	MobilePhone	varchar(20)	Y		
ΦΑΞ	FAX	varchar(20)	Y		
Ηλεκτρονική Διεύθυνση	Email	varchar(50)	Y		
Σχόλια	Comments	varchar(1000)	Y		
Χρήστης Εισαγωγής Εγγραφής	AddUser	varchar(10)	Y		
Ημ/νία Εισαγωγής Εγγραφής	AddDate	datetime	Y		

Επικεφαλίδες Εντολών Πώλησης - tbSalesOrderMaster

Name	Code	Data Type	Allow Null	Primary Key	Foreign Key
Αριθμός Εντολής	OrderNum	numeric(7)		Y	
Τύπος Εντολής	OrderType	char(2)			
Κατάσταση Εντολής	OrderStatus	char(1)			
Κωδικός Πελάτη	CustomerCode	varchar(20)			Y
Αρχική Προθεσμία Εντολής	OriginalDueDate	datetime			
Τρέχουσα Προθεσμία Εντολής	CurrentDueDate	datetime			
Ημ/νία Καταχώρησης Εντολής	OrderCreationDate	datetime			
Τρόπος Αποστολής	ShipViaCode	varchar(2)	Y		
Τρόπος Πληρωμής	TermsCode	varchar(2)	Y		
Κωδικός Νομίσματος	CurrencyCode	char(3)			
Κωδικός ΦΠΑ	VATCode	varchar(2)			
Κωδικός Πωλητή	SalesManCode	varchar(10)	Y		
Αναφορά Πελάτη	CustomerReference	varchar(20)	Y		

Σχόλια	Comments	varchar(1000)	Y		
Χρήστης Εισαγωγής Εγγραφής	AddUser	varchar(10)	Y		
Ημ/νία Εισαγωγής Εγγραφής	AddDate	datetime	Y		

Γραμμές Εντολών Πώλησης - tbSalesOrderDetail

Name	Code	Data Type	Allow Null	Primary Key	Foreign Key
Αριθμός Εντολής	OrderNum	numeric(7)		Y	Y
Αριθμός Γραμμής Εντολής	OrderLineNum	numeric(3)		Y	
Κωδικός Είδους	PartCode	varchar(20)			Y
Τύπος Γραμμής Εντολής	OrderLineType	char(2)			
Κατάσταση Γραμμής Εντολής	OrderLineStatus	char(1)			
Μονάδα Μέτρησης	UOMCode	varchar(3)			
Αρχική Ποσότητα	OriginalQuantity	numeric(16,3)			
Τρέχουσα Ποσότητα	CurrentQuantity	numeric(16,3)			
Εκκρεμής Ποσότητα	DueQuantity	numeric(16,3)			
Αρχική Προθεσμία	OriginalDueDate	datetime			
Προθεσμία	DueDate	datetime			
Ημ/νία Καταχώρησης Γραμμής Εντολής	OrderLineCreationDate	datetime			
Τιμή Πώλησης	SalesPrice	numeric(16,3)			
Ποσοστό Έκπτωσης	Discount	numeric(6,3)			
Χρήστης Εισαγωγής Εγγραφής	AddUser	varchar(10)	Y		
Ημ/νία Εισαγωγής Εγγραφής	AddDate	datetime	Y		

Επικεφαλίδες Εντολών Προμήθειας - tbPurchaseOrderMaster

Name	Code	Data Type	Allow Null	Primary Key	Foreign Key
Αριθμός Εντολής	OrderNum	numeric(7)		Y	
Τύπος Εντολής	OrderType	char(2)			
Κατάσταση Εντολής	OrderStatus	char(1)			
Κωδικός Προμηθευτή	VendorCode	varchar(20)			Y
Αρχική Προθεσμία Εντολής	OriginalDueDate	datetime			
Τρέχουσα Προθεσμία Εντολής	CurrentDueDate	datetime			
Ημ/νία Καταχώρησης Εντολής	OrderCreationDate	datetime			
Τρόπος Παραλαβής	ShipViaCode	varchar(2)	Y		
Τρόπος Πληρωμής	TermsCode	varchar(2)	Y		

Κωδικός Νομίσματος	CurrencyCode	char(3)			
Κωδικός ΦΠΑ	VATCode	varchar(2)			
Κωδικός Αγοραστή	BuyerCode	varchar(10)	Y		
Αίτηση Προμήθειας	RequestReference	varchar(20)	Y		
Σχόλια	Comments	varchar(1000)	Y		
Χρήστης Εισαγωγής Εγγραφής	AddUser	varchar(10)	Y		
Ημ/νία Εισαγωγής Εγγραφής	AddDate	datetime	Y		

Γραμμές Εντολών Προμήθειας - tbPurchaseOrderDetail

Name	Code	Data Type	Allow Null	Primary Key	Foreign Key
Αριθμός Εντολής	OrderNum	numeric(7)		Y	Y
Αριθμός Γραμμής Εντολής	OrderLineNum	numeric(3)		Y	
Κωδικός Είδους	PartCode	varchar(20)			Y
Τύπος Γραμμής Εντολής	OrderLineType	char(2)			
Κατάσταση Γραμμής Εντολής	OrderLineStatus	char(1)			
Μονάδα Μέτρησης	UOMCode	varchar(3)			
Αρχική Ποσότητα	OriginalQuantity	numeric(16,3)			
Τρέχουσα Ποσότητα	CurrentQuantity	numeric(16,3)			
Εκκρεμής Ποσότητα	DueQuantity	numeric(16,3)			
Αρχική Προθεσμία	OriginalDueDate	datetime			
Προθεσμία	DueDate	datetime			
Ημ/νία Καταχώρησης Γραμμής Εντολής	OrderLineCreationDate	datetime			
Τιμή Αγοράς	PurchasePrice	numeric(16,3)			
Ποσοστό Έκπτωσης	Discount	numeric(6,3)			
Χρήστης Εισαγωγής Εγγραφής	AddUser	varchar(10)	Y		
Ημ/νία Εισαγωγής Εγγραφής	AddDate	datetime	Y		

Εντολές Παραγωγής - tbProductionOrder

Name	Code	Data Type	Allow Null	Primary Key	Foreign Key
Αριθμός Εντολής	OrderNum	numeric(7)		Y	
Αριθμός Γραμμής Εντολής	OrderLineNum	numeric(3)			
Κωδικός Είδους	PartCode	varchar(20)			Y
Τύπος Γραμμής Εντολής	OrderLineType	char(2)			
Κατάσταση Γραμμής Εντολής	OrderLineStatus	char(1)			

Αρχική Ποσότητα	OriginalQuantity	numeric(16,3)			
Τρέχουσα Ποσότητα	CurrentQuantity	numeric(16,3)			
Εκκρεμής Ποσότητα	DueQuantity	numeric(16,3)			
Αρχική Προθεσμία	OriginalDueDate	datetime			
Προθεσμία	DueDate	datetime			
Ημ/νία Καταχώρησης Γραμμής Εντολής	OrderLineCreationDate	datetime			
Κωδικός Υπεύθυνου Εντολής	PlannerCode	varchar(20)	Y		
Αναφορά Εντολής	OrderReference	varchar(20)	Y		
Χρήστης Εισαγωγής Εγγραφής	AddUser	varchar(10)	Y		
Ημ/νία Εισαγωγής Εγγραφής	AddDate	datetime	Y		

Στον πίνακα αυτό αποθηκεύονται όλες οι Εντολές Παραγωγής του συστήματος, είτε αφορούν τελικά προϊόντα (εντολές τύπου 'MS'), είτε αφορούν ημιέτοιμα υλικά (εντολές τύπου 'MF') με κατάσταση εντολής '3' (εντολές σε εξέλιξη) ή '4' (κλειστές ή ακυρωθείσες εντολές).

Εντολές Εξαρτημένης Ζήτησης - tbRequirements

Name	Code	Data Type	Allow Null	Primary Key	Foreign Key
Αριθμός Εντολής	OrderNum	numeric(7)		Y	Y
Αριθμός Γραμμής Εντολής	OrderLineNum	numeric(3)		Y	
Κωδικός Είδους	PartCode	varchar(20)			Y
Τύπος Γραμμής Εντολής	OrderLineType	char(2)			
Κατάσταση Εντολής	OrderLineStatus	char(1)			
Αρχική Ποσότητα Απαίτησης	OriginalQuantity	numeric(16,3)			
Τρέχουσα Ποσότητα Απαίτησης	CurrentQuantity	numeric(16,3)			
Εκκρεμής Ποσότητα	DueQuantity	numeric(16,3)			
Αρχική Προθεσμία Απαίτησης	OriginalDueDate	datetime			
Προθεσμία Απαίτησης	DueDate	datetime			
Ημ/νία Δημιουργίας Απαίτησης	OrderLineCreationDate	datetime			
Χρήστης Εισαγωγής Εγγραφής	AddUser	varchar(10)	Y		
Ημ/νία Εισαγωγής Εγγραφής	AddDate	datetime	Y		

Στον πίνακα αυτό αποθηκεύονται όλες οι Εντολές Εξαρτημένης Ζήτησης του συστήματος, δηλαδή εντολές που εκφράζουν τη ζήτηση η οποία προκύπτει από τις εντολές (τύπου 'MS' ή 'MF') των ειδών του αμέσως ανώτερου επιπέδου στους Πίνακες Υλικών. Ο τύπος των Εντολών Εξαρτημένης Ζήτησης είναι 'RQ' και η κατάσταση των εντολών '3' (εντολές σε εξέλιξη) ή '4' (κλειστές ή ακυρωθείσες εντολές). Ουσιαστικά ο πίνακας tbRequirements αποτελεί Detail του πίνακα tbProductionOrder.

Προτεινόμενες Εντολές - tbPlannedOrder

Name	Code	Data Type	Allow Null	Primary Key	Foreign Key
Αριθμός Εντολής	OrderNum	numeric(7)		Y	
Αριθμός Γραμμής Εντολής	OrderLineNum	numeric(3)			
Κωδικός Είδους	PartCode	varchar(20)			Y
Τύπος Γραμμής Εντολής	OrderLineType	char(2)			
Κατάσταση Γραμμής Εντολής	OrderLineStatus	char(1)			
Αρχική Ποσότητα	OriginalQuantity	numeric(16,3)			
Τρέχουσα Ποσότητα	CurrentQuantity	numeric(16,3)			
Εκκρεμής Ποσότητα	DueQuantity	numeric(16,3)			
Αρχική Προθεσμία	OriginalDueDate	datetime			
Προθεσμία	DueDate	datetime			
Ημ/νία Καταχώρησης Γραμμής Εντολής	OrderLineCreationDate	datetime			
Κωδικός Υπεύθυνου Εντολής	PlannerCode	varchar(20)	Y		
Αναφορά Εντολής	OrderReference	varchar(20)	Y		
Χρήστης Εισαγωγής Εγγραφής	AddUser	varchar(10)	Y		
Ημ/νία Εισαγωγής Εγγραφής	AddDate	datetime	Y		

Στον πίνακα αυτό αποθηκεύονται όλες οι Εντολές που προτείνει το σύστημα, τόσο κατά την εκτέλεση του MPS, όσο και του MRP. Οι εντολές του πίνακα αυτού μπορεί να είναι τύπου 'MS', 'MF' ή 'PO' (Εντολές Προμήθειας), ανάλογα με τον τύπο του είδους στο οποίο αναφέρονται. Η κατάσταση των εντολών παίρνει τιμές '1' (Προγραμματισμένη εντολή) ή '2' (Εγκεκριμένη εντολή).

Όταν μία Προτεινόμενη Εντολή τύπου 'MS' ή 'MF' εκδοθεί, η κατάστασή της γίνεται '3' (σε εξέλιξη) και η εντολή μεταφέρεται στον πίνακα tbProductionOrder. Όταν μία Προτεινόμενη Εντολή τύπου 'PO' εκδοθεί, η κατάστασή της επίσης γίνεται '3' (σε εξέλιξη) και η εντολή μεταφέρεται στο ζεύγος των πινάκων που αποθηκεύονται οι Εντολές Προμήθειας.

Προτεινόμενες Εντολές Εξαρτημένης Ζήτησης - tbPlannedRequirements

Name	Code	Data Type	Allow Null	Primary Key	Foreign Key
Αριθμός Εντολής	OrderNum	numeric(7)		Y	Y
Αριθμός Γραμμής Εντολής	OrderLineNum	numeric(3)		Y	
Κωδικός Είδους	PartCode	varchar(20)			Y
Τύπος Γραμμής Εντολής	OrderLineType	char(2)			
Κατάσταση Εντολής	OrderLineStatus	char(1)			
Αρχική Ποσότητα Απαίτησης	OriginalQuantity	numeric(16,3)			
Τρέχουσα Ποσότητα Απαίτησης	CurrentQuantity	numeric(16,3)			
Εκκρεμής Ποσότητα	DueQuantity	numeric(16,3)			

Αρχική Προθεσμία Απαίτησης	OriginalDueDate	datetime			
Προθεσμία Απαίτησης	DueDate	datetime			
Ημ/νία Δημιουργίας Απαίτησης	OrderLineCreationDate	datetime			
Χρήστης Εισαγωγής Εγγραφής	AddUser	varchar(10)	Y		
Ημ/νία Εισαγωγής Εγγραφής	AddDate	datetime	Y		

Στον πίνακα αυτό αποθηκεύονται όλες οι Προτεινόμενες Εντολές Εξαρτημένης Ζήτησης του συστήματος, δηλαδή εντολές που εκφράζουν τη ζήτηση η οποία προκύπτει από τις εντολές (τύπου 'MS' ή 'MF') του προηγούμενου πίνακα tbPlannedOrder. Ο τύπος των εντολών είναι 'RQ' και η κατάσταση των εντολών '1' (Προγραμματισμένη εντολή) ή '2' (Εγκεκριμένη εντολή). Ουσιαστικά ο πίνακας tbPlannedRequirements αποτελεί Detail του πίνακα tbPlannedOrder.

Όταν μία Προτεινόμενη Εντολή τύπου 'MS' ή 'MF' του πίνακα tbPlannedOrder εκδοθεί, πέραν των όσων αναφέρθηκαν στο σχολιασμό του πίνακα αυτού, οι σχετικές Προτεινόμενες Εντολές Εξαρτημένης Ζήτησης, αλλάζουν επίσης κατάσταση σε '3' (σε εξέλιξη) και μεταφέρονται στον πίνακα tbRequirements.

Εντολές Πρόγνωσης Ζήτησης - tbForecastOrder

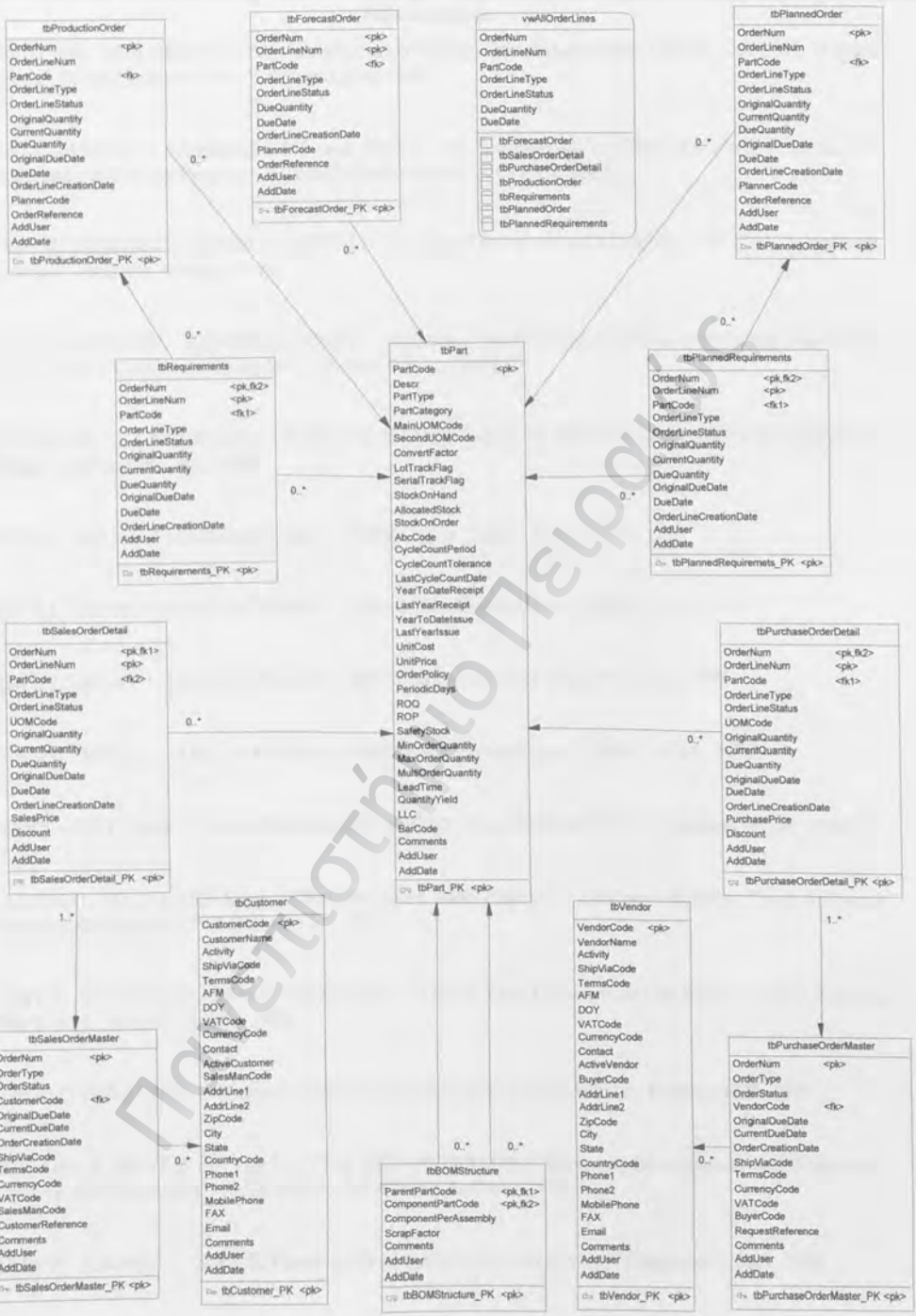
Name	Code	Data Type	Allow Null	Primary Key	Foreign Key
Αριθμός Εντολής	OrderNum	numeric(7)		Y	
Αριθμός Γραμμής Εντολής	OrderLineNum	numeric(3)		Y	
Κωδικός Είδους	PartCode	varchar(20)			Y
Τύπος Γραμής Εντολής	OrderLineType	char(2)			
Κατάσταση Γραμμής Εντολής	OrderLineStatus	char(1)			
Ποσότητα Εντολής	DueQuantity	numeric(16,3)			
Προθεσμία Εντολής	DueDate	datetime			
Ημ/νία Καταχώρησης Γραμμής Εντολής	OrderLineCreationDate	datetime			
Κωδικός Υπεύθυνου Εντολής	PlannerCode	varchar(20)	Y		
Αναφορά Εντολής	OrderReference	varchar(20)	Y		
Χρήστης Εισαγωγής Εγγραφής	AddUser	varchar(10)	Y		
Ημ/νία Εισαγωγής Εγγραφής	AddDate	datetime	Y		

View : vwAllOrderLines

Πρόκειται για ένα Ενοποιημένο View (Union View), το οποίο παρουσιάζει συνολικά τις κύριες ιδιότητες όλων των εντολών του συστήματος.

Πίνακας 9. Συσχετίσεις μεταξύ των Βασικών Πινάκων του Συστήματος.

Code	Parent Table	Child Table	Parent Key	Foreign Key Columns
TbBOMStructure_RF1	tbPart	tbBOMStructure	tbPart_PK	ParentPartCode
TbBOMStructure_RF2	tbPart	tbBOMStructure	tbPart_PK	ComponentPartCode
TbSalesOrderMaster_RF1	tbCustomer	tbSalesOrderMaster	tbCustomer_PK	CustomerCode
TbPurchaseOrderMaster_RF1	tbVendor	tbPurchaseOrderMaster	tbVendor_PK	VendorCode
TbSalesOrderDetail_RF2	tbSalesOrderMaster	tbSalesOrderDetail	tbSalesOrderMaster_PK	OrderNum
TbSalesOrderDetail_RF1	tbPart	tbSalesOrderDetail	tbPart_PK	PartCode
TbPurchaseOrderDetail_RF2	tbPurchaseOrderMaster	tbPurchaseOrderDetail	tbPurchaseOrderMaster_PK	OrderNum
TbPurchaseOrderDetail_RF1	tbPart	tbPurchaseOrderDetail	tbPart_PK	PartCode
TbProductionOrder_RF1	tbPart	tbProductionOrder	tbPart_PK	PartCode
TbForecastOrder_RF1	tbPart	tbForecastOrder	tbPart_PK	PartCode
TbRequirements_RF2	tbProductionOrder	tbRequirements	tbProductionOrder_PK	OrderNum
TbRequirements_RF1	tbPart	tbRequirements	tbPart_PK	PartCode
TbPlannedOrder_RF1	tbPart	tbPlannedOrder	tbPart_PK	PartCode
TbPlannedRequiremets_RF1	tbPart	tbPlannedRequiremets	tbPart_PK	PartCode
TbPlannedRequiremets_RF2	tbPlannedOrder	tbPlannedRequiremets	tbPlannedOrder_PK	OrderNum



Σχήμα 30. Γραμμογράφηση Βάσης Δεδομένων (Data Base Schema).

Βιβλιογραφία

DAVID WILLIAM BROWN : "An Introduction to Object-Oriented Analysis, Objects and UML in Plain English", Second Edition, John Wiley and Sons, 2002.

CRAIG LARMAN : "Applying UML and Patterns, An Introduction to Object-Oriented Analysis and Design and the Unified Process", Second Edition, Prentice Hall PTR, 2002.

DOUG ROSENBERG, KENDALL SCOTT : "Use Case Driven Object Modeling with UML, a Practical Approach", Addison Wesley, 1999.

DOUG ROSENBERG, KENDALL SCOTT : "Applying Use Case Driven Object Modeling with UML, an Annotated e-Commerce Example", Addison Wesley, 2001.

JIM ARLOW, ILA NEUSTADT : "UML and the Unified Process, Practical Object-Oriented Analysis & Design", Addison Wesley, 2002.

KEWILL ERP Inc : "MAX for Windows – MRP User's Guide", USA, 1999.

KEWILL ERP Inc : "MAX for Windows – Master Scheduling User's Guide", USA, 1999.

KEWILL ERP Inc : "MAX for Windows – Bill of Materials User's Guide", USA, 1999.

KEWILL ERP Inc : "MAX for Windows – Database Reference User's Guide", USA, 1999.

TERRY QUATRANI : "Visual Modeling with Rational Rose 2000 and UML", Addison Wesley, 2000.

RATIONAL SOFTWARE Corp. : "Rational Rose 2000 Enterprise Edition – Rational Unified Process Version 2001A.04.00, ON LINE HELP", 2001.

GARY K. EVANS, WILLIAM F. NAZZARO : "Killing Your Project with Use Cases !, The 5 Top Use Case Killers", Internet Article, 2002.

RONALD H. BALLOU : "Business Logistics Management", Fourth Edition, Prentice Hall, 1999.

EDWARD A. SILVER, DAVID F. PYKE, REIN PETERSON : "Inventory Management and Production Planning and Scheduling", Third Edition, John Wiley and Sons, 1998.

JOHN W. TOOMEY : "MRP II, Planning for Manufacturing Excellence", Chapman & Hall, 1996.

Η. Π. ΤΑΤΣΙΟΠΟΥΛΟΣ : «Προγραμματισμός και Έλεγχος Παραγωγής ΙΙ», ΕΜΠ, Αθήνα 1990.

Η. Π. ΤΑΤΣΙΟΠΟΥΛΟΣ : «Πληροφοριακά Συστήματα Διοικήσεως Παραγωγής», ΕΜΠ-ΟΠΑ, Διαπανεπιστημιακό Πρόγραμμα Μεταπτυχιακών Σπουδών - Διοίκηση Επιχειρήσεων.

Κ. Π. ΠΑΠΠΗΣ : «Προγραμματισμός Παραγωγής», Εκδόσεις Α. Σταμούλης, Αθήνα-Πειραιάς 1995.

Δ. ΞΗΡΟΚΩΣΤΑΣ : «Προγραμματισμός και Έλεγχος Παραγωγής μέσω Συστήματος CAPM στα Πλαίσια Ολοκληρωμένου Συστήματος CIM», Άρθρο, ΕΜΠ – Τομέας Β.Δ.Ε.Ε, Σεμινάριο Διοίκησης Παραγωγής με τη Χρήση Η/Υ, Αθήνα 1993.

Δρ. Γ. ΧΑΡΑΛΑΜΠΙΔΗΣ, Δρ. Δ. ΑΣΚΟΥΝΗΣ : «Μετά τα Συστήματα ERP : Dynamic Enterprise Modeling & Workflow στα Logistics», Άρθρο, Infopublica AE, PLANT Management / Ανάπτυξη 1998-99, Ετήσια Έκδοση.

Γ. Η. ΖΟΜΠΟΛΑΣ : «Χονδρικός Προγραμματισμός Δυναμικότητας (RCCP) & Βελτιστοποίηση Κύριου Προγράμματος Παραγωγής (MPS) με χρήση Γενετικών Αλγορίθμων», Διπλωματική Εργασία, ΕΜΠ, Αθήνα 2001.

Μ. ΔΕΡΔΕΛΑΚΟΥ, Γ. ΑΡΓΥΡΑΚΗΣ : «Ανάπτυξη και Εφαρμογή Συστήματος Υποστήριξης Αποφάσεων Ελέγχου Αποθεμάτων (SC) και Προγραμματισμού Απαιτήσεων Υλικών (MRP)», Διπλωματική Εργασία, ΕΜΠ, Αθήνα 2000.

Internet Sites

http://www.craiglarman.com/

http://www.williamnazzaro.com/

ΠΑΡΑΡΤΗΜΑΤΑ

```
/*==============================================================*/
/* Database name:  MPS-MRP Physical Data Model                  */
/* DBMS name     :  Microsoft SQL Server 2000                   */
/* Created on    :  1/10/2002 8:30:00 μμ                        */
/*==============================================================*/


alter table tbBOMStructure
   drop constraint TbBOMStructure_RF1
go

alter table tbBOMStructure
   drop constraint TbBOMStructure_RF2
go

alter table tbForecastOrder
   drop constraint TbForecastOrder_RF1
go

alter table tbPlannedOrder
   drop constraint TbPlannedOrder_RF1
go

alter table tbPlannedRequirements
   drop constraint TbPlannedRequirements_RF1
go

alter table tbPlannedRequirements
   drop constraint TbPlannedRequirements_RF2
go

alter table tbProductionOrder
   drop constraint TbProductionOrder_RF1
go

alter table tbPurchaseOrderDetail
   drop constraint TbPurchaseOrderDetail_RF1
go

alter table tbPurchaseOrderDetail
   drop constraint TbPurchaseOrderDetail_RF2
go

alter table tbPurchaseOrderMaster
   drop constraint TbPurchaseOrderMaster_RF1
go

alter table tbRequirements
   drop constraint TbRequirements_RF1
go

alter table tbRequirements
   drop constraint TbRequirements_RF2
go

alter table tbSalesOrderDetail
   drop constraint TbSalesOrderDetail_RF1
go

alter table tbSalesOrderDetail
   drop constraint TbSalesOrderDetail_RF2
go
```

```
alter table tbSalesOrderMaster
   drop constraint TbSalesOrderMaster_RF1
go


if exists (select 1
            from  sysobjects
           where  id = object_id('vwAllOrderLines')
            and   type = 'V')
   drop view vwAllOrderLines
go

if exists (select 1
            from  sysobjects
           where  id = object_id('tbBOMStructure')
            and   type = 'U')
   drop table tbBOMStructure
go

if exists (select 1
            from  sysobjects
           where  id = object_id('tbCustomer')
            and   type = 'U')
   drop table tbCustomer
go

if exists (select 1
            from  sysobjects
           where  id = object_id('tbForecastOrder')
            and   type = 'U')
   drop table tbForecastOrder
go

if exists (select 1
            from  sysobjects
           where  id = object_id('tbPart')
            and   type = 'U')
   drop table tbPart
go

if exists (select 1
            from  sysobjects
           where  id = object_id('tbPlannedOrder')
            and   type = 'U')
   drop table tbPlannedOrder
go

if exists (select 1
            from  sysobjects
           where  id = object_id('tbPlannedRequirements')
            and   type = 'U')
   drop table tbPlannedRequirements
go

if exists (select 1
            from  sysobjects
           where  id = object_id('tbProductionOrder')
            and   type = 'U')
   drop table tbProductionOrder
go
```

```
if exists (select 1
            from  sysobjects
           where  id = object_id('tbPurchaseOrderDetail')
            and   type = 'U')
   drop table tbPurchaseOrderDetail
go

if exists (select 1
            from  sysobjects
           where  id = object_id('tbPurchaseOrderMaster')
            and   type = 'U')
   drop table tbPurchaseOrderMaster
go

if exists (select 1
            from  sysobjects
           where  id = object_id('tbRequirements')
            and   type = 'U')
   drop table tbRequirements
go

if exists (select 1
            from  sysobjects
           where  id = object_id('tbSalesOrderDetail')
            and   type = 'U')
   drop table tbSalesOrderDetail
go

if exists (select 1
            from  sysobjects
           where  id = object_id('tbSalesOrderMaster')
            and   type = 'U')
   drop table tbSalesOrderMaster
go

if exists (select 1
            from  sysobjects
           where  id = object_id('tbVendor')
            and   type = 'U')
   drop table tbVendor
go



/*==============================================================*/
/* Table: tbBOMStructure                                        */
/*==============================================================*/
/* Δομή Πινάκων Υλικών */

create table tbBOMStructure (
ParentPartCode        varchar(20)          not null,
ComponentPartCode     varchar(20)          not null,
ComponentPerAssembly numeric(16,3)         not null
     constraint CKC_COMPONENTPERASSEMBLY_TBBOMSTRUCTURE check
     (ComponentPerAssembly >= 0),
ScrapFactor           numeric(6,3)         not null default 0
     constraint CKC_SCRAPFACTOR_TBBOMSTRUCTURE check (ScrapFactor between 0
     and 100),
Comments              varchar(1000)        null,
AddUser               varchar(10)          null,
AddDate               datetime             null,
constraint tbBOMStructure_PK primary key  (ParentPartCode, ComponentPartCode)
)
go
```

```
/*==============================================================*/
/* Table: tbCustomer                                            */
/*==============================================================*/
/* Πελάτες */
create table tbCustomer (
CustomerCode          varchar(20)          not null,
CustomerName          varchar(50)          not null,
Activity              varchar(50)          null,
ShipViaCode           varchar(2)           null,
TermsCode             varchar(2)           null,
AFM                   varchar(10)          null,
DOY                   varchar(20)          null,
VATCode               varchar(2)           not null,
CurrencyCode          char(3)              not null,
Contact               varchar(50)          null,
ActiveCustomer        char(1)              not null default 'Y'
     constraint CKC_ACTIVECUSTOMER_TBCUSTOMER check ActiveCustomer in
     ('Y','N')),
SalesManCode          varchar(10)          null,
AddrLine1             varchar(50)          null,
AddrLine2             varchar(50)          null,
ZipCode               varchar(20)          null,
City                  varchar(20)          null,
State                 varchar(20)          null,
CountryCode           char(2)              null,
Phone1                varchar(20)          null,
Phone2                varchar(20)          null,
MobilePhone           varchar(20)          null,
FAX                   varchar(20)          null,
Email                 varchar(50)          null,
Comments              varchar(1000)        null,
AddUser               varchar(10)          null,
AddDate               datetime             null,
constraint tbCustomer_PK primary key  (CustomerCode)
)
go

/*==============================================================*/
/* Table: tbForecastOrder                                       */
/*==============================================================*/
/* Εντολές Πρόγνωσης Ζήτησης */
create table tbForecastOrder (
OrderNum              numeric(7)           not null
     constraint CKC_ORDERNUM_TBFORECASTORDER check (OrderNum >= 1),
OrderLineNum          numeric(3)           not null
     constraint CKC_ORDERLINENUM_TBFORECASTORDER check (OrderLineNum >= 1),
PartCode              varchar(20)          not null,
OrderLineType         char(2)              not null default 'FO'
     constraint CKC_ORDERLINETYPE_TBFORECASTORDER check (OrderLineType in
     ('FO')),
OrderLineStatus       char(1)              not null default '3'
     constraint CKC_ORDERLINESTATUS_TBFORECASTORDER check (OrderLineStatus in
     ('3','4')),
DueQuantity           numeric(16,3)        not null
     constraint CKC_DUEQUANTITY_TBFORECASTORDER check (DueQuantity >= 0),
DueDate               datetime             not null,
OrderLineCreationDate datetime              not null,
PlannerCode           varchar(20)          null,
OrderReference        varchar(20)          null,
AddUser               varchar(10)          null,
AddDate               datetime             null,
constraint tbForecastOrder_PK primary key  (OrderNum, OrderLineNum)
)
go
```

```
/*==============================================================*/
/* Table: tbPart                                                */
/*==============================================================*/
/* Είδη */

create table tbPart (
PartCode              varchar(20)          not null,
Descr                 varchar(50)          not null,
PartType              char(1)              not null
     constraint CKC_PARTTYPE_TBPART check (PartType in
     ('M','A','B','C','D')),
PartCategory          varchar(20)          null,
MainUOMCode           varchar(3)           not null,
SecondUOMCode         varchar(3)           null,
ConvertFactor         numeric(16,3)        null
     constraint CKC_CONVERTFACTOR_TBPART check (ConvertFactor is null or
     (ConvertFactor >= 0 )),
LotTrackFlag          char(1)              not null default 'N'
     constraint CKC_LOTTRACKFLAG_TBPART check (LotTrackFlag in ('Y','N')),
SerialTrackFlag       char(1)              not null default 'N'
     constraint CKC_SERIALTRACKFLAG_TBPART check (SerialTrackFlag in
     ('Y','N')),
StockOnHand           numeric(16,3)        null
     constraint CKC_STOCKONHAND_TBPART check (StockOnHand is null or
     (StockOnHand >= 0 )),
AllocatedStock        numeric(16,3)        null
     constraint CKC_ALLOCATEDSTOCK_TBPART check (AllocatedStock is null or
     (AllocatedStock >= 0 )),
StockOnOrder          numeric(16,3)        null
     constraint CKC_STOCKONORDER_TBPART check (StockOnOrder is null or
     (StockOnOrder >= 0 )),
AbcCode               char(1)              null,
CycleCountPeriod      numeric(3)           null default 1
     constraint CKC_CYCLECOUNTPERIOD_TBPART check (CycleCountPeriod is null
     or (CycleCountPeriod >= 0 )),
CycleCountTolerance   numeric(16,3)        null default 0
     constraint CKC_CYCLECOUNTTOLERANCE_TBPART check (CycleCountTolerance is
     null or (CycleCountTolerance >= 0 )),
LastCycleCountDate    datetime             null,
YearToDateReceipt     numeric(16,3)        null
     constraint CKC_YEARTODATERECEIPT_TBPART check (YearToDateReceipt is null
     or (YearToDateReceipt >= 0 )),
LastYearReceipt       numeric(16,3)        null
     constraint CKC_LASTYEARRECEIPT_TBPART check (LastYearReceipt is null or
     (LastYearReceipt >= 0 )),
YearToDateIssue       numeric(16,3)        null
     constraint CKC_YEARTODATEISSUE_TBPART check (YearToDateIssue is null or
     (YearToDateIssue >= 0 )),
LastYearIssue         numeric(16,3)        null
     constraint CKC_LASTYEARISSUE_TBPART check (LastYearIssue is null or
     (LastYearIssue >= 0 )),
UnitCost              numeric(16,3)        null
     constraint CKC_UNITCOST_TBPART check (UnitCost is null or (UnitCost >= 0
     )),
UnitPrice             numeric(16,3)        null
     constraint CKC_UNITPRICE_TBPART check (UnitPrice is null or (UnitPrice
     >= 0 )),
OrderPolicy           char(1)              not null default 'L'
     constraint CKC_ORDERPOLICY_TBPART check (OrderPolicy in
     ('L','F','P','R')),
PeriodicDays          numeric(3)           not null default 1
     constraint CKC_PERIODICDAYS_TBPART check (PeriodicDays >= 1),
ROQ                   numeric(16,3)        not null default 0
     constraint CKC_ROQ_TBPART check (ROQ >= 0),
```

```
ROP                  numeric(16,3)          not null default 0
     constraint CKC_ROP_TBPART check (ROP >= 0),
SafetyStock          numeric(16,3)          not null default 0
     constraint CKC_SAFETYSTOCK_TBPART check (SafetyStock >= 0),
MinOrderQuantity     numeric(16,3)          not null default 0
     constraint CKC_MINORDERQUANTITY_TBPART check (MinOrderQuantity >= 0),
MaxOrderQuantity     numeric(16,3)          not null default 0
     constraint CKC_MAXORDERQUANTITY_TBPART check (MaxOrderQuantity >= 0),
MultiOrderQuantity   numeric(16,3)          not null default 0
     constraint CKC_MULTIORDERQUANTITY_TBPART check (MultiOrderQuantity >=
     0),
LeadTime             numeric(3)             not null default 0
     constraint CKC_LEADTIME_TBPART check (LeadTime >= 0),
QuantityYield        numeric(6,3)           not null default 100
     constraint CKC_QUANTITYYIELD_TBPART check (QuantityYield between 0 and
     100),
LLC                  numeric(3)             not null default -1
     constraint CKC_LLC_TBPART check (LLC >= -1),
BarCode              varchar(50)            null,
Comments             varchar(1000)          null,
AddUser              varchar(10)            null,
AddDate              datetime               null,
constraint tbPart_PK primary key  (PartCode)
)
go

/*==============================================================*/
/* Table: tbPlannedOrder                                        */
/*==============================================================*/
/* Προγραμματισμένες Εντολές */

create table tbPlannedOrder (
OrderNum             numeric(7)             not null
     constraint CKC_ORDERNUM_TBPLANNEDORDER check (OrderNum >= 1),
OrderLineNum         numeric(3)             not null default 0
     constraint CKC_ORDERLINENUM_TBPLANNEDORDER check (OrderLineNum in (0)),
PartCode             varchar(20)            not null,
OrderLineType        char(2)                not null
     constraint CKC_ORDERLINETYPE_TBPLANNEDORDER check (OrderLineType in
     ('MS','MF','PO')),
OrderLineStatus      char(1)                not null default '1'
     constraint CKC_ORDERLINESTATUS_TBPLANNEDORDER check (OrderLineStatus in
     ('1','2')),
OriginalQuantity     numeric(16,3)          not null
     constraint CKC_ORIGINALQUANTITY_TBPLANNEDORDER check (OriginalQuantity
     >= 0),
CurrentQuantity      numeric(16,3)          not null
     constraint CKC_CURRENTQUANTITY_TBPLANNEDORDER check (CurrentQuantity >=
     0),
DueQuantity          numeric(16,3)          not null
     constraint CKC_DUEQUANTITY_TBPLANNEDORDER check (DueQuantity >= 0),
OriginalDueDate      datetime               not null,
DueDate              datetime               not null,
OrderLineCreationDate datetime              not null,
PlannerCode          varchar(20)            null,
OrderReference       varchar(20)            null,
AddUser              varchar(10)            null,
AddDate              datetime               null,
constraint tbPlannedOrder_PK primary key  (OrderNum)
)
go
```

```
/*==============================================================*/
/* Table: tbPlannedRequirements                                 */
/*==============================================================*/
/* Προγραμματισμένη Εξαρτημένη Ζήτηση */

create table tbPlannedRequirements (
OrderNum              numeric(7)          not null
     constraint CKC_ORDERNUM_TBPLANNEDREQUIREMENTS check (OrderNum >= 1),
OrderLineNum          numeric(3)          not null
     constraint CKC_ORDERLINENUM_TBPLANNEDREQUIREMENTS check (OrderLineNum >=
     1),
PartCode              varchar(20)         not null,
OrderLineType         char(2)             not null default 'RQ'
     constraint CKC_ORDERLINETYPE_TBPLANNEDREQUIREMENTS check (OrderLineType
     in ('RQ')),
OrderLineStatus       char(1)             not null
     constraint CKC_ORDERLINESTATUS_TBPLANNEDREQUIREMENTS check
     (OrderLineStatus in ('1','2')),
OriginalQuantity      numeric(16,3)       not null
     constraint CKC_ORIGINALQUANTITY_TBPLANNEDREQUIREMENTS check
     (OriginalQuantity >= 0),
CurrentQuantity       numeric(16,3)       not null
     constraint CKC_CURRENTQUANTITY_TBPLANNEDREQUIREMENTS check
     (CurrentQuantity >= 0),
DueQuantity           numeric(16,3)       not null
     constraint CKC_DUEQUANTITY_TBPLANNEDREQUIREMENTS check (DueQuantity >=
     0),
OriginalDueDate       datetime            not null,
DueDate               datetime            not null,
OrderLineCreationDate datetime            not null,
AddUser               varchar(10)         null,
AddDate               datetime            null,
constraint tbPlannedRequiremets_PK primary key  (OrderNum, OrderLineNum)
)
go

/*==============================================================*/
/* Table: tbProductionOrder                                     */
/*==============================================================*/
/* Εντολές Παραγωγής */

create table tbProductionOrder (
OrderNum              numeric(7)          not null
     constraint CKC_ORDERNUM_TBPRODUCTIONORDER check (OrderNum >= 1),
OrderLineNum          numeric(3)          not null default 0
     constraint CKC_ORDERLINENUM_TBPRODUCTIONORDER check (OrderLineNum in
     (0)),
PartCode              varchar(20)         not null,
OrderLineType         char(2)             not null
     constraint CKC_ORDERLINETYPE_TBPRODUCTIONORDER check (OrderLineType in
     ('MS','MF')),
OrderLineStatus       char(1)             not null default '3'
     constraint CKC_ORDERLINESTATUS_TBPRODUCTIONORDER check (OrderLineStatus
     in ('3','4')),
OriginalQuantity      numeric(16,3)       not null
     constraint CKC_ORIGINALQUANTITY_TBPRODUCTIONORDER check
     (OriginalQuantity >= 0),
CurrentQuantity       numeric(16,3)       not null
     constraint CKC_CURRENTQUANTITY_TBPRODUCTIONORDER check (CurrentQuantity
     >= 0),
DueQuantity           numeric(16,3)       not null
     constraint CKC_DUEQUANTITY_TBPRODUCTIONORDER check (DueQuantity >= 0),
OriginalDueDate       datetime            not null,
DueDate               datetime            not null,
```

```
OrderLineCreationDate datetime                not null,
PlannerCode           varchar(20)             null,
OrderReference        varchar(20)             null,
AddUser               varchar(10)             null,
AddDate               datetime                null,
constraint tbProductionOrder_PK primary key  (OrderNum)
)
go

/*==============================================================*/
/* Table: tbPurchaseOrderDetail                                 */
/*==============================================================*/
/* Γραμμές Εντολών Προμήθειας */

create table tbPurchaseOrderDetail (
OrderNum              numeric(7)              not null
     constraint CKC_ORDERNUM_TBPURCHASEORDERDETAIL check (OrderNum >= 1),
OrderLineNum          numeric(3)              not null
     constraint CKC_ORDERLINENUM_TBPURCHASEORDERDETAIL check (OrderLineNum >=
     1),
PartCode              varchar(20)             not null,
OrderLineType         char(2)                 not null default 'PO'
     constraint CKC_ORDERLINETYPE_TBPURCHASEORDERDETAIL check (OrderLineType
     in ('PO')),
OrderLineStatus       char(1)                 not null default '3'
     constraint CKC_ORDERLINESTATUS_TBPURCHASEORDERDETAIL check
     (OrderLineStatus in ('3','4')),
UOMCode               varchar(3)              not null,
OriginalQuantity      numeric(16,3)           not null
     constraint CKC_ORIGINALQUANTITY_TBPURCHASEORDERDETAIL check
     (OriginalQuantity >= 0),
CurrentQuantity       numeric(16,3)           not null
     constraint CKC_CURRENTQUANTITY_TBPURCHASEORDERDETAIL check
     (CurrentQuantity >= 0),
DueQuantity           numeric(16,3)           not null
     constraint CKC_DUEQUANTITY_TBPURCHASEORDERDETAIL check (DueQuantity >=
     0),
OriginalDueDate       datetime                not null,
DueDate               datetime                not null,
OrderLineCreationDate datetime                not null,
PurchasePrice         numeric(16,3)           not null
     constraint CKC_PURCHASEPRICE_TBPURCHASEORDERDETAIL check (PurchasePrice
     >= 0),
Discount              numeric(6,3)            not null default 0
     constraint CKC_DISCOUNT_TBPURCHASEORDERDETAIL check (Discount >= 0),
AddUser               varchar(10)             null,
AddDate               datetime                null,
constraint tbPurchaseOrderDetail_PK primary key  (OrderNum, OrderLineNum)
)
go

/*==============================================================*/
/* Table: tbPurchaseOrderMaster                                 */
/*==============================================================*/
/* Επικεφαλίδες Εντολών Προμήθειας */

create table tbPurchaseOrderMaster (
OrderNum              numeric(7)              not null
     constraint CKC_ORDERNUM_TBPURCHASEORDERMASTER check (OrderNum >= 1),
OrderType             char(2)                 not null default 'PO'
     constraint CKC_ORDERTYPE_TBPURCHASEORDERMASTER check (OrderType in
     ('PO')),
```

```
OrderStatus          char(1)              not null default '3'
     constraint CKC_ORDERSTATUS_TBPURCHASEORDERMASTER check (OrderStatus in
     ('3','4')),
VendorCode           varchar(20)          not null,
OriginalDueDate      datetime             not null,
CurrentDueDate       datetime             not null,
OrderCreationDate    datetime             not null,
ShipViaCode          varchar(2)           null,
TermsCode            varchar(2)           null,
CurrencyCode         char(3)              not null,
VATCode              varchar(2)           not null,
BuyerCode            varchar(10)          null,
RequestReference     varchar(20)          null,
Comments             varchar(1000)        null,
AddUser              varchar(10)          null,
AddDate              datetime             null,
constraint tbPurchaseOrderMaster_PK primary key  (OrderNum)
)
go

/*==============================================================*/
/* Table: tbRequirements                                        */
/*==============================================================*/
/* Εξαρτημένη Ζήτηση */

create table tbRequirements (
OrderNum             numeric(7)           not null
     constraint CKC_ORDERNUM_TBREQUIREMENTS check (OrderNum >= 1),
OrderLineNum         numeric(3)           not null
     constraint CKC_ORDERLINENUM_TBREQUIREMENTS check (OrderLineNum >= 1),
PartCode             varchar(20)          not null,
OrderLineType        char(2)              not null default 'RQ'
     constraint CKC_ORDERLINETYPE_TBREQUIREMENTS check (OrderLineType in
     ('RQ')),
OrderLineStatus      char(1)              not null
     constraint CKC_ORDERLINESTATUS_TBREQUIREMENTS check (OrderLineStatus in
     ('3','4')),
OriginalQuantity     numeric(16,3)        not null
     constraint CKC_ORIGINALQUANTITY_TBREQUIREMENTS check (OriginalQuantity
     >= 0),
CurrentQuantity      numeric(16,3)        not null
     constraint CKC_CURRENTQUANTITY_TBREQUIREMENTS check (CurrentQuantity >=
     0),
DueQuantity          numeric(16,3)        not null
     constraint CKC_DUEQUANTITY_TBREQUIREMENTS check (DueQuantity >= 0),
OriginalDueDate      datetime             not null,
DueDate              datetime             not null,
OrderLineCreationDate datetime             not null,
AddUser              varchar(10)          null,
AddDate              datetime             null,
constraint tbRequirements_PK primary key  (OrderNum, OrderLineNum)
)
go


/*==============================================================*/
/* Table: tbSalesOrderDetail                                    */
/*==============================================================*/
/* Γραμμές Εντολών Πώλησης */

create table tbSalesOrderDetail (
OrderNum             numeric(7)           not null
     constraint CKC_ORDERNUM_TBSALESORDERDETAIL check (OrderNum >= 1),
```

```
OrderLineNum          numeric(3)          not null
     constraint CKC_ORDERLINENUM_TBSALESORDERDETAIL check (OrderLineNum >=
     1),
PartCode              varchar(20)         not null,
OrderLineType         char(2)             not null default 'SO'
     constraint CKC_ORDERLINETYPE_TBSALESORDERDETAIL check (OrderLineType in
     ('SO')),
OrderLineStatus       char(1)             not null default '3'
     constraint CKC_ORDERLINESTATUS_TBSALESORDERDETAIL check (OrderLineStatus
     in ('3','4')),
UOMCode               varchar(3)          not null,
OriginalQuantity      numeric(16,3)       not null
     constraint CKC_ORIGINALQUANTITY_TBSALESORDERDETAIL check
     (OriginalQuantity >= 0),
CurrentQuantity       numeric(16,3)       not null
     constraint CKC_CURRENTQUANTITY_TBSALESORDERDETAIL check (CurrentQuantity
     >= 0),
DueQuantity           numeric(16,3)       not null
     constraint CKC_DUEQUANTITY_TBSALESORDERDETAIL check (DueQuantity >= 0),
OriginalDueDate       datetime            not null,
DueDate               datetime            not null,
OrderLineCreationDate datetime             not null,
SalesPrice            numeric(16,3)       not null
     constraint CKC_SALESPRICE_TBSALESORDERDETAIL check (SalesPrice >= 0),
Discount              numeric(6,3)        not null default 0
     constraint CKC_DISCOUNT_TBSALESORDERDETAIL check (Discount >= 0),
AddUser               varchar(10)         null,
AddDate               datetime            null,
constraint tbSalesOrderDetail_PK primary key  (OrderNum, OrderLineNum)
)
go

/*==============================================================*/
/* Table: tbSalesOrderMaster                                    */
/*==============================================================*/
/* Επικεφαλίδες Εντολών Πώλησης */

create table tbSalesOrderMaster (
OrderNum              numeric(7)          not null
     constraint CKC_ORDERNUM_TBSALESORDERMASTER check (OrderNum >= 1),
OrderType             char(2)             not null default 'SO'
     constraint CKC_ORDERTYPE_TBSALESORDERMASTER check (OrderType in ('SO')),
OrderStatus           char(1)             not null default '3'
     constraint CKC_ORDERSTATUS_TBSALESORDERMASTER check (OrderStatus in
     ('3','4')),
CustomerCode          varchar(20)         not null,
OriginalDueDate       datetime            not null,
CurrentDueDate        datetime            not null,
OrderCreationDate     datetime            not null,
ShipViaCode           varchar(2)          null,
TermsCode             varchar(2)          null,
CurrencyCode          char(3)             not null,
VATCode               varchar(2)          not null,
SalesManCode          varchar(10)         null,
CustomerReference     varchar(20)         null,
Comments              varchar(1000)       null,
AddUser               varchar(10)         null,
AddDate               datetime            null,
constraint tbSalesOrderMaster_PK primary key  (OrderNum)
)
go
```

```
/*==============================================================*/
/* Table: tbVendor                                              */
/*==============================================================*/
/* Προμηθευτές */

create table tbVendor (
VendorCode              varchar(20)          not null,
VendorName              varchar(50)          not null,
Activity                varchar(50)          null,
ShipViaCode             varchar(2)           null,
TermsCode               varchar(2)           null,
AFM                     varchar(10)          null,
DOY                     varchar(20)          null,
VATCode                 varchar(2)           not null,
CurrencyCode            char(3)              not null,
Contact                 varchar(50)          null,
ActiveVendor            char(1)              not null default 'Y'
     constraint CKC_ACTIVEVENDOR_TBVENDOR check (ActiveVendor in ('Y','N')),
BuyerCode               varchar(10)          null,
AddrLine1               varchar(50)          null,
AddrLine2               varchar(50)          null,
ZipCode                 varchar(20)          null,
City                    varchar(20)          null,
State                   varchar(20)          null,
CountryCode             char(2)              null,
Phone1                  varchar(20)          null,
Phone2                  varchar(20)          null,
MobilePhone             varchar(20)          null,
FAX                     varchar(20)          null,
Email                   varchar(50)          null,
Comments                varchar(1000)        null,
AddUser                 varchar(10)          null,
AddDate                 datetime             null,
constraint tbVendor_PK primary key  (VendorCode)
)
go

/*==============================================================*/
/* View: vwAllOrderLines                                        */
/*==============================================================*/
/* Προβολή όλων των Εντολών */

create view vwAllOrderLines as
SELECT      OrderNum, OrderLineNum, PartCode, OrderLineType, OrderLineStatus,
            DueQuantity, DueDate
FROM        tbForecastOrder

UNION

SELECT      OrderNum, OrderLineNum, PartCode, OrderLineType, OrderLineStatus,
            DueQuantity, DueDate
FROM        tbSalesOrderDetail

UNION

SELECT      OrderNum, OrderLineNum, PartCode, OrderLineType, OrderLineStatus,
            DueQuantity, DueDate
FROM        tbPurchaseOrderDetail

UNION

SELECT      OrderNum, OrderLineNum, PartCode, OrderLineType, OrderLineStatus,
            DueQuantity, DueDate
FROM        tbProductionOrder
```

```
UNION

SELECT      OrderNum, OrderLineNum, PartCode, OrderLineType, OrderLineStatus,
            DueQuantity, DueDate
FROM        tbRequirements

UNION

SELECT      OrderNum, OrderLineNum, PartCode, OrderLineType, OrderLineStatus,
            DueQuantity, DueDate
FROM        tbPlannedOrder

UNION

SELECT      OrderNum, OrderLineNum, PartCode, OrderLineType, OrderLineStatus,
            DueQuantity, DueDate
FROM        tbPlannedRequirements
go


alter table tbBOMStructure
   add constraint TbBOMStructure_RF1 foreign key (ParentPartCode)
      references tbPart (PartCode)
go

alter table tbBOMStructure
   add constraint TbBOMStructure_RF2 foreign key (ComponentPartCode)
      references tbPart (PartCode)
go

alter table tbForecastOrder
   add constraint TbForecastOrder_RF1 foreign key (PartCode)
      references tbPart (PartCode)
go

alter table tbPlannedOrder
   add constraint TbPlannedOrder_RF1 foreign key (PartCode)
      references tbPart (PartCode)
go

alter table tbPlannedRequirements
   add constraint TbPlannedRequirements_RF1 foreign key (PartCode)
      references tbPart (PartCode)
go

alter table tbPlannedRequirements
   add constraint TbPlannedRequirements_RF2 foreign key (OrderNum)
      references tbPlannedOrder (OrderNum)
go

alter table tbProductionOrder
   add constraint TbProductionOrder_RF1 foreign key (PartCode)
      references tbPart (PartCode)
go

alter table tbPurchaseOrderDetail
   add constraint TbPurchaseOrderDetail_RF1 foreign key (PartCode)
      references tbPart (PartCode)
go

alter table tbPurchaseOrderDetail
   add constraint TbPurchaseOrderDetail_RF2 foreign key (OrderNum)
      references tbPurchaseOrderMaster (OrderNum)
go
```

```
alter table tbPurchaseOrderMaster
   add constraint TbPurchaseOrderMaster_RF1 foreign key (VendorCode)
      references tbVendor (VendorCode)
go

alter table tbRequirements
   add constraint TbRequirements_RF1 foreign key (PartCode)
      references tbPart (PartCode)
go

alter table tbRequirements
   add constraint TbRequirements_RF2 foreign key (OrderNum)
      references tbProductionOrder (OrderNum)
go

alter table tbSalesOrderDetail
   add constraint TbSalesOrderDetail_RF1 foreign key (PartCode)
      references tbPart (PartCode)
go

alter table tbSalesOrderDetail
   add constraint TbSalesOrderDetail_RF2 foreign key (OrderNum)
      references tbSalesOrderMaster (OrderNum)
go

alter table tbSalesOrderMaster
   add constraint TbSalesOrderMaster_RF1 foreign key (CustomerCode)
      references tbCustomer (CustomerCode)
go


/*==============================================================*/
/*                END OF GENERATION SCRIPT                      */
/*==============================================================*/
```

```
'*************************************************************************
' Module:  Part.vb
' Author:  christos
' Created: 1/10/2002 8:30:00 pm
' Purpose: Definition of Part
'*************************************************************************

Option Strict Off
Option Explicit On

Imports System

Public Class Part
    ' Attributes
    Public PartCode As String
    Public Descr As String
    Public PartType As Char
    Public MainUOMCode As String
    Public StockOnHand As Double
    Public OrderPolicy As Char
    Public PeriodicDays As Integer
    Public SafetyStock As Double
    Public ROQ As Double
    Public ROP As Double
    Public MinOrderQuantity As Double
    Public MaxOrderQuantity As Double
    Public MultiOrderQuantity As Double
    Public LeadTime As Integer
    Public QuantityYield As Double
    Public LLC As Integer
    Public Relation As BOM
    Public OrderLine() As OrderLine
    ' Properties

    ' Operations
    '---------------------------------------------------------------------
    ' Name:        Update()
    '---------------------------------------------------------------------
    Public Overridable Sub Update()
    ' TODO : implement
    End Sub

    '---------------------------------------------------------------------
    ' Name:        Delete()
    '---------------------------------------------------------------------
    Public Overridable Sub Delete()
    ' TODO : implement
    End Sub

    '---------------------------------------------------------------------
    ' Name:        CheckIfExist(ByRef PartCode)
    ' Parameters:
    '     [ByRef] PartCode -
    '---------------------------------------------------------------------
    Public Overridable Sub CheckIfExist(ByRef PartCode)
    ' TODO : implement
    End Sub

    '---------------------------------------------------------------------
    ' Name:        GetByCode()
    '---------------------------------------------------------------------
    Public Overridable Sub GetByCode()
    ' TODO : implement
    End Sub

    '---------------------------------------------------------------------
    ' Name:        GetRelations()
    '---------------------------------------------------------------------
    Public Overridable Sub GetRelations()
```

```
' TODO : implement
End Sub

'-------------------------------------------------------------------
' Name:         CreateRelation()
'-------------------------------------------------------------------
Public Overridable Sub CreateRelation()
' TODO : implement
End Sub

'-------------------------------------------------------------------
' Name:         GetRelationItem(ByRef Index As Integer)
' Parameters:
'     [ByRef] Index -
' Return: BOM
'-------------------------------------------------------------------
Public Overridable Function GetRelationItem(ByRef Index As Integer) As BOM
' TODO : implement
End Function

'-------------------------------------------------------------------
' Name:         GetStructure()
'-------------------------------------------------------------------
Public Overridable Sub GetStructure()
' TODO : implement
End Sub

'-------------------------------------------------------------------
' Name:         GetWhereUsed()
'-------------------------------------------------------------------
Public Overridable Sub GetWhereUsed()
' TODO : implement
End Sub

'-------------------------------------------------------------------
' Name:         GetReverseStructure()
'-------------------------------------------------------------------
Public Overridable Sub GetReverseStructure()
' TODO : implement
End Sub

'-------------------------------------------------------------------
' Name:         DeleteMSOrders(ByRef OrderLineStatus As Char)
' Parameters:
'     [ByRef] OrderLineStatus -
'-------------------------------------------------------------------
Public Overridable Sub DeleteMSOrders(ByRef OrderLineStatus As Char)
' TODO : implement
End Sub

'-------------------------------------------------------------------
' Name:         DeleteRQOrders(ByRef OrderLineStatus As Char)
' Parameters:
'     [ByRef] OrderLineStatus -
'-------------------------------------------------------------------
Public Overridable Sub DeleteRQOrders(ByRef OrderLineStatus As Char)
' TODO : implement
End Sub

'-------------------------------------------------------------------
' Name:         CreateRQOrder()
'-------------------------------------------------------------------
Public Overridable Sub CreateRQOrder()
' TODO : implement
End Sub
'-------------------------------------------------------------------
' Name:         GetOrders(ByRef OrderType As String, ByRef OrderLineStatus As Char)
' Parameters:
'     [ByRef] OrderType -
'     [ByRef] OrderLineStatus -
'-------------------------------------------------------------------
```

```
    Public Overridable Sub GetOrders(ByRef OrderType As String, ByRef OrderLineStatus
        As Char)
    ' TODO : implement
    End Sub

    '--------------------------------------------------------------------
    ' Name:         DeleteMRPOrders(ByRef OrderLineStatus As Char)
    ' Parameters:
    '     [ByRef] OrderLineStatus -
    '--------------------------------------------------------------------
    Public Overridable Sub DeleteMRPOrders(ByRef OrderLineStatus As Char)
    ' TODO : implement
    End Sub

    '--------------------------------------------------------------------
    ' Name:         SortOrderLinesByDueDate()
    '--------------------------------------------------------------------
    Public Overridable Sub SortOrderLinesByDueDate()
    ' TODO : implement
    End Sub

    '--------------------------------------------------------------------
    ' Name:         CalculetePartLLC(ByRef Part As Part, ByRef LLC As Integer)
    ' Parameters:
    '     [ByRef] Part -
    '     [ByRef] LLC -
    '--------------------------------------------------------------------
    Public Overridable Sub CalculetePartLLC(ByRef Part As Part, ByRef LLC As Integer)
    ' TODO : implement
    End Sub

End Class

'************************************************************************
' Module:  BOM.vb
' Author:  christos
' Created: 1/10/2002 8:30:00 pm
' Purpose: Definition of BOM
'************************************************************************

Option Strict Off
Option Explicit On

Imports System

Public Class BOM
    ' Attributes
    Public ParentPartCode As String
    Public ComponentPartCode As String
    Public ComponentPerAssembly As Double
    Public ScrapFactor As Double
    Public ParentPart As Part
    Public PartComponent As Part

    ' Properties

    ' Operations
    '--------------------------------------------------------------------
    ' Name:         GetByCode(ByRef ParentPartCode As String,ByRef ComponentPartCode As
    '               String)
    ' Parameters:
    '     [ByRef] ParentPartCode -
    '     [ByRef] ComponentPartCode -
    '--------------------------------------------------------------------
    Public Overridable Sub GetByCode(ByRef ParentPartCode As String, ByRef
        ComponentPartCode As String)
    ' TODO : implement
    End Sub
    '--------------------------------------------------------------------
    ' Name:         GetParentPart(ByRef PartCode As String)
    ' Parameters:
```

```
    '    [ByRef] PartCode -
    ' Return: Part
    '-------------------------------------------------------------------
    Public Overridable Function GetParentPart(ByRef PartCode As String) As Part
    ' TODO : implement
    End Function
    '-------------------------------------------------------------------
    ' Name:        GetComponentPart(ByRef PartCode As String)
    ' Parameters:
    '    [ByRef] PartCode -
    ' Return: Part
    '-------------------------------------------------------------------
    Public Overridable Function GetComponentPart(ByRef PartCode As String) As Part
    ' TODO : implement
    End Function

    '-------------------------------------------------------------------
    ' Name:        Update()
    '-------------------------------------------------------------------
    Public Overridable Sub Update()
    ' TODO : implement
    End Sub

    '-------------------------------------------------------------------
    ' Name:        Delete()
    '-------------------------------------------------------------------
    Public Overridable Sub Delete()
    ' TODO : implement
    End Sub

End Class

'*********************************************************************
' Module:  PartManager.vb
' Author:  christos
' Created: 1/10/2002 8:30:00 pm
' Purpose: Definition of PartManager
'*********************************************************************

Option Strict Off
Option Explicit On

Imports System

Public Class PartManager
    ' Attributes
    Public Part As Part

    ' Properties

    ' Operations
    '-------------------------------------------------------------------
    ' Name:        GetROPReplenismentParts(ByRef PartType As Char)
    ' Parameters:
    '    [ByRef] PartType -
    '-------------------------------------------------------------------
    Public Overridable Sub GetROPReplenismentParts(ByRef PartType As Char)
    ' TODO : implement
    End Sub

    '-------------------------------------------------------------------
    ' Name:        GetPartItem(ByRef Index As Integer)
    ' Parameters:
    '    [ByRef] Index -
    ' Return: Part
    '-------------------------------------------------------------------
    Public Overridable Function GetPartItem(ByRef Index As Integer) As Part
    ' TODO : implement
    End Function

End Class
```

```
'*************************************************************************
' Module:  MPS.vb
' Author:  christos
' Created: 1/10/2002 8:30:00 pm
' Purpose: Definition of MPS
'*************************************************************************

Option Strict Off
Option Explicit On

Imports System

Public Class MPS
    ' Attributes
    Public Part() As Part
    Public OrderLine() As OrderLine
    Public MsOrder() As MSOrder
    ' Properties

    ' Operations
    '-----------------------------------------------------------------
    ' Name:         RunNonRescheduling(ByRef LastDate As DateTime)
    ' Parameters:
    '     [ByRef] LastDate -
    '-----------------------------------------------------------------
    Public Overridable Sub RunNonRescheduling(ByRef LastDate As DateTime)
    ' TODO : implement
    End Sub

    '-----------------------------------------------------------------
    ' Name:         RunRescheduling(ByRef LastDate As DateTime)
    ' Parameters:
    '     [ByRef] LastDate -
    '-----------------------------------------------------------------
    Public Overridable Sub RunRescheduling(ByRef LastDate As DateTime)
    ' TODO : implement
    End Sub

    '-----------------------------------------------------------------
    ' Name:         SortOrdersByDueDate()
    '-----------------------------------------------------------------
    Public Overridable Sub SortOrdersByDueDate()
    ' TODO : implement
    End Sub

    '-----------------------------------------------------------------
    ' Name:         CreateMSOrders(ByRef Part As Part, ByRef DateOrders As OrderLine())
    ' Parameters:
    '     [ByRef] Part -
    '     [ByRef] DateOrders -
    '-----------------------------------------------------------------
    Public Overridable Sub CreateMSOrders(ByRef Part As Part, ByRef DateOrders As
        OrderLine())
    ' TODO : implement
    End Sub

    '-----------------------------------------------------------------
    ' Name:         GetDateMSOrders(ByRef Date As DateTime)
    ' Parameters:
    '     [ByRef] Date -
    '-----------------------------------------------------------------
    Public Overridable Sub GetDateMSOrders(ByRef Date As DateTime)
    ' TODO : implement
    End Sub

    '-----------------------------------------------------------------
    ' Name:         GetDateOrders(ByRef Date As DateTime)
    ' Parameters:
    '     [ByRef] Date -
    '-----------------------------------------------------------------
    Public Overridable Sub GetDateOrders(ByRef Date As DateTime)
```

```
    ' TODO : implement
    End Sub

    '--------------------------------------------------------------------
    ' Name:         GetMSOrders(ByRef OrderLineStatus As Char)
    ' Parameters:
    '     [ByRef] OrderLineStatus -
    '--------------------------------------------------------------------
    Public Overridable Sub GetMSOrders(ByRef OrderLineStatus As Char)
    ' TODO : implement
    End Sub

    '--------------------------------------------------------------------
    ' Name:         GetMSOrderItem(ByRef Index As Integer)
    ' Parameters:
    '     [ByRef] Index -
    ' Return: MSOrder
    '--------------------------------------------------------------------
    Public Overridable Function GetMSOrderItem(ByRef Index As Integer) As MSOrder
    ' TODO : implement
    End Function

    '--------------------------------------------------------------------
    ' Name:         UpdateMSOrders()
    '--------------------------------------------------------------------
    Public Overridable Sub UpdateMSOrders()
    ' TODO : implement
    End Sub

End Class


'**********************************************************************
' Module:  MRP.vb
' Author:  christos
' Created: 1/10/2002 8:30:00 pm
' Purpose: Definition of MRP
'**********************************************************************

Option Strict Off
Option Explicit On

Imports System

Public Class MRP
    ' Attributes
    Public Part() As Part
    Public OrderLine() As OrderLine
    Public MFOrder() As MFOrder

    ' Properties

    ' Operations
    '--------------------------------------------------------------------
    ' Name:         RunNonRescheduling(ByRef LastDate As DateTime)
    ' Parameters:
    '     [ByRef] LastDate -
    '--------------------------------------------------------------------
    Public Overridable Sub RunNonRescheduling(ByRef LastDate As DateTime)
    ' TODO : implement
    End Sub

    '--------------------------------------------------------------------
    ' Name:         RunReScheduling(ByRef LastDate As DateTime)
    ' Parameters:
    '     [ByRef] LastDate -
    '--------------------------------------------------------------------
    Public Overridable Sub RunReScheduling(ByRef LastDate As DateTime)
    ' TODO : implement
    End Sub
```

```
'-----------------------------------------------------------------
' Name:         CalculateLLC()
'-----------------------------------------------------------------
Public Overridable Sub CalculateLLC()
' TODO : implement
End Sub

'-----------------------------------------------------------------
' Name:         SortByLLC()
'-----------------------------------------------------------------
Public Overridable Sub SortByLLC()
' TODO : implement
End Sub

'-----------------------------------------------------------------
' Name:         SortOrdersByDueDate()
'-----------------------------------------------------------------
Public Overridable Sub SortOrdersByDueDate()
' TODO : implement
End Sub

'-----------------------------------------------------------------
' Name:         GetMRPOrders(ByRef OrderLineStatus As Char, ByRef OrderLineType As
'               String)
' Parameters:
'     [ByRef] OrderLineStatus -
'     [ByRef] OrderLineType -
'-----------------------------------------------------------------
Public Overridable Sub GetMRPOrders(ByRef OrderLineStatus As Char, ByRef
       OrderLineType As String)
' TODO : implement
End Sub

'-----------------------------------------------------------------
' Name:         GetMRPOrderItem(ByRef Index As Integer)
' Parameters:
'     [ByRef] Index -
' Return: OrderLine
'-----------------------------------------------------------------
Public Overridable Function GetMRPOrderItem(ByRef Index As Integer) As OrderLine
' TODO : implement
End Function

'-----------------------------------------------------------------
' Name:         CreateMFOrders(ByRef Part As Part, ByRef DateOrders As OrderLine())
' Parameters:
'     [ByRef] Part -
'     [ByRef] DateOrders -
'-----------------------------------------------------------------
Public Overridable Sub CreateMFOrders(ByRef Part As Part, ByRef DateOrders As
       OrderLine())
' TODO : implement
End Sub

'-----------------------------------------------------------------
' Name:         CreatePOOrders(ByRef Part As Part, ByRef DateOrders As OrderLine())
' Parameters:
'     [ByRef] Part -
'     [ByRef] DateOrders -
'-----------------------------------------------------------------
Public Overridable Sub CreatePOOrders(ByRef Part As Part, ByRef DateOrders As
       OrderLine())
' TODO : implement
End Sub

'-----------------------------------------------------------------
' Name:         GetDateOrders(ByRef Date As DateTime)
' Parameters:
'     [ByRef] Date -
'-----------------------------------------------------------------
```

```
    Public Overridable Sub GetDateOrders(ByRef Date As DateTime)
    ' TODO : implement
    End Sub

    '--------------------------------------------------------------------
    ' Name:         GetDateMFOrders(ByRef Date As DateTime)
    ' Parameters:
    '    [ByRef] Date -
    '--------------------------------------------------------------------
    Public Overridable Sub GetDateMFOrders(ByRef Date As DateTime)
    ' TODO : implement
    End Sub

End Class

'************************************************************************
' Module:  POMaster.vb
' Author:  christos
' Created: 1/10/2002 8:30:00 pm
' Purpose: Definition of POMaster
'************************************************************************

Option Strict Off
Option Explicit On

Imports System

Public Class POMaster
    ' Attributes
    Public OrderNum As Integer
    Public OrderType As String
    Public OrderStatus As Char
    Public CurrentDueDate As DateTime
    Public OrderCreationDate As DateTime
    Public ShipViaCode As String
    Public TermsCode As String
    Public CurrencyCode As String
    Public VATCode As String
    Public BuyerCode As String
    Public CustomerReference As String
    Public POOrder() As POOrder
    Public Vendor As Vendor

    ' Properties

    ' Operations
    '--------------------------------------------------------------------
    ' Name:         GetByCode()
    '--------------------------------------------------------------------
    Public Overridable Sub GetByCode()
    ' TODO : implement
    End Sub

    '--------------------------------------------------------------------
    ' Name:         Update()
    '--------------------------------------------------------------------
    Public Overridable Sub Update()
    ' TODO : implement
    End Sub

End Class

'************************************************************************
' Module:  SOMaster.vb
' Author:  christos
' Created: 1/10/2002 8:30:00 pm
' Purpose: Definition of SOMaster
'************************************************************************

Option Strict Off
Option Explicit On
```

```
Imports System

Public Class SOMaster
   ' Attributes
   Public OrderNum As Integer
   Public OrderType As String
   Public OrderStatus As Char
   Public CurrentDueDate As DateTime
   Public OrderCreationDate As DateTime
   Public ShipViaCode As String
   Public TermsCode As String
   Public CurrencyCode As String
   Public VATCode As String
   Public SalesManCode As String
   Public CustomerReference As String
   Public SOOrder() As SOOrder
   Public Customer As Customer

   ' Properties

   ' Operations
   '--------------------------------------------------------------
   ' Name:        GetByCode()
   '--------------------------------------------------------------
   Public Overridable Sub GetByCode()
   ' TODO : implement
   End Sub

   '--------------------------------------------------------------
   ' Name:        Update()
   '--------------------------------------------------------------
   Public Overridable Sub Update()
   ' TODO : implement
   End Sub

End Class


'*********************************************************************
' Module:  OrderLine.vb
' Author:  christos
' Created: 1/10/2002 8:30:00 pm
' Purpose: Definition of OrderLine
'*********************************************************************

Option Strict Off
Option Explicit On

Imports System

Public Class OrderLine
   ' Attributes
   Public OrderNum As Integer
   Public OrderLineNum As Integer
   Public OrderLineType As String
   Public OrderLineStatus As Char
   Public DueQuantity As Double
   Public DueDate As DateTime

   ' Properties
   ' Operations
   '--------------------------------------------------------------
   ' Name:        GetByCode(ByRef OrderNum As Integer, ByRef OrderLineNum As Integer)
   ' Parameters:
   '    [ByRef] OrderNum -
   '    [ByRef] OrderLineNum -
   '--------------------------------------------------------------
   Public Overridable Sub GetByCode(ByRef OrderNum As Integer, ByRef OrderLineNum As
         Integer)
   ' TODO : implement
   End Sub
```

```
    '-----------------------------------------------------------------
    ' Name:         Update()
    '-----------------------------------------------------------------
    Public Overridable Sub Update()
    ' TODO : implement
    End Sub

    '-----------------------------------------------------------------
    ' Name:         Delete()
    '-----------------------------------------------------------------
    Public Overridable Sub Delete()
    ' TODO : implement
    End Sub

    '-----------------------------------------------------------------
    ' Name:         GetRQOrders()
    '-----------------------------------------------------------------
    Public Overridable Sub GetRQOrders()
    ' TODO : implement
    End Sub

    '-----------------------------------------------------------------
    ' Name:         GetPart()
    ' Return: Part
    '-----------------------------------------------------------------
    Public Overridable Function GetPart() As Part
    ' TODO : implement
    End Function

    '-----------------------------------------------------------------
    ' Name:         GetRelation(ByRef ParentPartCode As String,ByRef PartComponentCode
    '               As String)
    ' Parameters:
    '     [ByRef] ParentPartCode -
    '     [ByRef] PartComponentCode -
    ' Return: BOM
    '-----------------------------------------------------------------
    Public Overridable Function GetRelation(ByRef ParentPartCode As String, ByRef
          PartComponentCode As String) As BOM
    ' TODO : implement
    End Function

    '-----------------------------------------------------------------
    ' Name:         GetRQOrderItem(ByRef Index As Integer)
    ' Parameters:
    '     [ByRef] Index -
    ' Return: RQOrder
    '-----------------------------------------------------------------
    Public Overridable Function GetRQOrderItem(ByRef Index As Integer) As RQOrder
    ' TODO : implement
    End Function

End Class

'*************************************************************************
' Module:  SOOrder.vb
' Author:  christos
' Created: 1/10/2002 8:30:00 pm
' Purpose: Definition of SOOrder
'*************************************************************************
Option Strict Off
Option Explicit On

Imports System

Public Class SOOrder
Inherits OrderLine
   ' Attributes
   Public UOMCode As String
   Public SalesPrice As Double
   Public Discount As Double
```

```
    ' Properties

    ' Operations

End Class

'*************************************************************************
' Module:  POOrder.vb
' Author:  christos
' Created: 1/10/2002 8:30:00 pm
' Purpose: Definition of POOrder
'*************************************************************************

Option Strict Off
Option Explicit On

Imports System

Public Class POOrder
Inherits OrderLine
    ' Attributes
    Public UOMCode As String
    Public PurchasePrice As Double
    Public Discount As Double
    Public Ventor As Ventor

    ' Properties

    ' Operations

End Class

'*************************************************************************
' Module:  FOOrder.vb
' Author:  christos
' Created: 1/10/2002 8:30:00 pm
' Purpose: Definition of FOOrder
'*************************************************************************

Option Strict Off
Option Explicit On

Imports System

Public Class FOOrder
Inherits OrderLine
    ' Attributes

    ' Properties

    ' Operations

End Class

'*************************************************************************
' Module:  MSOrder.vb
' Author:  christos
' Created: 1/10/2002 8:30:00 pm
' Purpose: Definition of MSOrder
'*************************************************************************
Option Strict Off
Option Explicit On

Imports System

Public Class MSOrder
Inherits OrderLine
```

```
    ' Attributes
    Public PlannerCode As String
    Public OrderReference As String
    Public RQOrder () As RQOrder
      ' Properties

    ' Operations

End Class

'*************************************************************************
' Module:  MFOrder.vb
' Author:  christos
' Created: 1/10/2002 8:30:00 pm
' Purpose: Definition of MFOrder
'*************************************************************************

Option Strict Off
Option Explicit On

Imports System

Public Class MFOrder
Inherits OrderLine
    ' Attributes
    Public PlannerCode As String
    Public OrderReference As String
    Public RQOrder() As RQOrder

    ' Properties

    ' Operations

End Class

'*************************************************************************
' Module:  RQOrder.vb
' Author:  christos
' Created: 1/10/2002 8:30:00 pm
' Purpose: Definition of RQOrder
'*************************************************************************

Option Strict Off
Option Explicit On

Imports System

Public Class RQOrder
Inherits OrderLine
    ' Attributes

    ' Properties

    ' Operations

End Class

'*************************************************************************
' Module:  Vendor.vb
' Author:  christos
' Created: 1/10/2002 8:30:00 pm
' Purpose: Definition of Vendor
'*************************************************************************

Option Strict Off
Option Explicit On

Imports System
```

```
Public Class Vendor
    ' Attributes
    Public VendorCode As String
    Public VendorName As String
    Public Activity As String
    Public VATCode As String
    Public CurrencyCode As String
    Public ActiveVendor As Char
    Public AddrLine1 As String
    Public AddrLine2 As String
    Public City As String
    Public CountryCode As String
    Public POMaster() As POOrder

    ' Properties

    ' Operations

End Class

'************************************************************************
' Module:  Customer.vb
' Author:  christos
' Created: 1/10/2002 8:30:00 pm
' Purpose: Definition of Customer
'************************************************************************

Option Strict Off
Option Explicit On

Imports System

Public Class Customer
    ' Attributes
    Public CustomerCode As String
    Public CustomerName As String
    Public Activity As String
    Public VATCode As String
    Public CurrencyCode As String
    Public ActiveCustomer As Char
    Public AddrLine1 As String
    Public AddrLine2 As String
    Public City As String
    Public CountryCode As String
    Public SOMaster() As SOMaster

    ' Properties

    ' Operations

End Class


'************************************************************************
'
' End of Code
'
'************************************************************************
```