ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΠΑΤΡΑΣ

Τμήμα Επιχειρηματικού Σχεδιασμού και Πληροφοριακών Συστημάτων

Τ.Ε.Ι. Πατρών

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

# *«Μελέτη της **UML** γλώσσας, της ιστορικής εξέλιξής της και των διαθέσιμων **CASE** εργαλείων για τη χρησιμοποίησή της»*


*ΚΑΡΙΩΤΟΓΛΟΥ ΧΡΗΣΤΟΣ*

*ΜΠΟΚΑΣ ΑΛΕΞΑΝΔΡΟΣ-ΠΑΡΑΣΚΕΥΑΣ*

Επιβλέπων Καθηγητής: Στάμος Κωνσταντίνος

**2012**

# Περίληψη

Η παρούσα εργασία έχει ως στόχο τη παρουσίαση της γλώσσας μοντελοποίησης **UML**, της ιστορικής εξέλιξής της, των διαγραμμάτων μοντελοποίησης που την αποτελούν και των διαθέσιμων εργαλείων **CASE**. Η γλώσσα **UML** μπορεί να καθιερώθηκε ως πρότυπο μόλις το **1997** αλλά διαθέτει πολύ μεγάλη προϊστορία, καθώς αποτελεί τον συνδυασμό μιας σειράς μεθόδων μοντελοποίησης.

Η μεγάλη της αξία είναι εμφανής από το βαθμό χρήσης της αλλά και το γεγονός ότι πλέον αποτελεί πρότυπο που εξελίσσεται μεθοδικά. Για να γίνει μια όσο το δυνατόν πιο πλήρης παρουσίασης της, η εργασία στο Κεφάλαιο **1** παρουσιάζει την ιστορική της αναδρομή, ώστε να αναδειχθεί το γεγονός ότι αποτέλεσε το αποτέλεσμα συγκεκριμένων συνεργασιών και ενοποίησης μεθόδων μοντελοποίησης, κάτι στο οποίο οφείλεται εν μέρει και το πρώτο συνθετικό του ονόματος της. Επιπλέον, παρουσιάζονται τα κύρια στοιχεία σήμανσης της γλώσσας και εξηγούνται τα πλεονεκτήματα και τα μειονεκτήματα της υιοθέτησης της, ενώ γίνεται και αναφορά σε εναλλακτικές λύσεις που υπάρχουν και εφαρμόζονται.

Στο Κεφάλαιο **2** παρουσιάζεται η Ενοποιημένη διεργασία ανάπτυξης αντικειμενοστραφών συστημάτων καθώς η Ενοποιημένη Γλώσσα Μοντελοποίησης **UML**, αποτελεί βασικό πυλώνα της Διεργασίας και αναπτύχθηκε παράλληλα με αυτή. Συνεπώς, η παρουσίαση της Ενοποιημένης Διεργασίας και των βασικών στοιχείων της, αποτελεί το μέσο για τη βαθύτερη κατανόηση και τη πληρέστερη παρουσίαση της **UML**. Έτσι επεξηγούνται οι κύκλοι εργασιών με τις τέσσερις συμπεριλαμβανόμενες φάσεις, η σύλληψη **(inception)**, η επεξεργασία **(elaboration)**, η κατασκευή **(construction)** και η μετάβαση **(transition)** και επιπλέον σε κάθε φάση πραγματοποιείται μια αντιστοίχιση διαγραμμάτων **UML**.

Το Κεφάλαιο **3** αποτελεί το κύριο μέρος της εργασίας καθώς παρουσιάζει τα δεκατέσσερα διαγράμματα μοντελοποίησης της **UML**, διαχωρισμένα στις τρεις βασικές κατηγορίες: διαγράμματα μοντελοποίησης δομής, μοντελοποίησης συμπεριφοράς και μοντελοποίησης αλληλεπίδρασης. Στο σημείο αυτό πρέπει να σημειωθεί ότι έχει ακολουθηθεί η σήμανση του προτύπου **UML 2.0** και έπειτα. Σε κάθε περίπτωση διαγράμματος παρουσιάζονται τα βασικά συστατικά του και αναπτύσσονται παραδείγματα, τα οποία επεξηγούνται.

Συμπληρωματικά του προηγούμενου Κεφαλαίου λειτουργεί το Κεφάλαιο **4** που παρουσιάζει ένα πραγματικό πρόβλημα που αφορά τη μοντελοποίηση ενός ηλεκτρονικού καταστήματος. Γίνεται περιγραφή του σεναρίου εφαρμογής και της λειτουργικότητας που απαιτείται από το σύστημα. Έπειτα, παρουσιάζονται τα διαγράμματα που μοντελοποιούν το σύστημα αυτό. Έχουν χρησιμοποιηθεί τα κυριότερα διαγράμματα **UML** που εφαρμόζονται και στη πράξη.

Τέλος, στο Κεφάλαιο **5** παρουσιάζονται τα εργαλεία **CASE**. Στην αγορά αυτή τη στιγμή υπάρχει πληθώρα τέτοιων εργαλείων με διαφορετικές δυνατότητες. Η εργασία επικεντρώνεται σε εργαλεία τα οποία αφενός υποστηρίζουν την εξαγωγή κώδικα από τα

διαγράμματα και επιπλέον ενσωματώνουν τουλάχιστον το πρότυπο **UML 2.0.** Παρουσιάζονται επίσης οι βασικότερες λειτουργίες των εργαλείων **CASE,** τα πλεονεκτήματα και μειονεκτήματά τους και τέλος, παρατίθενται τα δέκα πιο ευρέως χρησιμοποιούμενα **UML CASE** εργαλεία, με στοιχεία σχετικά με τις δυνατότητες που προσφέρουν.

# Πίνακας Περιεχομένων

# Λίστα Σχημάτων

# Λίστα Πινάκων

# Ακρώνυμα

**API (Application Programming Interface)**

**BPMN (Business Process Model and Notation)**

**CASE (Computer Aided Software Engineering)**

**EPC (Event-driven Process Chain)**

**ERD (Entity Relationship Diagrams)**

**E-R (Entity - Relationship)**

**IDE (Integrated Development Environment)**

**MDD (Model-Driven Development)**

**MOF (Microsoft Operating Framework)**

**OCL (Object Constraint Language)**

**OMG (Object Management Group)**

**OMT (Object Modeling Technique)**

**OOAD (Object-Oriented Analysis & Design)**

**OOSE (Object-Oriented Software Engineering)**

**SDL (Specification & Description Language)**

**SysML (Systems Modeling Language)**

**UML (Unified Modeling Language)**

**UMLDI (UML Diagram Interchange)**

**UPDM (Unified Profile for DoDAF/MODAF)**

# Κεφάλαιο 1: Εισαγωγή στη γλώσσα UML

## *1.1 Τι είναι η γλώσσα UML*

Η **UML (Unified Modeling Language)** αποτελεί το διάδοχο ενός κύματος μεθόδων που αποσκοπούσαν στην ανάλυση και σχεδιασμό πληροφοριακών συστημάτων, και πρωτοεμφανιστήκανε τη δεκαετία του **'70** έως τις αρχές του **'90.** Αποτελεί κυρίως μια ενοποίηση των μεθόδων των **Booch, Rumbaugh** και **Jacobson** αλλά το εύρος και οι δυνατότητές της είναι μεγαλύτερες.

Πλέον, η **UML** είναι μια τυποποιημένη γενικής χρήσης γλώσσα μοντελοποίησης στον τομέα της αντικειμενοστραφούς τεχνολογίας λογισμικού, καθώς αποτελεί πρότυπο το οποίο δημιουργήθηκε και εξελίσσεται από το **OMG. Η UML** εγκρίθηκε από το **OMG** το **1997**, και έκτοτε έχει γίνει το βιομηχανικό πρότυπο για τη μοντελοποίηση συστημάτων λογισμικού.**[1]**

Σύμφωνα με το **OMG** η **UML** είναι μια γραφική γλώσσα για την οπτικοποίηση, το καθορισμό, τη κατασκευή και τη τεκμηρίωση των στοιχείων ενός συστήματος λογισμικού. Επιπλέον, η **UML** προσφέρει ένα τυποποιημένο τρόπο για την απεικόνιση των σχεδίων ενός συστήματος, συμπεριλαμβανομένων τόσο εννοιολογικών στοιχείων όπως επιχειρηματικές διαδικασίες και λειτουργίες του συστήματος, όσο και συγκεκριμένα στοιχεία, όπως δηλώσεις γλώσσας προγραμματισμού, σχήματα βάσεων δεδομένων, και επαναχρησιμοποιήσιμα τμήματα λογισμικού.**[2]**

Η **UML** είναι γλώσσα μοντελοποίησης και όχι μέθοδος. Οι περισσότερες μέθοδοι αποτελούνται από μια γλώσσα μοντελοποίησης και μια διαδικασία μοντελοποίησης. Η γλώσσα μοντελοποίησης είναι κυρίως ένας (γραφικός) τρόπος σήμανσης που χρησιμοποιείται από τις μεθόδους για να εκφράσουν το τελικό σχεδιασμό του τελικού συστήματος. Η διαδικασία μοντελοποίησης αντιθέτως, είναι ο τρόπος υλοποίησης του σχεδιασμού. Στη πράξη όμως, για την ολοκλήρωση και τη παρουσίαση του σχεδίου ενός προϊόντος, πρέπει οι εμπλεκόμενοι να κατανοούν τη γλώσσα μοντελοποίησης που χρησιμοποιήθηκε για το σχέδιο και όχι τη διαδικασία που οδήγησε σε αυτό. Συνεπώς, από άποψη επικοινωνίας, το πιο σημαντικό τμήμα μιας μεθόδου είναι η γλώσσα μοντελοποίησης και όχι τόσο η διαδικασία μοντελοποίησης.

## *1.2 Ιστορία της UML*

Μετά την ευρεία εξάπλωση του αντικειμενοστραφούς προγραμματισμού κατά τη δεκαετία του **'90**, το αντικειμενοστραφές μοντέλο σχεδίασης (με κλάσεις, κληρονομικότητα, αντικείμενα και τυποποιημένες αλληλεπιδράσεις μεταξύ τους)

επικράτησε ακόμη και στις περιπτώσεις μοντελοποίησης που δεν περιλαμβάνανε προγραμματισμό **(**π.χ. σχήματα βάσεων δεδομένων**).**

Έτσι αναπτύχθηκαν διάφορες πρότυπες γλώσσες μοντελοποίησης λογισμικού οι οποίες τυποποιούσαν οπτικά σύμβολα και συμπεριφορές με στόχο την αφαιρετική περιγραφή της λειτουργίας και της δομής ενός υπολογιστικού συστήματος**.** Οι γλώσσες αυτές είχαν εξαρχής έναν εμφανή αντικειμενοστραφή προσανατολισμό**.** Τελικά οι πιο δημοφιλείς από αυτές ενοποιήθηκαν στο κοινό πρότυπο **UML** που η πρώτη του έκδοση οριστικοποιήθηκε το **1997.**

**Η UML** έχει αναπτυχθεί από τη **Rational Software** και τους εταίρους της**.** Επίσης πολλές εταιρίες έχουν ενσωματώσει τη **UML** ως πρότυπο στη διαδικασία ανάπτυξής τους και στα προϊόντα τους**,** τα οποία καλύπτουν περιοχές όπως το **business modeling,** η διαχείριση απαιτήσεων**,** η ανάλυση και ο σχεδιασμός**,** ο προγραμματισμός και ο έλεγχος**.** Από τα μέσα της δεκαετίας του **'70** έως τα τέλη της δεκαετίας του **'80** άρχισαν να εμφανίζονται αντικειμενοστραφείς γλώσσες μοντελοποίησης**,** καθώς οι διάφοροι ερευνητές είχαν αρχίσει να πειραματίζονται με διαφορετικές προσεγγίσεις στην αντικειμενοστραφή ανάλυση και σχεδιασμό**.** Οι γλώσσες αυτές επηρεάστηκαν από άλλες τεχνικές**,** όπως η μοντελοποίηση Οντοτήτων**-**Συσχετίσεων**,** η **SDL (Specification & Description Language -** Γλώσσα προδιαγραφών και περιγραφής**)** και άλλες τεχνικές**.** Το πλήθος των γλωσσών μοντελοποίησης αυξήθηκε από λιγότερες από **10** σε πάνω από **50,** στην περίοδο **1989-1994.** Πολλοί από τους χρήστες των αντικειμενοστραφών μεθόδων δεν ικανοποιούντο πλήρως από μία από αυτές τις γλώσσες μοντελοποίησης**,** δημιουργώντας **«**πολέμους μεθοδολογίας**».**

Στα μέσα της δεκαετίας του **90,** είχαν αρχίσει να εμφανίζονται νέες εκδόσεις των μεθοδολογιών αυτών**,** όπως η **Booch 93,** η συνεχιζόμενη εξέλιξη της **OMT (Object Modeling Technique)** και η **Fusion.** Οι μεθοδολογίες αυτές άρχισαν να ενσωματώνουν η μία τις τεχνικές της άλλης και άρχισαν να εμφανίζονται μεθοδολογίες που ήταν αρκετά υποσχόμενες όπως οι **OOSE (Object-oriented software engineering), OMT-2** και **Booch 93.** Κάθε μία από αυτές ήταν μία πλήρης μεθοδολογία και είχε τα δικά της ισχυρά σημεία**.** Με απλά λόγια**,** η **OOSE** ήταν μία προσέγγιση που βασιζόταν σε περιπτώσεις χρήσης και προσέφερε άριστη υποστήριξη για **business engineering** και ανάλυση απαιτήσεων**.** Η **OMT-2** ήταν ιδιαίτερη εκφραστική για ανάλυση και για πληροφοριακά συστήματα με έμφαση στα δεδομένα**.** Η **Booch 93** ήταν ιδιαίτερα εκφραστική κατά τις φάσεις του σχεδιασμού και της υλοποίησης για έργα**,** ενώ ήταν διάσημη για εφαρμογές που είχαν έμφαση στην τεχνολογία**.**

Η ανάπτυξη της **UML** ξεκίνησε τον Οκτώβριο του **1994** όταν οι **Grady Booch** και **Jim Rumbaugh** της **Rational Software Corporation** άρχισαν να δουλεύουν για την ενοποίηση των μεθοδολογιών **Booch (**O**bject-**O**riented** A**nalysis &** Δ**esign - OOAD)** και **OMT.** Δεδομένου του ότι οι δύο μεθοδολογίες ήδη αναπτύσσονταν ξεχωριστά και αναγνωρίζονταν ήδη ως οι κυρίαρχες αντικειμενοστραφείς μεθοδολογίες παγκόσμια**,** οι **Booch** και **Rumbaugh** ένωσαν τις δυνάμεις τους για να επιτευχθεί μία πλήρης ενοποίηση των μεθοδολογιών τους**.** Το πρώτο σχέδιο της έκδοση **0.8** της **Unified Method,** όπως ονομαζόταν τότε**,** εμφανίστηκε τον Οκτώβριο του **1995.** Στο φθινόπωρο του **1995** ο

**Ivar Jacobson** και η εταιρία του **Objectory** ενώθηκε με την **Rational** και την προσπάθεια ενοποίησης, συνδυάζοντας και τη μεθοδολογία **OOSE (Object-Oriented Software Engineering).** Το όνομα της **Objectory** χρησιμοποιείται πλέον στη **Rational** κυρίως για να δηλωθεί η συμβατή με τη **UML** διαδικασία που παρέχει, η **Rational Unified Process.** Ως βασικοί συγγραφείς των μεθοδολογιών **Booch, OMT** και **OOSE,** οι **Grady Booch, Jim Rumbaugh** και **Ivar Jacobson** είχαν το κίνητρο για να δημιουργήσουν μία ενοποιημένη γλώσσα μοντελοποίησης για τρεις λόγους.

Κατ' αρχάς οι μέθοδοι ήδη εξελίσσονταν προς τις άλλες ανεξάρτητα. Είχε νόημα να συνεχίσουν την προσπάθεια εξέλιξης μαζί, αντί να το κάνουν ο καθένας ξεχωριστά, απαλείφοντας έτσι την πιθανότητα για διαφορές που δε θα είχαν νόημα και απλά θα μπέρδευαν τους χρήστες. Δεύτερον, ενοποιώντας τη σημασιολογία και το συμβολισμό θα υπήρχε μία σταθερότητα στην αντικειμενοστραφή αγορά, που θα επέτρεπε στα έργα να χρησιμοποιήσουν μία ώριμη γλώσσα μοντελοποίησης και να αφήσουν όσους θα έφτιαχναν τα εργαλεία να δώσουν βάρος στο πώς θα παρέχουν τα χρήσιμα χαρακτηριστικά. Τρίτον, περίμεναν ότι η συνεργασία τους θα προσέφερε βελτιώσεις και στις τρεις προηγούμενες μεθόδους, βοηθώντας τους να συλλάβουν τα μαθήματα που είχαν ήδη πάρει και να αντιμετωπίσουν προβλήματα που δεν μπορούσε να χειριστεί ορθά καμία από τις προηγούμενες μεθοδολογίες.

Καθώς ξεκίνησαν τη διαδικασία ενοποίησης, έβαλαν τέσσερις στόχους για να εστιάσουν τις προσπάθειές τους:

**1** Να διευκολύνουν τη μοντελοποίηση των συστημάτων **(**όχι μόνο του λογισμικού**)** χρησιμοποιώντας αντικειμενοστραφείς έννοιες.

**2** Να δημιουργήσουν μία ρητή σύζευξη προς τα σημασιολογικά αλλά και τα εκτελέσιμα στοιχεία.

**3** Να αντιμετωπίσουν ζητήματα κλίμακας που είναι έμφυτα στα σύνθετα και κρίσιμα συστήματα.

**4** Να δημιουργήσουν μία γλώσσα μοντελοποίησης που θα μπορούν να χρησιμοποιήσουν τόσο άνθρωποι όσο και μηχανές.

Το να ανακαλύπτει κάποιος ένα συμβολισμό για χρήση στην αντικειμενοστραφή ανάλυση και σχεδιασμό δε διαφέρει και πολύ από το σχεδιασμό μίας προγραμματιστικής γλώσσας. Πρώτον πρέπει να αντιμετωπιστεί ένα πρόβλημα: θα πρέπει ο συμβολισμός να ενσωματώνει τον καθορισμό των απαιτήσεων; **(**Ναι, εν μέρει.**)** Θα πρέπει ο συμβολισμός να επεκτείνεται στο επίπεδο μίας οπτικής γλώσσας προγραμματισμού; **(**Όχι.**)** Δεύτερον, θα πρέπει να υπάρχει μία ισορροπία ανάμεσα σε απλότητα και εκφραστικότητα: ένας πολύ απλός συμβολισμός θα περιορίσει το πλήθος των προβλημάτων που θα μπορούν να επιλυθούν, ένας πολύ σύνθετος θα περιπλέξει τον απλό αναλυτή. Στην περίπτωση της ενοποίησης υπαρχουσών μεθοδολογιών, θα πρέπει να υπάρχει μία ευαισθησία ως προς την υπάρχουσα βάση χρηστών: αν υπάρχουν πολλές αλλαγές θα μπερδευτούν οι υπάρχοντες χρήστες. Αν δεν προωθηθεί ο συμβολισμός θα χαθεί η ευκαιρία να μπορεί να απευθυνθεί σε μεγαλύτερο σύνολο χρηστών. Ο ορισμός της **UML** προσπαθεί να πετύχει μέσα σε όλες αυτές τις περιοχές.

Οι προσπάθειες των **Booch, Rumbaugh** και **Jacobson** είχαν ως αποτέλεσμα την εμφάνιση της τεκμηρίωσης της **UML 0.9** και **0.91** τον Ιούνιο και τον Οκτώβριο του **1996.** Το **1996** οι συγγραφείς της **UML** ζήτησαν και έλαβαν σχόλια από την κοινότητα. Ενσωμάτωσαν τα σχόλια αλλά ήταν σαφές ότι ήταν απαραίτητη επιπλέον εστιασμένη προσοχή.

Το **1996** έγινε σαφές ότι αρκετοί οργανισμοί έβλεπαν τη **UML** ως στρατηγική επιλογή για τις εταιρίες τους. Μία αίτηση για προτάσεις **(Request for Proposal – RFP)** από το **Object Management Group (OMG)** ήταν ο καταλύτης ώστε οι οργανισμοί αυτοί να ενώσουν τις δυνάμεις τους και να παρουσιάσουν μία κοινή απάντηση στο **RFP.** Η **Rational** δημιούργησε την κοινοπραξία των εταίρων της **UML** με αρκετούς οργανισμούς που είχαν τη διάθεση να διαθέσουν τους πόρους για να δημιουργηθεί ένας ισχυρός ορισμός της **UML.** Εκείνοι που συμμετείχαν περισσότερο στον ορισμό της **UML** περιλαμβάνουν τους: **Digital Equipment Corp., HP, i-Logix , IntelliCorp, IBM, ICON Computing, MCI Systemhouse, Microsoft, Oracle, Rational Software, TI** και **Unisys.** Η συνεργασία αυτή παρήγαγε τη **UML,** μία γλώσσα μοντελοποίησης που ήταν καλά ορισμένη, εκφραστική, ισχυρή και γενικά εφαρμόσιμη.

Τον Ιανουάριο του **1997** η **IBM** με την **ObjecTime,** η **Platinum Technology,** η **Ptech,** οι **Taskon** και **Reich Technologies** και τέλος η **Softeam** υπέβαλλαν με τη σειρά τους απαντήσεις στο **RFP** του **OMG.** Οι εταιρίες αυτές ενώθηκαν με τους εταίρους της **UML** για να υποβάλλουν τις ιδέες τους και το αποτέλεσμα ήταν η **UML 1.1.** Η **UML 1.1** εστίασε στο να ξεκαθαριστεί η σημασιολογία της **UML 1.0** και να ενσωματωθούν οι συνεισφορές των νέων εταίρων. Αυτή είναι και η έκδοση που υιοθετήθηκε από το **OMG** ως επίσημο πρότυπο.**[3]** Από τότε υπήρξαν πολλές αναθεωρήσεις μέχρι το **2001** φτάνοντας στην έκδοση **1.5** η οποία διόρθωνε ελλείψεις και σφάλματα της αρχικής έκδοσης. **[4]**

Το **2005** το **OMG** προχώρησε σε γενικότερη ενημέρωση του προτύπου, υιοθετώντας την έκδοση **2.0** της **UML,** η οποία ουσιαστικά επανακαθορίζει τα διαγράμματα δραστηριοτήτων **(activity diagrams)** και εισάγει ένα νέο τύπο, τα Σύνθετα Διαγράμματα **(Composite Structures).** Η έκδοση **1.5** θα συνεχίσει να χρησιμοποιείται και να θεωρείται η επίσημη έκδοση μέχρι να ολοκληρωθούν και να επικυρωθούν τα τέσσερα συστατικά της **2.0.** Το πρώτο μέρος της έκδοσης **2.0** της **UML,** η υπερδομή, η οποία περιγράφει τα διαθέσιμα νέα διαγράμματα και τα στοιχεία μοντελοποίησης.

Από την έκδοση **2.2** και έπειτα, η **UML** περιλαμβάνει **14** διαγράμματα που χωρίζονται σε τρεις κατηγορίες: Δομικά **(Structure Diagrams),** Συμπεριφοράς **(Behavior Diagrams)** και Αλληλεπίδρασης **(Interaction Diagrams),** και είτε παρέχουν πληροφορίες Δομής **(Structure Diagrams),** είτε περιγράφουν τύπους Συμπεριφοράς **(Behavior & Interaction Diagrams).** **[5]**

Τα υπόλοιπα τρία μέρη της νέας έκδοσης έχουν επίσης υιοθετηθεί, αλλά δεν θεωρούνται ακόμα διαθέσιμα. Τα τρία μέρη είναι τα εξής:

⇒ Υποδομή της **UML 2.0 (UML 2.0 Infrastructure):** Η υποδομή ορίζει τις βασικές κλάσεις και διαμορφώνει τη βάση για την υπερδομή της **UML 2.0.**

⇒ Γλώσσα Περιορισμών Αντικειμένου της **UML 2.0 (UML 2.0 Object Constraint Language (OCL))**: Επιτρέπει τον προσδιορισμό συνθηκών και σταθερών.

⇒ Ανταλλαγή Διαγραμμάτων **UML 2.0 (UML 2.0 Diagram Interchange)**: Το μέρος αυτό επεκτείνει το μεταμοντέλο **UML** με τη χρήση ενός επιπρόσθετου πακέτου για πληροφορίες γραφικού περιεχομένου, επιτρέποντας με αυτόν τον τρόπο την ανταλλαγή και την αποθήκευση μοντέλων, καθώς και την αναπαράστασή τους στην αρχική τους μορφή.

Μέχρι το Αύγουστο του **2011**, το πρότυπο είχε φτάσει την έκδοση **2.4**.**[4]**

## *1.3 UML και Μοντελοποίηση Συστημάτων*

Γενικά, ως σύστημα εννοείται μια τεχνική ή κοινωνιολογική δομή που αποτελείται από μια ομάδα στοιχείων, τα οποία συνδυάζονται και αλληλεπιδρούν με τέτοιο τρόπο για να σχηματίσουν ένα λειτουργικό σύνολο.**[6]**

Η δόμηση ενός συστήματος, είτε πρόκειται για λογισμικό είτε πρόκειται για ένα πληροφοριακό σύστημα, αποτελεί μια πολύπλοκη διαδικασία. Μάλιστα, όσο πιο πολύπλοκο είναι το υπό δόμηση σύστημα, τόσο πιο σημαντική είναι η επικοινωνία ανάμεσα στον καθένα που συμμετέχει στη δημιουργία και την ανάπτυξή του. Ένας αποτελεσματικός τρόπος για να υπερκεραστεί αυτή η πολυπλοκότητα είναι η αφαιρετικότητα ώστε να απομονωθούν τα κυριότερα δομικά και λειτουργικά στοιχεία του συστήματος. Αυτή η λογική βρίσκεται και στο επίκεντρο της Μοντελοποίησης Συστημάτων.

**Η** σημασία της μοντελοποίησης είναι εξαιρετικά σπουδαία. Με τη βοήθεια των μοντέλων, υπάρχει η δυνατότητα για τη παρουσίαση μιας απλουστευμένης εκδοχής της πραγματικότητας. Φτιάχνουμε μοντέλα για να καταλάβουμε καλύτερα το σύστημα που πρόκειται να αναπτυχθεί, ειδικά σε περιπτώσεις μεγάλων, πολύπλοκων συστημάτων. Το μοντέλο του συστήματος προσφέρει γραφική απεικόνιση, προσδιορίζει τη δομή και τη συμπεριφορά του συστήματος και τέλος βοηθά στην οργάνωση της φάσης κατασκευής του συστήματος. Υπάρχουν κάποιες γενικές αρχές που εφαρμόζονται και ισχύουν κατά τη μοντελοποίηση συστημάτων:

⇒ Κατασκευάζουμε μοντέλα συστημάτων για να βοηθήσουμε στη κατανόησή τους

⇒ Ένα μοντέλο είναι μια απλοποιημένη αναπαράσταση ενός συστήματος

⇒ Δεν υπάρχει ένα μοναδικό μοντέλο που να καλύπτει όλες τις πλευρές του συστήματος. Διαφορετικοί τύποι μοντέλων προσεγγίζουν διαφορετικές πλευρές

⇒ Το ίδιο σύστημα μπορεί να αναπαρασταθεί σε διαφορετικά επίπεδα λεπτομέρειας από κάποιο μοντέλο.

Η μοντελοποίηση ενός συστήματος μπορεί να αυξήσει την αξιοπιστία και να μειώσει το κόστος ανάπτυξης, καθώς καθιστά ευκολότερο το χτίσιμο των συστημάτων.

Αυτό συμβαίνει διότι, τα τμήματα που προκύπτουν ως προϊόν μοντελοποίησης, μπορούν να επαναχρησιμοποιηθούν σε νέα συστήματα, ή μπορούν εύκολα να διαφοροποιηθούν ώστε να ταιριάξουν σε μεταβαλλόμενες απαιτήσεις, όπως η λειτουργική αναβάθμιση και αλλαγές πλατφόρμας. Με αυτόν τον τρόπο, ένα μοντέλο συστήματος μπορεί να ικανοποιήσει διαφορετικές ανάγκες.

Έτσι, η μοντελοποίηση των συστημάτων χρησιμοποιείται για να εξασφαλίσει ότι ένα αναπτυσσόμενο κομμάτι λογισμικού εξελίσσεται κατά τρόπο συνεκτικό με απώτερο στόχο την απλούστερη ενσωμάτωση του.

Η **UML** αποτελεί, πλέον την πιο διαδεδομένη γλώσσα μοντελοποίησης για την ανάπτυξη λογισμικού, καθώς χρησιμοποιείται από τους αναλυτές, τους σχεδιαστές και τους προγραμματιστές. Η **UML** δίνει στον κάθε συμμετέχοντα, από τον αναλυτή επιχειρήσεων μέχρι τον προγραμματιστή, ένα κοινό λεξιλόγιο για την επικοινωνία τους σχετικά με το σχεδιασμό λογισμικού. Μάλιστα. η χρήση της **UML** έχει επικρατήσει σε κάθε φάση του κύκλου ζωής ενός συστήματος, όπως θα εξηγηθεί και στο επόμενο Κεφάλαιο.

Γενικά, η **UML** επιτυγχάνει τη μοντελοποίηση των συστημάτων με μια σειρά εργαλείων όπως τα στοιχεία του μοντέλου **(model elements)**, τις συσχετίσεις **(relationships)**, τους μηχανισμούς **(mechanisms)**, τα διαγράμματα **(diagrams)** και τις αρχιτεκτονικές όψεις **(architectural views)**

Ένα μοντέλο **UML** δεν πρέπει να συγχέεται με το σετ διαγραμμάτων ενός συστήματος, αφού και το τελευταίο περιέχει **«**σημασιολογικές πληροφορίες**»** ώστε να καθοδηγεί σωστά τα στοιχεία και τα διαγράμματά του.

Είναι σημαντικό να γίνεται διάκριση μεταξύ ενός μοντέλου **UML** και του συνόλου των διαγραμμάτων του συστήματος. Ένα διάγραμμα είναι μια μερική γραφική αναπαράσταση του μοντέλου ενός συστήματος, ενώ το μοντέλο περιλαμβάνει ευρύτερες πληροφορίες. Οι κύριες εκφάνσεις ενός μοντέλου ενός συστήματος είναι τρεις:

- ⇒ Λειτουργικό Μοντέλο. Παρουσιάζει τη λειτουργικότητα του συστήματος από την οπτική γωνία του χρήστη. Θα πρέπει να λαμβάνει υπόψη το περιβάλλον του συστήματος.
- ⇒ Μοντέλο Αντικειμένων. Παρουσιάζει την δομή και την αρχιτεκτονική του συστήματος, χρησιμοποιώντας αντικείμενα, χαρακτηριστικά, λειτουργίες και συσχετίσεις.
- ⇒ Δυναμικό Μοντέλο. Παρουσιάζει την εσωτερική συμπεριφορά του συστήματος.

## *1.4 Κύρια στοιχεία & χαρακτηριστικά της UML*

### 1.4.1 Συστατικά της UML

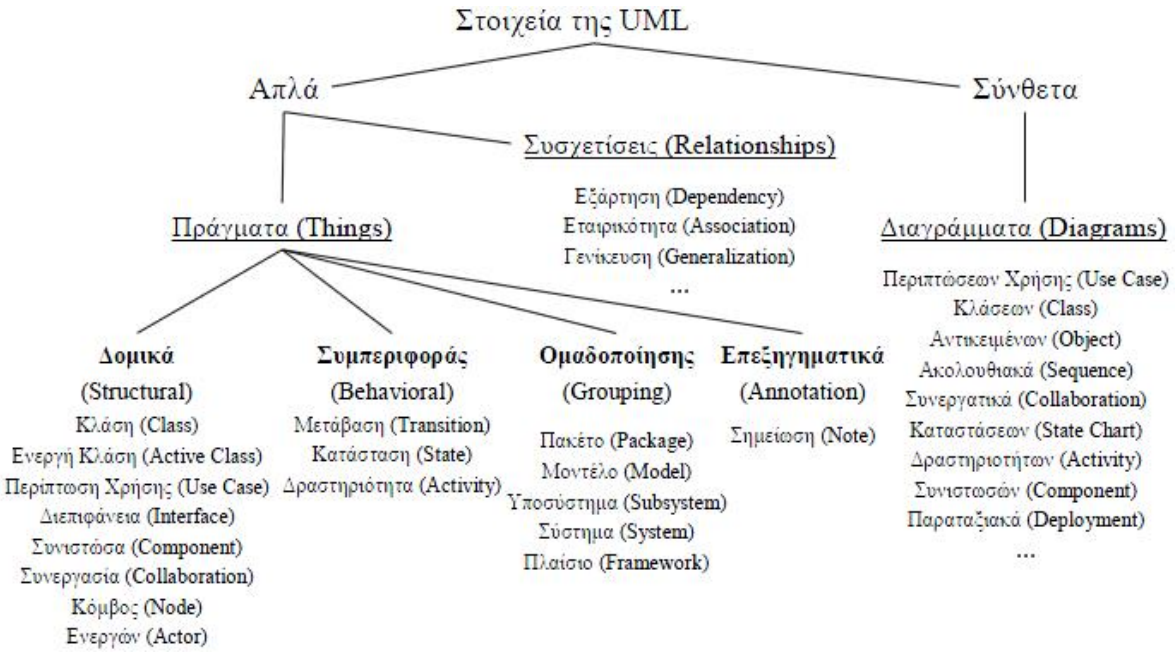
Αυτή τη στιγμή η **UML 2.4** καθορίζει **4** τμήματα προδιαγραφών.**[4]**

1. Το **Superstructure** που καθορίζει το συμβολισμό και τη σημασιολογία των διαγραμμάτων και των στοιχείων τους
2. Το **Infrastructure** που καθορίζει το πυρήνα του μεταμοντέλου πάνω στο οποίο στηρίζεται το **Superstructure**
3. Η γλώσσα **Object Constraint Language (OCL)** για τον καθορισμό των κανόνων για τα στοιχεία της **UML**
4. Το **UML Diagram Interchange (UMLDI)** που ορίζει τον τρόπο για την ομαλή και απρόσκοπτη ανταλλαγή εγγράφων που πληρούν το πρότυπο **UML (**που αναφέρονται ως **UML** μοντέλα**)** μεταξύ των διαφόρων εργαλείων λογισμικού

Στα πλαίσια της συγκεκριμένης πτυχιακής εργασίας**,** θα παρουσιαστεί το **Superstructure,** μαζί με κάποια στοιχεία από το **Infrastructure,** ώστε ο αναγνώστης να αποκτήσει μια σφαιρική άποψη των δυνατοτήτων της **UML** και του εννοιολογικού περιεχομένου που μπορεί να συμπεριλάβει μέσω των διαγραμμάτων **UML.**

### 1.4.1.1 Γενικά στοιχεία UML

Μια κατηγοριοποίηση των πιο σημαντικών στοιχείων της **UML,** παρουσιάζεται στο επόμενο σχήμα**,** όπου φαίνεται ότι τα στοιχεία της **UML** διακρίνονται σε Απλά και Σύνθετα**.** Τα απλά περιλαμβάνουν στοιχεία δομικά**,** συμπεριφοράς**,** ομαδοποίησης**,** επεξηγηματικά**,** καθώς και τις συσχετίσεις αυτών**.** Τα σύνθετα στοιχεία είναι ουσιαστικά τα διαγράμματα της **UML.[2]**



**Εικόνα 1 Κατηγοριοποίηση στοιχείων UML**

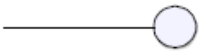
Αντικείμενο **(Object):** Οτιδήποτε πραγματικό ή αφηρημένο στο οποίο αποθηκεύουμε δεδομένα και έχει μεθόδους που τα διαχειρίζονται. Ένα αντικείμενο έχει κατάσταση **(state),** συμπεριφορά **(behavior)** και ταυτότητα **(identity).** Παραδείγματα: Τιμολόγιο, Πελάτης. **Τ**α αντικείμενα παριστάνονται με τον συμβολισμό των κλάσεων, με το όνομά τους υπογραμμισμένο.

Κλάση **(Class):** Σύνολο αντικειμένων με κοινή δομή και συμπεριφορά. Μια κλάση είναι μια κατηγορία αντικειμένων και ένα αντικείμενο είναι στιγμιότυπο **(instance)** μιας κλάσης. Παράδειγμα: Κλάση: εργαζόμενος **&** Αντικείμενα: μηχανικός, λογιστής, πωλητής κ.λ.π. **Ο**ι κλάσεις παριστάνονται με ένα παραλληλόγραμμο με τρία διαμερίσματα. Στο πρώτο υπάρχει το όνομα της κλάσης και στα άλλα δύο οι ιδιότητες και οι λειτουργίες. **Ε**ναλλακτικά, μπορεί να περιέχεται μόνο το όνομα της κλάσης.

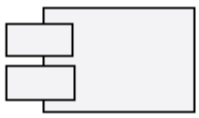

Ενεργή κλάση **(Active class):** Μια κλάση που περιγράφει μια διεργασία ή ένα νήμα εκτέλεσης και αλληλεπιδρά με άλλες.

Περίπτωση χρήσης **(Use case):** Μια λειτουργία που επιτελεί ένα σύστημα και είναι διαθέσιμη στο χρήστη. Είναι συμπεριφορά του συστήματος που συνεπάγεται τη συνεργασία ενός συνόλου αντικειμένων.

Διεπαφή **(Interface):** Ένα σύνολο από λειτουργίες που ορίζουν την εξωτερική συμπεριφορά ενός αντικειμένου.

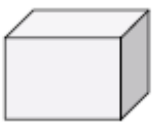


Συστατικό **(Component):** Ένα φυσικό και επαναχρησιμοποιήσιμο τμήμα ενός συστήματος, με λογική και φυσική υπόσταση που συνήθως υλοποιεί κάποιες διεπαφές **(interfaces)**

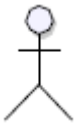


Συνεργασία **(Collaboration):** Η περιγραφή μιας διάδρασης μεταξύ ενός συνόλου αντικειμένων.

Κόμβος **(Node):** Ένας υπολογιστικός πόρος που έχει κάποια μνήμη και υπολογιστική ικανότητα, οπότε εκεί αποθηκεύεται ή**/**και εκτελείται το λογισμικό.

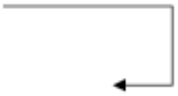


Ενεργών **(Actor):** Εξωτερική του συστήματος οντότητα που χρησιμοποιεί τη λειτουργικότητα και τις διεπαφές του.



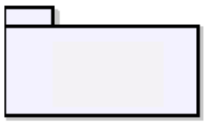
Κατάσταση **(State):** Μια συνθήκη ή περίπτωση στο χρόνο ζωής ενός αντικειμένου, όπου ικανοποιεί κάποιους περιορισμούς, εκτελεί κάποια δραστηριότητα ή αναμένει κάποιο γεγονός.

Μετάβαση **(Transition):** Μια σχέση μεταξύ δύο καταστάσεων ενός αντικειμένου που υποδηλώνει αλλαγή στην κατάσταση του αντικειμένου με την εμφάνιση ενός γεγονότος.

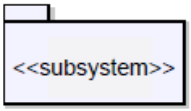


Δραστηριότητα **(Activity):** Μια εκτέλεση λειτουργίας κατά τη διάρκεια ζωής ενός αντικείμενου.

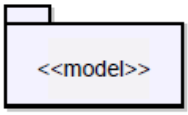
Πακέτο **(Package):** Ένα δομικό στοιχείο γενικής χρήσης για την οργάνωση άλλων δομικών στοιχείων, διαγραμμάτων ή και άλλων πακέτων της **UML** σε ομάδες.



Υποσύστημα **(Subsystem):** Μια μονάδα στην ιεραρχική αποσύνθεση ενός μεγάλου συστήματος. Επικοινωνεί με το περιβάλλον του μέσω διαπροσωπειών.

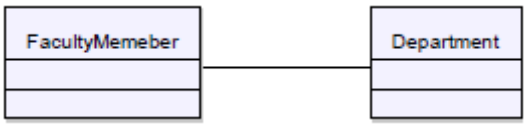



Μοντέλο **(Model):** Μια όψη του συστήματος.

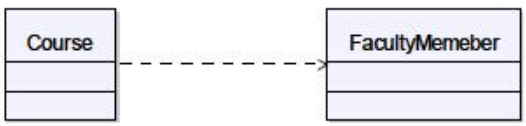



Σημείωση **(Note):** Ένα δομικό στοιχείο κειμενικού σχολιασμού για την περιγραφή ή επεξήγηση ενός άλλου δομικού στοιχείου ή μιας ομάδας δομικών στοιχείων. Αποτελεί σχόλιο ή επεξήγηση ή κείμενο αναφοράς.

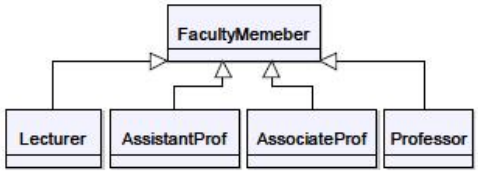
Συσχέτιση **(Association):** Μια δομική σχέση που περιγράφει ένα σύνολο συνδέσεων μεταξύ αντικειμένων.




Εξάρτηση **(Dependency):** Μια σχέση μεταξύ δυο δομικών στοιχείων, όπου μια αλλαγή στο πρώτο επιδρά στο δεύτερο.



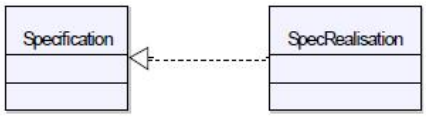
Γενίκευση**/** Κληρονομικότητα **(Generalization):**
Μια σχέση μεταξύ ενός δομικού στοιχείου **(**πατέρας**)** και ενός δεύτερου **(**παιδί**)** που εξειδικεύει το πρώτο**. Η** κληρονομικότητα **/** γενίκευση παριστάνεται με ένα βέλος από την ειδικότερη προς τη γενικότερη κλάση




Συσσωμάτωση **(Aggregation):** Μια σχέση μεταξύ δυο δομικών στοιχείων**,** όπου το πρώτο μπορεί να περιέχει το δεύτερο**.**

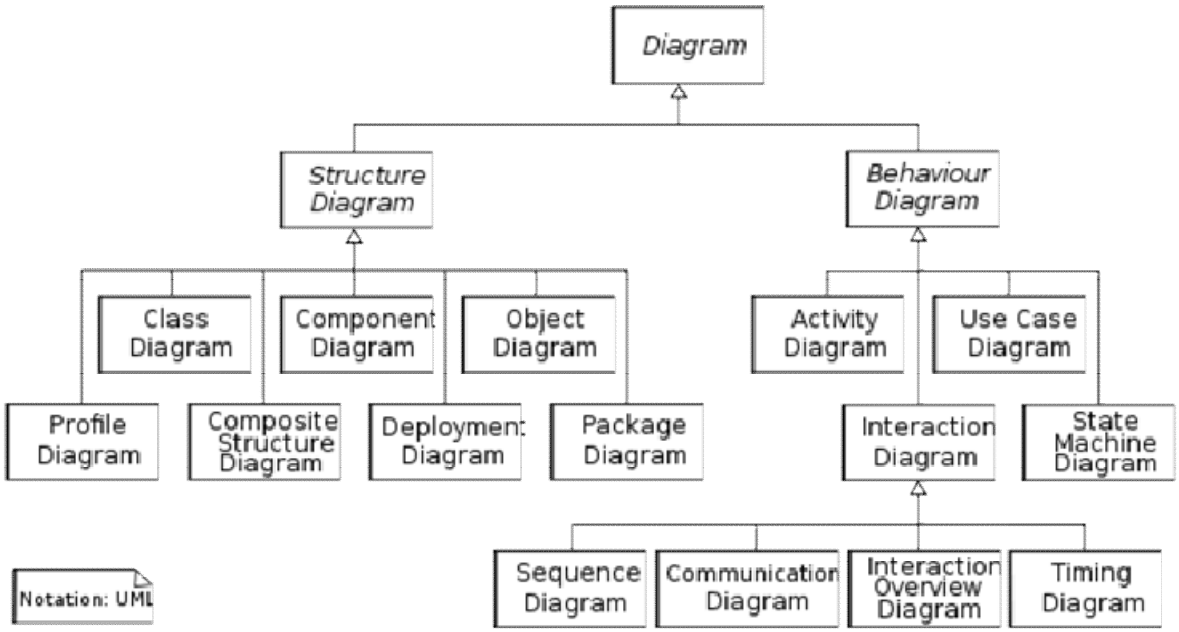

Σύνθεση **(Composition):** Μια σχέση μεταξύ δυο δομικών στοιχείων**,** όπου το πρώτο εντάσσεται αναπόσπαστα στο δεύτερο και δεν μπορεί να ανήκει σε κανένα άλλο του ιδίου τύπου**.**


Πραγματοποίηση **(Realization):** Μια σχέση μεταξύ δυο δομικών στοιχείων**,** όπου το πρώτο προδιαγράφει μια συμπεριφορά και το δεύτερο την υλοποιεί**.**




### 1.4.1.2 Διαγράμματα της UML

Από την έκδοση **2.2** και έπειτα της **UML,** η γλώσσα περιλαμβάνει **14** διαγράμματα που χωρίζονται σε δυο βασικές κατηγορίες**.** Επτά τύποι διαγραμμάτων αναπαριστούν δομική πληροφορία και άλλα επτά αναπαριστούν γενικούς τύπους συμπεριφοράς**,** συμπεριλαμβανομένων και τεσσάρων τύπων διαγραμμάτων που αντιπροσωπεύουν διαφορετικές πτυχές αλληλεπίδρασης**.[5]**




**Εικόνα 2** **Τα διαγράμματα της UML και η κατηγοριοποίηση τους**

Η **UML** δεν περιορίζει τους τύπους στοιχείων σε συγκεκριμένους τύπους διαγραμμάτων. Αντιθέτως, οποιοδήποτε στοιχείο **UML** μπορεί να εμφανιστεί σε οποιοδήποτε τύπο διαγράμματος, αν και αυτή η ευελιξία έχει κάπως περιοριστεί στην **UML 2.0**.

Στη συνέχεια ακολουθεί μια συνοπτική περιγραφή των τύπων διαγραμμάτων της **UML**. Αναλυτική, περιγραφή και παραδείγματα περιλαμβάνονται στο Κεφάλαιο **3**.

Δομικά διαγράμματα

Το δομικά διαγράμματα δίνουν έμφαση στα στοιχεία που πρέπει να είναι παρόντα στο υπό μοντελοποίηση σύστημα. Καθώς τα συγκεκριμένα διαγράμματα αναπαριστούν τη δομή του συστήματος, συναντώνται ιδιαιτέρως συχνά για τη περιγραφή της αρχιτεκτονικής λογισμικού του συστήματος.

Συνοπτικά τα δομικά διαγράμματα είναι:

- ⇒ Διαγράμματα Κλάσης **(Class Diagrams)**. Περιγράφουν τη δομή ενός συστήματος, παρουσιάζοντας τις κλάσεις του συστήματος, τις ιδιότητες τους και τις σχέσεις τους με άλλες κλάσεις.
- ⇒ Διαγράμματα Συστατικών **(Component Diagrams)**. Περιγράφουν πώς ένα σύστημα λογισμικού διαιρείται σε επιμέρους συστατικά και παρουσιάζει τις εξαρτήσεις μεταξύ αυτών.
- ⇒ Διαγράμματα Σύνθετης Δομής **(Composite Structure Diagrams)**. Περιγράφουν την εσωτερική δομή μιας κλάσης και τις συνεργασίες που καθιστά δυνατές.
- ⇒ Διαγράμματα Ανάπτυξης **(Deployment Diagrams)**. Περιγράφουν το υλικό που χρησιμοποιείται σε υλοποιήσεις του συστήματος καθώς και τα

περιβάλλοντα εκτέλεσης και τα αντικείμενα που χρησιμοποιήθηκαν στο υλικό.

⇒ Διαγράμματα Αντικειμένων **(Object Diagrams).** Δείχνουν μια πλήρη ή μερική άποψη της δομής ενός παραδείγματος μοντελοποιημένου συστήματος σε συγκεκριμένη χρονική στιγμή.

⇒ Διαγράμματα Πακέτων **(Package Diagrams).** Περιγράφουν πώς ένα σύστημα χωρίζεται σε λογικές ομάδες, δείχνοντας τις εξαρτήσεις μεταξύ αυτών των ομάδων.

⇒ Διάγραμμα Προφίλ **(Profile Diagrams).** Λειτουργούν σε επίπεδο μετα-μοντέλου ώστε να δείξουν τα στερεότυπα ως κλάσεις με το στερεότυπο **"stereotype"**, και τα προφίλ ως πακέτα με το στερεότυπο **"profile".**

Διαγράμματα Συμπεριφοράς

Τα διαγράμματα συμπεριφοράς τονίζουν τι πρέπει να συμβεί στο υπό μοντελοποίηση σύστημα. Δεδομένου ότι τα διαγράμματα αυτά απεικονίζουν τη συμπεριφορά ενός συστήματος, χρησιμοποιούνται ευρέως για να περιγράψουν τη λειτουργικότητα των συστημάτων λογισμικού.

Συνοπτικά τα διαγράμματα συμπεριφοράς είναι:

⇒ Διαγράμματα Δραστηριοτήτων **(Activity Diagrams).** Περιγράφουν βήμα προς βήμα, την επιχειρηματική και επιχειρησιακή ροή εργασίας των συνιστωσών ενός συστήματος. Ένα διάγραμμα δραστηριότητας δείχνει τη συνολική ροή του ελέγχου.

⇒ Διαγράμματα σταθερής κατάστασης **(State Machine Diagrams).** Περιγράφουν τις καταστάσεις και τις μεταβάσεις μεταξύ των καταστάσεων του συστήματος.

⇒ Διαγράμματα Περιπτώσεων **(Use case Diagrams).** Περιγράφουν τη λειτουργικότητα που παρέχεται από ένα σύστημα όσον αφορά τους **actors**, τους στόχούς τους ως περιπτώσεις χρήσης, και τις εξαρτήσεις μεταξύ των περιπτώσεων χρήσης.

Διαγράμματα Αλληλεπίδρασης

Τα διαγράμματα αλληλεπίδρασης, είναι ένα υποσύνολο των διαγραμμάτων συμπεριφοράς, και δίνουν έμφαση στη ροή του ελέγχου και των δεδομένων μεταξύ των στοιχείων του υπό μοντελοποίηση συστήματος.

⇒ Διάγραμμα Επικοινωνίας **(Communication Diagrams).** Δείχνουν τις αλληλεπιδράσεις μεταξύ των αντικειμένων ή τμημάτων του συστήματος από την άποψη της αλληλουχίας μηνυμάτων. Αντιπροσωπεύουν ένα συνδυασμό πληροφοριών που προκύπτουν από διαγράμματα Κλάσης, Αλληλουχίας και Περιπτώσεων Χρήσης, που περιγράφουν τόσο τη στατική όσο και τη δυναμική συμπεριφορά του συστήματος.

⇒ Διάγραμμα Επισκόπησης Αλληλεπίδρασης **(Interaction Overview Diagram).** Παρέχουν μια γενική εικόνα στην οποία οι κόμβοι αντιπροσωπεύουν διαγράμματα επικοινωνίας.

⇒ Διαγράμματα Αλληλουχίας **(Sequence Diagrams).** Δείχνουν πώς τα αντικείμενα επικοινωνούν μεταξύ τους με σειρά μηνυμάτων. Επίσης, δείχνουν τη διάρκεια ζωής των αντικειμένων σε σχέση με αυτά τα μηνύματα.

⇒ Διαγράμματα χρονισμού **(Timing Diagrams).** Ένας συγκεκριμένος τύπος του διαγράμματος αλληλεπίδρασης, όπου η εστίαση είναι στους χρονικούς περιορισμούς.

## 1.4.2 Πλεονεκτήματα & Μειονεκτήματα

Είναι μια γλώσσα καθορισμένη, όπου κάθε στοιχείο της έχει συγκεκριμένο εννοιολογικό περιεχόμενο, γεγονός που δεν επιτρέπει παρερμηνείες ως προς τη κατανόηση του μοντέλου του συστήματος. Επίσης, η σήμανση της είναι απλή, κάτι που την κάνει κατανοητή.

Είναι μια γλώσσα επεκτάσιμη και ευέλικτη, που σημαίνει ότι μπορεί να χρησιμοποιηθεί για τη μοντελοποίησης μεγάλων συστημάτων με ακρίβεια αλλά και μικρών συστημάτων χωρίς περιττές λεπτομέρειες. Γενικά, μπορεί να περιγράψει κάθε χαρακτηριστικό ενός συστήματος, επιτρέποντας έτσι την ανάλυση και επανασχεδιασμό διευκολύνοντας την ανακάλυψη πιθανών αδυναμιών.

Επίσης, είναι αποτέλεσμα συσσωρευμένης **15**ετούς εμπειρίας στη μοντελοποίηση αντικειμενοστραφών συστημάτων λογισμικού, συνδυάζοντας όλες τις βέλτιστες λύσεις που υπήρχαν μέχρι τότε **(Booch, OMT and OOSE).**

Κυριότερα όμως, αποτελεί γλώσσα πρότυπο, που ελέγχεται από το **OMG**, που αποτελεί ένα **group** ανοικτών προτύπων που υποστηρίζεται από πολυάριθμους ακαδημαϊκούς και κατασκευαστές. Το **OMG** εγγυάται τη "μεταφερσιμότητα" της **UML** και τη συμβατότητά της ανεξαρτήτως πλατφόρμας, χωρίς να εξαρτάται συγκεκριμένα από κάποιο εμπορικό προϊόν. Πρακτικά, αυτό σημαίνει ότι η **UML** μπορεί να χρησιμοποιηθεί για να μοντελοποιήσει σχεδόν οποιοδήποτε τύπο εφαρμογής που "τρέχει" σε οποιοδήποτε συνδυασμό υλικού, λειτουργικού συστήματος, γλώσσας προγραμματισμού και δικτύου δεδομένων.

Το γεγονός ότι αποτελεί πρότυπο σημαίνει ότι επιτρέπει τη συνεργασία μεταξύ πολλών κατασκευαστών ενός συστήματος λογισμικού, δίνοντάς τους ένα κοινό και κατανοητό τρόπο για να μοιράζονται τα σχέδιά τους. Σε άλλη περίπτωση, επιτρέπει τη σύντομη "επάνοδο" σε ένα μοντέλο συστήματος, ακόμα και αν έχει μεσολαβήσει μεγάλη περίοδος από την εποχή κατασκευής του.

Μεγάλο της πλεονέκτημα είναι ότι υποστηρίζεται από εργαλεία **Computer-Aided Software Engineering (CASE)** που επιτρέπουν την αυτόματη δημιουργία κώδικα βάσει

των **UML** διαγραμμάτων. Ο σχεδιαστής το μόνο που έχει να κάνει είναι να σχεδιάσει με λεπτομέρεια τα διαγράμματα και τα εργαλεία **CASE** θα δημιουργήσουν τον αντίστοιχο κώδικα. Βέβαια, σε αυτή τη περίπτωση δεν επιτρέπεται απόκλιση από τους κανόνες σύνταξης της **UML**, ή πιο σωστά από τους κανόνες που εφαρμόζει το κάθε **CASE** εργαλείο, αφού τότε θα υπάρχει απόκλιση μεταξύ του παραγόμενου κώδικα και των απαιτήσεων που έχουν αρχικά τεθεί και έχουν μοντελοποιηθεί με το διάγραμμα.

Σε κάθε περίπτωση η **UML** υπερτερεί και για ένα αριθμό από τεχνικούς λόγους, όπως το γεγονός ότι αντί για οντότητες χρησιμοποιεί κλάσεις, διευκολύνοντας τη κατανόηση του συστήματος καθώς φαίνονται ξεκάθαρα οι σχέσεις μεταξύ των κλάσεων.

Η **UML** μπορεί να χρησιμοποιηθεί για τη μοντελοποίηση **middleware layers** κάτι ιδιαίτερα χρήσιμο σε μεγάλα, σύνθετα συστήματα λογισμικού. Επίσης έχει βασιστεί πάνω στο μετά-μοντέλο **Microsoft Operating Framework (MOF)**, που είναι μια σειρά από οδηγίες, για τη δημιουργία νέων, αξιόπιστων και αποδοτικών υπηρεσιών.

Τέλος, τα διάφορα **profiles** της **UML (**αποτελούν υποσύνολα της **UML** για που εξυπηρετούν ειδικούς σκοπούς**)**, παρέχουν τη δυνατότητα μοντελοποίησης συστημάτων συναλλαγής **(transactional)**, πραγματικού χρόνου **(real-time)** και ανεκτικών σε σφάλματα **(fault-tolerant)** συστημάτων, με ένα πιο φυσικό και κατανοητό τρόπο.

Πέρα όμως από τα αδιαμφισβήτητα πλεονεκτήματα, υπάρχουν και κάποια μειονεκτήματα.

Η **UML** δεν καθορίζει κάποιο συγκεκριμένο **format** αρχείων, έτσι ο κάθε κατασκευαστής εργαλείου **UML (**μεταξύ αυτών και οι κατασκευαστές **CASE** εργαλείων**)**, αποθηκεύει την αναπαράσταση του **UML** μοντέλου του εργαλείου του με δικό του τρόπο.

Γενικά, ο σχεδιασμός των διαγραμμάτων καταλαμβάνει πολύ χρόνο, κάτι που πρέπει να αντικατοπτρίζεται και στο παραγόμενο κώδικα. Πολλές φορές, όταν το σύστημα είναι σχετικά απλό, δεν αξίζει να δαπανηθεί χρόνος στη κατασκευή των διαγραμμάτων.

Επιπλέον, το γεγονός ότι παρέχει τη δυνατότητα για μεγάλη λεπτομέρεια των δυνατοτήτων του συστήματος, πολλές φορές οδηγεί τους σχεδιαστές, στο να αποτυπώνουν κάθε πιθανή λεπτομέρεια, χάνοντας έτσι το νόημα των διαγραμμάτων, που είναι η απλουστευμένη οπτικοποίηση ενός σύνθετου συστήματος.

### 1.4.3 Εναλλακτικές λύσεις

Υπάρχουν δύο ανταγωνιστικές τάσεις στο χώρο της μοντελοποίησης **-** η γλώσσα **UML (**ή κλώνοι της με παρόμοιο συμβολισμό**)** και οι γλώσσες **DSL (Domain Specific Languages)**.

**UML** προσέγγιση προσπαθεί να μοντελοποιήσει κάθε δυνατό πραγματικό σενάριο κόσμο με μια ενοποιημένη γλώσσα μοντελοποίησης, ενώ η προσέγγιση **DSL** τείνει να δημιουργεί μια νέα ειδικού σκοπού γλώσσα για κάθε περιοχή ενδιαφέροντος.

Έτσι, πολύ συχνά απαντώνται τα **E-R** διαγράμματα **(Entity - Relationship)** για τη δόμηση βάσεων δεδομένων.**[6]** Στην τεχνολογία λογισμικού, ένα μοντέλο Οντοτήτων-Συσχετίσεων **(ER** μοντέλο για συντομία**)** είναι μια αφηρημένη και εννοιολογική αναπαράσταση των δεδομένων. Η **E-R** είναι μια μέθοδος μοντελοποίησης βάσεων δεδομένων, που χρησιμοποιείται για την παραγωγή ενός τύπου του εννοιολογικού σχήματος ή ενός μοντέλου σημασιολογικών δεδομένων του συστήματος, καθώς και των σχετικών απαιτήσεων με μια **top-down** μεθοδολογία.

Επίσης μεγάλη αποδοχή έχουν τα **Petri Net**ς που αφορούν τα ασύγχρονα συστήματα.**[7]** Τα δίκτυα **Petri** προσφέρουν μια γραφική αναπαράσταση για σταδιακές διαδικασίες που περιλαμβάνουν επιλογή, επανάληψη, και ταυτόχρονη εκτέλεση. Σε αντίθεση με πρότυπα όπως το **BPMN** και **EPC**, τα δίκτυα **Petri** διαθέτουν ένα ακριβή μαθηματικό ορισμό για τα εκτελεστικά στοιχεία τους, με μια καλά αναπτυγμένη μαθηματική θεωρία για την ανάλυση διαδικασιών.

Η **SysML** για συστήματα μηχανικής.**[8]** Η **Systems Modeling Language (SysML)** είναι μια γενικής χρήσης γλώσσα μοντελοποίησης συστημάτων για εφαρμογές μηχανικής. Υποστηρίζει την δημιουργία προδιαγραφών, την ανάλυση, το σχεδιασμό, την επαλήθευση και την επικύρωση ενός μεγάλου εύρους συστημάτων. Η **SysML** ορίζεται ως μια επέκταση ενός υποσυνόλου της **UML**, που χρησιμοποιεί το μηχανισμό **profile** της **UML**. Η **SysML**, όπως και η **UML**, αποτελεί πρότυπο του **OMG**.

Η **Event-driven Process Chain (EPC)** είναι ένας τύπος **flowchart** που χρησιμοποιείται για τη μοντελοποίηση της επιχειρηματικής διαδικασίας.**[9]** Η **EPC** μπορεί να χρησιμοποιηθεί για τη διαμόρφωση ενός σχεδίου διαχείρισης επιχειρηματικών πόρων **(ERP)** και για τη βελτίωση των επιχειρηματικών διαδικασιών.

Η **Agile Modeling (AM)** είναι μια μεθοδολογία που βασίζεται στην αποτελεσματική μοντελοποίηση και τεκμηρίωση συστημάτων λογισμικού.**[10]** Σε υψηλό επίπεδο η **AM** είναι μια συλλογή των βέλτιστων πρακτικών μοντελοποίησης. Σε πιο αναλυτικό επίπεδο **AM** είναι μια συλλογή από αξίες, αρχές και πρακτικές για το λογισμικό προσομοίωσης που μπορούν να εφαρμοστούν σε ένα έργο ανάπτυξης λογισμικού κατά τρόπο αποτελεσματικό.

Το **Business Process Model and Notation (BPMN)** αποτελεί ένα πρότυπο για την μοντελοποίηση των επιχειρηματικών διαδικασιών που παρέχει μια γραφική παράσταση για τον καθορισμό των επιχειρηματικών διαδικασιών.**[11]** Βασίζεται σε μια τεχνική διαγραμμάτων ροής, πολύ παρόμοια με τα διαγράμματα δραστηριότητας της **UML**. Ο στόχος της **BPMN** είναι η υποστήριξη της διαχείρισης επιχειρηματικών διαδικασιών, τόσο για τεχνικούς χρήστες όσο και για επιχειρηματικούς χρήστες, παρέχοντας ένα συμβολισμό που είναι πιο διαισθητικός για τους επιχειρησιακούς χρήστες. **Το BPMN**, όπως και η **UML**, αποτελεί πρότυπο του **OMG**.

### 1.4.4 Βαθμός χρήσης της UML σήμερα

Η **UML** μπορεί να εφαρμοστεί σε πολλούς τομείς, όπως στα ενσωματωμένα συστήματα, τις δικτυακές εφαρμογές, τις εμπορικές εφαρμογές κλπ. Μπορεί να χρησιμοποιηθεί για την μοντελοποίηση ολόκληρου του συστήματος, ανεξαρτήτως γλώσσας προγραμματισμού και πλατφόρμας. Είναι μια γραφική γλώσσα για την οπτικοποίηση, προσδιορισμό, την κατασκευή, και την τεκμηρίωση των πληροφοριών σχετικά με τα συστήματα λογισμικού. Η **UML** παρέχει έναν τυποποιημένο τρόπο περιγραφής ενός συστήματος, που καλύπτει τόσο τις επιχειρηματικές διαδικασίες και τις λειτουργίες του συστήματος, όσο και κλάσεις γραμμένες σε συγκεκριμένες γλώσσες προγραμματισμού, σχήματα βάσεων δεδομένων, καθώς και επαναχρησιμοποιήσιμα τμήματα λογισμικού.

Για τους παραπάνω λόγους η **UML** έχει αποδειχθεί χρήσιμη σε κλάδους όπως:

- ⇒ Πληροφοριακά συστήματα επιχειρήσεων
- ⇒ Τραπεζικές και χρηματοοικονομικές υπηρεσίες
- ⇒ Τηλεπικοινωνίες
- ⇒ Μεταφορές
- ⇒ Αμυνα
- ⇒ Λιανικό εμπόριο
- ⇒ Ιατρικά ηλεκτρονικά
- ⇒ Κατανεμημένες **web** υπηρεσίες
- ⇒ κ.α.

Επιπλέον, σχετικές έρευνες που έχουν πραγματοποιηθεί αναφέρουν:

- ⇒ Έρευνα των **Scottish Developers** το **2005,** αναφέρει χρήση της **UML** από **30%** των προγραμματιστών. **[12]**
- ⇒ Το περιοδικό **Computerworld,** το Μάρτιο του **2005** επίσης αναφέρει **33%.[13]**
- ⇒ Στην Αυστρία **to 2002** επίσης αναφέρεται **30%** χρήση της **UML** μικρές και πολύ μικρές επιχειρήσεις.**[14]**
- ⇒ Άλλη έρευνα του **2002,** της **BZ Research** αναφέρει επίσης **32%.[15]**
- ⇒ **Το** Μάιο του **2006** το περιοδικό **Communications of the ACM** πραγματοποίησε αντίστοιχη έρευνα για τη χρήση της **UML.[16]** Μάλιστα οι περισσότεροι συμμετέχοντες βρέθηκαν μέσω του **OMG,** συνεπώς τα αποτελέσματα αναφέρονται σε προγραμματιστές που γνωρίσουν τα πρότυπα του **OMG** και όχι γενικότερα σε όλους τους προγραμματιστές. Τα αποτελέσματα κατέδειξαν:

- Μόνο τα Διαγράμματα Κλάσης χρησιμοποιούνται αρκετά συχνά, ενώ τα διαγράμματα Αλληλουχίας και Περιπτώσεων χρησιμοποιούνται από τους μισούς. Το λιγότερο χρησιμοποιούμενο διάγραμμα είναι το **Collaboration** διάγραμμα (το **25%** των **UML** χρηστών δεν το έχει χρησιμοποιήσει ποτέ).
- Σε επίπεδο επικοινωνίας με τον πελάτη, η **UML** βοήθησε σε μέτριο βαθμό σύμφωνα με το **55%** των ερωτηθέντων.
- Οι δύο κυριότεροι λόγοι που αναφέρθηκαν για τη μη ευρεία χρήση της **UML** ήταν: **(a)** ότι δεν είναι κατανοητή από τους αναλυτές, και **(b)** ότι δεν προσθέτουν ιδιαίτερη αξία ώστε να δικαιολογούν το κόστος αγοράς ενός **uml** εργαλείου.
- Οι περιγραφές **Use Case** χρησιμοποιούνται περισσότερο για την επαλήθευση των προδιαγραφών των πελατών.
- Από αυτούς που δηλώσαν ότι δεν χρησιμοποιούν τη **UML**, αναφέρθηκε ως κυριότερος λόγος ότι υπάρχουν πολύ λίγοι άνθρωποι που είναι γνώστες της **UML**

Επιπλέον, ο **Booch**, ένας εκ των "πατεράδων" της **UML**, θεωρεί ότι ο βαθμός υιοθέτησης της **UML** είναι περίπου στο **10%**.**[17]** Συγκεκριμένα, δηλώνει ότι

**"10%** δεν είναι απαραίτητα κάτι κακό, αν λάβουμε υπόψη την παγκόσμια αγορά προγραμματιστών που σύμφωνα με την **IDC** είναι περίπου **13** εκατομμύρια. Δέκα τοις εκατό είναι ένα συμπαθητικό υγιές νούμερο. Αποτελεί μεγάλη διείσδυση, με την έννοια ότι έχουμε να κάνουμε εδώ με την κοινότητα των ανθρώπων που σχετίζονται με το σχεδιασμό και την αφαιρετικότητα, και αν θέλουμε να περιοριστούμε μόνο στην εν λόγω κοινότητα, θα τολμούσα να πω ότι η **UML** έχει μια τεράστια διείσδυση."

# Κεφάλαιο 2: UML και Ενοποιημένη Διεργασία ανάπτυξης αντικειμενοστραφών συστημάτων λογισμικού

Τη τελευταία δεκαετία η κυρίαρχη τάση στην ανάπτυξη λογισμικού είναι ο αντικειμενοστραφής προγραμματισμός, σύμφωνα με τον οποίο η ανάπτυξη των πληροφοριακών συστημάτων στηρίζεται σε αντικείμενα. Τα βασικά πλεονεκτήματα αυτής της προσέγγισης είναι η επαναχρησιμοποίηση τμημάτων κώδικα, τα οποία χαρακτηρίζονται από την απλότητα τους και την αξιοπιστία τους.

Η βασική μέθοδος ανάπτυξης αντικειμενοστραφών συστημάτων λογισμικού είναι η "Ενοποιημένη Διεργασία" **(Unified Process)** καθώς και η Ενοποιημένη Γλώσσα Μοντελοποίησης, η γνωστή **UML**, που αποτελεί βασικό πυλώνα της Διεργασίας και αναπτύχθηκε παράλληλα με αυτή. Συνεπώς, η παρουσίαση της Ενοποιημένης Διεργασίας και των βασικών στοιχείων της, αποτελεί το μέσο για τη βαθύτερη κατανόηση και τη πληρέστερη παρουσίαση της **UML**.

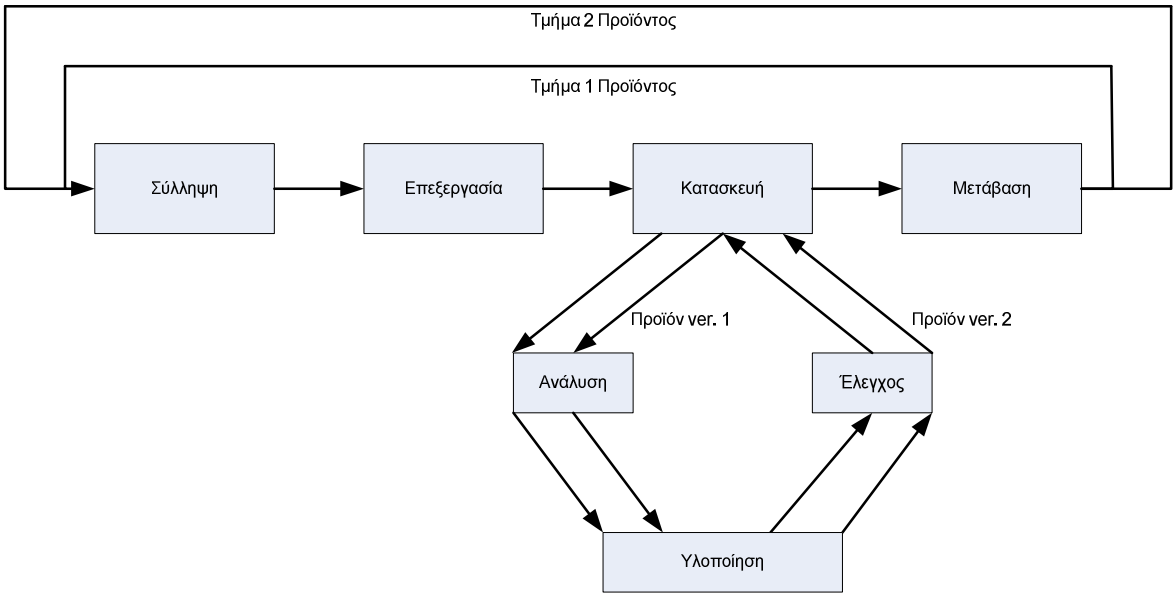
Η Ενοποιημένη Διεργασία προέκυψε ως λύση στο πρόβλημα των υψηλών απαιτήσεων στην υλοποίηση λογισμικού. Οι τελικοί χρήστες και η αγορά απαιτούσε συνεχώς πιο πολύπλοκα συστήματα και σε πιο σύντομο χρονικό διάστημα, εν μέρει ως αποτέλεσμα και της εξέλιξης των υπολογιστικών δυνατοτήτων των προσωπικών υπολογιστών και της εξάπλωσης του **Internet**.

Αυτή η απαίτηση για σύνθετα συστήματα σε πιο σύντομα χρονικά διαστήματα δεν μπορούσε να καλυφθεί από τις παρωχημένες μεθόδους ανάπτυξης λογισμικού, όπως ο ακολουθιακός προγραμματισμός, συνεπώς οι προγραμματιστές στράφηκαν και υιοθέτησαν τον αντικειμενοστραφή προγραμματισμό. Όμως ο αντικειμενοστραφής προγραμματισμός από μόνος του μπορεί να "κουβαλάει" πλεονεκτήματα όπως αναφέρθηκε παραπάνω, αλλά δεν τα "παρέχει", με την έννοια ότι απαιτείται να υιοθετηθεί και μια προσέγγιση επαναληπτικού προγραμματισμού, που είναι το κλειδί για την εκμετάλλευση των πλεονεκτημάτων του αντικειμενοστραφούς προγραμματισμού.

Σε αυτό ακριβώς το σημείο βρίσκεται η ουσία και η χρησιμότητα της Ενοποιημένης Διεργασίας, η οποία είναι μια διαδικασία, ένα πλαίσιο ανάπτυξης λογισμικού που απαρτίζεται από ένα σύνολο ενεργειών για τη μετατροπή των απαιτήσεων του χρήστη σε ένα σύστημα λογισμικού. Όμως το λογισμικό δεν προκύπτει ξαφνικά στο τέλος μιας μακράς διαδικασίας, αλλά αντιθέτως είναι το αποτέλεσμα μιας σταδιακής και επαναληπτικής διαδικασίας ανάπτυξης. Σε κάθε επανάληψη παράγεται ένα τμήμα λογισμικού, το οποίο καλύπτει ένα υποσύνολο των απαιτήσεων του χρήστη και αλληλεπιδρά με άλλα τμήματα λογισμικού μέσω καλά καθορισμένων διεπαφών. Η ενοποιημένη διεργασία χρησιμοποιεί την ενοποιημένη γλώσσα μοντελοποίησης **UML** κατά τη φάση σχεδίασης του συστήματος λογισμικού.

## *2.1 Κύκλος ζωής Ενοποιημένης Διεργασίας και επανάληψη φάσεων*

Η Ενοποιημένη Διεργασία εμπεριέχει την επανάληψη μιας σειράς κύκλων εργασιών που σταδιακά οδηγούν στο ολοκληρωμένο, τελικό σύστημα. Κάθε κύκλος εργασιών περιλαμβάνει τέσσερις φάσεις που είναι: η σύλληψη **(inception)**, η επεξεργασία **(elaboration)**, η κατασκευή **(construction)** και η μετάβαση **(transition)**. Οι φάσεις αυτές αναλύονται στις επόμενες ενότητες. Στο τέλος κάθε κύκλου, παράγεται ένα λογισμικό που μπορεί να είναι το τελικό σύστημα και προορίζεται για τους τελικούς χρήστες ή ένα τμήμα του συστήματος που προορίζεται για χρήση από τους κατασκευαστές του συστήματος. Κάθε φάση χωρίζεται σε επαναλήψεις, αν και η φάση της κατασκευής διαθέτει σίγουρα επαναλήψεις που ενσωματώνουν το γνωστό κύκλο ζωής της ανάλυσης, υλοποίησης και ελέγχου του λογισμικού.**[2]**




**Εικόνα 3** **Φάσεις και επαναλήψεις του κύκλου ζωής της υλοποίησης συστήματος**

Κατά τη πρώτη φάση της Σύλληψης, ο στόχος είναι να προκύψει μια γενική εικόνα για το σύστημα και να καθοριστεί το γενικό επιχειρηματικό σκεπτικό πίσω από τη δημιουργία του συστήματος. Πρέπει να απαντηθούν ερωτήματα όπως:

- Ø Ποιες θα είναι οι βασικές λειτουργίες του συστήματος και ποιοι οι δυνητικοί χρήστες του;
- Ø Ποια θα μπορούσε να είναι η αρχιτεκτονική του ώστε να προκύπτει η απαιτούμενη λειτουργικότητα με το ελάχιστο δυνατό ρίσκο;
- Ø Ποιο το κόστος υλοποίησης και ποιο το πιθανό εμπορικό κέρδος από το σύστημα;

Κατά τη φάση της Επεξεργασίας θα πρέπει να συλλεχθούν λεπτομερείς προδιαγραφές και να πραγματοποιηθεί μια εις βάθος ανάλυση και σχεδίαση, ώστε να ορισθεί μια βασική αρχιτεκτονική που θα ικανοποιεί τις κύριες περιπτώσεις χρήσης, και βάσει αυτής της αρχιτεκτονικής να μπορεί να καθορισθεί πιο λεπτομερώς το πρόγραμμα της επόμενης φάσης Κατασκευής. Το κύριο ερώτημα που πρέπει να απαντηθεί είναι:

- Σε ποιο βαθμό είναι καθορισμένη η βασική αρχιτεκτονική, ώστε να υπάρξει έναρξη της κατασκευής χωρίς ρίσκο;
- Πώς εξασφαλίζεται η υλοποίηση του συστήματος, ως προς το ρίσκο των προδιαγραφών. Έχουν επιλεγεί σωστές προδιαγραφές;
- Πώς εξασφαλίζεται η υλοποίηση του συστήματος, ως προς το ρίσκο των τεχνολογιών. Έχουν επιλεγεί οι σωστές τεχνολογίες και πώς θα συνδυαστούν μεταξύ τους;
- Πώς εξασφαλίζεται η υλοποίηση του συστήματος, ως προς το ρίσκο των εξειδικευμένων προσόντων. Έχει βρεθεί το απαιτούμενο εξειδικευμένο προσωπικό;

Το σύστημα υλοποιείται κατά τη φάση της κατασκευής. Σε αυτήν τη φάση η κεντρική αρχιτεκτονική του συστήματος επεκτείνεται, συνεπώς καταναλώνεται το μεγαλύτερο τμήμα των πόρων που έχουν δεσμευτεί και η εικόνα του τελικού συστήματος αρχίζει να ολοκληρώνεται.

Η φάση αυτή αποτελείται από επαναλήψεις, όπου κάθε μια στοχεύει να προσδώσει επιπλέον λειτουργικότητα στο σύστημα. Κάθε επανάληψη έχει ως αποτέλεσμα τόσο τη βήμα - βήμα διεύρυνση του συστήματος, χτίζοντας πάνω σε προηγούμενα τμήματα, όσο και τη βελτίωση της ευελιξίας του, ξαναγράφοντας τμήματα κώδικα. Αυτό πραγματοποιείται μέσω της εφαρμογής ενός κύκλου υπο-φάσεων που περιλαμβάνει την ανάλυση του κώδικα, την υλοποίηση **/** βελτίωση του κώδικα και το έλεγχο του κώδικα. Συνήθως, η κάθε επανάληψη σχετίζεται με μια περίπτωση χρήσης, χωρίς όμως αυτό να είναι κανόνας. Στο τέλος της φάσης πρέπει να μπορεί να απαντηθεί επιτυχώς το ερώτημα:

- Το σύστημα διαθέτει όλη τη λειτουργικότητα που είχε ζητηθεί σε επίπεδο προδιαγραφών ώστε να παραδοθεί στους χρήστες;

Ακόμη και μετά αυτή την επαναληπτική διαδικασία κατασκευής, κάποια δουλειά πρέπει να μείνει για τη τελική φάση, τη φάση της Μετάβασης. Η φάση αυτή μπορεί να περιλαμβάνει έλεγχο δοκιμαστική έκδοσης **(beta testing)**, ρύθμιση απόδοσης, εκπαίδευση χρηστών κ.α. Ο λόγος που θεωρείται διαφορετική φάση και όχι απλά μια επανάληψη της φάσης Κατασκευής, είναι ότι ο έλεγχος και προτροπές για βελτιώσεις, δεν προκύπτουν εσωτερικά από την ίδια την ομάδα υλοποίησης, αλλά από εξωτερικούς συνεργάτες ή έμπειρους χρήστες που αναφέρουν τυχόν ατέλειες και ελλείψεις.

Όπως αναφέρθηκε, κάθε κύκλος ζωής ολοκληρώνεται **με** μια νέα έκδοση του συστήματος. Ένα τέτοιο σύστημα για να θεωρείται ολοκληρωμένο, θα πρέπει να περιλαμβάνει το πηγαίο κώδικα σε κατάλληλη μορφή ώστε να μπορεί να μεταγλωττιστεί

και να εκτελεσθεί. Επιπλέον όμως πρέπει να συνοδεύεται από πληροφορίες ως προς την αρχιτεκτονική του, καθώς και τα μοντέλα που χρησιμοποιήθηκαν κατά τη κατασκευή του. Μάλιστα, αυτές οι συνοδευτικές πληροφορίες πρέπει να συμπεριλαμβάνονται καθ΄ όλη τη διαδικασία εξέλιξης του συστήματος, δηλαδή κατά τις διάφορες φάσεις και επαναλήψεις του κύκλου ζωής της υλοποίησής του. Ο λόγος είναι ότι τόσο οι χρήστες όσο και άλλοι ενδιάμεσοι συντελεστές π.χ. αναλυτές συστημάτων, προγραμματιστές, δοκιμαστές κ.λ.π. θα πρέπει να έχουν όλη τη διαθέσιμη πληροφορία ώστε να είναι σε θέση να χρησιμοποιήσουν ή να εξελίξουν το σύστημα.

Ένας τρόπος ενοποίησης και ταξινόμησης όλης αυτής της συνοδευτικής πληροφορίας είναι μέσω μοντέλων, που μπορούν να εκφρασθούν από τα διαγράμματα της **UML**, που στο σύνολο τους μπορεί να φτάσουν τα δεκατέσσερα. Πάντως, γενικά για να θεωρείται πλήρης μια περιγραφή συστήματος προτείνεται η συμπερίληψη των εξής: **[18]**

- Ένα μοντέλο που να περιγράφει τη λειτουργικότητα του συστήματος. Λειτουργικότητα: περιγράφεται τι πρέπει να κάνει το σύστημα. Κυριότερο παράδειγμα τέτοιου διαγράμματος είναι το διάγραμμα περίπτωσης χρήσης **(use case)**. Τα διαγράμματα αυτά είναι πιο κοντά στην ανάλυση απαιτήσεων, παρά στη σχεδίαση και προκύπτουν από περιγραφές σε κείμενο.
- Ένα μοντέλο που να περιγράφει τη συμπεριφορά του συστήματος. Συμπεριφορά: περιγράφεται η δυναμική συμπεριφορά του συστήματος, δηλ. πώς γίνονται οι μεταβάσεις ανάμεσα στις διάφορες καταστάσεις **(states)**. Αποτελεί εξέλιξη της λειτουργικότητας, μόνο που εξαντλεί όλα τα μονοπάτια που μπορεί να ακολουθήσει το σύστημα. Είναι ένα διαγράμματα ροής.
- Ένα μοντέλο δομής του συστήματος. Δομή: στατικά διαγράμματα που ασχολούνται με τις οντότητες που αποτελούν το σύστημα και τη σχέση τους.

Γενικά, η λογική σειρά ανάπτυξης των μοντέλων και των αντίστοιχων διαγραμμάτων είναι να περιγράφεται πρώτα η λειτουργικότητα, δηλ. τι πρέπει να κάνει το σύστημα, έπειτα να αναλύεται η δυναμική συμπεριφορά του και τέλος να σχεδιάζονται τα δομικά στοιχεία που αλληλεπιδρούν για να υλοποιήσουν τη συμπεριφορά.

Τα προαναφερθέντα μοντέλα συσχετίζονται μεταξύ τους ώστε να αναπαραστήσουν και να μοντελοποιήσουν το σύστημα στο σύνολό του και σε κάθε περίπτωση εξειδικεύονται ακόμα περισσότερο με τη χρήση των διαγραμμάτων **UML**. Η λειτουργικότητα, η συμπεριφορά και η δομή αποτελούν τους **3** άξονες της μοντελοποίησης συστημάτων. Στις επόμενες παραγράφους παρουσιάζεται μια ταξινόμηση των διαγραμμάτων της **UML** σε κάθε φάση, ενώ παράλληλα παρουσιάζονται και οι κύριοι στόχοι των φάσεων αυτών.

## *2.2 Φάση Σύλληψης*

Κατά τη φάση Σύλληψης όπως ειπώθηκε πρέπει να απαντηθούν ερωτήματα σχετικά με το ποιοι θα είναι οι χρήστες και ποιες θα είναι οι βασικές λειτουργίες του συστήματος, ποια θα μπορούσε να είναι η αρχιτεκτονική του και ποιο το κόστος υλοποίησης και το πιθανό κέρδος. Συνεπώς, οι σχεδιαστές θα πρέπει να καθορίσουν τους ρόλους των χρηστών του συστήματος, καθώς και τις λειτουργίες που θα επιτελεί. Επιπλέον, πρέπει να ορίσουν τα επιχειρηματικά αντικείμενα του συστήματος και μια πρόχειρη, αρχική αρχιτεκτονική.

Τα διαγράμματα που συνήθως χρησιμοποιούνται για να απαντηθούν τα παραπάνω ερωτήματα είναι τα διαγράμματα Περίπτωσης Χρήσης, Κλάσεων, Δραστηριοτήτων και Παράταξης. Στο παρακάτω πίνακα παρουσιάζεται και η σχετική αντιστοίχηση.

**Πίνακας 1 Εργασίες στη φάση Σύλληψης και αντίστοιχα διαγράμματα**

| **Εργασία για τη Σύλληψη συστήματος** | **Διάγραμμα UML** |
|---|---|
| Χρήστες **&** ρόλοι | Περίπτωσης Χρήσης |
| Λειτουργίες | Δραστηριοτήτων |
| Αρχιτεκτονική **&** Διεπαφές | Παράταξης |
| Επιχειρηματικό αντικείμενο | Κλάσεων |

Για τη δημιουργία του διαγράμματος Περίπτωσης Χρήσης απαιτείται η καταγραφή των λειτουργικών και μη λειτουργικών απαιτήσεων. Οι λειτουργικές απαιτήσεις θα καταγραφούν και θα εκφραστούν ως περιπτώσεις χρήσης. Οι **μ**η λειτουργικές απαιτήσεις που είναι συγκεκριμένες για κάποια περίπτωση χρήσης προσαρτώνται σε εκείνη την περίπτωση χρήσης, ενώ εκείνες που είναι πιο γενικές, καταγράφονται ως συμπληρωματικές απαιτήσεις. Κάποιες από τις τελευταίες μπορεί να είναι πολύ σημαντικές για την επιλογή της αρχιτεκτονικής.

Γενικά σε αυτή τη φάση οι κατασκευαστές πρέπει να βάλουν σε μια σειρά το υποσύνολο των περιπτώσεων χρήσης που χρειάζονται για την ολοκλήρωση αυτής της φάσης και έπειτα να συμπληρώσουν τυχόν λεπτομέρειες. Πρέπει λοιπόν να γίνει μια επιλογή περιπτώσεων χρήσης που να αναλογούν στην εμβέλεια του συστήματος, και δεν συγκεκριμενοποιούν ιδιαίτερα το επιχειρηματικό πλάνο, ώστε να είναι όσο πιο αντιπροσωπευτικές γίνεται.

Συνήθως, λεπτομερώς συμπληρώνεται περίπου το **10%** των περιπτώσεων χρήσης. Δηλαδή, ενώ μπορεί αρχικά να έχουν διερευνηθεί πολλές περιπτώσεις χρήσης, εντούτοις μπορεί να χρειάζεται ένα περιορισμένο ποσοστό λεπτομερούς περιγραφής των σεναρίων κάθε μιας. Ο βασικός στόχος είναι να κρατηθούν σε χαμηλό επίπεδο το κόστος και ο χρόνος αυτής της φάσης. Το τελικό αποτέλεσμα είναι το διάγραμμα περιπτώσεων χρήσης.

Το επόμενο θέμα που πρέπει να μοντελοποιηθεί είναι οι λειτουργίες του συστήματος. Ο καθορισμός των λειτουργικών απαιτήσεων στην αρχή της φάσης της

σύλληψης ποικίλουν από μια γενική εικόνα έως εκτενείς περιγραφές. Αυτές οι αρχικές λειτουργικές απαιτήσεις θα περιέχουν πολλά αμφίβολα χαρακτηριστικά που εξαρτώνται και από την αντιπροσωπευτικότητα των περιπτώσεων χρήσης και των σχετικών απαιτήσεων. Πάντως η φάση της σύλληψης είναι αρχική και φτάνει μέχρι και το καθορισμό της αρχιτεκτονικής ώστε να διευκρινιστεί η εφικτότητα του συστήματος και να γίνει μια αρχική εκτίμηση κόστους. κατά τα άλλα, η διαδικασία είναι πλήρως δυναμικής και μπορεί τα δεδομένα και οι παραδοχές που έχουν γίνει να διαφοροποιούνται. Το διάγραμμα δραστηριοτήτων αποτελεί μια ικανή αντιπροσώπευση των λειτουργιών του συστήματος, ενώ η χρήση του επιτρέπει και την εύκολη μετατροπή του σε περιπτώσεις που σε κάποια επόμενη επανάληψη διαφοροποιηθούν η παραδοχές που έχουν γίνει.

Ο σκοπός εδώ είναι να αναπτυχθεί μια αρχική περιγραφή του μοντέλου σχεδίασης, που θα αποτελέσει το πρώτο βήμα για τη δημιουργία μιας εικόνας αρχιτεκτονικής του συστήματος, που θα υποστηρίζει τις λειτουργίες του προηγούμενου βήματος. Η αρχιτεκτονική θα περιλαμβάνει και τις αλληλεπιδράσεις με το χρήστη ή άλλα συστήματα, δηλαδή θα περιλαμβάνει και διεπαφές. Το αποτέλεσμα είναι το διάγραμμα παράταξης.

Στο τέλος της φάσης της σύλληψης όπου πλέον έχει καθορισθεί και μια υποψήφια αρχιτεκτονική, μπορεί πλέον να εξεταστεί η επιχειρηματική πλευρά του συστήματος, δηλαδή από οικονομικής πλευράς, λαμβάνοντας υπόψη τις απαιτήσεις σε πόρους και την αποδοχή της αγοράς. Από τη μια πλευρά του επιχειρηματικού πλάνου θα είναι το κόστος του έργου και από την άλλη τα οφέλη από τυχόν χρήση του έργου. Το αποτέλεσμα δύναται να αναπαρασταθεί εν μέρει με ένα διάγραμμα κλάσεων ανά και το καταλληλότερο διάγραμμα σε αυτή τη περίπτωση είναι κάποιο διάγραμμα **BPMN (Business Process Model and Notation)** που όμως δεν συμπεριλαμβάνεται στην **UML.**

## *2.3 Φάση Επεξεργασίας*

Ξεκινώντας τη φάση της επεξεργασίας, οι κατασκευαστές έχουν στα χέρια τους πλέον τρία επιτεύγματα. Το πρώτο είναι μια αρχική, υποψήφια αρχιτεκτονική, που καταδεικνύει τη γνώση για το τρόπο κατασκευής του συστήματος ώστε να περικλείει όλα τα απαραίτητα τμήματα. Το δεύτερο είναι η λειτουργικότητα του συστήματος, δηλαδή τι λειτουργίες θα επιτελεί και προς ποιους. Τρίτον το επιχειρηματικό πλάνο που καταδεικνύει τη βιωσιμότητα του εγχειρήματος.

Όπως προαναφέρεται τα ζητήματα που πρέπει να διευκρινιστούν σε αυτή τη φάση αφορούν την πιο λεπτομερή περιγραφή της αρχιτεκτονικής του συστήματος ώστε να αποκτήσει μια σταθερή δομή που θα υπερκαλύπτει και τυχόν κινδύνους από σχεδιαστικά λάθη και αμέλειες, ενώ παράλληλα να παραμένει εντός των πλαισίων που ορίζονται από το επιχειρηματικό πλάνο.

Αυτά μεταφράζονται σε επέκταση ουσιαστικά των απαιτήσεων του συστήματος ώστε να δημιουργηθεί ένας κορμός αρχιτεκτονικής, ελαχιστοποιώντας τυχόν κινδύνους. Στο τέλος βέβαια απαιτείται και μια επανεκτίμηση του επιχειρηματικού πλάνου. Σε επίπεδο διαγραμμάτων **UML**, λεπτομέρεια στην αρχιτεκτονική μπορεί να αποτυπωθεί σχεδόν με οποιοδήποτε διάγραμμα δομικής μοντελοποίησης ή ακόμα και μοντελοποίησης συμπεριφοράς.

**Πίνακας 2 Εργασίες στη φάση Επεξεργασίας και αντίστοιχα διαγράμματα**

| **Εργασία για τη Σύλληψη συστήματος** | **Διάγραμμα UML** |
|---|---|
| Επέκταση απαιτήσεων | Περίπτωσης Χρήσης |
| Κορμός αρχιτεκτονικής | Κλάσης, Καταστάσεων, Δραστηριοτήτων, Αλληλουχίας, Επισκόπησης Αλληλεπίδρασης |
| Ελαχιστοποίηση κινδύνων | Κλάσης, Καταστάσεων, Δραστηριοτήτων, Αλληλουχίας, Επισκόπησης Αλληλεπίδρασης |

Η επέκταση των απαιτήσεων ουσιαστικά γίνεται διευρύνοντας τις περιπτώσεις χρήσης. Συνεπώς τα πιο συχνά χρησιμοποιούμενα διαγράμματα είναι αυτών των Περιπτώσεων Χρήσεων.

Κατά την αρχιτεκτονική σχεδίαση θα πρέπει να ελεγχθούν **μ**όνο οι αρχιτεκτονικά απαραίτητες απαιτήσεις. Πληροφορίες που δεν είναι αρχιτεκτονικά σημαντικές καλό είναι να μην συμπεριλαμβάνονται, αφού αυτές θα χρησιμοποιηθούν στη φάση της κατασκευής. Ο κορμός της αρχιτεκτονικής που θα είναι το αποτέλεσμα αυτών των προσπαθειών θα είναι απλά ο σκελετός του συστήματος. Κάποια σημεία βέβαια της αρχιτεκτονικής απαιτούν πιο λεπτομερή περιγραφή αφού ουσιαστικά αυτά επιβεβαιώνουν ότι η αρχιτεκτονική θα δουλέψει στη πράξη. Εξετάζονται οι περιπτώσεις χρήσης και πραγματοποιείται επεξεργασία σε αρχιτεκτονικό επίπεδο. Αναλύονται οι κλάσεις και τα πακέτα και σχεδιάζονται τα υποσυστήματα. Συνεπώς, για τη περιγραφή του κορμού της αρχιτεκτονικής μπορεί να χρησιμοποιηθεί οποιοδήποτε διάγραμμα δομικής μοντελοποίησης ή ακόμα και μοντελοποίησης συμπεριφοράς.

Ο σκοπός της διερεύνησης των κινδύνων σε αυτή τη φάση δεν είναι η εξάλειψη τους αλλά η μείωση τους σε ένα αποδεκτό επίπεδο για τη φάση της κατασκευής. Σε αυτή τη φάση πιο αντιμετωπίσιμοι είναι οι τεχνικοί κίνδυνοι αρχιτεκτονικής φύσεως, αφού πραγματοποιείται η επεξεργασία της αρχιτεκτονικής. Η διερεύνηση των κινδύνων δεν μπορεί να είναι εξαντλητική, αλλά αντιθέτως θα φτάνει ως το βαθμό του καθορισμού τρόπων αντιμετώπισης τους με χρόνο και κόστος που να μπορεί να εκτιμηθεί. Πραγματική εξάλειψη του κινδύνου πραγματοποιείται κατά την υλοποίηση των περιπτώσεων χρήσης που τους εμπεριέχουν στη φάση της κατασκευής. Η ελαχιστοποίηση των κινδύνων δεν απεικονίζεται με κάποιο διάγραμμα, αλλά υπονοείται

κυρίως από τα δομικά διαγράμματα που παρουσιάζουν την αρχιτεκτονική το συστήματος.

## *2.4 Φάση Κατασκευής*

Ο κυρίαρχος στόχος αυτής της φάσης είναι να δημιουργήσει ένα προϊόν λογισμικού που θα αποτελεί μια αρχική λειτουργική έκδοση, που πολλές φορές ονομάζεται δοκιμαστική έκδοση **(beta release)**. Το προϊόν αυτό πρέπει να είναι ικανοποιητικής ποιότητας και να ικανοποιεί τις απαιτήσεις. Η διάρκεια κατασκευής πρέπει να είναι μέσα στα όρια του επιχειρηματικού πλάνου. Ουσιαστικά πλέον η διαδικασία περνά στην πραγματική υλοποίηση.

Θα αναρωτηθεί λοιπόν κάποιος, τι χρειάζεται η μοντελοποίηση σε αυτή τη φάση. Η απάντηση έχει ήδη δοθεί και σε προηγούμενο κεφάλαιο: για μια σειρά από λόγους όπως ότι επιτρέπει τη συνεργασία μεταξύ πολλών προγραμματιστών, επιτρέπει την μεταφερσιμότητα, διευκολύνει τον επανασχεδιασμό, παρέχει ένα εποπτικό τρόπο κατανόησης του συστήματος κ.α. Σε αυτή τη φάση είναι δυνατή η χρήση οποιουδήποτε διαγράμματος. Συνηθέστερα, πραγματοποιείται μια ενημέρωση στα διαγράμματα που έχουν σχεδιαστεί σε προηγούμενες φάσεις.

**Πίνακας 3 Εργασίες στη φάση Κατασκευής και αντίστοιχα διαγράμματα**

| **Εργασία για τη Σύλληψη συστήματος** | **Διάγραμμα UML** |
|---|---|
| Υλοποίηση δομής | Πακέτων, Συστατικών, Αντικειμένων |
| Υλοποίηση κλάσεων και λειτουργιών | Κλάσης, Καταστάσεων, Δραστηριοτήτων, Αλληλουχίας, Επισκόπησης Αλληλεπίδρασης, Χρονισμού |
| Υλοποίηση επιχειρηματικού σχεδίου | Κλάσης |

Η ουσιώδης διαφορά μεταξύ της φάσης της επεξεργασίας και της φάσης της κατασκευής είναι ότι τώρα στη φάση της κατασκευής συμπληρώνεται το μοντέλο ανάλυσης. Το μοντέλο ανάλυσης που στο τέλος της φάσης της επεξεργασίας ήταν η αρχιτεκτονική όψη, και ήταν σε μεγάλο βαθμό σχετικό **μ**ε την αρχιτεκτονική. τώρα η αρχιτεκτονική όψη του μοντέλου ανάλυσης θα είναι **μ**όνο ένα μέρος του ολόκληρου μοντέλου ανάλυσης. Στο τέλος της φάσης της κατασκευής θα προκύψει ολοκληρωμένο το μοντέλο ανάλυσης. Η αρχιτεκτονική όψη θα είναι **μ**όνο ένα υποσύνολο του.

Μέχρι αυτό το σημείο έχει ετοιμαστεί η αρχιτεκτονική όψη του συστήματος. Συνεπώς στα αντίστοιχα διαγράμματα της φάσης επεξεργασίας θα πρέπει απλά να γίνουν ενημερώσεις για τυχόν αλλαγές που επηρεάζουν την αρχιτεκτονική. Όσον αφορά τις περιπτώσεις χρήσης, στη προηγούμενη φάση χρησιμοποιήθηκαν μόνο αυτές

που ήταν αρχιτεκτονικά σημαντικές. Σε κάθε επανάληψη στη φάση της κατασκευής το μοντέλο ανάλυσης θα εμπλουτιστεί **μ**ε εκείνες τις περιπτώσεις χρήσης που δεν είχαν αρχικά συμπεριληφθεί. Επιπλέον, θα πρέπει να αναλυθούν περαιτέρω οι κλάσεις και να βελτιωθούν τα πακέτα, ανάλογα **μ**ε τις καινούργιες περιπτώσεις χρήσης.

Κατά την υπό-φάση Δοκιμής θα πρέπει να δημιουργηθεί ένα πλάνο ελέγχου με στόχους, βάσει των οποίων θα ελεγχθούν οι διαδοχικές κατασκευές και τελικώς το ολόκληρο σύστημα. Θα πρέπει να καθορισθεί ο τρόπος που θα δοκιμαστούν οι περιπτώσεις χρήσης και να καθορισθούν και τα συστατικά που πρέπει να ελεγχθούν μαζί. Ο σκοπός των ενοποιημένων ελέγχων είναι να επιβεβαιώσουν τη σωστή λειτουργία των διεπαφών μεταξύ των συστατικών που υπόκεινται στον έλεγχο, δηλαδή να επιβεβαιωθεί ότι τα συστατικά δουλεύουν σωστά μεταξύ τους. Ουσιαστικά η φάση ολοκληρώνεται με τη εκτέλεση των ελέγχων των ενοποιημένων ελέγχων του συστήματος και την αξιολόγηση των αποτελεσμάτων των ελέγχων. Σε περίπτωση που τα αποτελέσματα δεν είναι ικανοποιητικά, θα πρέπει να πραγματοποιηθεί και νέα επανάληψη της φάσης, μέχρι να επιτευχθεί το επιθυμητό αποτέλεσμα.

## *2.5 Φάση Μετάβασης*

Αυτή η φάση εστιάζει στην εγκατάσταση του προϊόντος στο περιβάλλον λειτουργίας του. Καθώς το έργο εισέρχεται στη φάση της μετάβασης, το σύστημα έχει αποκτήσει μια αρχική λειτουργική ικανότητα και θεωρείται αξιόπιστο για να λειτουργήσει μέσα στο περιβάλλον του χρήστη. Βέβαια κάποια προβλήματα, κίνδυνοι και ελαττώματα δεν θα έχουν γίνει ορατά κατά την υπό-φάση Δοκιμής της φάσης Κατασκευής, οπότε μπορεί να εμφανιστούν στο περιβάλλον του χρήστη. Μπορεί να υπάρχουν χαρακτηριστικά τα οποία οι χρήστες καθυστερημένα ανακάλυψαν ότι τα χρειάζονται. Η συγκεκριμένη φάση λοιπόν έχει σχεδιαστεί για να ανταποκρίνεται σε τέτοιες ανάγκες.

**Πίνακας 4 Εργασίες στη φάση Μετάβασης και αντίστοιχα διαγράμματα**

| **Εργασία για τη Σύλληψη συστήματος** | **Διάγραμμα UML** |
|---|---|
| Παραγωγή Οδηγιών εγκατάστασης | Παράταξης |
| Δημιουργία εκπαιδευτικού υλικού | Οποιοδήποτε διάγραμμα |

# Κεφάλαιο 3: Διαγράμματα μοντελοποίησης

Όπως αναφέρθηκε στο **1**ο Κεφάλαιο**,** από το πρότυπο **UML 2.0** και έπειτα καθορίζονται **14** τύποι διαγραμμάτων **UML.** Εδώ παρουσιάζονται οι τύποι αυτοί**,** μαζί με παραδείγματα για την ευκολότερη κατανόησή τους**.**

## *3.1 Διαγράμματα μοντελοποίησης Δομής*

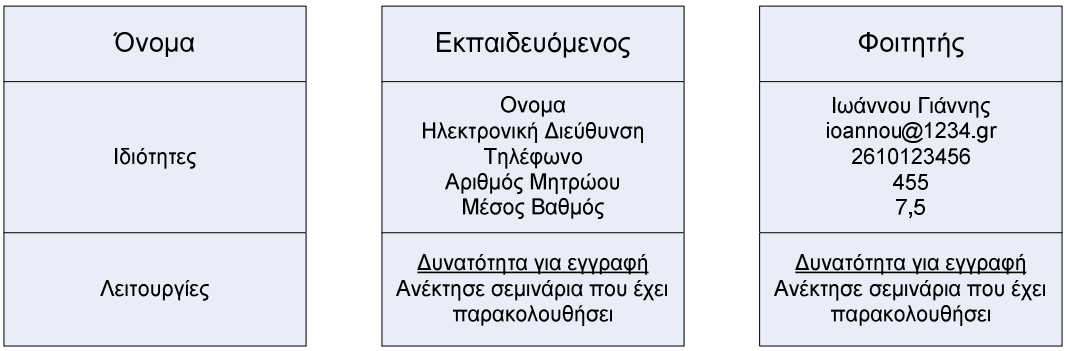
Τα διαγράμματα μοντελοποίησης δομής**,** αναπαριστούν τη στατική δομή του συστήματος και των τμημάτων του**,** από διαφορετικές οπτικές γωνίες και επίπεδα εφαρμογής**,** και το πώς αυτά τα τμήματα συνδέονται μεταξύ τους**.** Τα στοιχεία σε ένα διάγραμμα δομής αποτελούν τις σημασιολογικές έννοιες του συστήματος**,** και μπορεί να περιλαμβάνουν έννοιες αφηρημένες**,** πραγματικές ή σχετικές με την εφαρμογή τους**.**

Τα διαγράμματα δομής δεν αξιοποιούν σχετικές με το χρόνο έννοιες και δεν εμφανίζουν τις λεπτομέρειες της δυναμικής συμπεριφοράς**.** Ωστόσο**,** μπορούν να εμφανίσουν τις σχέσεις με τις συμπεριφορές των στοιχείων που συμπεριλαμβάνονται στα διαγράμματα δομής**.**

### 3.1.1 Διαγράμματα Κλάσεων (Class)

Όπως έχει ειπωθεί και παραπάνω**,** τα αντικει**μ**ενοστραφή συστήματα λειτουργούν ως μια συλλογή συνεργαζόμενων αντικειμένων**.** Τα αντικείμενα αποτελούν ουσιαστικά στιγμιότυπα κλάσεων**.** Συνεπώς**,** σε αυτό το σημείο είναι χρήσιμο να γίνει μια σύντομη αναφορά στις κλάσεις και τα αντικείμενα του αντικειμενοστραφούς προγραμματισμού**,** ώστε να είναι πιο κατανοητό για τον αναγνώστη το διάγραμμα που ουσιαστικά περιγράφει τις κλάσεις του συστήματος και τις συσχετίσεις μεταξύ αυτών**.**

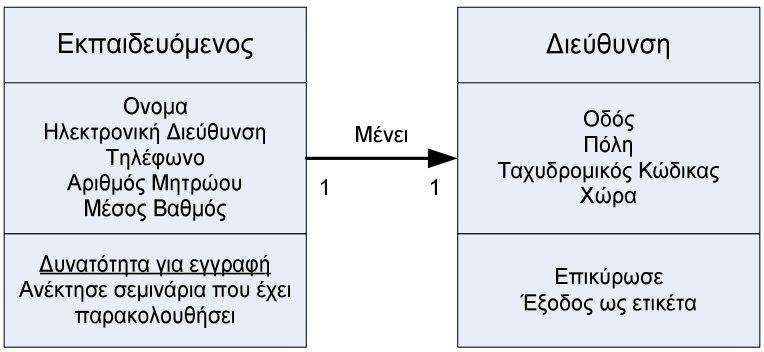
Οι κλάσεις αποτελούν τη βάση οποιουδήποτε αντικει**μ**ενοστραφούς συστήματος**.** Εμπεριέχουν τόσο τα δεδομένα όσο και τις λειτουργίες που επενεργούν στα δεδομένα αυτά**.** Ο συμβολισμός μιας κλάση είναι ένα παραλληλόγραμμο με τρία τμήματα**.** Το πάνω τμήμα περιλαμβάνει το όνομα της κλάσης**,** το μεσαίο τμήμα περιλαμβάνει τις ιδιότητες και το κάτω τμήμα τις λειτουργίες**.** Από την άλλη το αντικείμενο είναι ουσιαστικά ένα στιγμιότυπο **(instance)** μιας κλάσης**.**




**Εικόνα 4** **Δομή κλάσης και παράδειγμα κλάσης και αντικειμένου**

Το σύνηθες είναι σε μια κλάση οι ιδιότητες να είναι ιδιωτικές **(private)** και οι λειτουργίες δημόσιες **(public).** Ιδιωτικά είναι τα χαρακτηριστικά μιας κλάσης**,** ιδιότητες ή λειτουργίες**,** που δεν είναι προσπελάσιμα από άλλες κλάσεις**,** ενώ δημόσια αυτά που είναι προσπελάσιμα από άλλες κλάσεις**.** Το ότι τα δεδομένα είναι ιδιωτικά σημαίνει ότι αντικείμενα άλλων κλάσεων δεν έχουν τη δυνατότητα να προσπελάσουν απευθείας τις ιδιότητες μιας κλάσης**,** αλλά χρησιμοποιούν για αυτό τις μεθόδους της κλάσης**.** Η τακτική αυτή ονομάζεται αρχή απόκρυψης των δεδομένων **(information hiding principle).**

Συσχέτιση **(association)** μεταξύ κλάσεων ονομάζουμε μια στατική σχέση μεταξύ των κλάσεων**.** Η ανάγκη μιας συσχέτισης προκύπτει από τη διαπίστωση πως για τη λειτουργία μιας κλάσης απαιτείται η συνεργασία της **μ**ε **μ**ία ή περισσότερες άλλες κλάσεις**.** Αν αυτή η συνεργασία απαιτείται να είναι σε μόνιμη βάση**,** τότε χρησι**μ**οποιού**μ**ε συσχέτιση**,** αν είναι παροδική τότε χρησι**μ**οποιού**μ**ε εξάρτηση **(dependency).** Για την αναπαράσταση μια συσχέτισης χρησι**μ**οποιού**μ**ε **μ**ία γρα**μμ**ή μεταξύ των δύο κλάσεων**.** Κάθε συσχέτιση έχει δύο πέρατα**,** το καθένα από τα οποία μπορεί να ονομαστεί **μ**ε **μ**ία ετικέτα που λέγεται ρόλος**.** Το πέρας μιας συσχέτισης έχει επίσης πολλαπλότητα **(multiplicity),** που εκφράζει πόσα αντικείμενα από κάθε κλάση μπορούν να συ**μμ**ετέχουν σε αυτή τη σχέση**.** Οι γρα**μμ**ές συσχέτισης έχουν βέλη τα οποία δείχνουν τη δυνατότητα πλοήγησης και μπορεί να είναι μονόδρομη ή α**μφ**ίδρο**μ**η**.** Τα ίδια ισχύουν και για τις εξαρτήσεις με τη διαφορά ότι χρησιμοποιείται διακεκομμένη γραμμή**.**



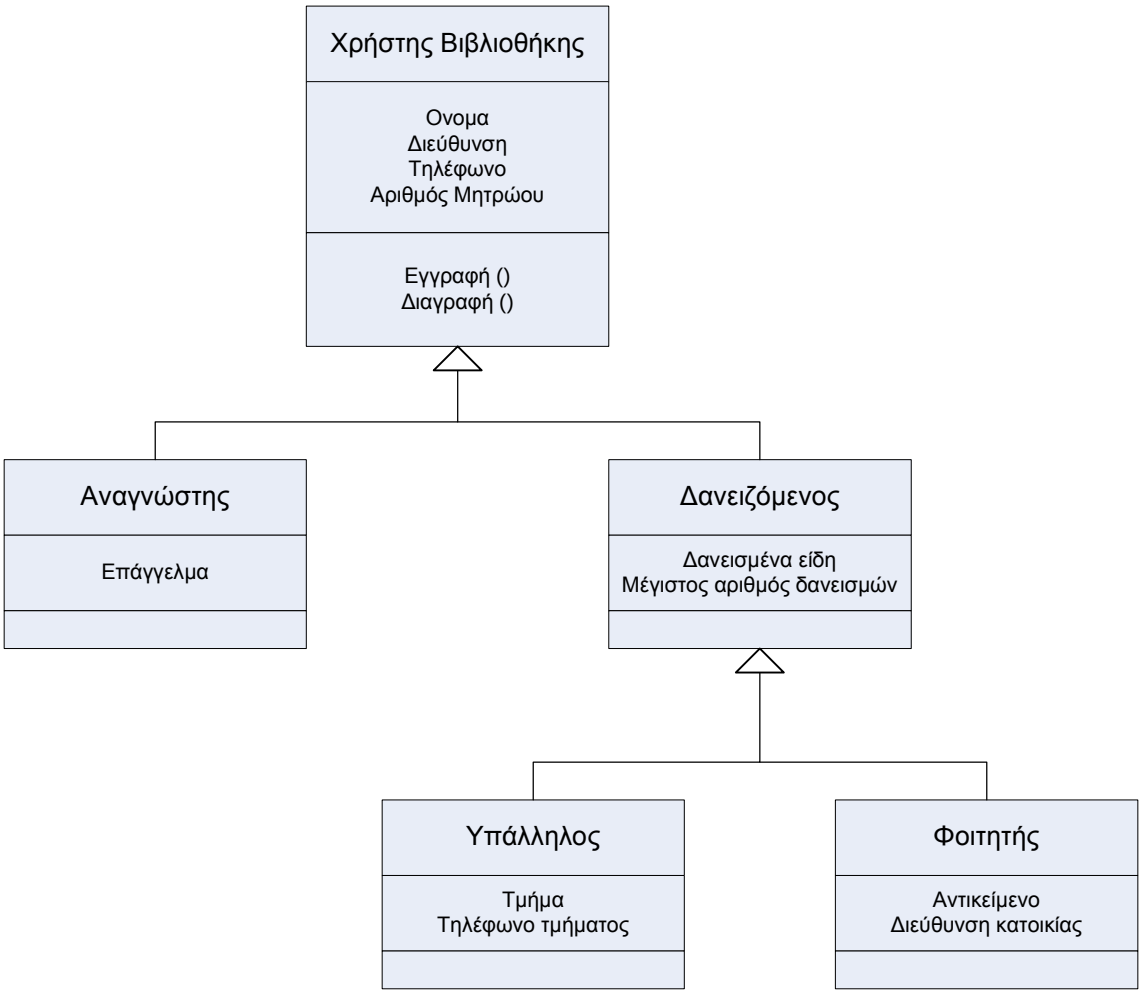

**Εικόνα 5** **Συσχέτιση μεταξύ δύο κλάσεων**

Η πολλαπλότητα περιγράφεται με δείκτες που επεξηγούνται στον ακόλουθο πίνακα.

**Πίνακας 5 Δείκτες πολλαπλότητας συσχετίσεων κλάσεων**

| **Δείκτης** | **Σημασία** |
|---|---|
| **1** | Ένα μόνο |
| **n** | **n** μόνο **(n>1)** |
| **0..1** | Μηδέν ή ένα |
| **0..*** | Μηδέν ή περισσότερο |
| **1..*** | Ένα ή περισσότερο |
| **0..n** | Μηδέν έως **n (n>1)** |
| **1..n** | Ένα έως **n (n>1)** |

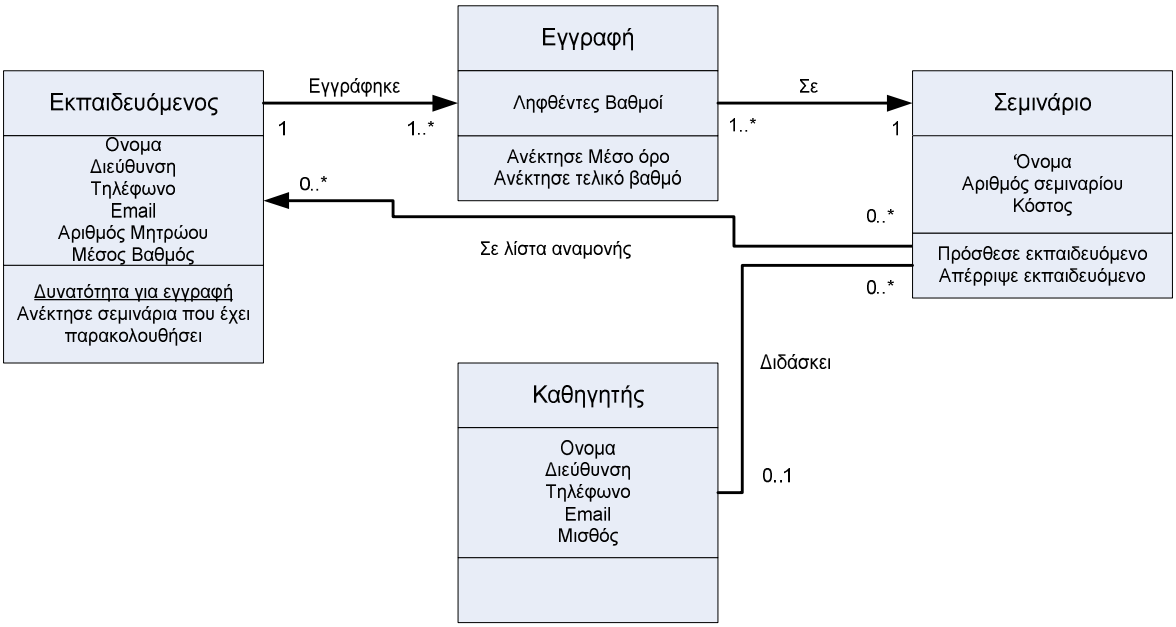
Η γενίκευση **(generalization)** ή κληρονομικότητα είναι μία ειδική μορφή συσχέτισης, κατά την οποία μια γενική κλάση αποτελεί τη βάση για τη δήλωση μιας ή περισσοτέρων ειδικότερων κλάσεων. Η γενική κλάση ονομάζεται υπερκλάση και ειδικές κλάσεις ονομάζονται υποκλάσεις. Η γενίκευση στις περισσότερες γλώσσες προγραμματισμού υλοποιείται με το μηχανισμό της κληρονομικότητας ή της επέκτασης. Η ιδέα είναι πως η γενική κλάση παρέχει λειτουργίες, ιδιότητες και συσχετίσεις που είναι χρήσιμες σε όλες τις υποκλάσεις. Από την άλλη μεριά, οι υποκλάσεις επεκτείνουν τη λειτουργικότητα της υπερκλάσης και παρέχουν επιπλέον λειτουργίες όπου αυτό είναι απαραίτητο ή εξειδικεύουν τη συμπεριφορά τους. Η γενίκευση συμβολίζεται με συνεχή γραμμή με βέλος στο πέρας και κατεύθυνση από την υποκλάση προς την υπερκλάση.




**Εικόνα 6 Γενίκευση κλάσεων**

Στη παραπάνω περίπτωση η κλάση **"**Χρήστης Βιβλιοθήκης**",** είναι υπερκλάση των υποκλάσεων **"**Αναγνώστης**"** και **"**Δανειζόμενος**",** ενώ η κλάση **"**Δανειζόμενος**"** έχει ως υποκλάσεις τις **"**Υπάλληλος**"** και **"**Φοιτητής**".**

Ακολουθεί ένα παράδειγμα διαγράμματος κλάσεων που περιλαμβάνει τα προαναφερθέντα**.**




**Εικόνα 7 Παράδειγμα διαγράμματος κλάσεων με συσχετίσεις**

Στο παραπάνω διάγραμμα υπάρχουν τέσσερις κλάσεις: “Εκπαιδευόμενος”, “Εγγραφή”, “Σεμινάριο”, “Καθηγητής”. Κάθε κλάση, περιλαμβάνει και κάποιες λειτουργίες που εφαρμόζονται στις ιδιότητες. Με το διάγραμμα, μοντελοποιείται η διαδικασία ανάκτησης των σεμιναρίων τα οποία έχει παρακολουθήσει ο εκπαιδευόμενος.

Η συσχέτιση "εγγράφηκε" μεταξύ “Εκπαιδευόμενου” και “Εγγραφής” είναι μονόδρομη και αντιπροσωπεύει την δυνατότητα των αντικειμένων της κλάσης “Εκπαιδευόμενος” να γνωρίζουν σε ποια αρχεία σεμιναρίων συμπεριλαμβάνονται. Προφανώς, κάθε εκπαιδευόμενος μπορεί να έχει βαθμούς από ένα ή περισσότερα σεμινάρια και μάλιστα αυτοί οι βαθμοί μπορούν να χρησιμοποιηθούν για την εξαγωγή του μέσου όρου και του μέγιστου βαθμού.

Επιπλέον, η συσχέτιση "σε" μεταξύ “Εγγραφής” και “Σεμιναρίου” είναι επίσης μονόδρομη και υποστηρίζει τη δυνατότητα των αντικειμένων της κλάσης “Εγγραφή” να συσχετίζουν τους βαθμούς του αρχείου κάθε σεμιναρίου με το σεμινάριο. Για ένα σεμινάριο προφανώς υπάρχουν ένα ή περισσότερα αρχεία βαθμών εκπαιδευομένων.

Η συσχέτιση "διδάσκει" μεταξύ κλάσεων “Καθηγητή” και “Σεμιναρίου” είναι αμφίδρομη αφού τα μεν αντικείμενα της κλάσης “Καθηγητή” γνωρίζουν ποια σεμινάρια διδάξανε, τα δε αντικείμενα της κλάσης “Σεμινάριο” γνωρίζουν ποιοι καθηγητές τα εισηγηθήκανε.

Τέλος, η συσχέτιση "σε λίστα αναμονής" μεταξύ “Σεμιναρίου” και “Εκπαιδευόμενου”, είναι μονόδρομη και εκφράζει τη δυνατότητα των αντικειμένων της κλάσης “Σεμινάριο” να γνωρίζουν ποια αντικείμενα της κλάσης “Εκπαιδευόμενοι” είναι σε λίστα αναμονής.

Κάποιες κλάσεις σε ένα σύστημα δηλώνονται ως αφαιρετικές **(abstract)**. Αυτές οι κλάσεις περιλαμβάνουν κώδικα για κάποιες από τις λειτουργίες τους, ενώ για κάποιες

άλλες δεν περιλαμβάνουν. Οι υποκλάσεις αυτών των κλάσεων θα αναλάβουν να πραγματοποιήσουν αυτές τις λειτουργίες παρέχοντας τον κατάλληλο κώδικα. Αυτές οι υποκλάσεις ονομάζονται συγκεκριμένες **(concrete)** και κληρονο**μ**ούν τα συγκεκρι**μ**ένα αφαιρετικά στοιχεία των υπερκλάσεων τους, ενώ ταυτόχρονα παρέχουν τον κώδικα για την υλοποίηση αυτών των αφαιρετικών στοιχείων.

Διεπαφή **(interface)** ονομάζεται ο τύπος που περιλαμβάνει εξ' ολοκλήρου αφαιρετικές λειτουργίες. Δεν έχει δεδο**μ**ένα, ούτε κατάσταση, άλλα ούτε συσχετίσεις **μ**ε κατεύθυνση από τη διεπαφή προς το άλλο άκρο της συσχέτισης. Αυτό εξάλλου αποτελεί συνέπεια της έλλειψης δεδο**μ**ένων. Η κλάση που πραγ**μ**ατώνει **(realizes)** τη διεπαφή έχει τη υποχρέωση να υλοποιήσει όλες τις λειτουργίες της διεπαφής. Το κέρδος για την κλάση είναι ότι τα αντικεί**μ**ενα της θα έχουν τον τύπο της διεπαφή και επο**μ**ένως θα μπορούν να χρησι**μ**οποιηθούν και αλλού. Μια διεπαφή στη **UML** σχεδιάζεται **μ**ε το σύ**μ**βολο της κλάσης και **μ**ε τη λέξη κλειδί **«Interface»** πάνω από το όνο**μ**α της κλάσης, για να υποδηλώνεται ότι πρόκειται για διεπαφή. Βέβαια, αφού δεν διαθέτει ιδιότητες, το αντίστοιχο τμήμα θα είναι κενό ή δεν θα υπάρχει. Η κλάση που υλοποιεί τη διεπαφή συσχετίζεται **μ**ε αυτή **μ**ε το σύ**μ**βολο της πραγ**μ**άτωσης **(realization)**, που είναι ίδιο **μ**ε το σύ**μ**βολο της γενίκευσης, άλλα **μ**ε γρα**μμ**ή που είναι διακεκο**μμ**ένη.

Οι αφηρη**μ**ένες κλάσεις μοιάζουν με τις διεπαφές με την έννοια ότι και οι δύο επιτρέπουν τη δήλωση μιας λειτουργίας χωρίς κώδικα. Όμως, οι αφηρη**μ**ένες κλάσεις επιτρέπουν την προσθήκη υλοποίησης για κάποιες από τις λειτουργίες, ενώ η διεπαφή εξαναγκάζει σε αναβολή την υλοποίηση όλων των λειτουργιών.

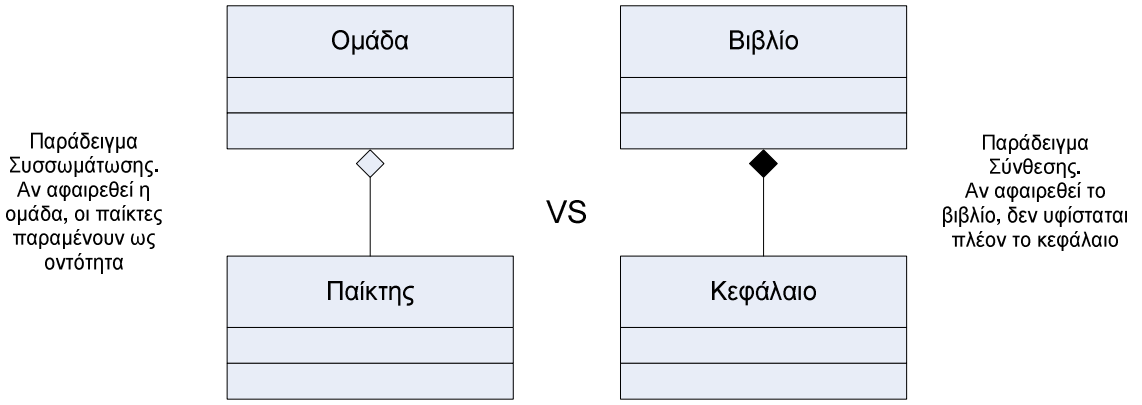
Η συσσωμάτωση **(aggregation)** και η σύνθεση **(composition)** είναι δύο περιπτώσεις συσχετίσεων μιας κλάσης **μ**ε κάποια άλλη κλάση που αποτελεί τμήμα της πρώτης.

Μια Συσσωμάτωση είναι ένας ειδικός τύπος συσχέτισης, στην οποία αντικείμενα διαφορετικών κλάσεων "συναρμολογούνται" ώστε να δημιουργηθεί ένα αντικείμενο μιας πιο σύνθετης κλάσης. Μια συσσωμάτωση περιγράφει μια ομάδα αντικειμένων και πώς γίνεται η αλληλεπίδραση μαζί τους. Στη Συσσωμάτωση τα αντικείμενα των κλάσεων που "συναρμολογούν" το νέο αντικείμενο διατηρούν την οντότητα τους ακόμα και αν η νέα σύνθετη κλάση καταργηθεί, κάτι που προφανώς συνεπάγεται ότι τα αντικείμενα μπορούν να συμμετέχουν και σε άλλες Συσσωματώσεις.

Η Σύνθεση είναι επίσης μια μορφή συσσωμάτωσης με τη διαφορά ότι στη περίπτωση που η πιο σύνθετη κλάση καταστραφεί, τότε εκτός από το αντικείμενο της σύνθετης κλάσης, καταστρέφονται και τα αντικείμενα από τα οποία "συναρμολογήθηκε". Επιπλέον, τα αντικείμενα που απαρτίζουν το σύνθετο αντικείμενο, δεν μπορούν να συμμετέχουν σε άλλες Συνθέσεις. Δηλαδή, η σύνθεση είναι **μ**ία πιο ισχυρή μορφή συσχέτισης, συγκρινόμενη με τη Συσσωμάτωση.

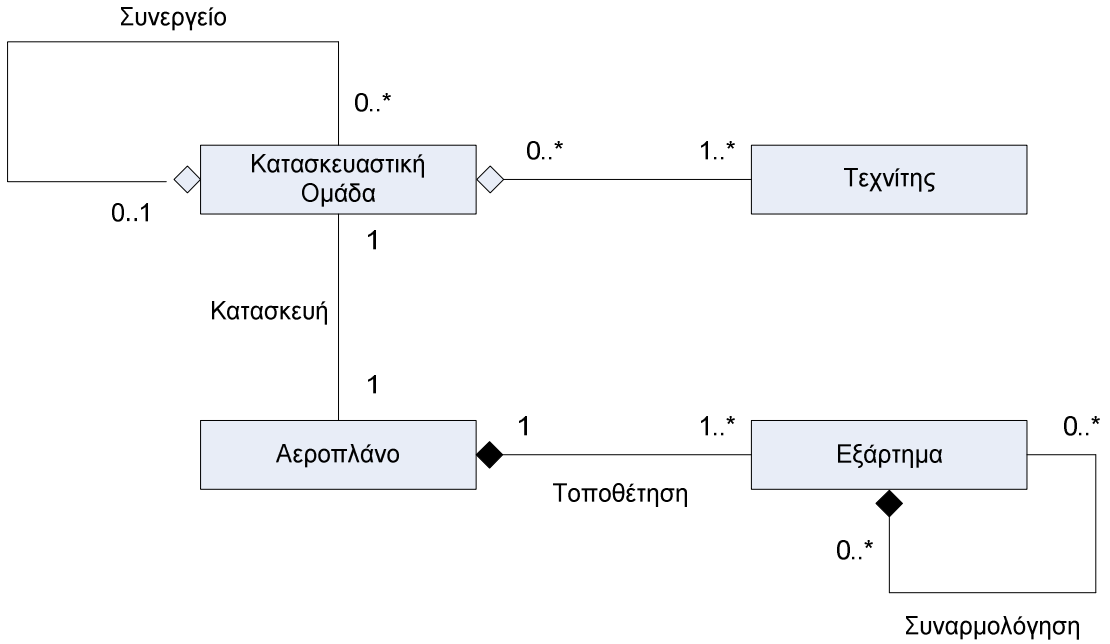
Μια συσσωμάτωση συμβολίζεται **μ**ε **μ**ία συσχέτιση από το τμήμα προς το όλο με ένα άσπρο ρόμβο στην πλευρά του όλου. Η σύνθεση συ**μ**βολίζεται **μ**ε **μ**ία συσχέτιση από το τμήμα προς το όλο με ένας μαύρο ρόμβο στην πλευρά του όλου.

Παρακάτω ακολουθεί ένα απλό παράδειγμα συσσωμάτωσης και ένα σύνθεσης.




**Εικόνα 8** **Παραδείγματα και διαφορά Συσσωμάτωσης και Σύνθεσης**

Το επόμενο διάγραμμα κλάσεων παρουσιάζει ένα σύστημα για τη κατασκευή ενός αεροσκάφους.




**Εικόνα 9** **Παράδειγμα διαγράμματος κλάσεων με συσσωματώσεις και συνθέσεις**

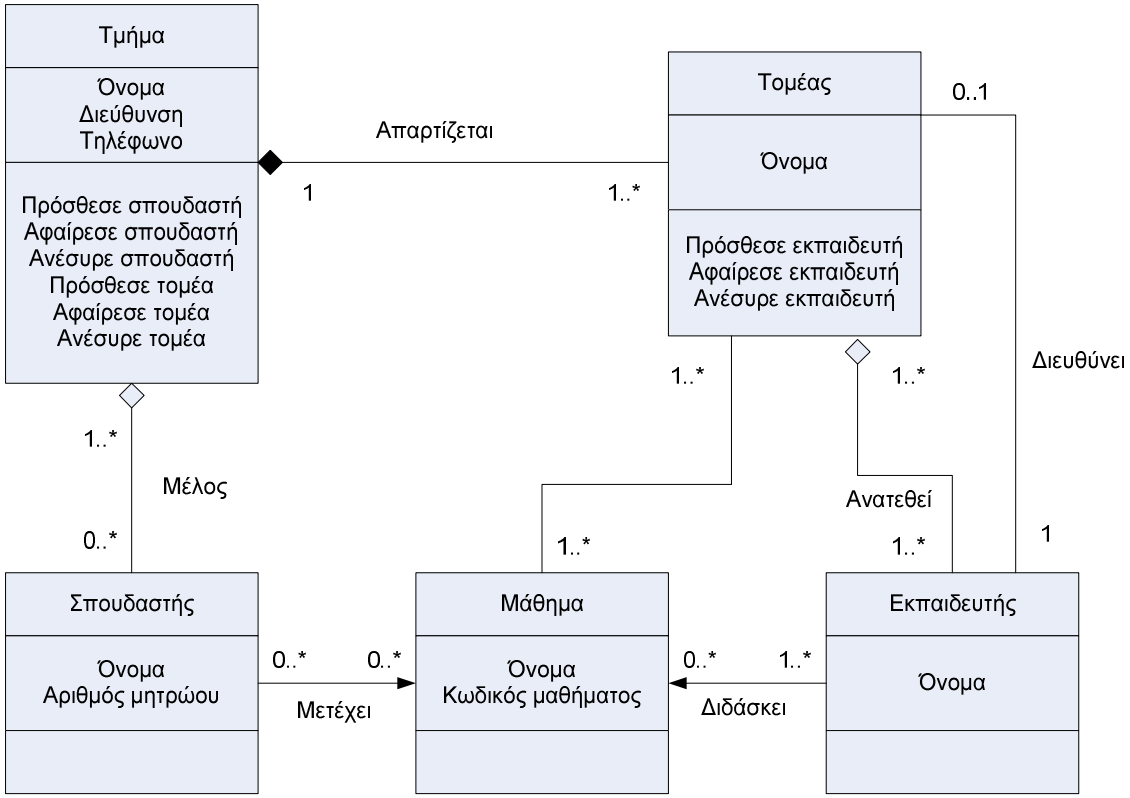
Στο παραπάνω σύστημα, μια κατασκευαστική ομάδα θα κατασκευάσει ένα αεροπλάνο. Για τη δημιουργία της ομάδας θα χρειαστούν ένας ή περισσότεροι τεχνίτες. Οι τεχνίτες λοιπόν είναι αντικείμενα της κλάσης "Τεχνίτης" που συσσωματώνονται για να δημιουργήσουν καμία ή περισσότερες ομάδες, δηλαδή ένα αντικείμενο της πιο σύνθετης κλάσης "Κατασκευαστική Ομάδα".

Επιπλέον, κανένας ή περισσότερες ομάδες μπορούν να αποτελέσουν κανένα ή περισσότερα συνεργεία της ομάδας. Τελικά, το αντικείμενο κατασκευαστική ομάδα συσχετίζεται με τη κατασκευή του αντικειμένου αεροπλάνο. Επιπλέον, ένα αντικείμενο αεροπλάνο αποτελεί σύνθεση ενός ή περισσοτέρων αντικειμένων της κλάσης

"Εξάρτημα". Ενώ κανένα ή περισσότερα εξαρτήματα συνθέτουν κανένα ή περισσότερα σύνθετα εξαρτήματα.

Γενικά, αν αφαιρεθεί η κλάση "Κατασκευαστική Ομάδα", τα αντικείμενα της κλάσης "Τεχνίτης" συνεχίζουν να υπάρχουν, ενώ αν αφαιρεθεί η κλάση "Αεροπλάνο", τα αντικείμενα της κλάσης "Εξάρτημα" δεν έχουν νόημα ύπαρξης. Επιπλέον, τα αντικείμενα εξαρτήματα χρησιμοποιούνται αποκλειστικά για τη σύνθεση του αντικειμένου αεροπλάνο, ενώ τα αντικείμενα τεχνίτες θα μπορούσαν να χρησιμοποιηθούν στη κατασκευή και άλλης κλάσης π.χ. "Ομάδα Συντηρήσεων".

Ακολουθεί ένα πιο σύνθετο διάγραμμα κλάσεων που περιλαμβάνει κλάσεις με ιδιότητες και λειτουργίες και πολλές από τις προαναφερθείσες συσχετίσεις μεταξύ τους.




**Εικόνα 10 Σύνθετο Παράδειγμα διαγράμματος κλάσεων**

Στο παραπάνω διάγραμμα κλάσεων κανένας ή περισσότεροι σπουδαστές μετέχουν σε κανένα ή περισσότερα μαθήματα, τα οποία διδάσκει ένας ή περισσότεροι εκπαιδευτές. Επιπλέον, ένα ή περισσότερα μαθήματα συσχετίζονται με έναν ή περισσότερους τομείς, ενώ ένας εκπαιδευτικός διευθύνει κανένα (άρα δεν διευθύνει) ή ένα τομέα. Ένας ή περισσότεροι τομείς συνθέτουν ένα τμήμα ή ένα τμήμα απαρτίζεται από ένα ή περισσότερους τομείς.

Συνεπώς, με ένα τέτοιο διάγραμμα κλάσεων μπορεί πολύ εποπτικά και σύντομα να μοντελοποιηθεί ένα σύστημα όπως ένα ίδρυμα τριτοβάθμιας εκπαίδευσης.

## 3.1.2 Διαγράμματα Αντικειμένων (Object)

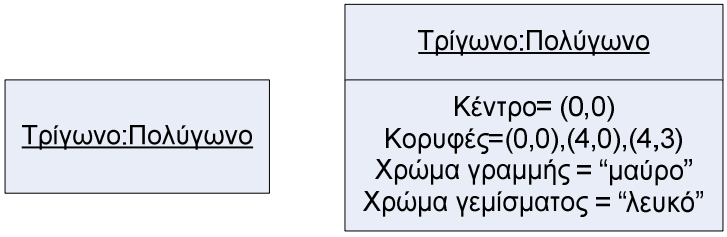
Τα διαγράμματα αντικειμένων της **UML,** που συχνά αναφέρονται και ως διαγράμματα στιγμιότυπων, είναι χρήσιμα για τη διερεύνηση **"**πραγματικών**"** προβλημάτων που θέλουμε να μοντελοποιήσουμε. Αναπαριστούν τις σχέσεις μεταξύ των στιγμιότυπων των κλάσεων και των ίδιων των κλάσεων. Ουσιαστικά είναι παρόμοια με τα διαγράμματα κλάσεων και χρησιμοποιούν και την ίδια σημειογραφία.

Τα διαγράμματα αντικειμένων δεν θεωρούνται τόσο σημαντικά αλλά χρησιμοποιούνται συχνά όταν τα αντίστοιχα διαγράμματα κλάσεων είναι σύνθετα. Επιπλέον λόγω ότι τα διαγράμματα κλάσεων συχνά θεωρούνται **"**ασαφή**"** και **"**αφηρημένα**",** τα διαγράμματα αντικειμένων αποτελούν μια πιο **"**κατανοητή**"** παρουσίαση του συστήματος και μάλιστα παρέχουν και ένα τρόπο ελέγχου της ορθότητας του διαγράμματος κλάσης.

Ο συμβολισμός του αντικειμένου προέρχεται από το συμβολισμό της κλάσης υπογραμμίζοντας τα στοιχεία επιπέδου στιγμιότυπου. Δηλαδή,

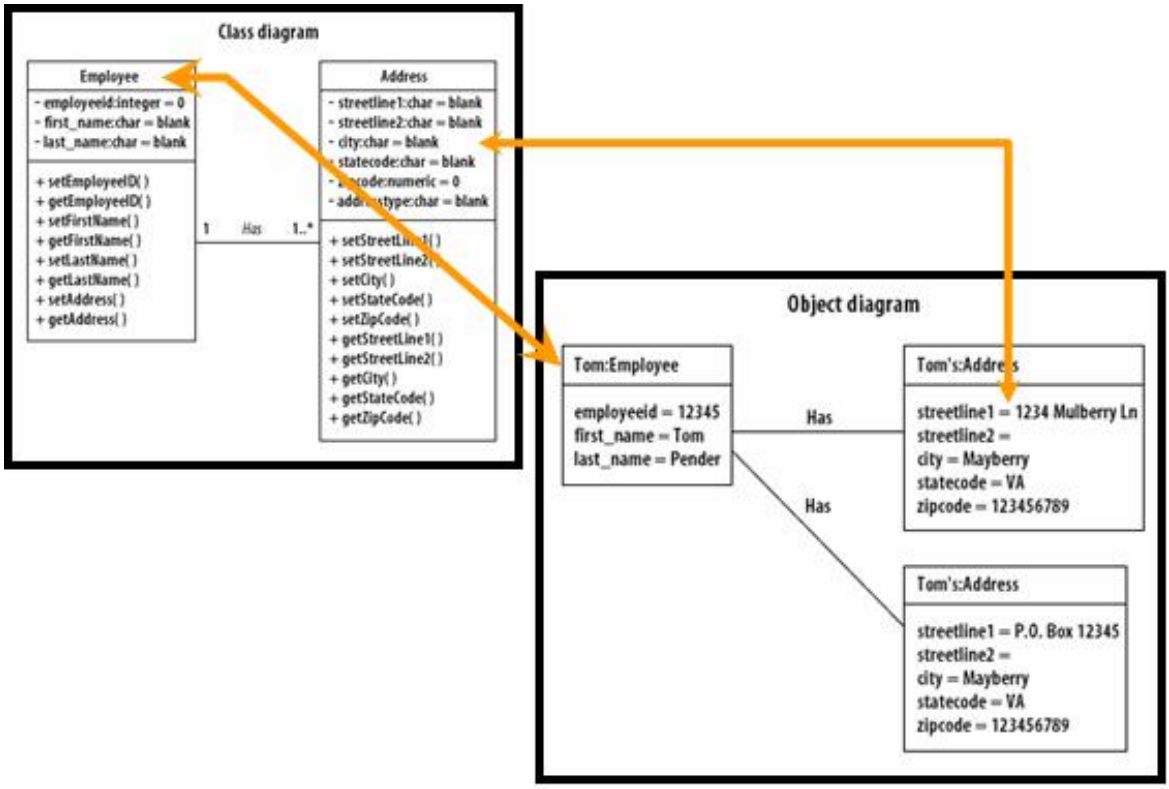
όνομα αντικειμένου : όνομα κλάσης

Ενίοτε, το αντικείμενο συμβολίζεται και με ένα ορθογώνιο, χωρισμένο σε δύο τμήματα, όπου στο πάνω τμήμα τοποθετείται το όνομα του αντικειμένου και στο κάτω τοποθετούνται τα χαρακτηριστικά του αντικείμένου με τη τιμή τους σε εισαγωγικά ή παρενθέσεις. Ακολουθεί, ένα παράδειγμα απεικονίσεων ενός αντικειμένου.



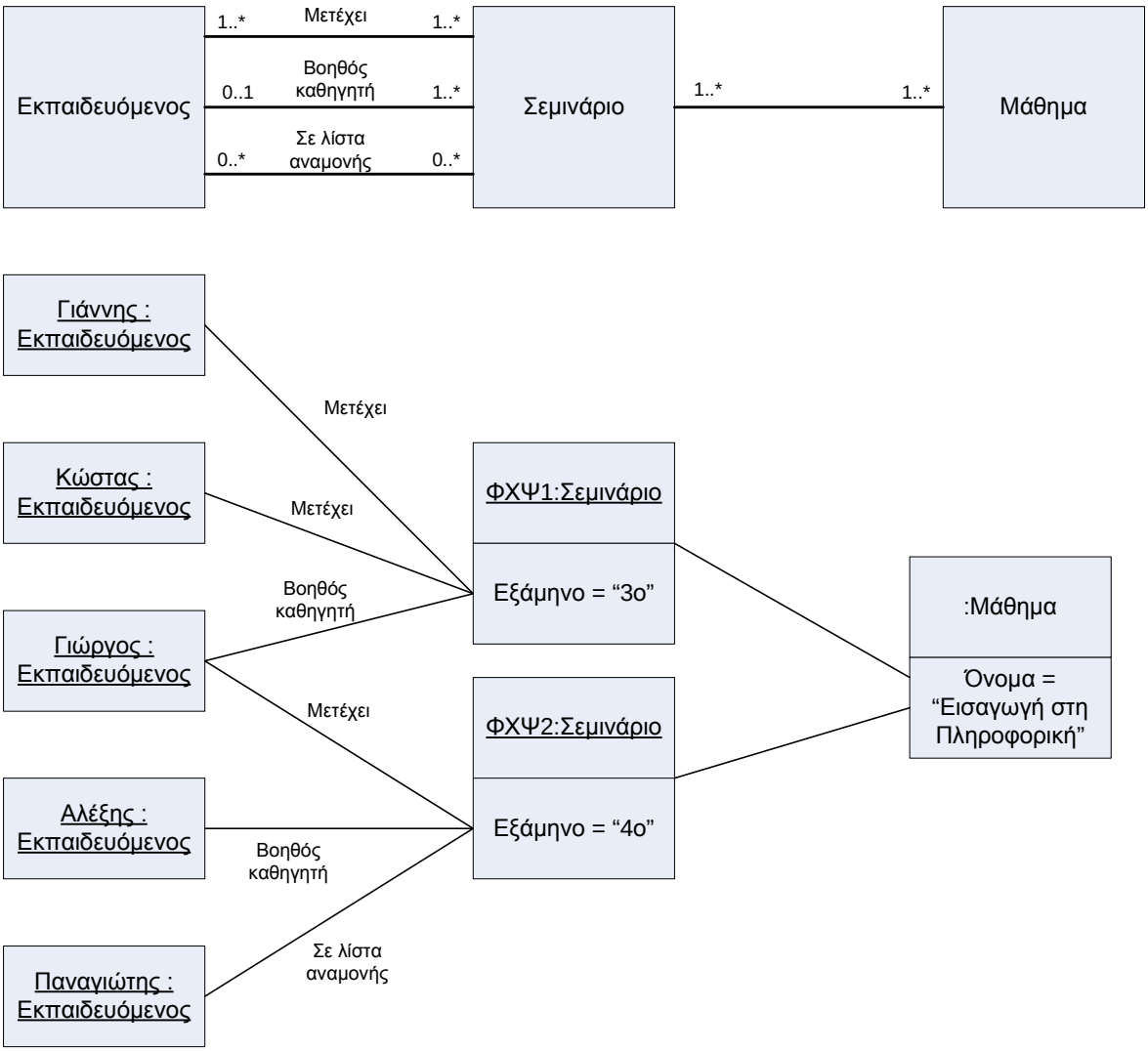

**Εικόνα 11 Τρόποι απεικόνισης αντικειμένου**

Ακολουθεί ένα παράδειγμα όπου παρουσιάζεται ένα διάγραμμα κλάσης και το αντίστοιχο διάγραμμα αντικειμένου ως στιγμιότυπο της κλάσης στην οποία ανήκει.




**Εικόνα 12 Παράδειγμα αντικειμένου ως στιγμιότυπο κλάσης**

Παρακάτω φαίνεται ένα διάγραμμα κλάσεων και το αντίστοιχο διάγραμμα αντικειμένων**.**




**Εικόνα 13 Αντιστοίχιση διαγράμματος αντικειμένων και κλάσεων**

Στο πάνω μέρος της εικόνας παρουσιάζεται το διάγραμμα κλάσεων που υποδηλώνει ότι ένας ή περισσότεροι εκπαιδευόμενοι συμμετέχουν σε ένα ή περισσότερα σεμινάρια, ενώ ο καθηγητής δύναται να έχει κανένα ή ένα βοηθό και υπάρχουν κανένας έως πολλοί σε λίστα αναμονής. Επίσης, ένα ή περισσότερα σεμινάριο αποτελούν ένα ή περισσότερα μαθήματα.

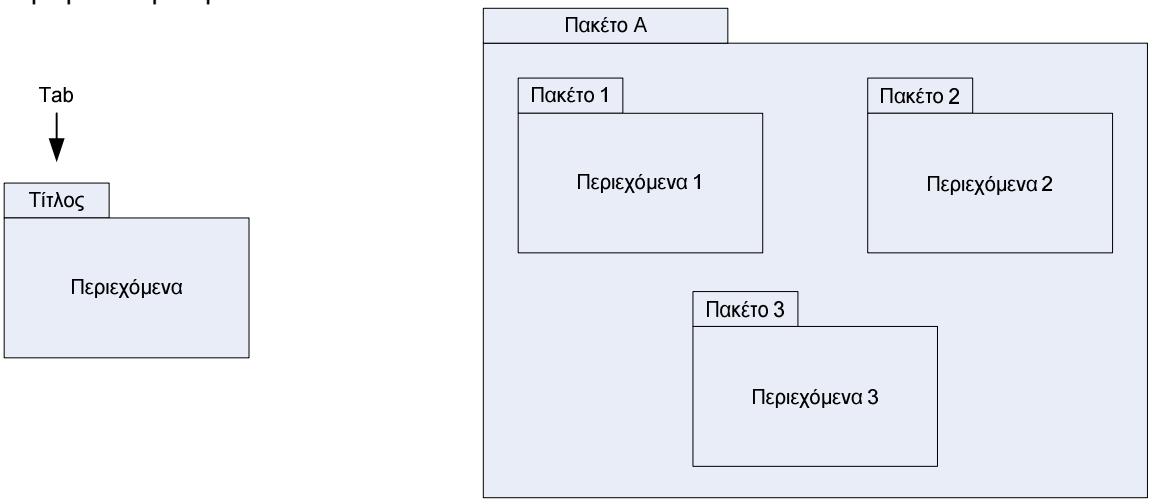
Στο κάτω μέρος του σχήματος παρουσιάζεται το αντίστοιχο διάγραμμα αντικειμένων. Είναι προφανές ότι παρουσιάζει πιο απλά και "απτά" τα αντικείμενα και το πώς συνδέονται μεταξύ τους. Κάποιοι εκπαιδευόμενοι απλά μετέχουν στα σεμινάρια, κάποιο είναι βοηθοί καθηγητή στα σεμινάρια και ένας βρίσκεται σε λίστα αναμονής. Επίσης, τα δύο σεμινάρια συσχετίζονται με το Μάθημα Εισαγωγή στη Πληροφορική.

### 3.1.3 Διαγράμματα Πακέτων (Package)

Ένα από τα πιο κλασικά ερωτήματα στα συστήματα λογισμικού, είναι πως αναλύεις ένα σύνθετο και μεγάλο σύστημα σε μικρότερα, ώστε να είναι πιο εύκολη η κατανόηση και η επέμβαση σε αυτό. Παλαιότερα, στο δομημένο **(**ακολουθιακό**)** προγραμματισμό το συνολικό σύστημα χαρτογραφούταν ως μια λειτουργία και έπειτα αναλύονταν σε υπό-λειτουργίες. Κάτι τέτοιο όμως δεν μπορεί να εφαρμοστεί στον αντικειμενοστραφή προγραμματισμό καθώς προϋποθέτει ότι υπάρχει πλήρης διαχωρισμός διαδικασιών και δεδομένων. Στον αντικειμενοστραφή προγραμματισμό αυτό που συνήθως εφαρμόζεται είναι η ομαδοποίηση κλάσεων ή άλλων στοιχείων μοντελοποίησης σε υψηλότερου επιπέδου μονάδες που ονομάζονται πακέτα.

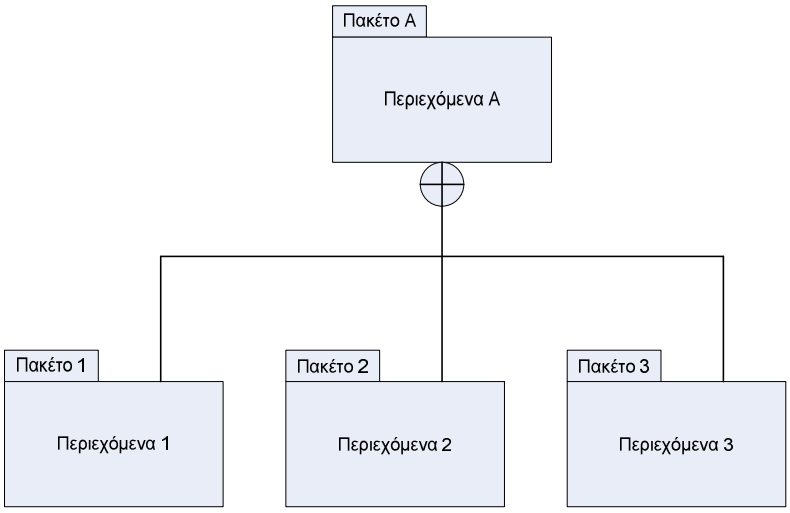
Ένα πακέτο είναι μία ομαδοποίηση στοιχείων μοντελοποίησης. Τα ίδια τα πακέτα μπορούν να είναι ένθετα **(nested)** σε άλλα πακέτα, δηλαδή ένα πακέτο μπορεί να περιέχει υφιστάμενα πακέτα, καθώς και άλλα στοιχεία μοντελοποίησης. Όλα τα είδη των στοιχείων μοντελοποίησης της **UML** μπορούν να οργανωθούν σε πακέτα.

Τα πακέτα συμβολίζονται με ένα ορθογώνιο, που στο πάνω αριστερό τμήμα του διαθέτει ένα μικρότερο ορθογώνιο **(tab)**. Τα περιεχόμενα του πακέτου τοποθετούνται στο μεγάλο ορθογώνιο.




**Εικόνα 14 Απλό και ένθετα πακέτα**

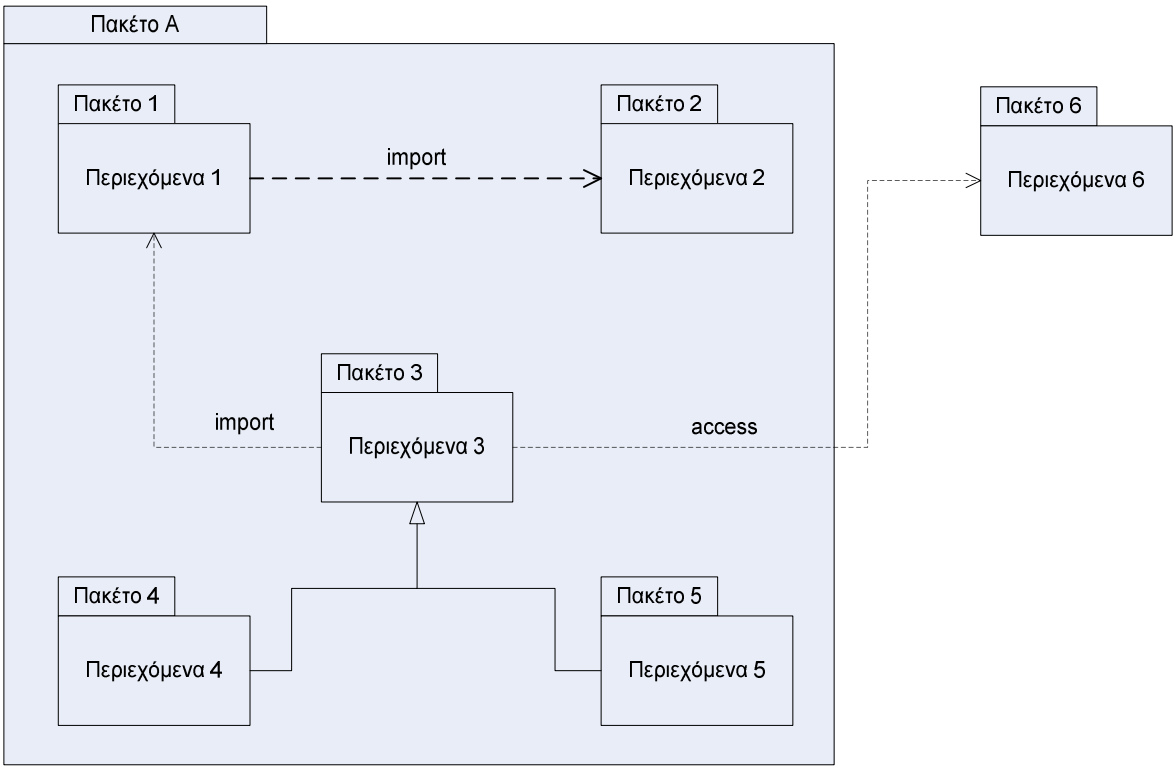
Τα περιεχόμενα των πακέτων μπορούν επίσης να παρουσιάζονται με διακλαδιζόμενες γραμμές προς τα περιεχόμενα στοιχεία, εκτός του πακέτου, όπως παρουσιάζεται στο παράδειγμα στη συνέχεια. Στην πλευρά της γραμμής που είναι προσκολλημένη στο περιέχον αντικείμενο σχεδιάζεται ένα σύμβολο συν **(+)** μέσα σε κύκλο.




**Εικόνα 15 Εναλλακτικός τρόπος παρουσίασης ένθετων πακέτων**

Αν τα περιεχόμενα του πακέτου δεν παρουσιάζονται μέσα στο μεγάλο ορθογώνιο, τότε το όνομα του μπορεί να τοποθετηθεί στο μεγάλο ορθογώνιο, διαφορετικά μπορεί να τοποθετηθεί στο **tab.**

Μπορούν να υπάρχουν σχέσεις ανάμεσα σε πακέτα και δηλώνουν ότι υπάρχει η αντίστοιχη σχέση ανάμεσα σε στοιχεία των δύο πακέτων. Μία σχέση εισαγωγής **(import)** ή προσπέλασης **(access)** ανάμεσα σε δύο πακέτα παρουσιάζεται με διακεκομμένη γραμμή και βέλος με ανοιχτή κεφαλή και την ετικέτα **«import»** ή **«access»**, αντίστοιχα.

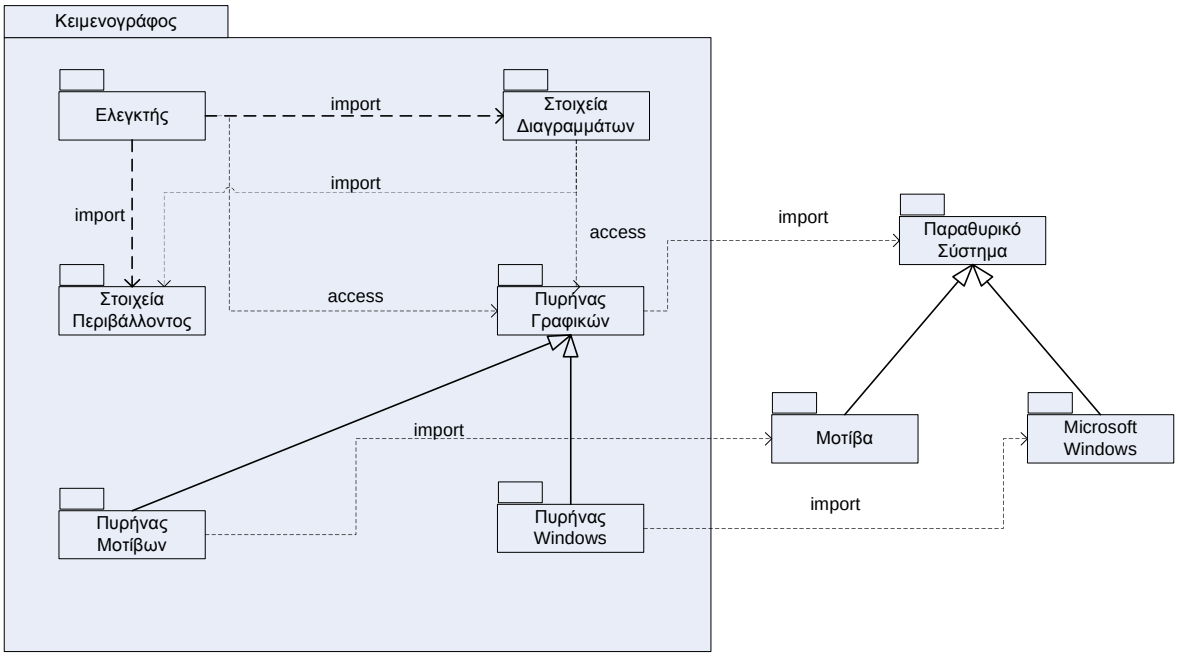



**Εικόνα 16 Ένθετα πακέτα με σχέσεις εισαγωγής , προσπέλασης και γενίκευσης**

Στο παραπάνω σχήμα εντός του Πακέτου Α υπάρχουν τα ένθετα Πακέτα **1** έως **5** ενώ εκτός του Πακέτου Α υπάρχει το Πακέτο **6.** Τα περιεχόμενα του Πακέτου **1** εισάγονται στα περιεχόμενα του Πακέτου **2,** ενώ ταυτόχρονα έχουν προσπελαύνουν τα περιεχόμενα του Πακέτου **6.** Επίσης, το Πακέτο **3** αποτελεί γενίκευση των Πακέτων **4** και **5.** Έτσι, στο παραπάνω μοντέλο, το Πακέτο **1** επηρεάζει το Πακέτο **2,** ενώ το Πακέτο **3** επηρεάζει το Πακέτο **1** και το Πακέτο **6.**

Παρακάτω ακολουθεί ακόμα ένα παράδειγμα, όπου ένας κειμενογράφος για λειτουργικό **Windows,** περιλαμβάνει τα πακέτα ενός Ελεγκτή, το πακέτο Στοιχείων Περιβάλλοντος, το πακέτο Στοιχείων Διαγραμμάτων, το πακέτο Γραφικών, το πακέτο Πυρήνας Μοτίβων και το πακέτο Πυρήνας **Windows.**

Εξωτερικά του πακέτου Κειμενογράφου υπάρχουν το πακέτο Παραθυρικό Σύστημα που είναι γενίκευση των πακέτων Μοτίβων και του πακέτου **Microsoft Windows.**




**Εικόνα 17 Παράδειγμα μοντελοποίησης κειμενογράφου με διάγραμμα πακέτων**

Το παραπάνω σχήμα μοντελοποιεί έναν ηλεκτρονικό Κειμενογράφο ως εξής: Ο Πυρήνας Γραφικών είναι αυτός που ουσιαστικά παρέχει τα στοιχεία που είναι απαραίτητα ώστε το Παραθυρικό Σύστημα των **Windows** να εμφανίσει το Κειμενογράφο. Ο Πυρήνας Γραφικών, ουσιαστικά ενεργοποιείται από τον Ελεγκτή και τα Στοιχεία Διαγραμμάτων, που έχουν δικαίωμα να τον προσπελάσουν, ενώ παράλληλα αποτελεί γενίκευση των Πυρήνων Μοτίβων και **Windows.**

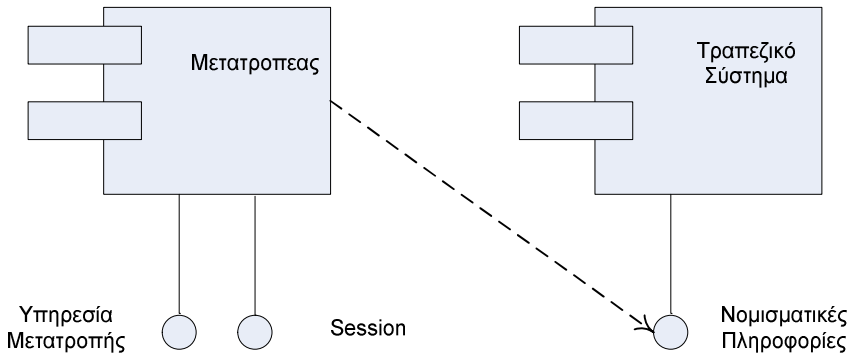
### 3.1.4 Διαγράμματα Συστατικών (Component)

Η **UML** παρέχει δύο είδη διαγραμμάτων που φανερώνουν την αρχιτεκτονική του υπό μοντελοποίηση συστήματος. Αυτά είναι τα διαγράμματα Συστατικών **(Component)** και Παράταξης **(Deployment)**. Ένα Συστατικό **(Component)** αναπαριστά ένα τμήμα κώδικα, μία ενότητα λογισμικού που εκθέτει τις λειτουργίες της μέσω ενός συνόλου δημόσιων διασυνδέσεων και μπορεί να συνδεθεί δυναμικά με άλλα συστατικά για τη δημιουργία μεγαλύτερων συστατικών και εφαρμογών.

Ένα Συστατικό λοιπόν πρέπει να διαθέτει τις λειτουργίες του μέσω διασυνδέσεων, διατηρώντας την εσωτερική του λειτουργία κρυφή, να συνδέεται δυναμικά με άλλα συστατικά και να είναι επαναχρησιμοποιήσιμο.

Η αναπαράσταση των Συστατικών στη **UML** πραγματοποιείται μέσω του διάγραμμα Συστατικών, όπου αυτά παριστάνονται σαν παραλληλόγραμμα, από την αριστερή πλευρά των οποίων εξέρχονται δύο μικρότερα παραλληλόγραμμα. Οι διεπαφές των Συστατικών συμβολίζονται με μικρούς λευκούς κύκλους, που συνδέονται μέσω ευθύγραμμων τμημάτων με το Συστατικό (ονομάζεται **lollipop**). Τα συστατικά μπορούν να εξαρτώνται από άλλα συστατικά με χρήση των διασυνδέσεων. Η εξάρτηση αυτή απεικονίζεται με ένα κατευθυνόμενο διακεκομμένο βέλος, με αρχή από το συστατικό που χρησιμοποιεί τη διασύνδεση και κατάληξη στη διασύνδεση του συστατικού που την παρέχει.

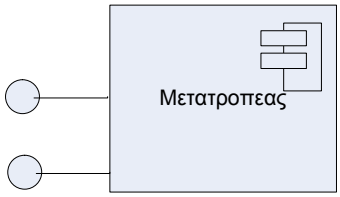
Στο παρακάτω σχήμα παρουσιάζεται ένα διάγραμμα Συστατικών για τη μοντελοποίηση ενός συστήματος μετατροπής νομισματικών ισοτιμιών.




**Εικόνα 18 Διάγραμμα συστατικών συστήματος μετατροπής νομισματικών ισοτιμιών**

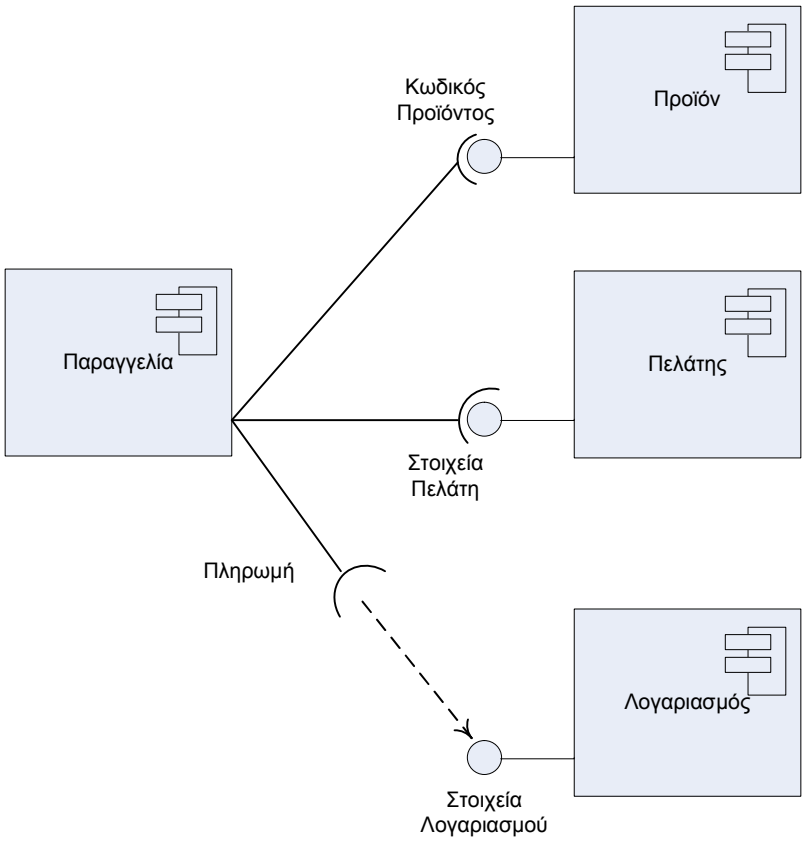
Αριστερά είναι το Συστατικό "Μετατροπέας" που διαθέτει δύο διεπαφές, μια για πρόσβαση στην Υπηρεσία Μετατροπής και μια δεύτερη για το **Session** (σύνοδο) της υπηρεσίας, ενώ δεξιά είναι το συστατικό "Τραπεζικό Σύστημα" που διαθέτει μια διεπαφή για Νομισματικές Πληροφορίες. Το συστατικό αριστερά μπορεί να χρησιμοποιηθεί από πελάτες, ανοίγοντας ένα **session** και καλώντας τις μεθόδους που υλοποιούν την Υπηρεσία Μετατροπής. Για να επιστρέψει αποτελέσματα η Υπηρεσία Μετατροπής, το συστατικό Μετατροπέας χρησιμοποιεί το συστατικό Τραπεζικό Σύστημα και συγκεκριμένα τη διεπαφή Νομισματικές Πληροφορίες ώστε να πάρει τις ισοτιμίες μεταξύ των διαφόρων νομισμάτων.

Εναλλακτικά και στα πλαίσια της **UML 2.x** ένα Συστατικό μπορεί να απεικονιστεί και με την παρακάτω σήμανση.




**Εικόνα 19 Συστατικό της UML 2.x**

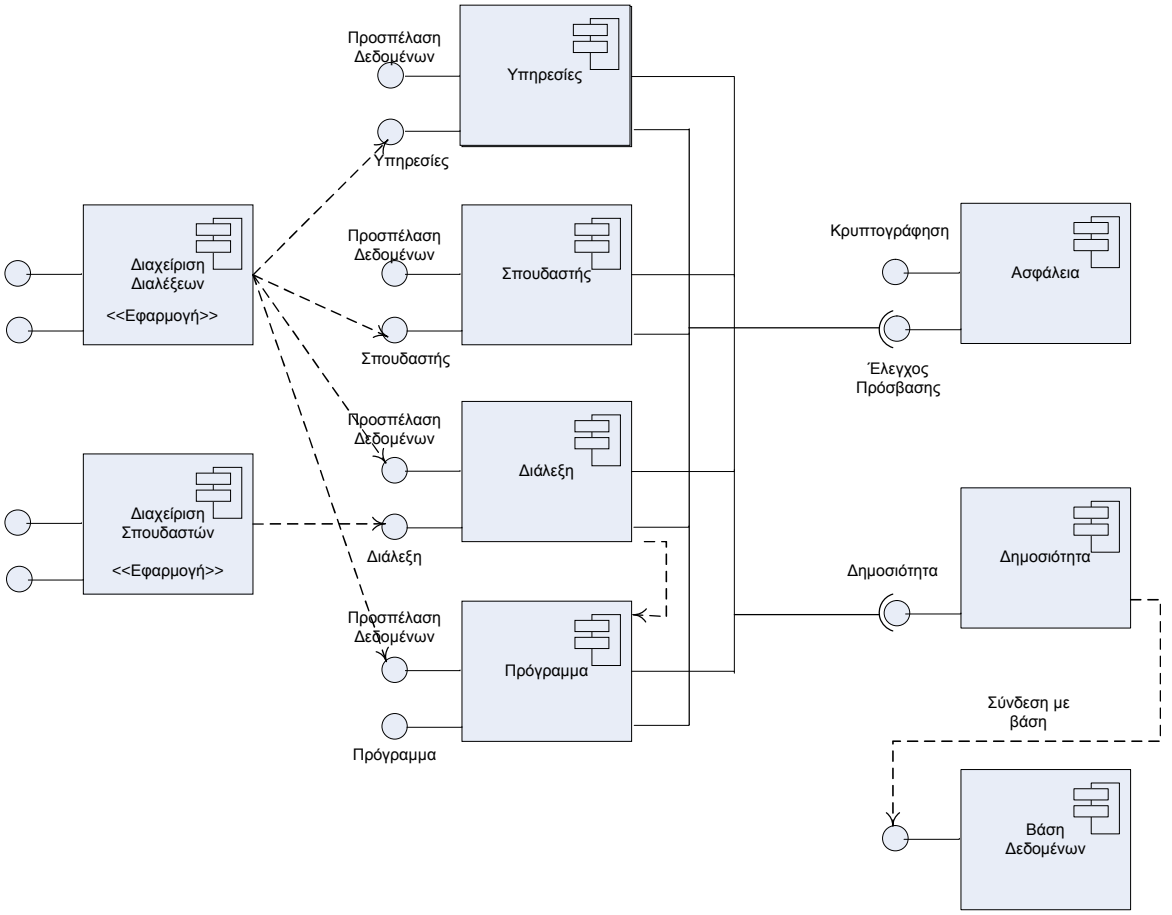
Ακολουθεί ένα παράδειγμα μοντελοποίησης της παραγγελίας ενός προϊόντος.




**Εικόνα 20 Μοντελοποίηση παραγγελίας προϊόντος**

Στο παραπάνω σχήμα, το Συστατικό παραγγελία προϋποθέτει την ύπαρξη διεπαφών με τα συστατικά Προϊόν και Πελάτη, ώστε να αντληθούν ο Κωδικός του Προϊόντος και τα Στοιχεία του Πελάτη. Επίσης, για να πραγματοποιηθεί η πληρωμή απαιτείται μια σχέση με το συστατικό Λογαριασμό.

Ένα πιο σύνθετο παράδειγμα διαγράμματος Συστατικών που μοντελοποιεί ένα ακαδημαϊκό σύστημα, φαίνεται στο παρακάτω σχήμα.



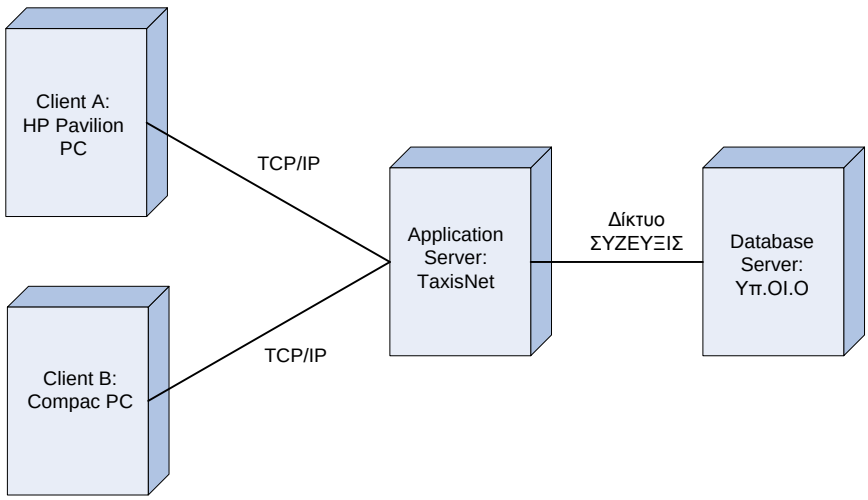

**Εικόνα 21 Μοντελοποίηση ακαδημαϊκού συστήματος**

## 3.1.5 Διαγράμματα Παράταξης (Deployment)

Τα διαγράμματα Παράταξης αναπαριστούν την αρχιτεκτονική του υλικού **(hardware)** και του λογισμικού **(software)** σε ένα σύστημα**.** Με το διάγραμμα αναπαριστώνται οι μηχανές και οι υπολογιστές **(**κόμβοι**)** που χρησιμοποιούνται από το σύστημα καθώς και οι διασυνδέσεις τους**.** Κόμβοι είναι φυσικά αντικείμενα που διαθέτουν κατ' ελάχιστο μνήμη και επεξεργαστικές δυνατότητες**.**

Στο παρακάτω παράδειγμα φαίνεται ένα απλουστευμένο διάγραμμα Παράταξης με τέσσερις κόμβους που ο καθένας αποτελεί ξεχωριστή υπολογιστική μονάδα και διασυνδέονται μεταξύ τους με πρωτόκολλα επικοινωνίας**.** Υπάρχουν δύο κόμβοι **(client PCs)** που μέσω πρωτοκόλλου **TCP/IP** έχουν πρόσβαση στον **Application Server** του **TaxisNet** και εκτελούν αντίστοιχες εφαρμογές**,** για παράδειγμα Περιοδική Δήλωση ΦΠΑ ή Συμπλήρωση Στοιχείων Ε**9.** Για την εκτέλεση των εφαρμογών**,** ο **Application Server** χρειάζεται να έχει πρόσβαση με τη σειρά του στα φορολογικά στοιχεία των κατόχων των δύο υπολογιστών**,** κάτι που είναι εφικτό μέσω τη σύνδεσης με τον **Database Server**

του Υπουργείου Οικονομικών, ο οποίος περιλαμβάνει όλα τα φορολογικά στοιχεία των πολιτών.




**Εικόνα 22 Διάγραμμα Παράταξης με κόμβους**

Οι κόμβοι συμπεριλαμβάνουν Συστατικά **(Components)**, δηλαδή τμήματα κώδικα που αναλαμβάνουν να εκτελέσουν συγκεκριμένες λειτουργίες. Τα Συστατικά των κόμβων μπορούν να εκτελούν λειτουργίες υποστήριξης των κόμβων και επιπλέον μπορούν να συνεργάζονται με συστατικά άλλων κόμβων. Έτσι, μέσω των διαγραμμάτων Παράταξης είναι εφικτή η μοντελοποίηση και απεικόνιση της αρχιτεκτονικής των συστημάτων.

Ένα στιγμιότυπο κόμβου έχει όνομα και ένα τύπο. Ο κόμβος μπορεί αν έχει υπογραμμισμένο το όνομά του και το όνομα μπορεί να βρίσκεται μέσα ή κάτω από τον κόμβο. Το όνομα έχει τη σύνταξη:

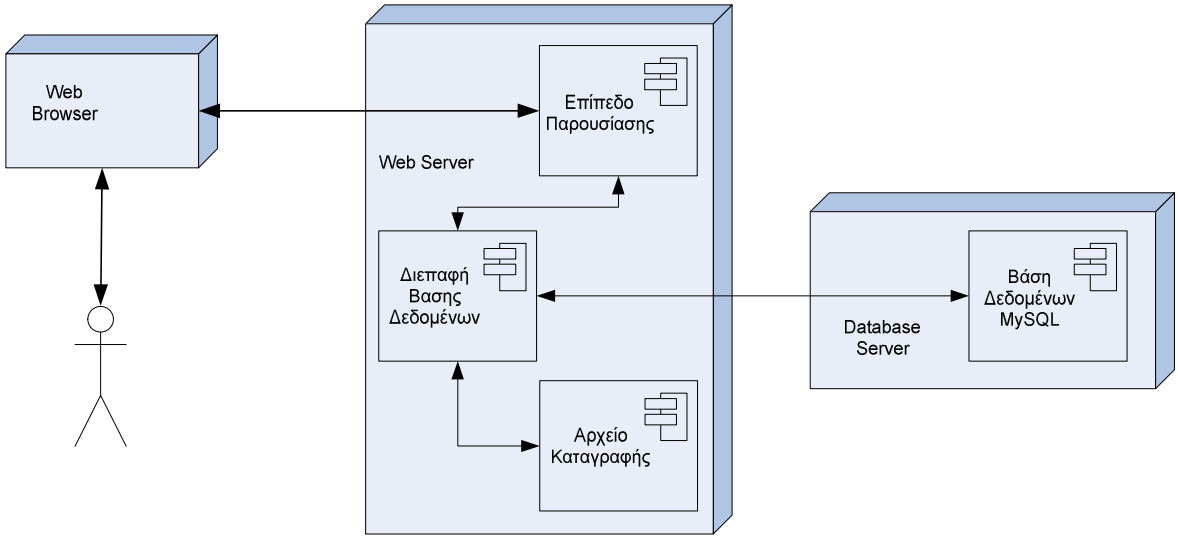
όνομα : τύπος-κόμβου

Το όνομα είναι το όνομα ενός οποιουδήποτε κόμβου **(**εφόσον υπάρχει**)**. Ο τύπος-κόμβου λέει τι είδους είναι ο κόμβος. Οποιοδήποτε ή και τα δύο στοιχεία είναι προαιρετικά. Αν παραληφθεί ο τύπος του κόμβου, μπορεί να παραληφθεί η άνω-κάτω τελεία.

Οι κόμβοι μπορεί να είναι συνδεδεμένοι με συσχετίσεις με άλλους κόμβους, όπως στο παραπάνω διάγραμμα. Μία συσχέτιση ανάμεσα σε κόμβους δηλώνει μία διαδρομή επικοινωνίας ανάμεσα στους κόμβους. Η συσχέτιση μπορεί να δηλώνει το είδος της διαδρομής επικοινωνίας **(**για παράδειγμα, το είδος του καναλιού ή του δικτύου**)**.

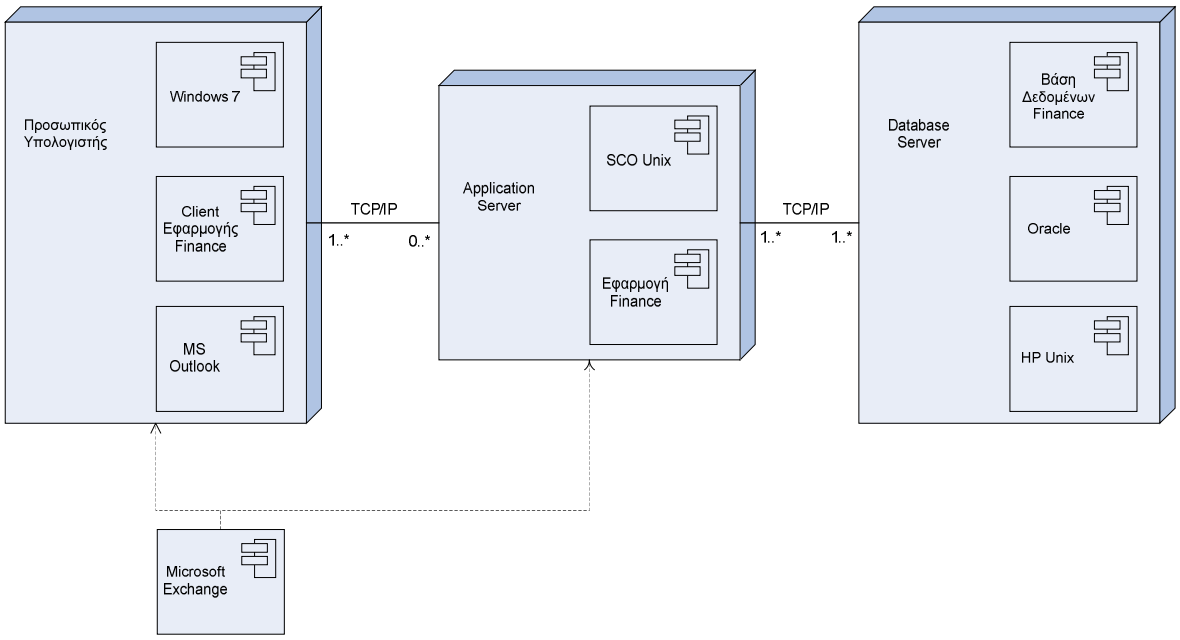
Στιγμιότυπα Συστατικών και αντικείμενα μπορούν να περιέχονται μέσα σε σύμβολα στιγμιότυπων κόμβων. Αυτό δηλώνει ότι τα στοιχεία βρίσκονται στα στιγμιότυπα των κόμβων. Σε αυτή τη περίπτωση, συχνά χρησιμοποιούνται διακεκομμένα βέλη με μια λέξη κλειδί που παρουσιάζουν τη δυνατότητα ενός τύπου κόμβου να υποστηρίξει έναν τύπο Συστατικού. Εναλλακτικά, αυτό μπορεί να φανεί με ένθεση του συμβόλου του συστατικού στο σύμβολο του κόμβου.

Παρακάτω, παρουσιάζεται ένα απλό παράδειγμα διαγράμματος Παράθεσης που παρουσιάζει τη διαδικασία πρόσβασης ενός χρήστη σε ένα δυναμικό **web site.** Ο χρήστης μέσω του **Web browser** του έχει πρόσβαση στο επίπεδο παρουσίασης του δυναμικού **web site,** το οποίο για να **"**φέρει**"** τα ζητούμενα δεδομένα**,** πρέπει να επικοινωνήσει με το **database server** που περιλαμβάνει τη βάση δεδομένων σε **mySQL.** Την επικοινωνία αυτή την αναλαμβάνει η διεπαφή βάσης δεδομένων που είναι τμήμα του **web server,** και παράλληλα αυτή η δραστηριότητα καταγράφεται σε ένα αρχείο καταγραφής **Log file.**




**Εικόνα 23 Διάγραμμα παράταξης για πρόσβαση σε δυναμικό web site**

Ακολουθεί ένα πιο σύνθετο παράδειγμα για τη πρόσβαση χρηστών σε μια δικτυακή εφαρμογή του οικονομικού κλάδου**.**




**Εικόνα 24 Διάγραμμα παράταξης για οικονομική εφαρμογή client - server**

Πολλαπλοί χρήστες χρησιμοποιούν τους προσωπικούς τους υπολογιστές με **windows 7,** τρέχουν τον **client** της εφαρμογής **Finance** στους υπολογιστές τους και συνδέονται μέσω **TCP/IP** σε κάποιον από τους πολλούς **application servers.** Οι **application servers** διαθέτουν λειτουργικό **SCO Unix** και έχουν εγκατεστημένη και την εφαρμογή **Finance,** ενώ επικοινωνούν μέσω **TCP/IP** με ένα κεντρικό **server** βάσης δεδομένων που τρέχει λειτουργικό **HP Unix** και **Oracle** την διαχείριση της βάσης δεδομένων που χρησιμοποιεί η εφαρμογή **Finance. Η** επικοινωνία μεταξύ των χρηστών και των **application servers** γίνεται μέσω **MS-Outlook** που υποστηρίζεται από τον **Microsfot Exchange.**

## 3.1.6 Σύνθετης Δομής (Composite Structure)

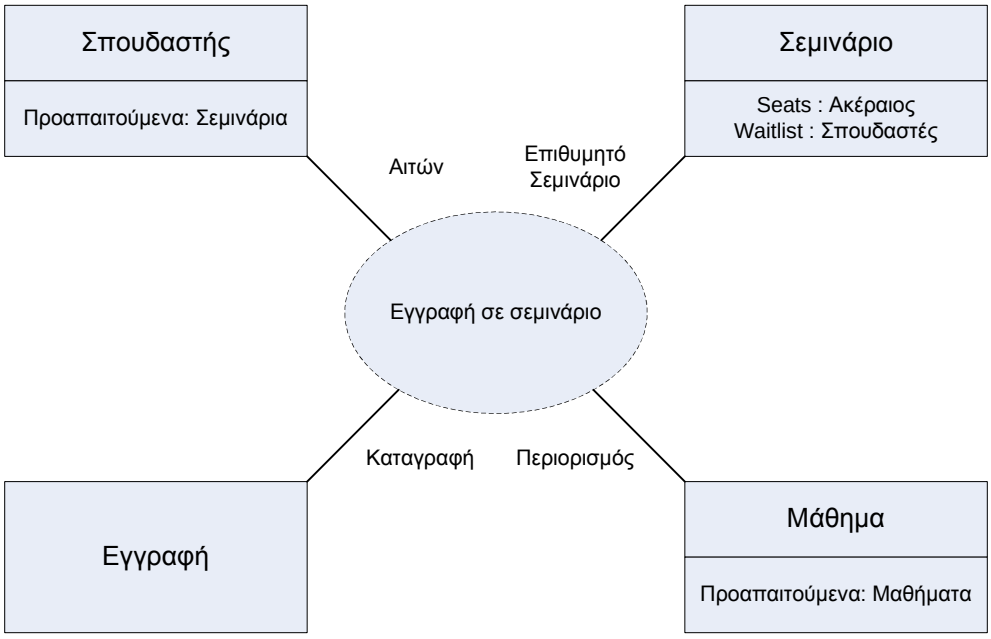
Τα διαγράμματα Σύνθετης Δομής είναι ένας τύπος διαγραμμάτων στατικής δομής και είναι παρόμοια με τα διαγράμματα κλάσεων και αναπαριστούν μέλη **(parts)** και συνδέσεις των κλάσεων. Τα μέλη δεν αναπαριστούν απαραίτητα κλάσεις ή στιγμιότυπα κλάσεων, αλλά ρόλους εντός του συστήματος και η σήμανση τους είναι σαν των αντικειμένων. Επίσης, μπορούν να αναπαραστήσουν σχέσεις ή συνδέσεις μεταξύ μελών (π.χ. περιπτώσεις χρήσης) ή ακόμα και εκτελέσιμα στοιχεία που χρειάζεται ένα μέλος του συστήματος.

Αυτά τα διαγράμματα μπορούν να περιλαμβάνουν τα εσωτερικά μέλη, τις πόρτες **(ports),** μέσω των οποίων τα μέλη αλληλεπιδρούν μεταξύ τους ή τα στιγμιότυπα των κλάσεων αλληλεπιδρούν με τα μέλη και με τον έξω κόσμο, καθώς και συνδέσεις **(connectors)** μεταξύ των μελών ή **ports.** Μια Σύνθετη Δομή είναι ένα σύνολο αλληλοσυνδεόμενων στοιχείων που συνεργάζονται **(collaboration)** κατά το χρόνο

εκτέλεσης για την επίτευξη κάποιου σκοπού. Κάθε στοιχείο έχει κάποιο συγκεκριμένο ρόλο στη συνεργασία αυτή.

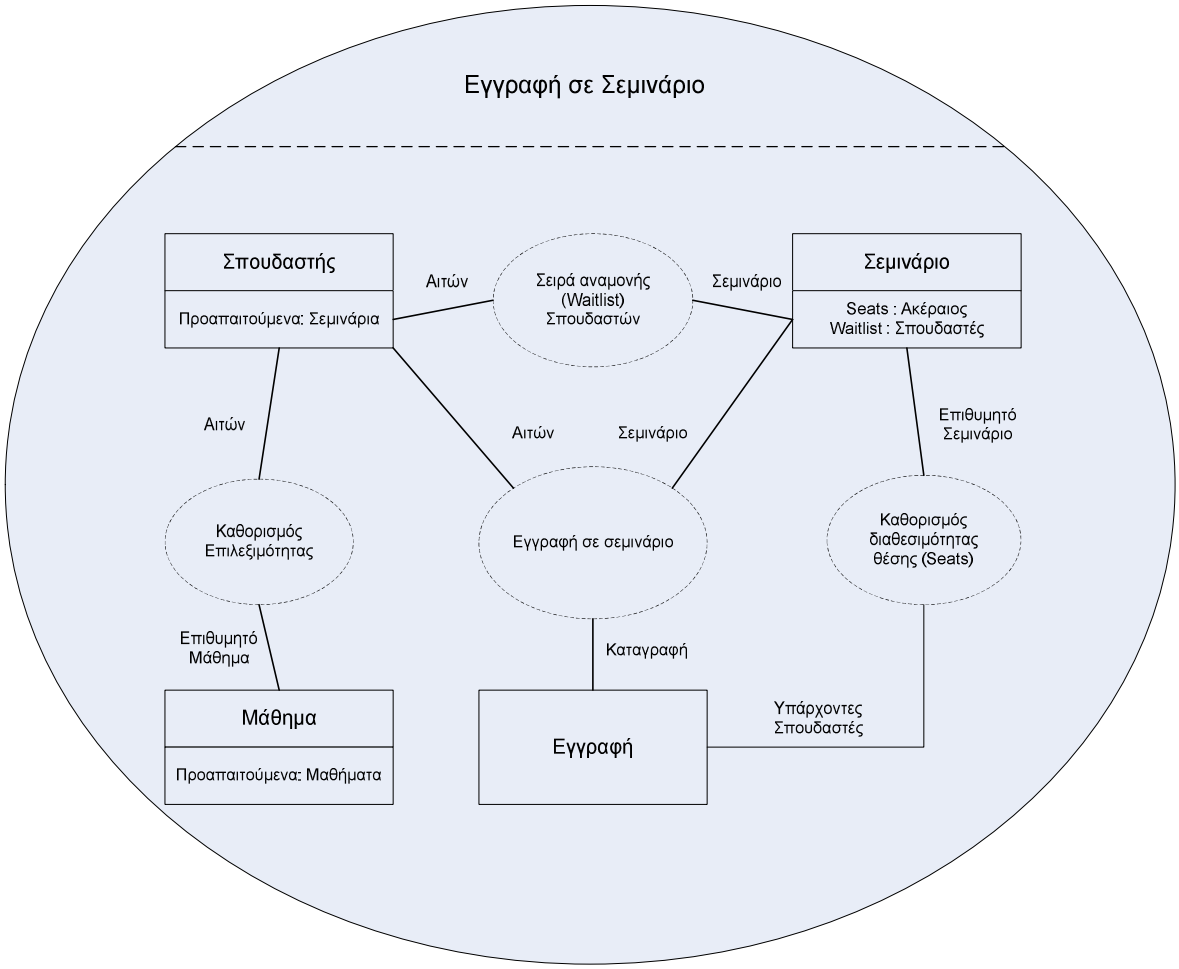
Μια συνεργασία είναι γενικά μια αφηρημένη κλάση, που συμβολίζεται με ένα οβάλ με διακεκομμένη γραμμή, που περιέχει ρόλους που μπορούν να αναλάβουν τα στιγμιότυπα στα πλαίσια της συνεργασίας.

Το παρακάτω διάγραμμα παρουσιάζει ένα διάγραμμα Σύνθετη Δομής που μοντελοποιεί τη διαδικασία εγγραφής σε ένα σεμινάριο. Στο κέντρο το οβάλ με τίτλο "Εγγραφή σε σεμινάριο" μοντελοποιεί μια συνεργασία. Μια συνεργασία δίνει τη δυνατότητα μοντελοποίησης των σχετικών πτυχών συνεργασίας μεταξύ των στιγμιότυπων, καθορίζοντας τα αντικείμενα και τους ρόλους που θα χρειαστούν στο πλαίσιο της συνεργασίας. Τα ορθογώνια αναπαριστούν στιγμιότυπα οποιουδήποτε τύπου του στοιχείου (ταξινομητή), συμπεριλαμβανομένων των κλάσεων, αντικειμένων, ή διεπαφών. Οι ιδιότητες που χρησιμοποιούνται για την υλοποίηση της συνεργασίας, όπως για παράδειγμα η προϋπόθεση ο φοιτητής να έχει συμμετάσχει στο παρελθόν σε άλλα σεμινάρια, μπορούν να αναπαρασταθούν προαιρετικά όπως κλάσεις.




**Εικόνα 25 Διάγραμμα Σύνθετης Δομής για τη διαδικασία εγγραφής σε σεμινάριο**

Μια εναλλακτική μορφή του πιο πάνω διαγράμματος φαίνεται στο κάτωθι σχήμα, κάτι που συχνά αναφέρεται και ως διάγραμμα Σύνθετης Δομής τύπου Συνεργασίας, κάτι όμως που μπορεί να οδηγήσει σε σύγχυση σε οποιονδήποτε είναι εξοικειωμένος με **UML 1.x** όπου τα διαγράμματα συνεργασίας που περιελάμβανε, τώρα πια ονομάζονται διαγράμματα επικοινωνίας στην **UML 2.x**.




**Εικόνα 26 Διάγραμμα Συνεργασίας για τη διαδικασία εγγραφής σε σεμινάριο**

Σε αυτό το διάγραμμα το σύμβολο της Συνεργασίας περιλαμβάνει λεπτομερώς το διάγραμμα Σύνθετης Δομής, δείχνοντας πώς τα διαγράμματα σύνθετης δομής μπορούν αποτελεσματικά να είναι ένθετα το ένα μέσα στο άλλο.

Πάντως, πολλοί χρήστες αντί για τα διαγράμματα Σύνθετης Δομής προτιμούν τα διαγράμματα Ακολουθίας (που παρουσιάζονται αργότερα) για την μοντελοποίηση μιας Συνεργασίας, καθώς ο συμβολισμός τους είναι πολύ πιο ισχυρός και πιο κατανοητός από τους προγραμματιστές.

### 3.1.7 Προφίλ (Profile)

Ένα διάγραμμα Προφίλ λειτουργεί σε επίπεδο μεταμοντέλου για να δείξει τα στερεότυπα ως κλάσεις με το στερεότυπο **<<**στερεότυπο**>>**, και τα προφίλ ως πακέτα με το στερεότυπο **<<**προφίλ**>>**. Η σχέση επέκτασης (συνεχής γραμμή με κλειστό, γεμισμένη αιχμή βέλους) δείχνει ποιο στοιχείο μεταμοντέλου επεκτείνει ένα στερεότυπο.

Το Στερεότυπο είναι ένας από τους τρεις μηχανισμούς επέκτασης της **UML.** Επιτρέπει στους σχεδιαστές να επεκτείνουν το λεξιλόγιο της **UML,** προκειμένου να δημιουργήσουν νέα στοιχεία-μοντέλα, που προέρχονται από τα υπάρχοντα, αλλά έχουν νέες ειδικές ιδιότητες, κατάλληλες για το συγκεκριμένο τομέα ή πρόβλημα, ή άλλη εξειδικευμένη χρήση. Γραφικά, ένα στερεότυπο συμβολίζεται με ένα όνομα που περικλείεται από εισαγωγικά **(«»)** και τοποθετείται πάνω από το όνομα ενός άλλου στοιχείου **UML.**

Στη **UML,** η λέξη "γίνε" **(become),** είναι μια λέξη-κλειδί για ένα συγκεκριμένο στερεότυπο **UML,** και ισχύει για μια εξάρτηση **(**μοντελοποιείται με διακεκομμένο βέλος**).** Το **become** δείχνει ότι το στοιχείο από το οποίο ξεκινάει το βέλος, μετατρέπεται στο στοιχείο που δείχνει το βέλο, διατηρώντας παράλληλα κάποιο είδος ταυτότητας, παρόλο που μπορεί να έχουν αλλάξει οι τιμές, η κατάσταση, ή ακόμα και η κλάση.

Από την άλλη, ένα Προφίλ στη **UML,** παρέχει ένα γενικό μηχανισμό επέκτασης για τη προσαρμογή **UML** μοντέλων σε συγκεκριμένες περιοχές και πλατφόρμες. Οι μηχανισμοί επέκτασης επιτρέπουν τον λεπτομερή καθορισμό σήμανσης με αυστηρά προσθετικό τρόπο, ώστε να μην έρχονται σε αντιπαράθεση με τη **standard** σήμανση του προτύπου.

Τα διαγράμματα Προφίλ διαγράμματα δομής που περιγράφουν ένα "ελαφρύ" μηχανισμό επέκτασης της **UML,** ορίζοντας προσαρμοσμένα στερεότυπα, τις τιμές τους και τους περιορισμούς τους. Τα Προφίλ επιτρέπουν με αυτό το τρόπο τη προσαρμογή του μεταμοντέλου της **UML** σε:

- ⇒ διαφορετικές πλατφόρμες **(**όπως **J2EE** και. **NET),** ή
- ⇒ διαφορετικά περιβάλλοντα **(**όπως σε μοντελοποιήσεις πραγματικού χρόνου ή επιχειρηματικών διαδικασιών**)**

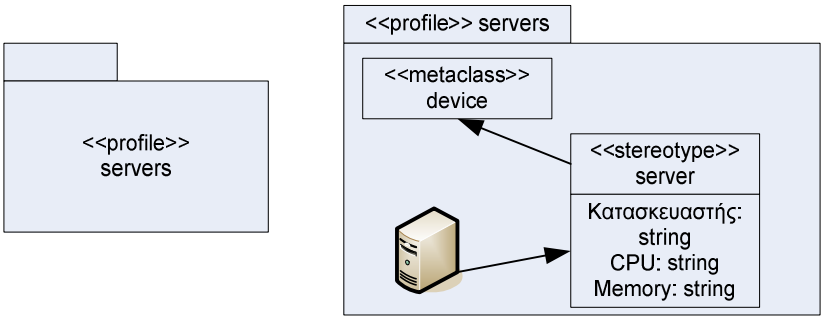
Για παράδειγμα, η σήμανση των **standard** στοιχείων της **UML** θα μπορούσε να εξειδικεύεται σε ένα Προφίλ π.χ. "μοντέλο της **Java,"** Ο μηχανισμός Προφίλ δεν είναι μια πρώτης τάξης μηχανισμός επέκτασης. Δεν επιτρέπει την τροποποίηση του υπάρχοντος **metamodel** ή τη δημιουργία ενός νέου. Επιτρέπει όμως την προσαρμογή ενός υπάρχοντος μεταμοντέλου για ένα ειδικό σκοπό ή ένα ειδικό τομέα, πλατφόρμα, ή μέθοδο. Επίσης, δεν είναι δυνατόν να αφαιρεθούν οι περιορισμοί που ισχύουν στο μεταμοντέλο, αλλά είναι δυνατή η προσθήκη νέων περιορισμών ειδικά για το προφίλ.

Τα προσαρμοσμένα **metamodels** ορίζονται σε ένα προφίλ, το οποίο εφαρμόζεται στη συνέχεια σε ένα Πακέτο. Τα Στερεότυπα είναι συγκεκριμένες μετακλάσεις, ενώ οι τιμές τους **(meta-tags)** είναι μεταχαρακτηριστικα **(meta-attributes),** και τα Προφίλ είναι ειδικοί τύποι πακέτων.

Τα προφίλ μπορούν να εφαρμοστούν δυναμικά ή να ανασυρθούν από ένα μοντέλο. Μπορούν επίσης να συνδυαστούν δυναμικά, έτσι ώστε περισσότερα από ένα προφίλ να εφαρμόζονται ταυτόχρονα στο ίδιο μοντέλο.

Τα Προφίλ υπήρχαν στη **UML 1.x.** αλλά τα διαγράμματα Προφίλ εισήχθησαν στη **UML 2.2.**

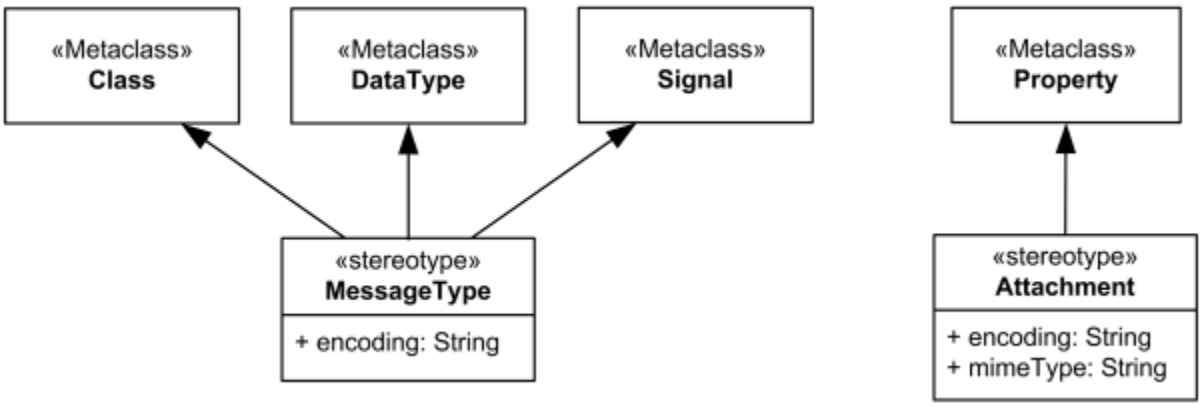
Το Προφίλ έχει τον ίδιο συμβολισμό όπως το Πακέτο, με την προσθήκη της λέξης-κλειδί «προφίλ» πριν ή πάνω από το όνομα του πακέτου. Ένα προφίλ μπορεί να καθορίσει κλάσεις, στερεότυπα, πρωτόγονες μορφές κ.λπ.




**Εικόνα 27 Δύο τρόποι απεικόνισης ενός Προφίλ**

Καθώς το Στερεότυπο είναι μια κλάση, μπορεί να έχει ιδιότητες. Οι ιδιότητες του στερεοτύπου αναφέρονται ως **tagged definitions.** Όταν ένα στερεότυπο εφαρμοστεί σε ένα στοιχείο, οι τιμές αναφέρονται ως **tagged values.** Στο παραπάνω σχήμα οι ιδιότητες Κατασκευαστής, **CPU** και **Memory** είναι **tagged definitions.**

Παρακάτω παρουσιάζεται ένα διάγραμμα Προφίλ για τα δεδομένα υπηρεσιών της **SoaML (Service Oriented Architecture Modeling Language)** που αποτελεί μια προδιαγραφή για **Service oriented** αρχιτεκτονικές.




**Εικόνα 28 Διάγραμμα Προφίλ για τα δεδομένα υπηρεσιών της SoaML**

## *3.2 Διαγράμματα μοντελοποίησης Συμπεριφοράς*

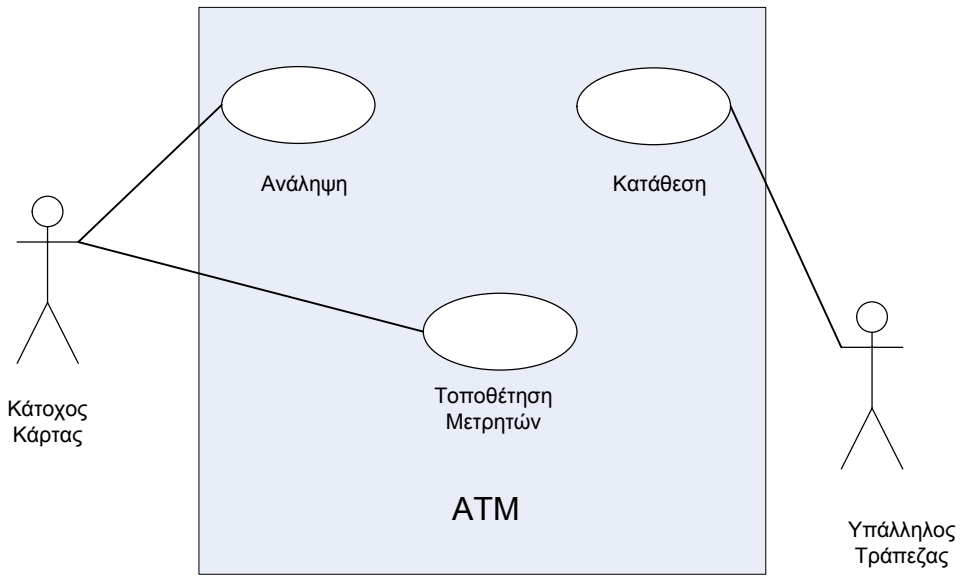
Τα διαγράμματα μοντελοποίησης συμπεριφοράς αναπαριστούν τη δυναμική συμπεριφορά των αντικειμένων σε ένα σύστημα, η οποία μπορεί να περιγραφθεί ως μια σειρά από αλλαγές στο σύστημα κατά τη πάροδο του χρόνου.

## 3.2.1 Περιπτώσεων Χρήσης (Use Case)

Ένα σύστημα θεωρείται ότι επιτελεί το σκοπό του όταν καλύπτει τις προδιαγραφές που έχουν αρχικά τεθεί. Οι προδιαγραφές αυτές ουσιαστικά προκύπτουν ως απαιτήσεις των χρηστών ως προς τη συμπεριφορά του συστήματος στα διάφορα σενάρια λειτουργίας ή χρήσης του. Συνεπώς, ένα διάγραμμα Περιπτώσεων χρήσης παρουσιάζει τη σχέση ανάμεσα στους χρήστες του συστήματος **(actors)** και τις περιπτώσεις χρήσης του, δίνοντας έμφαση στη λειτουργικότητά του όπως αυτή γίνεται αντιληπτή από τους χρήστες του.

Γενικά, η Περίπτωση Χρήσης **(use case)** είναι ένα σύνολο σεναρίων για την επίτευξη των σκοπών του χρήστη. Ως σενάριο νοείται μια ακολουθία ενεργειών που περιγράφουν την αλληλεπίδραση μεταξύ χρήστη και συστήματος, η οποία θα καταλέιξει είτε σε επιτυχία είτε σε αποτυχία. Τα διαγράμματα Περιπτώσεων Χρήσης εκτός από την αναπαράσταση των εξωτερικών **actors (**χρήστες, άλλη εφαρμογή, άλλο σύστημα**)** και την σχέση τους με το σύστημα στη διάρκεια των σεναρίων, ουσιαστικά χρησιμοποιούνται για τον προσδιορισμό των απαιτήσεων.

Ως **Actor** στις περιπτώσεις χρήσης, νοείται μια οντότητα που ορίζει ένα ρόλο που έχει ο χρήσης κατά την αλληλεπίδραση του με το σύστημα. **Actor** όπως προαναφέρθηκε μπορεί να είναι ένας άνθρωπος, μια εξωτερική εφαρμογή που συνεργάζεται με το υπό μοντελοποίηση σύστημα, ή ένα άλλο σύστημα. Ακολουθεί ένα απλοϊκό διάγραμμα Περίπτωσης Χρήσης που αφορά το **ATM** σύστημα μιας τράπεζας.




**Εικόνα 29 Διάγραμμα Περιπτώσεων Χρήσης ATM τράπεζας**

Στο παραπάνω απλό διάγραμμα, το σύστημα που μοντελοποιείται είναι το **ATM** της τράπεζας. Συνεπώς ως εξωτερικός **actor**, εκτός από το κάτοχο κάρτας συναλλαγών, λογίζεται και ο υπάλληλος της τράπεζας. Αν μοντελοποιούσαμε το τραπεζικό κατάστημα,

ο υπάλληλος δεν θα μπορούσε να εμφανίζεται ως **actor,** αφού είναι στοιχείο του συστήματος "τραπεζικό κατάστημα". Η σχέση που έχει σχεδιαστεί μεταξύ **actors** και περιπτώσεων χρήσης είναι μια συσχέτιση και συμβολίζεται με συμπαγή γραμμή.

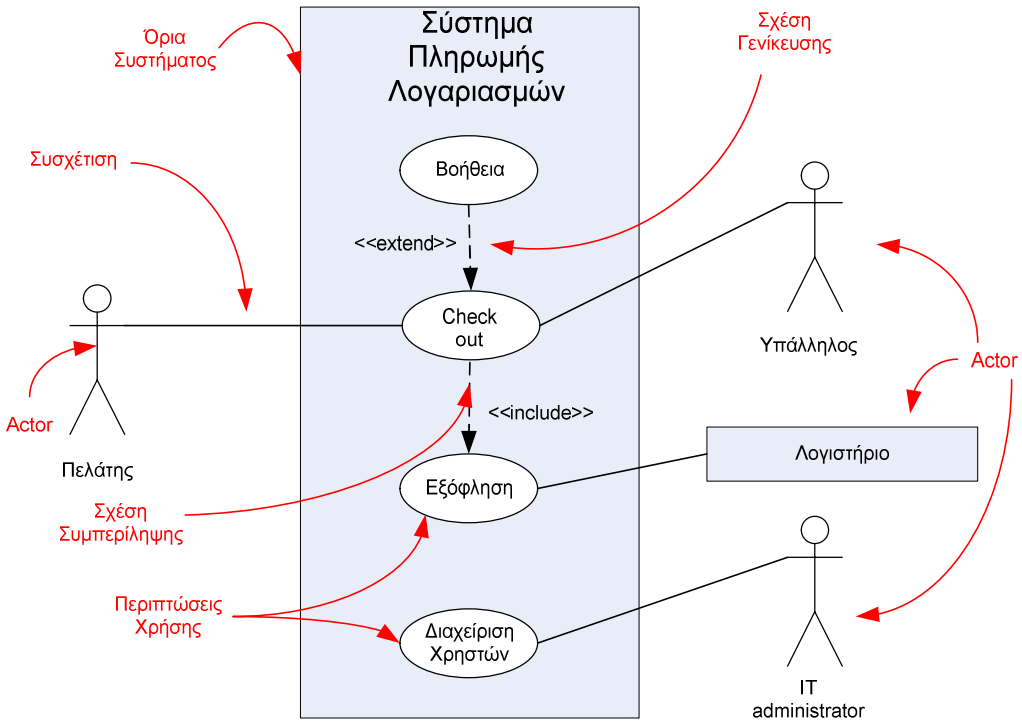
Εκτός από τις Συσχετίσεις ανάμεσα στους **actors** και τις περιπτώσεις χρήσης, υπάρχουν ακόμα οι Γενικεύσεις ανάμεσα σε **actors** και ανάμεσα σε περιπτώσεις χρήσεις, οι Επεκτάσεις **(extends)** και περιλήψεις **(includes)** ανάμεσα σε περιπτώσεις χρήσης. Οι περιπτώσεις χρήσης προαιρετικά μπορούν να περικλείονται σε ένα ορθογώνιο, το οποίο αναπαριστά τα όρια του περιέχοντος συστήματος.

Παρακάτω επεξηγούνται τα διάφορα είδη σχέσεων:

⇒ Συσχέτιση **(association) –** Η συμμετοχή ενός **actor** σε μία περίπτωση χρήσης, δηλαδή τα στιγμιότυπα του **actor** και τα στιγμιότυπα της περίπτωσης χρήσης επικοινωνούν μεταξύ τους. Είναι η μόνη σχέση ανάμεσα σε **actors** και σε περιπτώσεις χρήσης.

⇒ Επέκταση **(extend) –** Μία σχέση επέκτασης από την περίπτωση χρήσης Α στην περίπτωση χρήσης Β δηλώνει ότι ένα στιγμιότυπο της περίπτωσης χρήσης Β μπορεί να αυξηθεί **(**υπό συγκεκριμένες συνθήκες**)** από τη συμπεριφορά που καθορίζεται από την Α. Η συμπεριφορά εισάγεται στη θέση που καθορίζεται από το σημείο επέκτασης στη Β, στο οποίο αναφέρεται από τη σχέση **extend.**

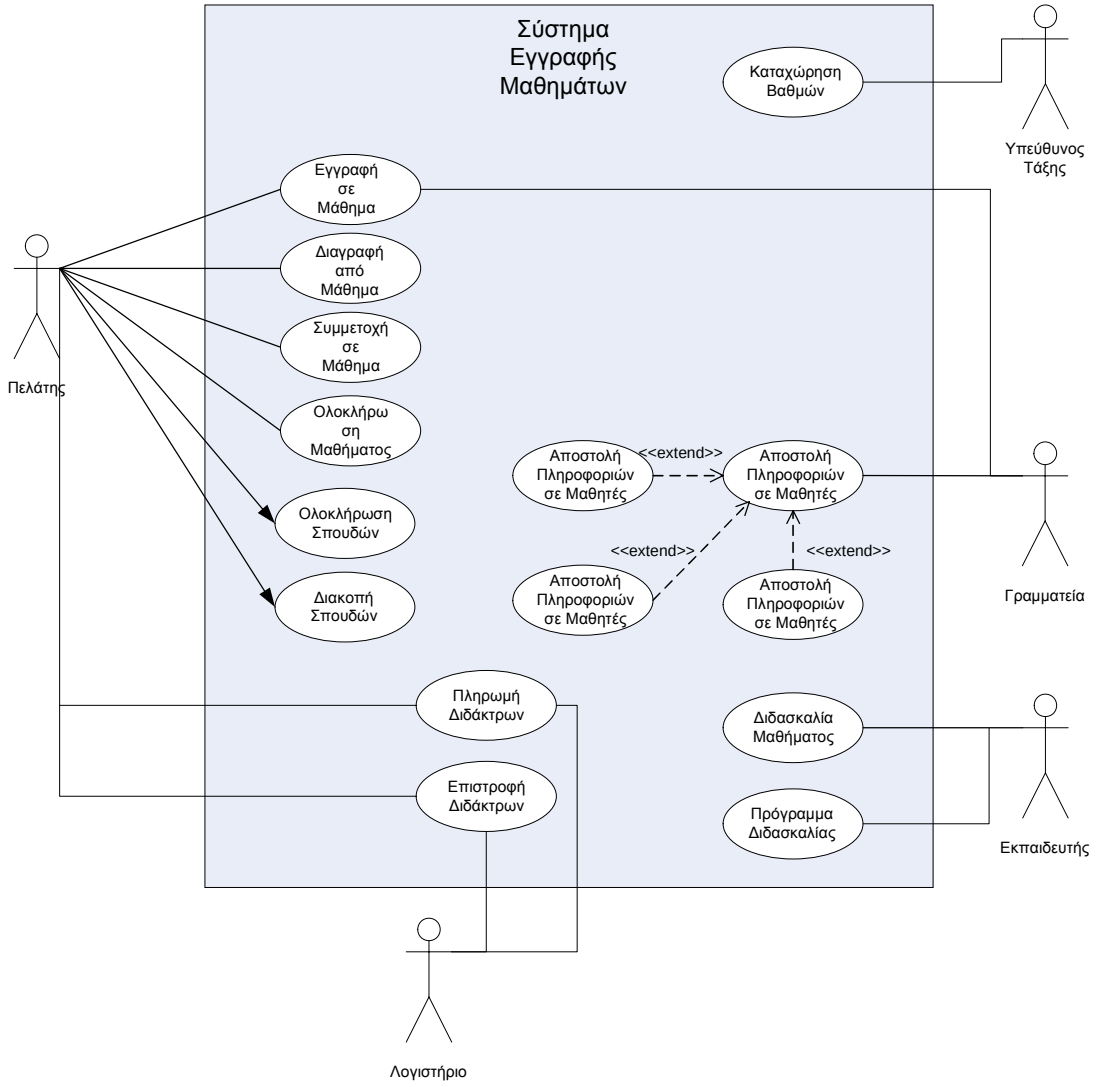
⇒ Συμπερίληψη **(include) –** Μία σχέση συμπερίληψης από την περίπτωση χρήσης Α στην περίπτωση χρήσης Β δηλώνει ότι ένα στιγμιότυπο της περίπτωσης χρήσης Α θα περιέχει επίσης τη συμπεριφορά που καθορίζει η Β. Η συμπεριφορά περιλαμβάνεται στη θέση που καθορίζει η Α.




**Εικόνα 30** **Διάγραμμα Περιπτώσεων Χρήσης με συσχετίσεις, γενικεύσεις, συμπεριλήψεις**

Το παραπάνω διάγραμμα μοντελοποιεί ένα ηλεκτρονικό σύστημα εξόφλησης λογαριασμών. Τα όρια του συστήματος καθορίζονται από το παραλληλόγραμμο. Εντός του συστήματος τοποθετούνται οι διάφορες περιπτώσεις χρήσης, ενώ εξωτερικά οι **actors**. Υπάρχουν **4** περιπτώσεις χρήσης, η Βοήθεια, που είναι το κλασσικό **help** των προγραμμάτων και το οποίο γενικεύεται στη περίπτωση χρήσης **Check out**, η οποία όμως πρέπει να συμπεριλαμβάνει και την Εξόφληση. Επιπλέον υπάρχει και η Περίπτωση Χρήσης Διαχείριση Χρηστών που συσχετίζεται μόνο με τον **IT Administrator** που ουσιαστικά επιβλέπει την ορθή λειτουργία του συστήματος. Οι **actors** έχουν απλές συσχετίσεις με τις περιπτώσεις χρήσης. Έτσι, ο Πελάτης σχετίζεται με τη περίπτωση **Check Out**, όπως και ο υπάλληλος και το Λογιστήριο το οποίο ενημερώνεται για την Εξόφληση. Στις περιπτώσεις που οι συσχετίσεις δεν έχουν βέλος, σημαίνει ότι δεν ξεκάθαρο ποιος εκκινεί την αλληλεπίδραση.

Στο επόμενο παράδειγμα απεικονίζονται οι μαθητές που εγγράφονται στα μαθήματα ενός ιδιωτικού σχολείου με τη βοήθεια της Γραμματείας.




**Εικόνα 31 Διάγραμμα Περιπτώσεων Χρήσης συστήματος εγγραφής μαθημάτων**

Οι Υπεύθυνοι Τάξης καταχωρούν τους βαθμούς που λαμβάνουν οι μαθητές στις εργασίες τους και οι Γραμματεία μοιράζει τα βαθμολόγια στους Μαθητές. Σε ορισμένες περιπτώσεις χρήσης, υπάρχει μόνο ένας **actor** που εμπλέκεται. Επίσης, υπάρχει συσχέτιση μεταξύ του Μαθητή και της περίπτωσης Εγγραφή στο Μάθημα, χωρίς να μεσολαβεί στο διάγραμμα η Γραμματεία. Αυτό δεν σημαίνει ότι όντως δεν μεσολαβεί. Προσοχή, το διάγραμμα δείχνει ότι ο συγκεκριμένος **actor** με κάποιο τρόπο σχετίζεται με τη περίπτωση χρήσης, δεν δείχνει ροή πληροφορίας**!** Επιπλέον, το διάγραμμα παρουσιάζει συνεχείς ανταλλαγές μεταξύ **actors** και περιπτώσεων χρήσης. Έτσι, για παράδειγμα, οι μαθητές θα πρέπει να αναφέρουν σε ποια μαθήματα θέλουν να εγγραφούν και το σύστημα θα πρέπει να δείξει στους μαθητές αν έχουν εγγραφεί. Όμως το διάγραμμα δεν δείχνει αυτή τη πληροφορία, γιατί επίσης σχετίζεται με ροή πληροφορίας, κάτι που δεν υποστηρίζεται από τα διαγράμματα Περιπτώσεων Χρήσης, αλλά από τα διαγράμματα Δραστηριοτήτων που θα τα δούμε αργότερα. Η συσχέτιση μεταξύ της περίπτωσης Εγγραφή σε Μάθημα και της Γραμματείας δεν περιλαμβάνει κάποιο βέλος κατεύθυνσης, κάτι που σημαίνει ότι δεν είναι ξεκάθαρο ποιος εκκινεί την αλληλεπίδραση. Θα μπορούσε ο Μαθητής να ζητήσει βοήθεια, ή η Γραμματεία να προσέφερε βοήθεια.

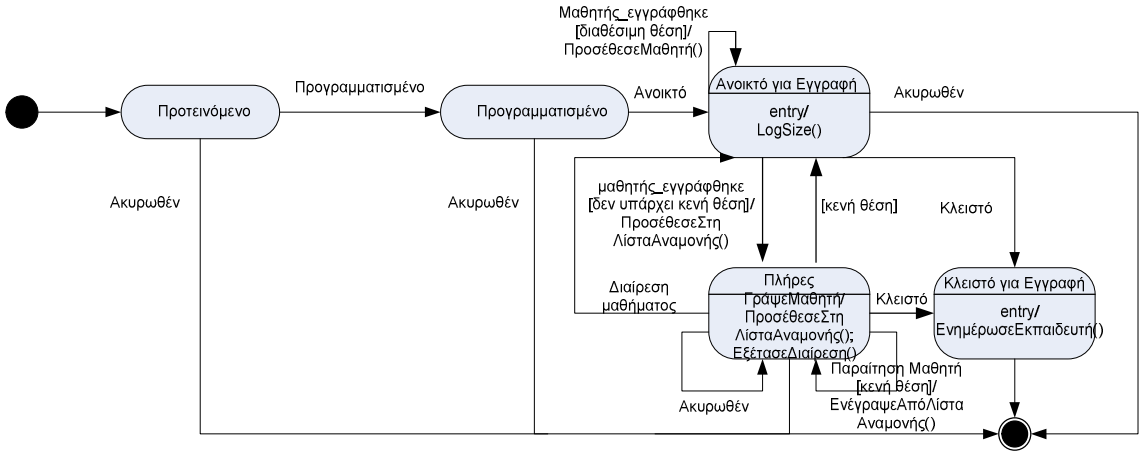
### 3.2.2 Διαγράμματα Καταστάσεων (State Machine)

Τα αντικείμενα έχουν και συμπεριφορά και κατάσταση ή λίγο διαφορετικά "κάνουν" πράγματα και "γνωρίζουν" αντιστοίχως. Υπάρχουν όμως αντικείμενα που είναι τόσο σύνθετα που ακόμα και οι προγραμματιστές δυσκολεύονται να τα κατανοήσουν. Για την κατανόηση τέτοιων σύνθετων αντικειμένων **(**ουσιαστικά των κλάσεων τους**)**, ειδικά αυτών που συμπεριφέρονται με διαφορετικό τρόπο, αναλόγως της κατάστασής τους, έχουν καθιερωθεί τα διαγράμματα Κατάστασης που περιγράφουν πως λειτουργούν τα στιγμιότυπα τους. Συγκεκριμένα, τα διαγράμματα Κατάστασης της **UML 2.0** απεικονίζουν τις διάφορες καταστάσεις στις οποίες μπορεί να βρεθεί ένα αντικείμενο, καθώς και τις μεταβάσεις του αντικειμένου μεταξύ των καταστάσεων αυτών.

Η Κατάσταση αντιπροσωπεύει μια συγκυρία στη μεταβαλλόμενη συμπεριφορά ενός αντικειμένου. Δύναται να υπάρχει αρχική και τελική κατάσταση. Η αρχική κατάσταση είναι αυτή στην οποία βρίσκεται το αντικείμενο όταν δημιουργείται, ενώ η τελική κατάσταση είναι αυτή από την οποία δεν υπάρχει έξοδος. Η μετάβαση είναι η πρόοδος από τη μια κατάσταση στην άλλη και πραγματοποιείται με “πυροδότηση” **(triggering)** από ένα γεγονός που μπορεί να λαμβάνει χώρα εσωτερικά ή εξωτερικά του αντικειμένου.

Στο παρακάτω σχήμα παρουσιάζεται ένα διάγραμμα καταστάσεων για τη κλάση *Μάθημα* κατά την εγγραφή των σπουδαστών. Τα παραλληλόγραμμα με τις στρογγυλεμένες γωνίες είναι καταστάσεις. Το στιγμιότυπο **(**αντικείμενο**)** της κλάσης *Μάθημα* μπορεί να βρεθεί στις καταστάσεις *Προτεινόμενο, Προγραμματισμένο, Ανοικτό*

*για Εγγραφή*, *Πλήρες*, *Κλειστό για Εγγραφή*. Ένα αντικείμενο, ξεκινάει από την αρχική κατάσταση που αναπαριστάται με τον μαύρο κύκλο και καταλήγει στην τελική κατάσταση που αναπαριστάται με τον μαύρο κύκλο σε περίγραμμα. Ο ίδιος συμβολισμό χρησιμοποιείται και στα διαγράμματα Δραστηριοτήτων, όπως θα δούμε και στη συνέχεια.




**Εικόνα 32 Διάγραμμα καταστάσεων για εγγραφή σε μάθημα**

Τα βέλη στο σχήμα αναπαριστούν τις μεταβάσεις από τη μια κατάσταση στην επόμενη. Για παράδειγμα, όταν ένα Μάθημα είναι στη κατάσταση *Προγραμματισμένο* μπορεί είτε να ανοίξει για εγγραφές είτε να ακυρωθεί. Η σήμανση για ετικέτες των μεταβάσεων είναι της μορφής *όνομα_μετάβασης**[συνθήκη]**[/λίστα μεθόδων]***. Είναι υποχρεωτικό να αναφέρεται το γεγονός που προκαλεί τη μετάβαση, όπως εδώ το γεγονός ανοικτό ή ακυρωθέν. Οι συνθήκες πρέπει να ικανοποιούνται ώστε να "πυροδοτηθεί" η μετάβαση, αλλά δεν είναι απαραίτητο να φαίνονται στο διάγραμμα. Η συνθήκη ***[****δεν υπάρχει κενή θέση****]*** εμφανίζεται στη μετάβαση *σπουδαστής_εγγράφθηκε* από τη κατάσταση *Ανοικτό για Εγγραφή* στη κατάσταση *Πλήρες*. Οι συνθήκες μπορούν να περιγράφονται και με άλλους τρόπους όπως απλή γλώσσα ή με τη γλώσσα προγραμματισμού που θα χρησιμοποιηθεί. Επίσης, η επίκληση μεθόδων όπως το *ΠροσέθεσεΣτηΛίσταΑναμονής****()***, δεν είναι υποχρεωτικό να εμφανίζονται στις μεταβάσεις. Πάντως όταν υπάρχει λίστα μεθόδων τότε με αυτό τρόπο υπονοείται και η σειρά επίκλησής τους.

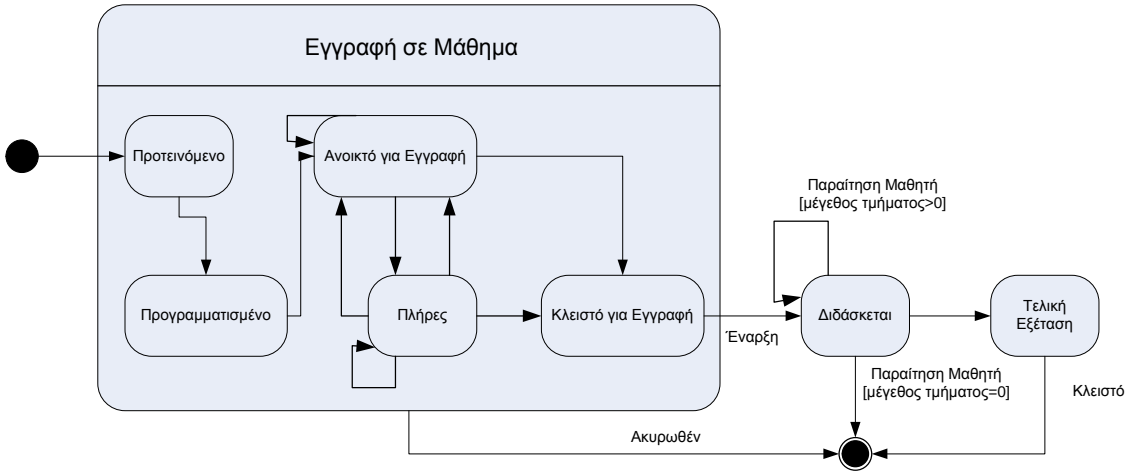
Οι Καταστάσεις εκφράζονται από τις τιμές που παίρνουν οι ιδιότητες του αντικειμένου. Για παράδειγμα, ένα Μάθημα βρίσκεται στην κατάσταση *Ανοικτό για Εγγραφή*, όταν έχει χαρακτηρισθεί ως διαθέσιμο και υπάρχουν θέσεις κενές. Είναι επίσης δυνατό να αναφέρεται η επίκληση μεθόδων εντός της κατάστασης, όπως για παράδειγμα όταν το Μάθημα εισέρχεται στην κατάσταση *Κλειστό για Εγγραφή* που περιλαμβάνει την επίκληση της μεθόδου *ΕνημέρωσεΕκπαιδευτή****()***. Η σήμανση που χρησιμοποιείται είναι όμοια με αυτή των μεταβάσεων, με τη διαφορά ότι η λίστα μεθόδων είναι υποχρεωτική ενώ το γεγονός - συνθήκη που την προκαλεί είναι προαιρετικό. Για παράδειγμα, στη κατάσταση *Πλήρες*, οι μέθοδοι

*ΠροσέθεσεΣτηΛίσταΑναμονής**()*** και *ΕξέτασεΔιαίρεση**()*** καλούνται όποτε ένας μαθητής εγγράφεται σε Μάθημα. Αν δεν υπήρχε το γεγονός - συνθήκη, τότε με βάση το διάγραμμα οι δύο μέθοδοι θα καλούνταν συνεχώς **(loop)** όποτε βρισκόμασταν στη συγκεκριμένη κατάσταση. Οι μέθοδοι που καλούνται με την είσοδο του αντικειμένου σε μια κατάσταση ορίζονται με το τη λέξη κλειδί ***entry***, όπως φαίνεται στις καταστάσεις *Ανοικτό για Εγγραφή* και *Κλειστό για Εγγραφή*. Οι μέθοδοι που καλούνται όταν ένα αντικείμενο βγαίνει από μια κατάσταση ορίζονται με τη λέξη - κλειδί ***exit***.

Οι μεταβάσεις είναι το αποτέλεσμα της επίκλησης μεθόδων που προκαλούν μια σημαντική αλλαγή κατάστασης. Πρέπει όμως να γίνει κατανοητό ότι οι μέθοδοι δεν προκαλούν πάντα μεταβάσεις. Για παράδειγμα, η προσπάθεια να εγγράφει ένας μαθητής σε ένα Μάθημα που είναι πλήρες, δεν οδηγεί σε αλλαγή της κατάστασης, εκτός και αν πρέπει να γίνει διαίρεση σε τμήματα. Είναι φανερό ότι οι μεταβάσεις ακολουθούν τους πραγματικούς επιχειρησιακούς κανόνες, δηλαδή μπορεί κάποιος να επιχειρήσει να εγγράψει ένα μαθητή σε ένα μάθημα όταν είναι *Ανοικτό για Εγγραφή* ή *Πλήρες* και το ένα τμήμα μαθήματος μπορεί να διαιρεθεί σε περισσότερα όταν η λίστα αναμονής είναι αρκετά μεγάλη. Επίσης, υπάρχουν επαναλαμβανόμενες μεταβάσεις που ονομάζονται αυτο-μεταβάσεις καθώς ξεκινούν και καταλήγουν στην ίδια κατάσταση. Ένα τέτοιο παράδειγμα είναι η μετάβαση *ΠαραίτησηΜαθητή* όταν το τμήμα μαθήματος είναι πλήρες.

Ένα άλλο σημείο που πρέπει να ξεκαθαριστεί για λόγους τυπικούς είναι ότι ένα αντικείμενο βρίσκεται σε μια μόνο κατάσταση κάθε φορά, αφού οι μεταβάσεις θεωρούνται στιγμιαίες. Βέβαια πάντα απαιτείται κάποιος χρόνος για την κλήση και τρέξιμο των μεθόδων αλλά τον θεωρούμε μηδενικό για λόγους ευκολίας.

Το πιο πάνω διάγραμμα παρουσιάζει μόνο ένα τμήμα του κύκλου ζωής του Μαθήματος, που είναι πολύ σύνθετος. Το επόμενο διάγραμμα παρουσιάζει το πλήρη κύκλο ζωής του Μαθήματος όπου τα προαναφερθέντα αποτελούν μόνο μια υπο-κατάσταση την Εγγραφή. Επίσης, στο διάγραμμα αυτό οι λεπτομέρειες δεν έχουν συμπεριληφθεί για λόγους απλότητας.

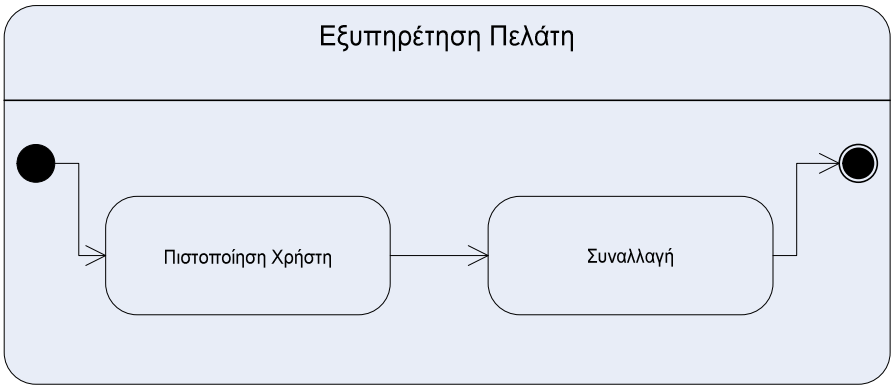



**Εικόνα 33 Διάγραμμα Καταστάσεων πλήρους κύκλου ζωής Μαθήματος**

Μέχρι αυτό το σημείο παρουσιάστηκαν Απλές Καταστάσεις. Αυτές ουσιαστικά όπως ειπώθηκε εκφράζουν μια συγκυρία κατά τη διάρκεια της ζωής ενός αντικειμένου όπου απλά αναμένεται κάτι να συμβεί. Αν δεν συμβεί κάτι, η κατάσταση δεν αλλάζει.

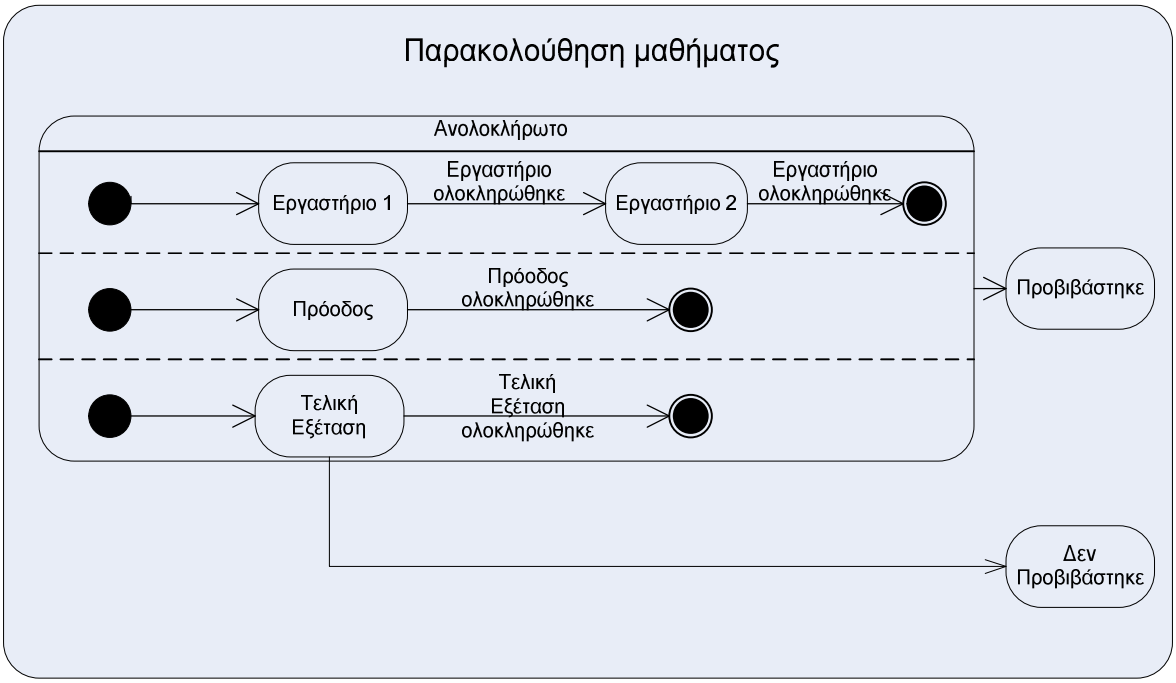
Υπάρχουν όμως και οι Σύνθετες Καταστάσεις που "αποσυντίθενται" σε δύο ή περισσότερες υποκαταστάσεις (οι οποίες ονομάζονται περιοχές), που μπορεί να είναι ακολουθιακές ή συντρέχουσες.

Παρακάτω παρουσιάζεται μια Σύνθετη Κατάσταση με δύο ακολουθιακές υποκαταστάσεις.



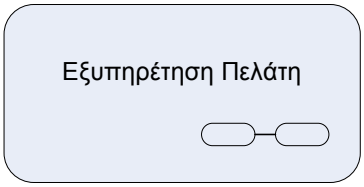

**Εικόνα 34 Σύνθετη Κατάσταση με δυο ακολουθιακές υποκαταστάσεις**

Στη περίπτωση Σύνθετης Κατάστασης με συντρέχουσες καταστάσεις, υπάρχουν δύο ή περισσότεροι "διάδρομοι" (τους ονομάζουν **swimlanes)** που εκπροσωπούν τις συντρέχουσες υποκαταστάσεις.



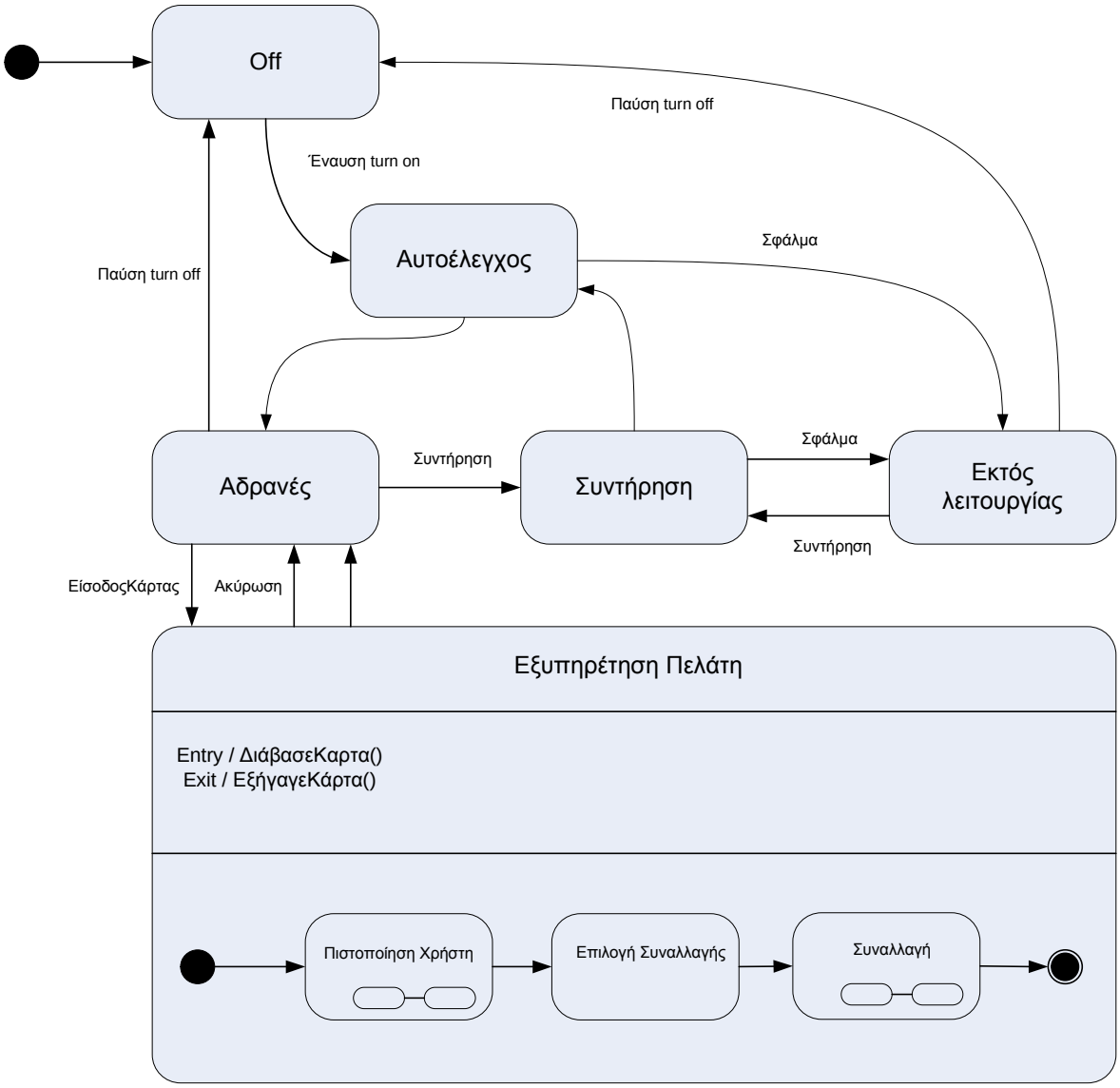

**Εικόνα 35 Σύνθετη Κατάσταση με συντρέχουσες υποκαταστάσεις**

Πολλές φορές για λόγους απλότητας θεωρείται σκόπιμο να αποκρύπτεται η "αποσύνθεση" σε υποκαταστάσεις, οπότε εναλλακτικά χρησιμοποιείται ο παρακάτω συμβολισμός.




**Εικόνα 36 Σύνθετη Κατάσταση με απόκρυψη υποκαταστάσεων**

Ακολουθεί το διάγραμμα καταστάσεων ενός τραπεζικού **ATM.** Το διάγραμμα συμπεριλαμβάνει Σύνθετες Καταστάσεις.




**Εικόνα 37 Διάγραμμα Καταστάσεων τραπεζικού ATM**

Το **ATM** αρχικά είναι σβηστό. Αφού γίνει η έναυση του, το ATM εισέρχεται σε κατάσταση *Αυτοελέγχου*. Αν ο έλεγχος αποτύχει τότε υπάρχει Σφάλμα και εισέρχεται σε κατάσταση *Εκτός Λειτουργίας*, διαφορετικά αν ο έλεγχος ολοκληρωθεί τότε το ATM χωρίς "σκανδαλισμό" εισέρχεται στη κατάσταση *Αδράνειας*. Σε αυτή τη κατάσταση το ATM αναμένει κάποια αλληλεπίδραση με το χρήστη.

Η κατάσταση του ATM αλλάζει από *Αδράνεια* σε *Εξυπηρέτηση Πελάτη* όταν ο τελευταίος εισαγάγει την κάρτα του στο μηχάνημα. Με την είσοδο στη νέα κατάσταση, το ATM εκτελεί την **entry** μέθοδο *ΔιάβασεΚάρτα()*. Η μετάβαση *Ακύρωση* που υπάρχει μεταξύ της κατάστασης *Εξυπηρέτηση Πελάτη* και *Αδράνειας*, περιγράφει το γεγονός ότι πελάτης μπορεί να ακυρώσει τη διαδικασία ανά πάσα στιγμή.

Η κατάσταση *Εξυπηρέτηση Πελάτη* είναι μια σύνθετη κατάσταση με ακολουθιακές υποκαταστάσεις *Πιστοποίηση Χρήστη, Επιλογή Συναλλαγής, Συναλλαγή*. Η *Πιστοποίηση Χρήστη* και η *Συναλλαγή* είναι και αυτές σύνθετες καταστάσεις όπως φαίνεται και από το σχετικό εικονίδιο κάτω δεξιά. Η μετάβαση από τη κατάσταση *Εξυπηρέτηση Πελάτη* στη κατάσταση *Αδράνεια*, πραγματοποιείται χωρίς σκανδαλισμό, μόλις ολοκληρωθεί η συναλλαγή. Πριν την έξοδο από την κατάσταση καλείται η μέθοδος *ΕξαγωγήΚάρτας()*, που την αφαιρεί από το μηχάνημα.

### 3.2.3 Διαγράμματα Δραστηριοτήτων (Activity)

Τα διαγράμματα δραστηριοτήτων είναι διαγράμματα συμπεριφοράς που φανερώνουν τη ροή ελέγχου ή τη ροή αντικειμένων, με έμφαση στην αλληλουχία και τις συνθήκες της ροής αυτής. Οι δράσεις που εμφανίζονται στο μοντέλο δραστηριοτήτων, μπορούν να εκκινούν όταν άλλες δράσεις ολοκληρώνονται, όταν νέα αντικείμενα είναι διαθέσιμα, είτε όταν συμβαίνουν ορισμένα εξωτερικά γεγονότα.

Τα διαγράμματα δραστηριοτήτων της **UML 2.0**, συνήθως χρησιμοποιούνται για μοντελοποίηση επιχειρηματικών δραστηριοτήτων, για μοντελοποίηση της λογικής ενός σεναρίου χρήσης ή για τη μοντελοποίηση της λογικής ενός επιχειρηματικού κανόνα. Παρόλο που με τα διαγράμματα δραστηριοτήτων θα μπορούσαν να αναπαρασταθούν σύνθετες λειτουργίες, εντούτοις είναι προτιμότερο να ξαναγραφτεί το πρόγραμμα της λειτουργίας, ώστε να είναι απλή και να μην απαιτείται διάγραμμα δραστηριοτήτων για να μοντελοποιηθεί. Γενικά τα διαγράμματα δραστηριοτήτων είναι για τον αντικειμενοστραφή προγραμματισμό, το ισοδύναμο των διαγραμμάτων ροής του δομημένου προγραμματισμού.

Η βασική σήμανση των διαγραμμάτων δραστηριοτήτων που περιλαμβάνεται και στα δυο παρακάτω παραδείγματα είναι η εξής:

Αρχικός κόμβος. Ο μαύρος κύκλος είναι η αρχή του διαγράμματος. Δεν απαιτείται αλλά διευκολύνει την ανάγνωση του.

Τελικός κόμβος. Ο μαύρος κύκλος με περίγραμμα. Ένα διάγραμμα μπορεί να μην έχει μηδέν ή περισσότερους τελικούς κόμβους.

Δραστηριότητα. Τα παραλληλόγραμμα με τις στρογγυλεμένες γωνίες αναπαριστούν δραστηριότητες που λαμβάνουν χώρα. Μια δραστηριότητα μπορεί να είναι πραγματική όπως η *Εξέτασε Φόρμες* ή θα μπορούσε να είναι και ψηφιακή π.χ. εξέτασε βάση δεδομένων.

Ροή. Είναι τα βέλη του διαγράμματος που δείχνουν τη ροή των διαδικασιών

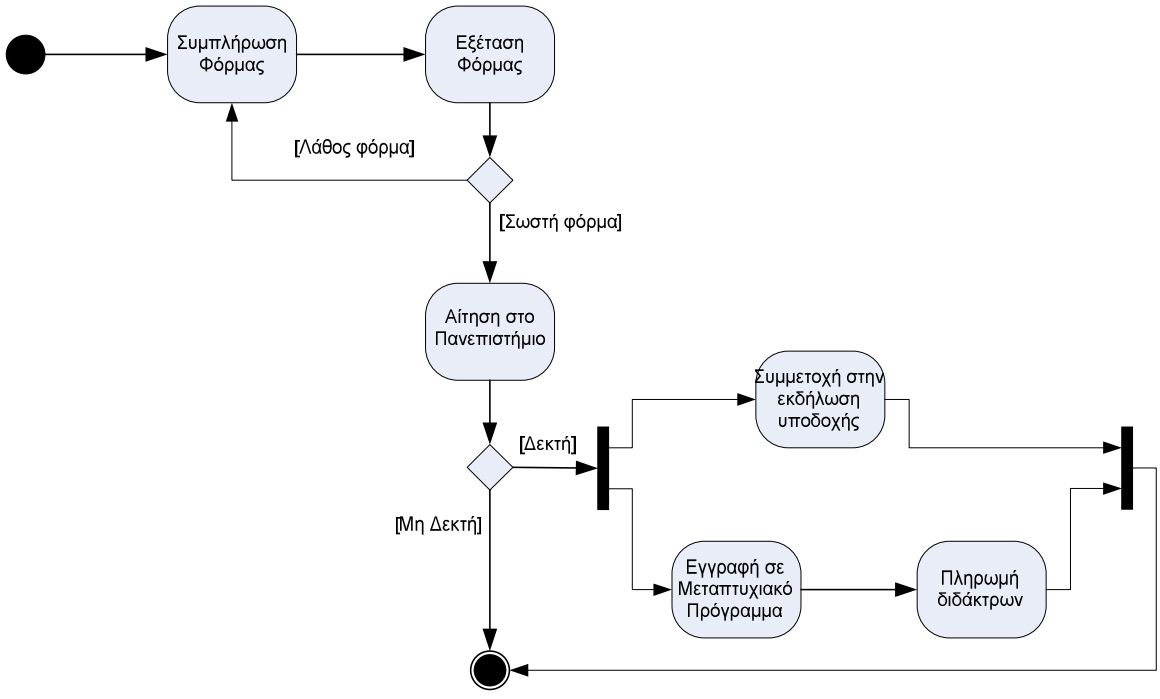
Διαχωριστής. Συμβολίζεται με μαύρη μπάρα στην οποία εισέρχεται μια ροή και εξέρχονται πολλές. Ουσιαστικά εκφράζει την παράλληλη επεξεργασία.

Συνδετήρας. Συμβολίζεται με μαύρη μπάρα στην οποία εισέρχονται πολλές ροές και εξέρχεται μια. Ουσιαστικά εκφράζει το τέλος της παράλληλης επεξεργασίας.

Συνθήκη. Κείμενο της μορφής **[**Λάθος Φόρμα**]** σε μια ροή, που αποτελεί συνθήκη για να μπορέσει να ισχύσει η ροή.

Απόφαση. Ένας ρόμβος στον οποίο εισέρχεται μια ροή και εξέρχονται πολλές οι οποίες συνοδεύονται από τις αντίστοιχες συνθήκες που πρέπει να ικανοποιούνται ώστε να ισχύσει η ροή.

Σύνθεση. Ένας ρόμβος στον οποίο εισέρχονται πολλές ροές και εξέρχεται μια. Το νόημα είναι ότι μία ή περισσότερες εισερχόμενες ροές πρέπει να φθάσουν σε αυτό το σημείο για την συνέχεια της επεξεργασίας, με βάση τις τυχόν συνθήκες στην εξερχόμενη ροή.




**Εικόνα 38 Διάγραμμα Καταστάσεων για εγγραφή σε μεταπτυχιακό πρόγραμμα**

Παρακάτω παρουσιάζεται ένα διάγραμμα δραστηριοτήτων που ακολουθεί κάπως διαφορετική προσέγγιση. Χρησιμοποιεί την έννοια των κατατμήσεων.

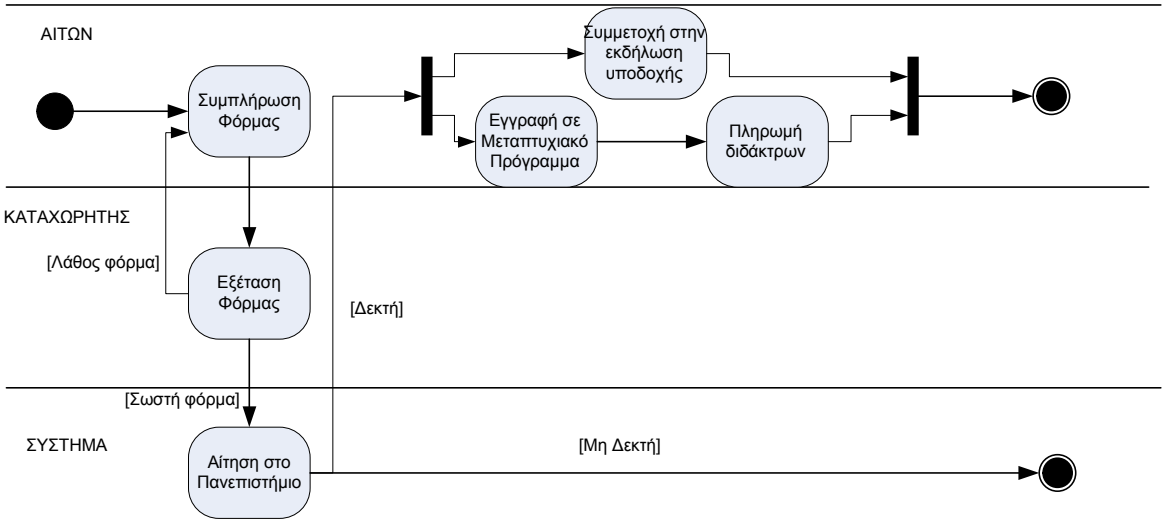
Κατατμήσεις **(partitions),** που αλλιώς ονομάζονται **swimlanes** λόγω της εμφάνισης τους, και δείχνουν ποιος */*τι ασκεί την δραστηριότητα.

Σε αυτό το διάγραμμα υπάρχουν **3 swimlanes,** ένα για τον Αιτών, ένα για τον Καταχωρητή και ένα για το Σύστημα, που ουσιαστικά είναι και οι **actors** του διαγράμματος. Τα **swimlanes** είναι χρήσιμα, επειδή παρέχουν περισσότερες πληροφορίες, αλλά επιμηκύνουν και το διάγραμμα.

Εκτός από τις Κατατμήσεις στο παρακάτω διάγραμμα υπάρχουν επιπλέον:

Σύμβολο υπο-δραστηριότητας. Το σύμβολο στη κάτω γωνία μιας δραστηριότητας, όπως στην *Αίτηση στο Πανεπιστήμιο*, σημαίνει ότι η δραστηριότητα περιγράφεται πιο λεπτομερώς σε άλλο διάγραμμα δραστηριοτήτων.

Τελικό σημείο. Ο κύκλος με το Χ στο εσωτερικό του, που υποδηλώνει ότι η διαδικασία σταματάει σε αυτό το σημείο

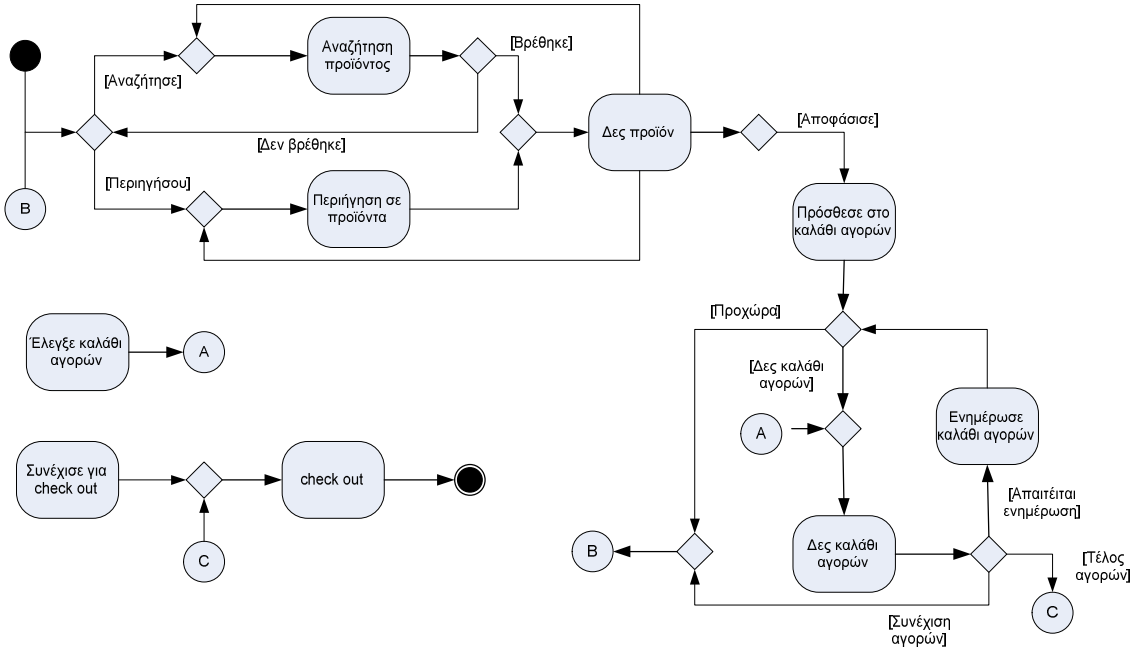



**Εικόνα 39 Διάγραμμα Καταστάσεων με swimlanes για εγγραφή σε μεταπτυχιακό πρόγραμμα**

Στο επόμενο παράδειγμα παρουσιάζεται το διάγραμμα καταστάσεων για **Online** αγορές. Ο **online** πελάτης μπορεί να περιηγηθεί και να αναζητήσει το προϊόν που τον ενδιαφέρει και κατόπιν να το τοποθετήσει στο καλάθι αγορών. Επίσης μπορεί να προβάλει και να ενημερώσει το καλάθι αγορών, να ολοκληρώσει την παραγγελία και να κάνει **check out.** Αυτό το παράδειγμα δεν χρησιμοποιεί κατατμήσεις, καθώς η θεώρηση γίνεται από τη πλευρά του πελάτη και δεν φαίνεται άμεσα η εμπλοκή του συστήματος.

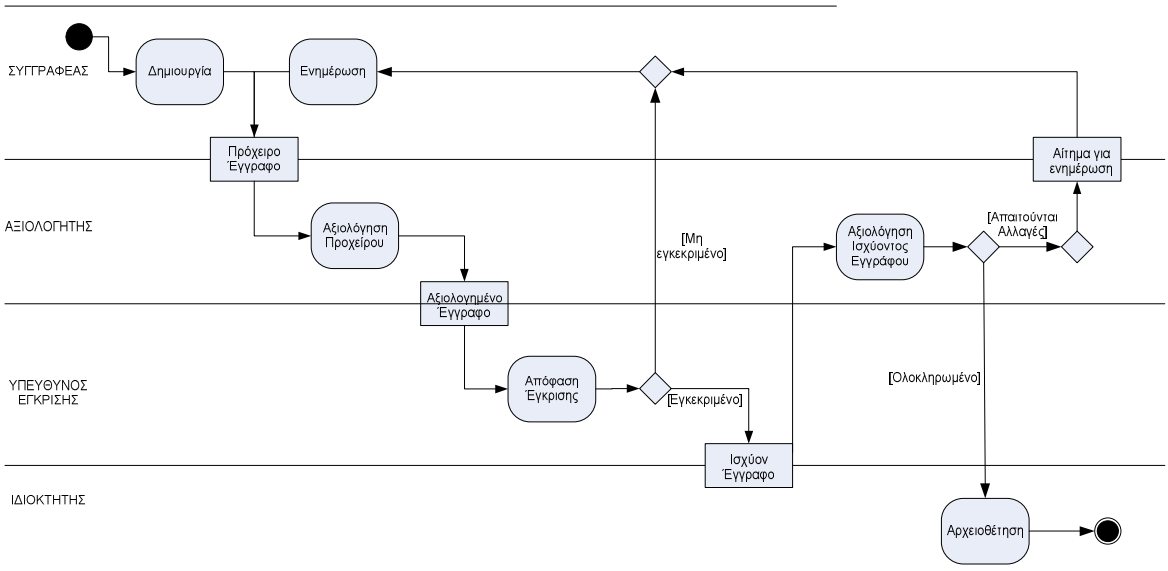
Το νέο στοιχείο στο διάγραμμα είναι η χρήση των **connectors (**κύκλος με γράμμα στο εσωτερικό**)** που χρησιμοποιούνται για την αποφυγή μεγάλων διαγραμμάτων. Όταν

φτάνουμε σε **connector** με κάποιο γράμμα**,** τότε η ροή συνεχίζεται στον δεύτερο **connector** με το ίδιο γράμμα**.**




**Εικόνα 40 Διάγραμμα Καταστάσεων για Online αγορές**

Τέλος ακολουθεί ένα παράδειγμα για διαχείριση εγγράφων**.** Το διάγραμμα ενσωματώνει **swimlanes** που διαχωρίζουν τους διαφορετικούς ρόλους της διαδικασίας**.** Αυτοί είναι Συγγραφέας**,** Αξιολογητής**,** Υπεύθυνος έγκρισης**,** και Ιδιοκτήτης**.**




**Εικόνα 41 Διάγραμμα Καταστάσεων για διαχείριση εγγράφων**

Όπως φαίνεται και από το προηγούμενο διάγραμμα, ένα έγγραφο δημιουργείται, αξιολογείται, ενημερώνεται αν είναι αναγκαίο και εγκρίνεται, και σε κάποιο σημείο αρχειοθετείται.

### 3.2.4 Αλληλεπίδρασης (Interaction)

Τα διαγράμματα Αλληλεπίδρασης **(interaction diagrams)** χρησιμοποιούνται για την οπτική αναπαράσταση των διαφόρων σεναρίων των περιπτώσεων χρήσης του συστήματος. Απεικονίζουν ένα σενάριο σαν στιγμιότυπα αντικειμένων που αλληλεπιδρούν μεταξύ τους ανταλλάσσοντας μηνύματα. Υπάρχουν τέσσερις τύποι διαγραμμάτων αλληλεπίδρασης, οι οποίοι είναι συμπληρωματικοί Το διάγραμμα ακολουθίας **(sequence diagram)**, το διάγραμμα επικοινωνίας **(communication diagram)**, το διάγραμμα χρονισμού **(timing diagram)** και το διάγραμμα επισκόπησης αλληλεπίδρασης **(interaction collaboration diagram)**. Το διάγραμμα ακολουθίας δίνει έμφαση στη χρονική ακολουθία των μηνυμάτων, όπως και το διάγραμμα επικοινωνίας, με τη διαφορά ότι το διάγραμμα επικοινωνίας δίνει μια πιο γενική εικόνα ων ανταλλασόμενων μηνυμάτων. Επίσης το διάγραμμα επικοινωνίας απαιτεί λιγότερο σχεδιαστικό χώρο και χρόνο, μιας και η διάταξη των αντικειμένων είναι ελεύθερη στο χώρο και συνεπώς προτιμάται όταν υπάρχουν πολλά αντικείμενα. Από την άλλη όμως επειδή το διάγραμμα ακολουθίας προσφέρει καλύτερη απεικόνιση της χρονικής αλληλουχίας ων μηνυμάτων, οι περισσότεροι σχεδιαστές τα προτιμούν.

Το διάγραμμα χρονισμού χρησιμοποιείται για την επισκόπηση αντικειμένων κατά τη πάροδο του χρόνου. Μοιάζουν με τα διαγράμματα καταστάσεων αλλά τα δεύτερα εστιάζουν περισσότερο στις λεπτομερείς μεταβάσεις ανάμεσα στις καταστάσεις λόγω των συμβάντων που λαμβάνουν χώρα στο σύστημα. Τα διαγράμματα επισκόπησης αλληλεπίδρασης στη πράξη δεν χρησιμοποιούνται ιδιαίτερα καθώς απαιτούν σημαντικό σχεδιαστικό χώρο και επιπλέον τείνουν να εγκαταλειφθούν για χάρη των διαγραμμάτων δραστηριότητας.

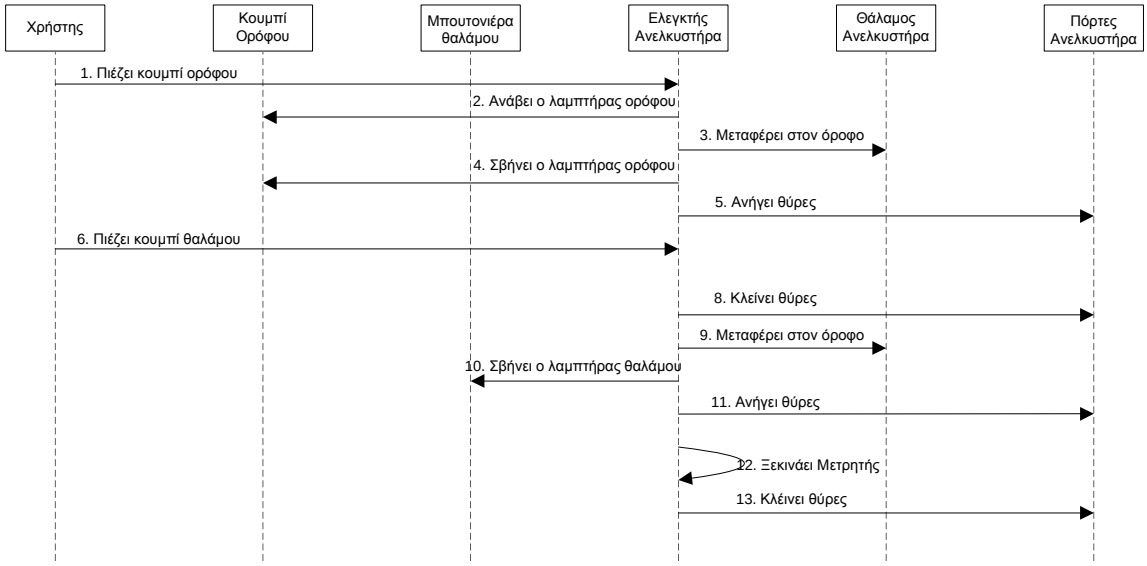
#### 3.2.4.1 Αλληλουχίας (Sequence)

Τα διαγράμματα Ακολουθίας της **UML**, μοντελοποιούν με ένα εποπτικό τρόπο, τη ροή της λογικής εντός του συστήματος που μοντελοποιείται, επιτρέποντας έτσι τόσο τη τεκμηρίωση όσο και την επικύρωση της εφαρμοζόμενης λογικής. Τα διαγράμματα ακολουθίας είναι τα πιο γνωστά διαγράμματα αλληλεπίδρασης και από τα πιο γνωστά της **UML**, και εστιάζουν στη συμπεριφορά του συστήματος.

Τα διαγράμματα αυτά χρησιμοποιούνται για να μοντελοποιήσουν κυρίως:

1. Σενάρια χρήσης. Ένα σενάριο χρήσης είναι μια περιγραφή της χρήσης του συστήματος. Η λογική ενός σεναρίου χρήσης μπορεί να είναι τμήμα μιας γενικότερης Περίπτωσης χρήσης ή και συνδυασμός πολλών τμημάτων Περιπτώσεων Χρήσης. Για παράδειγμα ένα σενάριο χρήσης είναι η εγγραφή ενός φοιτητή σε ένα Πανεπιστήμιο και ακολούθως σε **5** μαθήματα εξαμήνου. Εδώ έχουμε ουσιαστικά δύο Περιπτώσεις χρήσης που εμπεριέχονται στο σενάριο χρήσης.
2. Τη λογική των μεθόδων. Τα διαγράμματα ακολουθίας μπορούν να χρησιμοποιηθούν για να διερευνηθεί η λογική μιας σύνθετης λειτουργίας ή διαδικασίας.
3. Τη λογική των υπηρεσιών. Μια υπηρεσία είναι ουσιαστικά μια μέθοδος υψηλού επιπέδου που μπορεί να κληθεί από μια σειρά από **clients**. Οι υπηρεσίες μπορεί να είναι τόσο **web services** όσο και επιχειρηματικές διαδικασίες.

Παρακάτω παρουσιάζεται ένα παράδειγμα διαγράμματος ακολουθίας για τη λειτουργία ενός συστήματος ανελκυστήρα. Τα βήματα της ακολουθίας έχουν αριθμηθεί **(**κάτι που δεν ισχύει στη σήμανση της **UML)** ώστε να φανεί η χρησιμότητα του διαγράμματος.



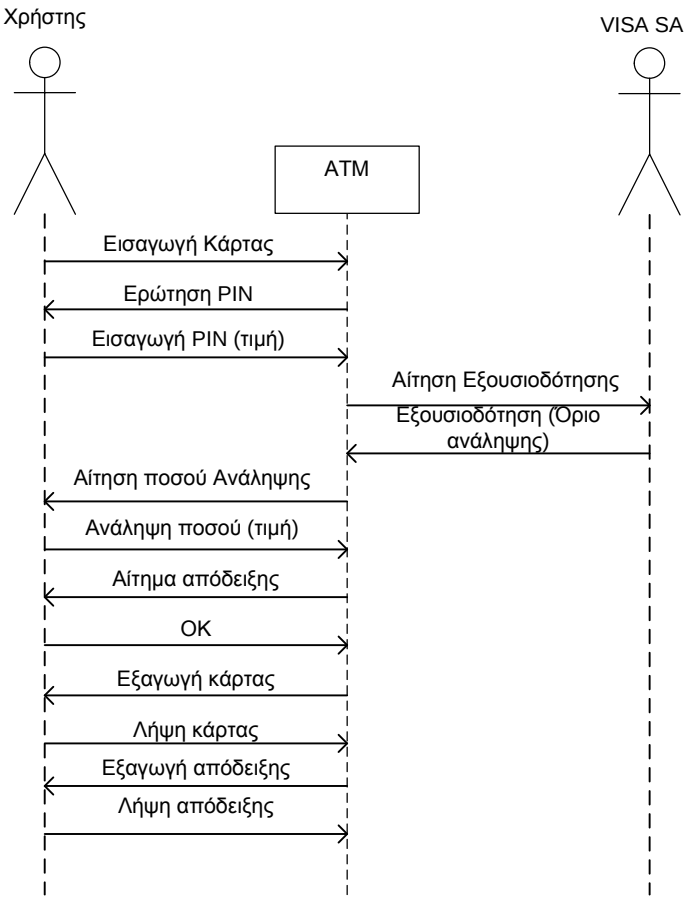

**Εικόνα 42 Παράδειγμα χρήσης διαγράμματος Αλληλουχίας**

Αυτό που μας δείχνει το παραπάνω διάγραμμα είναι η αλληλουχία των δράσεων. Ο χρήστης πιέζει το κουμπί κλήσης του ανελκυστήρα. Η κίνηση αυτή ενεργοποιεί τον ελεγκτή του ανελκυστήρα που δίνει εντολή να ανάψει ο λαμπτήρας του κουμπιού

κλήσης στον όροφο που βρίσκεται ο χρήστης και επιπλέον δίνε την εντολή στο σύστημα ανύψωσης του θαλάμου, να τον μετακινήσει στο όροφο του χρήστη. Όταν φτάσει στον όροφο του χρήστη, σβήνει ο λαμπτήρας του κουμπιού κλήσης, ο ελεγκτής στέλνει εντολή για να ανοίξουν οι πόρτες του ορόφου.

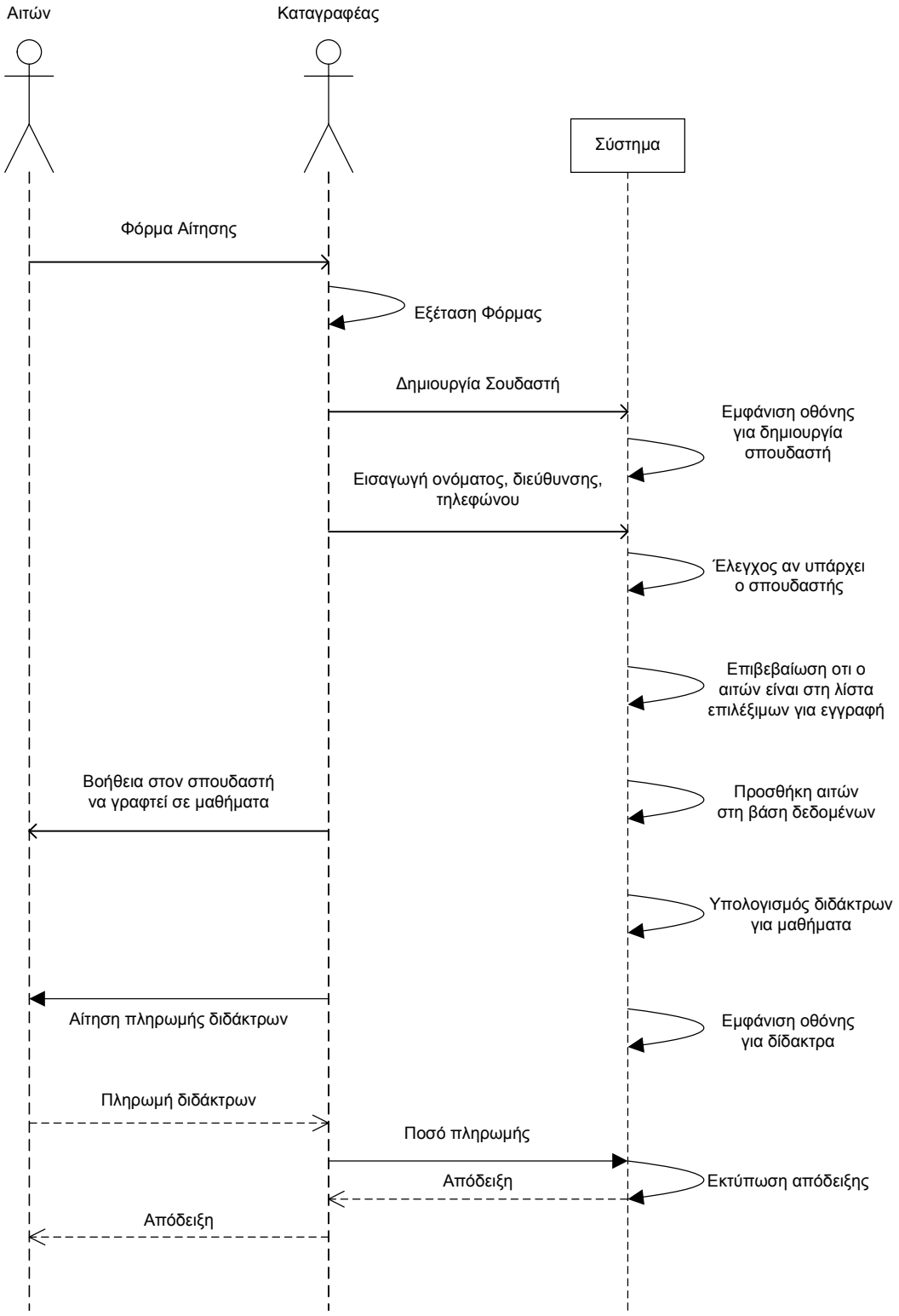
Ο χρήστης εισέρχεται και πιέζει το κουμπί του ορόφου που θέλει να πάει, οπότε ο ελεγκτής δίνει εντολή να ανάψει το σχετικό λαμπτήρα στη μπουτονιέρα εντός του θαλάμου, να κλείσουν οι πόρτες του ανελκυστήρα και να μετακινηθεί ο θάλαμος στον ζητούμενο όροφο. Με τη άφιξη του θαλάμου, σβήνει ο λαμπτήρας της μπουτονιέρας και ανοίγουν οι πόρτες του ορόφου. Τότε ξεκινά ένας χρονικός μετρητής με το πέρας του οποίου ξανακλείνουν οι πόρτες. Πλέον απαιτείται ξανά κάποια ενέργεια από το χρήστη, είτε χρήστη εντός του θαλάμου είτε εκτός σε κάποιο όροφο για να συνεχίσει με τον ίδιο τρόπο η ακολουθία.

Ακολουθεί ένα παράδειγμα για τη χρήση τραπεζικού μηχανήματος **ATM**.




**Εικόνα 43 Διάγραμμα Αλληλουχίας για μηχάνημα ATM**

Ακολουθεί ένα άλλο παράδειγμα, με ορθή χρήση της **UML** σήμανσης, σχετικά με τη Περίπτωση χρήσης εγγραφής σε ένα Μάθημα.




**Εικόνα 44 Διάγραμμα Αλληλουχίας για τη διαδικασία εγγραφής σε μάθημα**

Ο λόγος για τον οποίο τα διαγράμματα αυτά ονομάζονται διαγράμματα ακολουθίας είναι προφανής: η διαδοχική φύση της λογικής εμφανίζεται μέσω των μηνυμάτων

παραγγελίας **(order messages)** που συμβολίζονται με τα οριζόντια βέλη. Το πρώτο μήνυμα ξεκινά από την πάνω αριστερή γωνία, το επόμενο μήνυμα εμφανίζεται ακριβώς από κάτω κ.λ.π.

Το οριζόντιο βέλος με τη συνεχή γραμμή και συμπαγές μαύρο πέρας, ονομάζεται Σύγχρονη κλήση και αντιπροσωπεύει την αποστολή μηνύματος, αναστέλλοντας παράλληλα την ακολουθία μέχρι να έρθει απάντηση στο μήνυμα.

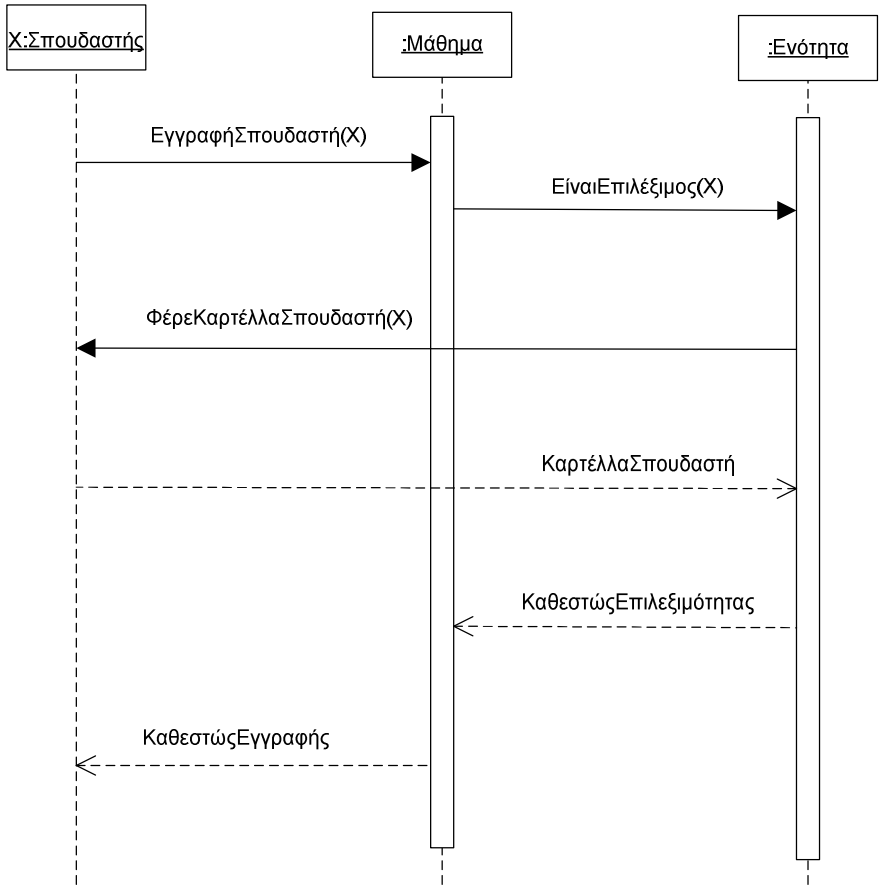
Το οριζόντιο βέλος με συνεχή γραμμή αλλά ανοικτό πέρας, ονομάζεται Ασύγχρονη κλήση και αντιπροσωπεύει την αποστολή μηνύματος με την παράλληλη συνέχιση της ακολουθίας χωρίς αν πρέπει πρώτα να επιστρέψει η απάντηση στο μήνυμα.

Το οριζόντιο βέλος με τη διακεκομμένη γραμμή και το ανοικτό πέρας, που έπεται μιας σύγχρονης/ασύγχρονης κλήσης, αντιπροσωπεύει ένα μήνυμα απάντηση στην σύγχρονη/ασύγχρονη κλήση.

Οι κάθετες διακεκομμένες γραμμές ονομάζονται **Lifelines** και είναι στοιχεία που αντιπροσωπεύουν τους συμμετέχοντες στην αλληλεπίδραση που παρουσιάζεται. Ένα **Lifeline** εμφανίζεται με ένα σύμβολο που αποτελείται από ένα ορθογώνιο **(**ή άλλο σύμβολο αναλόγως το στοιχείο**)** που αποτελεί **"**τη κεφαλή**"** ακολουθούμενο από μια κάθετη γραμμή, που αντιπροσωπεύει τη διάρκεια ζωής του στοιχείου κεφαλής. Στο πιο πάνω παράδειγμα υπάρχουν δύο **actors** και ένα σύστημα. Γενικά, η κεφαλή μπορεί να αναφέρεται σε Περίπτωση Χρήσης, Κλάση Αντικείμενο ή **Actor**.

Όταν στην κεφαλή υπάρχει αντικείμενο, τότε χρησιμοποιείται για ετικέτα ο κλασσικός τρόπος ονομασίας *όνομα:Κλάση* **(**με το όνομα να είναι προαιρετικό**)**, ενώ οι κλάσεις εμφανίζονται με το *ΚλάσηΌνομα* και οι δράστες με *ΔράστηςΟνομα*. Επιπλέον, οι ετικέτες των αντικειμένων εμφανίζονται υπογραμμισμένες, ενώ των κλάσεων και των **actors** όχι.

Ακολουθεί το διάγραμμα ακολουθίας που παρουσιάζει τη λογική της εγγραφής ενός σπουδαστή σε ένα μάθημα.




**Εικόνα 45 Διάγραμμα Αλληλουχίας για εγγραφή σε μάθημα θεματικής ενότητας**

Στο παραπάνω διάγραμμα το αντικείμενο της Κλάσης Σπουδαστής είναι το Χ, ενώ το αντικείμενο της κλάσης Μάθημα είναι ανώνυμο. Αυτό συμβαίνει γιατί ατη πρώτη περίπτωση το αντικείμενο του Σπουδαστή είναι συχνά χρησιμοποιούμενο ως παράμετρος των μηνυμάτων, ενώ κάτι τέτοιο δεν ισχύει για το αντικείμενο της κλάσης Μάθημα.

Επίσης στο συγκεκριμένο διάγραμμα περιλαμβάνονται και "κουτιά ενεργοποίησης", τα κάθετα παραλληλόγραμμα που αντικαθιστούν τις **Lifelines** στις περιπτώσεις των αντικειμένων Μάθημα και Ενότητα. Εναλλακτικά ονομάζονται και κουτιά επίκλησης μεθόδων και σημαίνουν ότι κάποια επεξεργασία λαμβάνει χώρα από το αντικείμενο/κλάση ώστε να εκπληρωθεί ο σκοπός ενός μηνύματος. Πολλές, φορές στο κάτω μέρος ενός κουτιού ενεργοποίησης υπάρχει ένα Χ, που σημαίνει ότι από αυτό το σημείο και έπειτα, το αντικείμενο έχει διαγραφτεί από τη μνήμη. Αυτό είναι χρήσιμο στις περιπτώσεις που ο προγραμματισμός γίνει σε κάποια γλώσσα που δεν υποστηρίζει αυτόματη διαχείριση μνήμης (π.χ. **C++**) οπότε πρέπει να φροντίζουμε για τη καταστροφή παραμέτρων που πλέον δεν χρειάζονται ώστε να εξοικονομούνται πόροι συστήματος. Όταν πρόκειται να χρησιμοποιηθεί γλώσσα που υποστηρίζει αυτόματη διαχείριση μνήμες (π.χ. **C#**) τότε δεν είναι απαραίτητο να μοντελοποιήσουμε τη

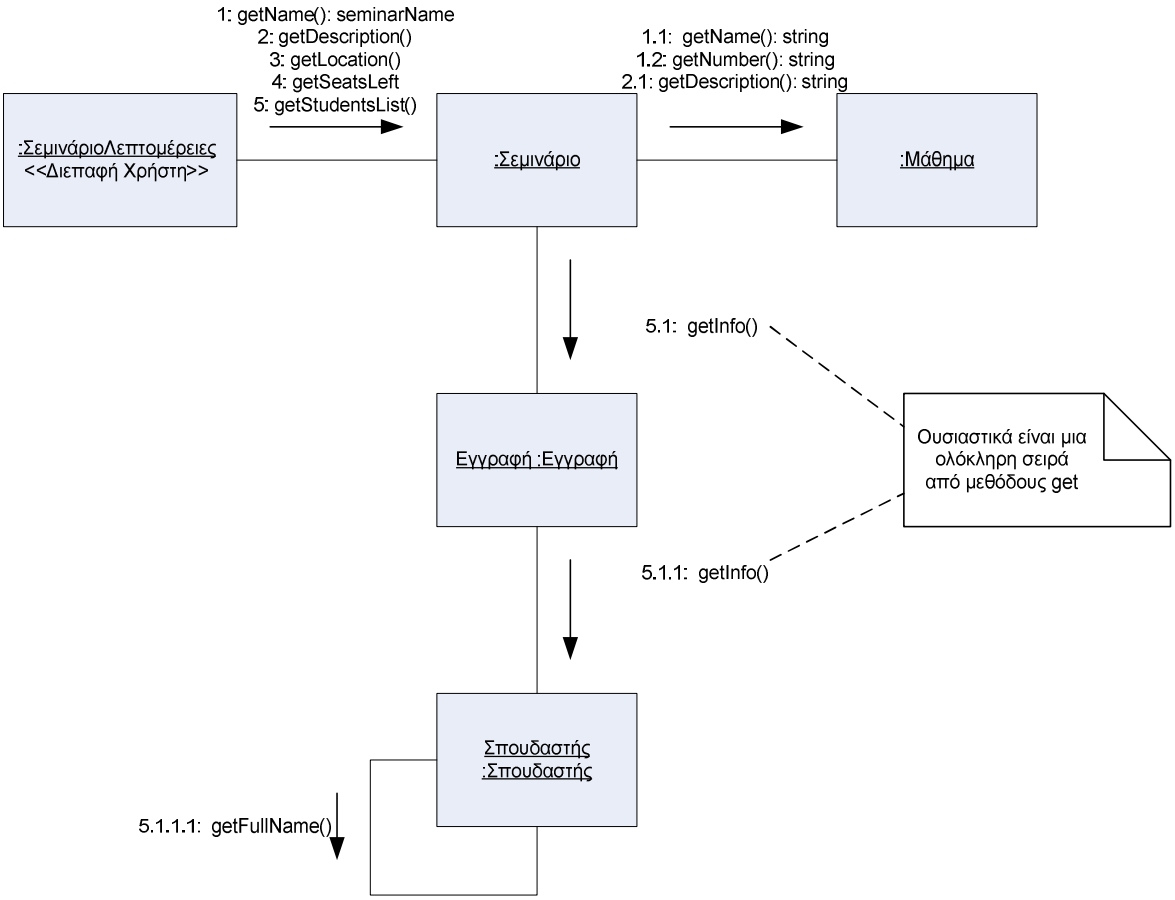
καταστροφή παραμέτρων, αφού αυτή λαμβάνει χώρα αυτόματα όταν η παράμετρος δεν χρησιμοποιείται πλέον.

### 3.2.4.2 Επικοινωνίας (Communication)

Μια βασική αρχή της **UML** είναι η χρήση διαφορετικών διαγραμμάτων για διαφορετικούς σκοπούς. Τα διαγράμματα Κλάσης χρησιμοποιούνται για να μοντελοποιήσουν τη στατική φύση του συστήματός, τα διαγράμματα Αλληλουχίας χρησιμοποιούνται για να μοντελοποιήσουν την ακολουθιακή λογική, και τα διαγράμματα Κατάστασης χρησιμοποιούνται για να αναπαραστήσουν της συμπεριφορά πολύπλοκων κλάσεων. Αλλά τι συμβαίνει όταν πρέπει να επιδειχθεί η συμπεριφορά διαφόρων αντικειμένων που συνεργάζονται μαζί για την επίτευξη ενός κοινού σκοπού; Σε αυτή τη περίπτωση χρησιμοποιούνται τα διαγράμματα Επικοινωνίας, τα οποία στη **UML 1.x** ονομάζονταν διαγράμματα συνεργασίας, τα οποία παρέχουν μια πιο υψηλού επιπέδου άποψη των συνεργαζόμενων αντικειμένων.

Τα διαγράμματα επικοινωνίας δείχνουν τη ροή μηνυμάτων μεταξύ των αντικειμένων σε μια αντικειμενοστραφή εφαρμογή και επιπλέον παρουσιάζουν τις βασικές σχέσεις μεταξύ των κλάσεων. Ουσιαστικά εστιάζουν στη δομική οργάνωση των αντικειμένων που στέλνουν και λαμβάνουν μηνύματα.

Στο παρακάτω σχήμα παρουσιάζεται ένα απλοποιημένο διάγραμμα Επικοινωνίας για την επίδειξη μιας στην οθόνη των λεπτομερειών ενός σεμιναρίου. Τα τετράγωνα αντιπροσωπεύουν τα διάφορα αντικείμενα που εμπλέκονται και αποτελούν την εφαρμογή. Οι γραμμές μεταξύ των κλάσεων εκπροσωπούν τις σχέσεις (συσχετίσεις, συνθέσεις, εξαρτήσεις, ή γενικεύσεις) μεταξύ τους. Η ίδια σημειογραφία για τις κλάσεις και τα αντικείμενα που χρησιμοποιείται στα διαγράμματα Αλληλουχίας, χρησιμοποιείται και στα διαγράμματα Επικοινωνίας, κάτι που φανερώνει και την συνέπεια της σήμανσης της **UML**. Τα μηνύματα απεικονίζονται ως βέλη με ετικέτα, και δείχνουν την κατεύθυνση του μηνύματος, όπως ακριβώς και στα διαγράμματα ακολουθίας.




**Εικόνα 46 Διάγραμμα Επικοινωνίας για το σύστημα εγγραφής σε σεμινάριο**

Παρακάτω, παρουσιάζεται η βασική σήμανση για τη μοντελοποίηση των μηνυμάτων στα διαγράμματα Επικοινωνίας.

***[****αριθμόςΑκολουθίας:****]****όνομαΜεθόδου****(****παράμετροι****)[:****επιστρεφόμενηΤιμή****]***

Όπως γίνεται φανερό, εναλλακτικά, μπορεί να αναφέρεται και ο αριθμός ακολουθίας που στέλνεται το μήνυμα, καθώς και μια επιστρεφόμενη τιμή. Συμβατικά όμως πρέπει να αναφέρονται το όνομα της μεθόδου καθώς και οι τιμές που δίνονται ως είσοδος (αν ισχύει) στη μέθοδο. Οι αριθμοί της ακολουθίας θα πρέπει να είναι στη μορφή Α.Β**.C.D** ώστε να καθορίζουν τη σειρά με την οποία στέλνονται τα μηνύματα.

Στη παραπάνω εικόνα, το μήνυμα **1** στέλνεται στο αντικείμενο *Σεμινάριο* που με τη σειρά του στέλνει τα μηνύματα **1.1** και **1.2** στο αντικείμενο *Μάθημα*. Επίσης, το μήνυμα **5** στέλνεται στο αντικείμενο *Σεμινάριο*, το οποίο στέλνει το μήνυμα **5.1** στο αντικείμενο *Εγγραφή* της ομώνυμης κλάσης, που με τη σειρά της στέλνει το μήνυμα **5.1.1** στο αντικείμενο *Σπουδαστής*, το οποίο στέλνει το μήνυμα **5.1.1.1** στον εαυτό του, όπως φαίνεται από την επαναληπτική σύνδεση που διαθέτει το αντικείμενο *Σπουδαστής*.

Στο παραπάνω διάγραμμα έχουν χρησιμοποιηθεί αριθμοί ακολουθίας για τα μηνύματα, κάτι που γενικά θα ήταν χρήσιμο στη σχεδίαση ενός διαγράμματος

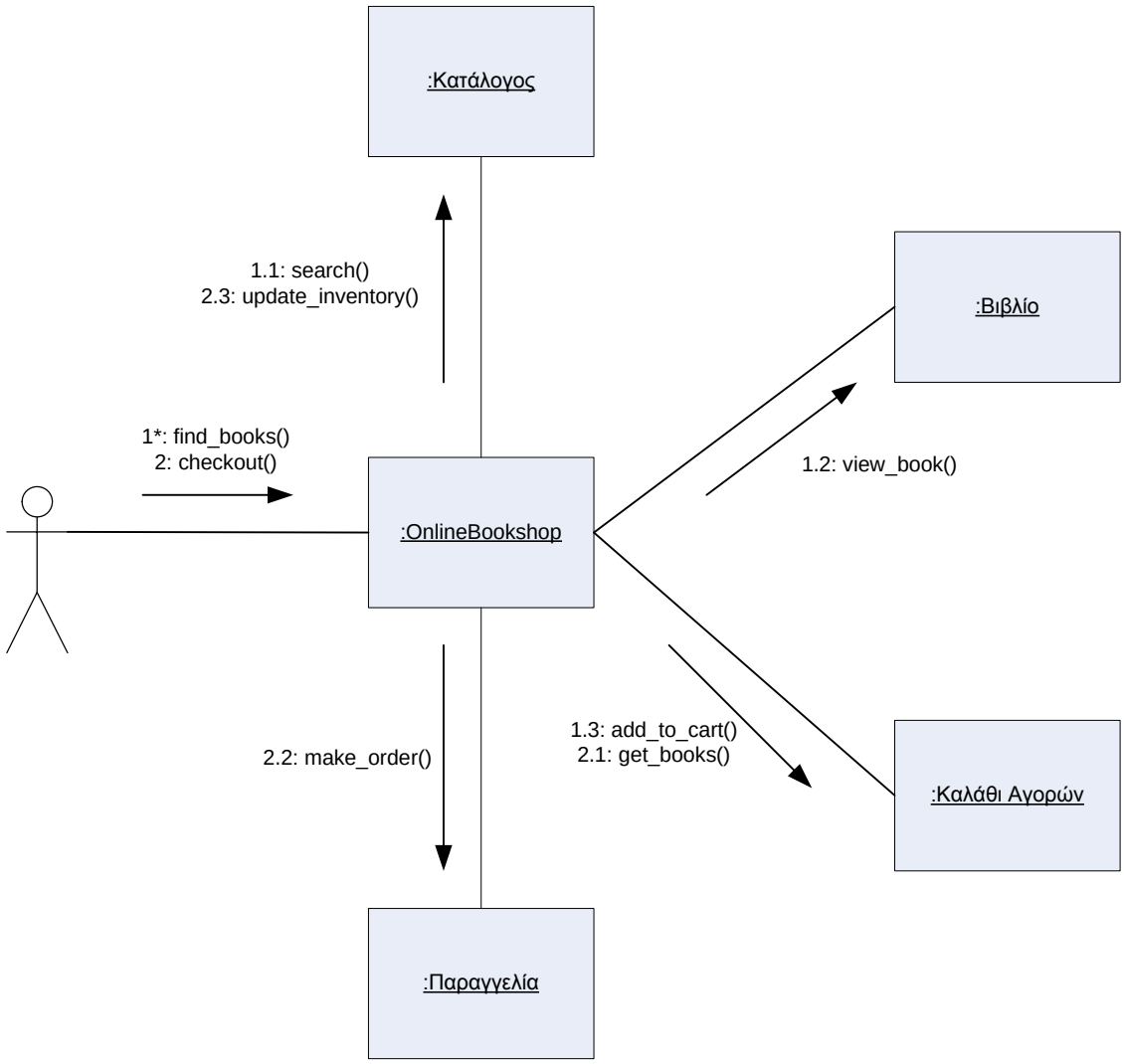
Αλληλουχίας. Η κύρια διαφορά μεταξύ διαγραμμάτων Επικοινωνίας και Αλληλουχίας είναι ότι τα δεύτερα απεικονίζουν μεν καλά τη διαδοχική λογική αλλά όχι τόσο καλά μια ευρύτερη άποψη της ανταλλαγής μηνυμάτων εντός του συστήματος, κάτι που προσφέρουν άψογα τα διαγράμματα Επικοινωνίας.

Επίσης, όπως φαίνεται το αντικείμενο της διεπαφής χρήστη ΣεμινάριοΛεπτομέρειες συνεργάζεται με το αντικείμενο Σεμινάριο για να αποκτήσει τις απαραίτητες πληροφορίες ώστε να τις εμφανίσει στον χρήστη. Αρχικά, καλεί τη μέθοδο **get** για να ανακτήσει το όνομα του σεμιναρίου. Για να γίνει αυτό, το αντικείμενο Σεμινάριο συνεργάζεται με το αντικείμενο Μάθημα για να ανακτήσει το όνομα και τον αριθμό του σεμιναρίου που εμπίπτει στο συγκεκριμένο μάθημα, όπως φαίνεται από τα μηνύματα **1.1** και **1.2.** Στα συγκεκριμένα μηνύματα έχει συμπεριληφθεί και η επιστρεφόμενη τιμή, δηλαδή πρόκειται για τιμή τύπου **string.** Θα μπορούσε εναλλακτικά να συμπεριλαμβάνεται το αποτέλεσμα π.χ. **seminarName.**

Ακολουθεί ένα παράδειγμα για ένα **online** Βιβλιοπωλείο. Ο πελάτης, που αναπαρίσταται ως **actor,** μπορεί να ψάξει, να δει και να αγοράσει βιβλία.

Η επικοινωνία ξεκινάει με το μήνυμα **1*: find_books(),** που μπορεί να είναι ένα επαναληπτικό μήνυμα **(**αυτό σημαίνει το ***),** καθώς ο πελάτης μπορεί να ψάξει για πολλά βιβλία. Αφού λοιπόν γίνει η διερεύνηση στο κατάλογο βιβλίων και εντοπιστεί κάποιο συγκεκριμένου ενδιαφέροντος, τότε ακολουθεί το μήνυμα **1.2: view_book().** Αν ο πελάτης αποφασίσει να αγοράσει το βιβλίο, τότε ακολουθεί το μήνυμα **1.3: add_to_cart().**

Η διαδικασία **check out** της εφαρμογής περιλαμβάνει την ανάκτηση της λίστας βιβλίων που έχουν ενταχθεί στο καλάθι αγορών, τη δημιουργία της παραγγελίας και την ανανέωση του καταλόγου, αν ολοκληρωθεί η παραγγελία. Συγκεκριμένα, τα μηνύματα που στέλνονται είναι τα **2: checkout()** για την έναρξη της διαδικασίας, **2.1: get_books()** για τη τοποθέτηση των βιβλίων στο καλάθι αγορών, **2.2: make_order()** για τη δημιουργία της παραγγελίας με τα βιβλία που περιέχονται στο καλάθι αγορών και **2.3: update_inventory()** για την ανανέωση του **online** καταλόγου όταν ολοκληρωθεί η παραγγελία.

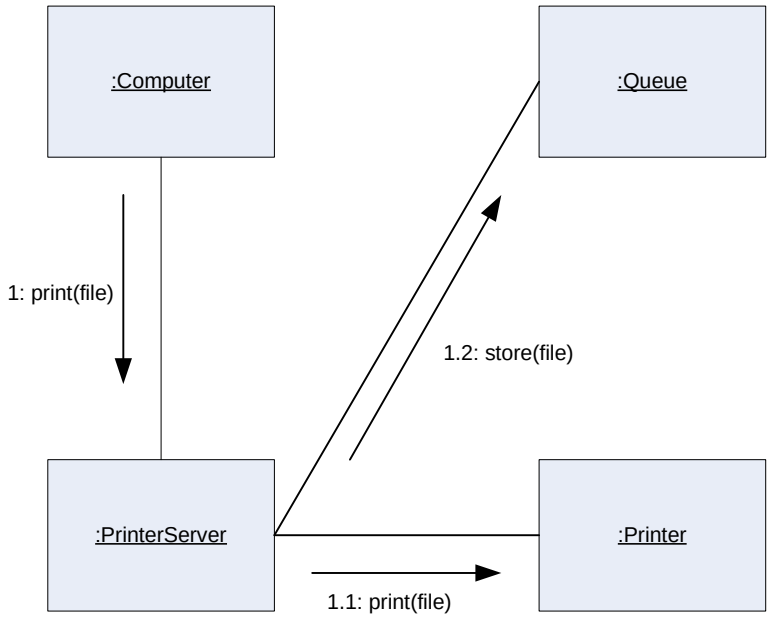



**Εικόνα 47 Διάγραμμα Επικοινωνίας για Online Βιβλιοπωλείο**

Τέλος παρουσιάζεται ένα παράδειγμα διαγράμματος επικοινωνίας για το σύστημα εκτύπωσης αρχείων από ένα ηλεκτρονικό υπολογιστή σε ένα δικτυακό εκτυπωτή**.**

Ο υπολογιστής καλεί τη μέθοδο **print** με παράμετρο το προς εκτύπωση αρχείο **file,** δηλαδή στέλνει το μήνυμα **1: print(file).** Ο εξυπηρετητής εκτυπώσεων λαμβάνει το μήνυμα και αν ο εκτυπωτής είναι ελεύθερος**,** του προωθεί το μήνυμα μέσω του **1.1: print(file),** ενώ αν είναι απασχολημένος τότε καλεί τη μέθοδο **store** με το μήνυμα **1.2 store(file)** και η εντολή εκτύπωσης τοποθετείται στην ουρά**.**

Βέβαια**,** αυτό το παράδειγμα δεν είναι ολοκληρωμένο καθώς δεν εξηγείται τι πρέπει να συμβαίνει όταν τοποθετηθεί μια εκτύπωση στην ουρά**.**



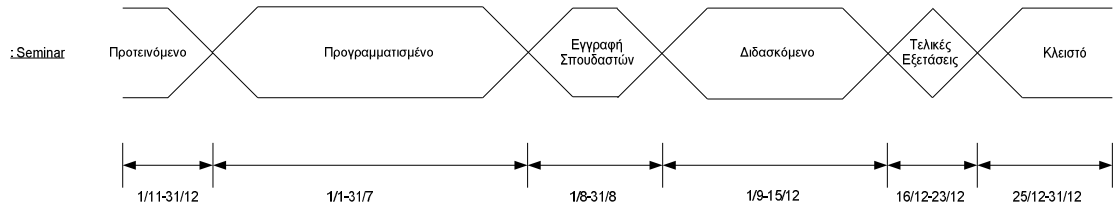

**Εικόνα 48 Διάγραμμα επικοινωνίας για εκτύπωση σε δικτυακό εκτυπωτή**

### 3.2.4.3 Χρονισμού (Timing)

Τα διαγράμματα Χρονισμού είναι από τα νέα στοιχεία που προστεθήκανε στη **UML 2.0.** χρησιμοποιούνται για τη διερεύνηση της συμπεριφοράς ενός ή περισσοτέρων αντικειμένων κατά τη διάρκεια μιας χρονικής περιόδου. Υπάρχουν δύο βασικοί τύποι διαγραμμάτων χρονισμού, η συνοπτική σήμανση **(concise notation)** που παρουσιάζεται στο επόμενο σχήμα και η ισχυρή σήμανση **(robust notation)** που περιγράφεται αμέσως μετά. Τα διαγράμματα χρονισμού χρησιμοποιούνται κατά το σχεδιασμό ενσωματωμένου λογισμικού, όπως στις περιπτώσεις λογισμικού ελέγχου, όπου ο χρόνος παίζει το κυρίαρχο ρόλο.

Το παρακάτω σχήμα απεικονίζει τον κύκλο ζωής ενός σεμιναρίου, δείχνοντας το χρονοδιάγραμμά του ξεκάθαρα. Η ετικέτα :Σεμινάριο υποδηλώνει ότι το χρονοδιάγραμμα που παρουσιάζεται αφορά ένα στιγμιότυπο ενός σεμιναρίου. Οι κρίσιμες καταστάσεις του σεμιναρίου που παρουσιάζονται στο διάγραμμα είναι Προτεινόμενο, Προγραμματισμένο, Εγγραφή Σπουδαστών, Διδασκόμενο, Τελικές Εξετάσεις, Κλειστό.

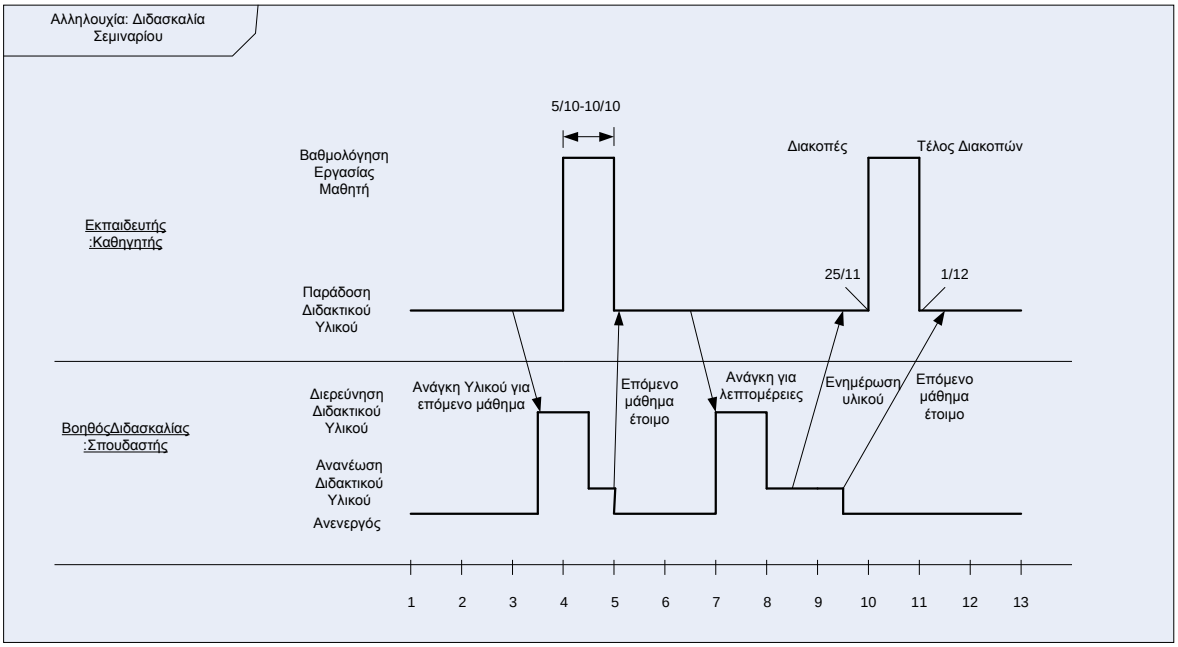
Οι δύο γραμμές που περικυκλώνουν τις καταστάσεις χρησιμοποιούνται για να δείξουν τη διάρκεια των καταστάσεων και τη μετάβαση από η μια κατάσταση στην επόμενη. Στο κάτω μέρος του διαγράμματος παρουσιάζονται οι χρονικοί περιορισμοί που στη προκειμένη περίπτωση δείχνουν το χρονικό διάστημα που το σεμινάριο βρίσκεται σε κάθε μια κατάσταση.




**Εικόνα 49 Διάγραμμα χρονισμού σεμιναρίου (συνοπτική σήμανση)**

Είναι ενδιαφέρον να σημειώσουμε τις διαφορές μεταξύ του παραπάνω διαγράμματος χρονισμού και του αντίστοιχου διαγράμματος καταστάσεων της Εικόνας **33.** Αρκετές καταστάσεις του διαγράμματος καταστάσεων έχουν ενσωματωθεί στη κατάσταση Εγγραφή Σπουδαστών του διαγράμματος χρονισμού. Αυτό είναι απολύτως φυσιολογικό καθώς το κάθε διάγραμμα χρησιμοποιείται για διαφορετικό σκοπό. Έτσι το διάγραμμα χρονισμού με συνοπτική σήμανση χρησιμοποιείται για την επισκόπηση ενός ή περισσοτέρων αντικειμένων κατά τη πάροδο του χρόνου, ενώ τα διαγράμματα καταστάσεων αντικατοπτρίζουν καλύτερα τις λεπτομερείς μεταβάσεις ανάμεσα σε καταστάσεις σαν αποτέλεσμα των εσωτερικών ή εξωτερικών συμβάντων **(events).**

Το επόμενο σχήμα απεικονίζει ένα διάγραμμα χρονισμού που παρουσιάζει με λεπτομέρεια τι ακριβώς συμβαίνει, όταν ένα σεμινάριο που διδάσκεται. Σε αυτό το παράδειγμα, ο εκπαιδευτής της κλάσης *Καθηγητής* παραδίδει στο σεμινάριο και βαθμολογεί τις εργασίες των σπουδαστών, ενώ ο βοηθός διδασκαλίας **(**που είναι επίσης *Σπουδαστής* και γα αυτό προέρχεται από αυτή τη κλάση**)** αναπτύσσει το εκπαιδευτικό υλικό για να διδαχθεί. Ένα πλαίσιο χρησιμοποιείται για να περιορίζει τις δύο διάρκειες ζωής, αυτή του καθηγητή και του βοηθού διδασκαλίας, ενώ είναι δυνατόν με τον ίδιο τρόπο να μοντελοποιηθούν και άλλοι **Actors.** Οι οριζόντιες και κατακόρυφες γραμμές ονομάζονται χρονογραμμές καταστάσεων και στην προκειμένη περίπτωση αναπαριστούν τις διακριτές μεταβάσεις μεταξύ των καταστάσεων. Σε άλλες περιπτώσεις όπου οι μεταβολές είναι συνεχούς φύσης **(**όπως π.χ. η αλλαγή της θερμοκρασίας**),** οι γραμμές θα ήτανε καμπυλωτές. Στη προκειμένη περίπτωση οι καταστάσεις που ισχύουν, όπως η *Βαθμολόγηση Εργασίας Σπουδαστή*, η κατάσταση *Ανενεργός* κ.λ.π. παρατίθενται στην αριστερή πλευρά του διαγράμματος, ενώ τα γεγονότα όπως *Διακοπές*, *Τέλος Διακοπών* μπορούν προαιρετικά να επισημαίνονται στα σημεία μετάβασης, ώστε να επιδεικνύουν και το λόγο της μετάβασης. Τα βέλη μεταξύ των χρονογραμμών, αναπαριστούν τα μηνυμάτων μεταξύ των αντικειμένων.




**Εικόνα 50 Διάγραμμα χρονισμού σεμιναρίου (ισχυρή σήμανση)**

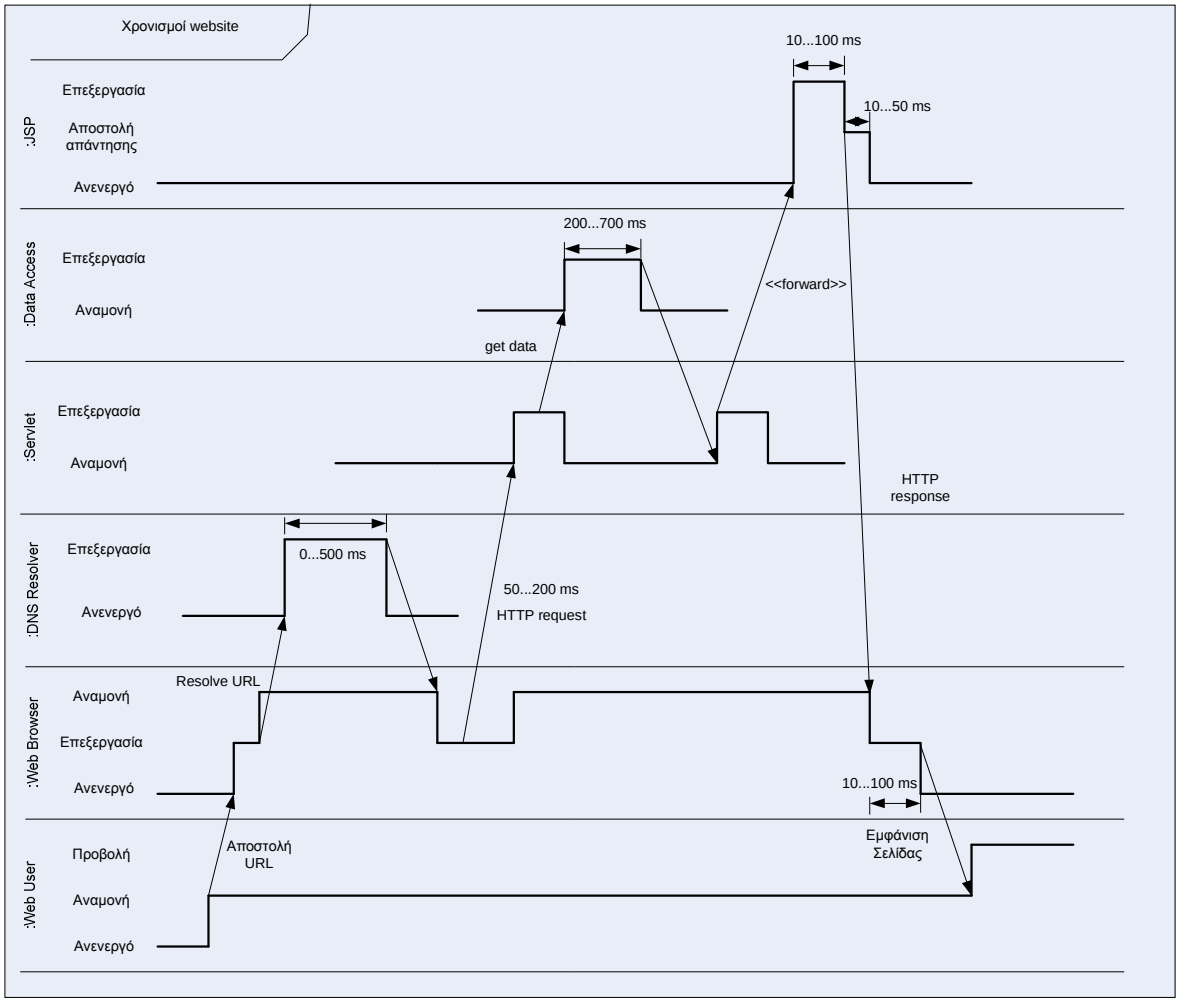
Υπάρχουν πολλοί τρόποι για να δηλωθεί ο χρόνος. Ο χρονικός περιορισμός, **5** Οκτωβρίου έως **10** Οκτωβρίου παρουσιάζεται όπως και στο προηγούμενο διάγραμμα, ενώ η χρονικές στιγμές Τ**=25** Νοεμβρίου και Τ **= 1** Δεκεμβρίου, φαίνονται με γραμμές. Οι χρονικοί περιορισμοί και οι χρονικές παρατηρήσεις μπορούν να εφαρμοστούν σε πληθώρα **UML** διαγραμμάτων, συμπεριλαμβανομένων όλων των τύπων διαγραμμάτων αλληλεπίδρασης, όπως διαγράμματα αλληλεπίδρασης, επικοινωνίας κ.λ.π. αν και η πιο σημαντική χρήση τους είναι στα διαγράμματα χρονισμού. Ένα μοναδικό χαρακτηριστικό των διαγραμμάτων χρονισμού είναι η χρονική κλίμακα στο κάτω μέρος του διαγράμματος.

Στο επόμενο διάγραμμα παρουσιάζεται η περίπτωση των χρονικών περιορισμών και καθυστερήσεων που εμφανίζει ένα **website**. Ο χρήστης εισάγει το **url** της ιστοσελίδας στο **browser** του, και περιμένει μέχρι το **URL** να επιλυθεί σε κάποια διεύθυνση **IP** από την υπηρεσία **DNS (domain name service)**. Αυτή η διαδικασία μπορεί να εισάγει μια σοβαρή καθυστέρηση στο χρόνο αναμονής, όπως τον αντιλαμβάνεται ο χρήστης. Οι καθυστερήσεις που οφείλονται στην επίλυση του **url** από το **DNS Resolver** μπορεί να κυμαίνονται από **1 ms** έως μερικά δευτερόλεπτα.

Το **Java servlet** παίρνει τον έλεγχο και αιτείται κάποια δεδομένα που συνήθως απαιτεί κάποιο αισθητό χρόνο. Αφού τα δεδομένα ληφθούν και επεξεργαστούν, το **servlet** προωθεί το **request** στη **JSP**, ενώ το **HTTP response** στέλνεται πίσω στον **browser**.

Οι **web browsers** απαιτούν κάποιο χρόνο για να διεκπεραιώσουν το **HTTP response** και να ξεκινήσει το **render** της **HTML** σελίδας για να προβληθεί στο χρήστη. Να σημειωθεί ότι κανονικά, απαιτείται ακόμα περισσότερος χρόνος ώστε ο **web browser** να

ζητήσει από άλλες πηγές στοιχεία όπως **CSS, JavaScript** Εικόνες, κάτι που δεν παρουσιάζεται στο συγκεκριμένο διάγραμμα.




**Εικόνα 51 Διάγραμμα χρονισμού website (ισχυρή σήμανση)**

### 3.2.4.4 Επισκόπησης αλληλεπίδρασης (Interaction Overview)

Τα διαγράμματα Επισκόπησης Αλληλεπίδρασης παρέχουν επισκόπηση της ροής του ελέγχου. Οι κόμβοι στο εσωτερικό του διαγράμματος είναι πλαίσια που μπορεί να είναι δύο ειδών: πλαίσια αλληλεπίδρασης που απεικονίζουν οποιοδήποτε τύπο **UML** διαγράμματος **(**διάγραμμα ακολουθίας, επικοινωνίας, χρονισμού κλπ**)** ή πλαίσια περιστατικού αλληλεπίδρασης που δείχνουν μια δραστηριότητα ή μια λειτουργία που έχει κληθεί. Τα πλαίσια αυτά συμβολίζονται με ένα παραλληλόγραμμο που πάνω αριστερά έχουν ένα εσωτερικό **tab** που περιέχει τ**ο** συνδυασμό γραμμάτων:

⇒ **sd** για διάγραμμα αλληλουχίας

⇒ **cd** για διάγραμμα επικοινωνίας

⇒ **td** για διάγραμμα χρονισμού

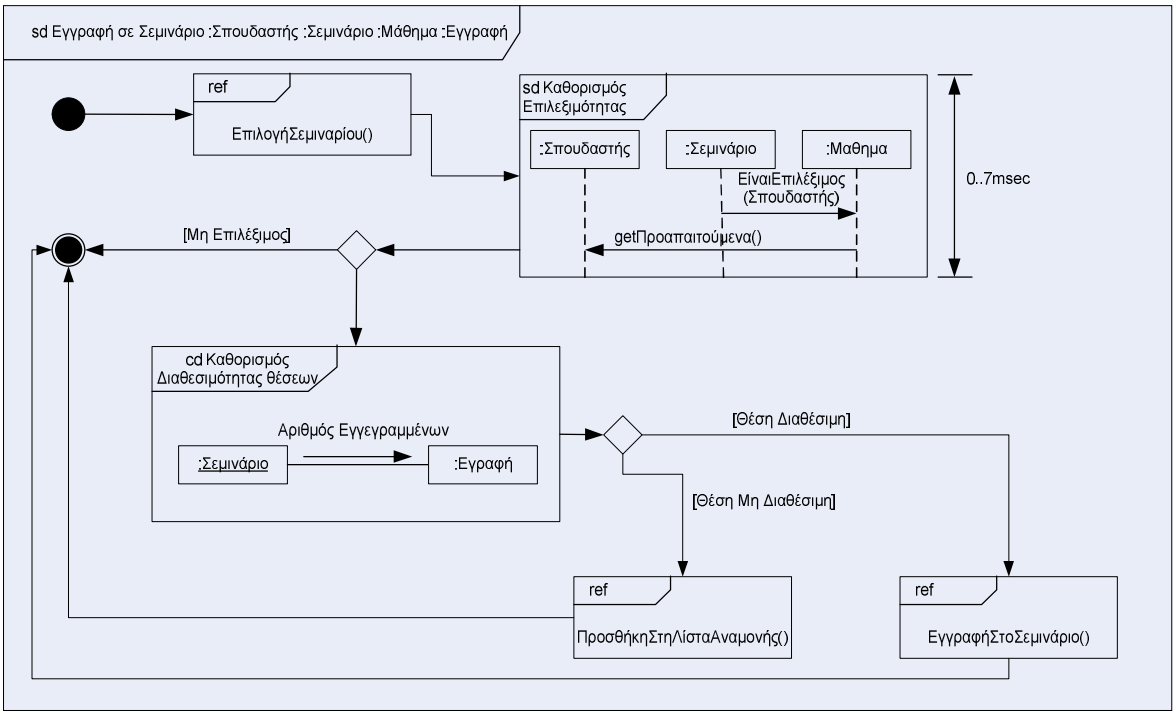
⇒ **iod** για διάγραμμα επισκόπησης αλληλεπίδρασης

⇒ **ref** για διάγραμμα περιστατικού αλληλεπίδρασης

και προαιρετικά το όνομα του διαγράμματος. Τα πλαίσια **ref** που αναπαριστούν την χρήση αλληλεπίδρασης γενικά όπως; π.χ. την επίκληση μεθόδου, είναι ανώνυμα.

Επιπλέον, στα διαγράμματα επισκόπησης αλληλεπίδρασης χρησιμοποιείται και ο συμβολισμός του σημείου απόφασης, που εμφανίζονται ως ρόμβοι, όπως ακριβώς και στα διαγράμματα δραστηριότητας. Συνήθως, υπάρχουν συνθήκες σε όλες τις ροές εξόδου αν και πολλές φορές κάτι τέτοιο μπορεί να παραβλέπεται όταν είναι προφανές τι σημαίνει. Οι χρονικοί περιορισμοί, όπως **0 .. 7 msec**, εμφανίζονται με τον ίδιο συμβολισμό όπως και στα λοιπά διαγράμματα αλληλεπίδρασης. Τα σημεία έναρξης και τέλους χρησιμοποιούν την ίδια σημειογραφία όπως και στα διαγράμματα καταστάσεων και τα διαγράμματα δραστηριότητας.

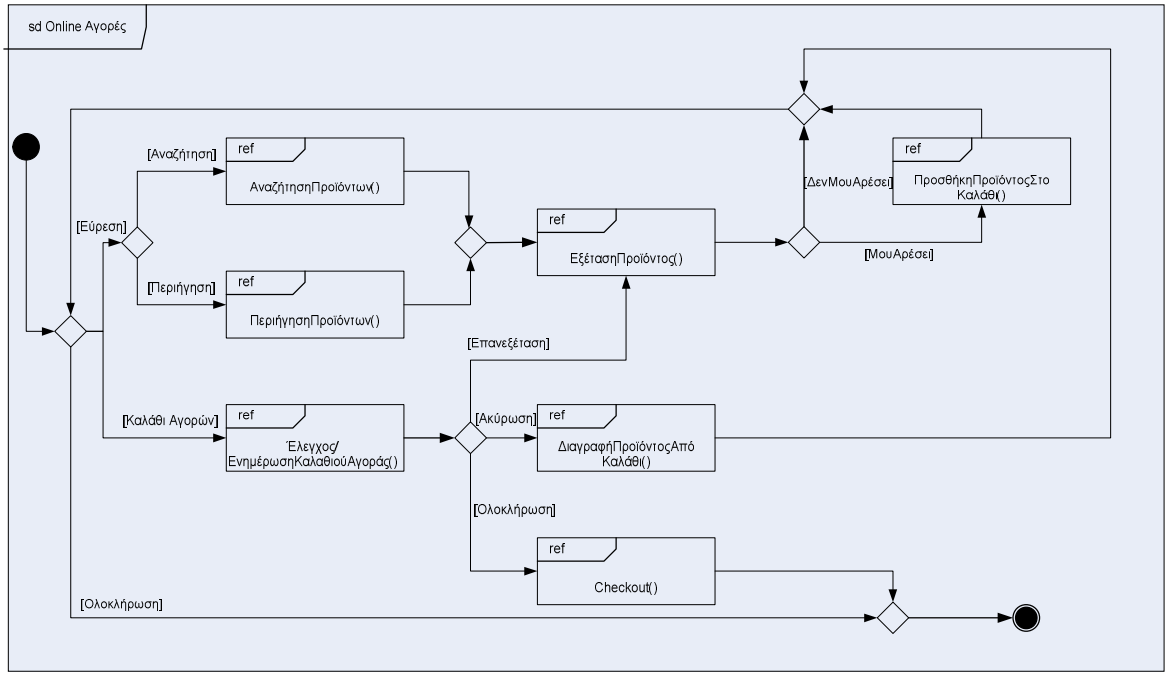
Ακολουθεί ένα διάγραμμα επισκόπησης αλληλεπίδρασης για τη περίπτωση εγγραφής σε σεμινάριο μαθήματος. Σε αυτό περιλαμβάνονται κόμβοι που αναπαριστούν διαγράμματα αλληλουχίας, επικοινωνία και χρήσης αλληλεπίδρασης.




**Εικόνα 52 Διάγραμμα Επισκόπησης Αλληλεπίδρασης για εγγραφή σε σεμινάριο**

Αν και διαγράμματα επισκόπησης αλληλεπίδρασης είναι μια ενδιαφέρουσα ιδέα, στη πράξη δεν χρησιμοποιούνται ιδιαίτερα. Τα πλαίσια αλληλεπίδρασης είναι σχεδόν δύσχρηστα, λόγω έλλειψης σχεδιαστικού χώρου, αφού τα διαγράμματα που μπορούν να απεικονιστούν μέσα στα πλαίσια θα είναι πολύ μικρά για να είναι εύχρηστα. Γενικά, τα διαγράμματα επισκόπησης αλληλεπίδρασης τείνουν να εγκαταλειφθούν για χάρη των διαγραμμάτων δραστηριότητας.

Ένα πιο απλό διάγραμμα είναι το παρακάτω που αναφέρεται στην αγορά προϊόντων από **Online** κατάστημα**.** Το συγκεκριμένο παράδειγμα περιλαμβάνει μόνο πλαίσια επίκλησης μεθόδων **ref.**




**Εικόνα 53 Διάγραμμα Επισκόπησης Αλληλεπίδρασης για online αγορές**

# Κεφάλαιο 4: Παράδειγμα μοντελοποίησης με UML

Σε αυτό το κεφάλαιο παρουσιάζεται η περίπτωση ενός ηλεκτρονικού καταστήματος. Το **portal** του ηλεκτρονικού καταστήματος θα επιτρέπει τη παρουσίαση και πώληση προϊόντων που ανήκουν σε διάφορες κατηγορίες και διαφορετικές φίρμες. Για την αποτελεσματική μοντελοποίηση του συστήματος θα χρησιμοποιηθούν τα πιο σημαντικά και χρηστικά διαγράμματα **UML.**

## *4.1 Το παράδειγμα ενός e-shop*

Το σενάριο πάνω στο οποίο θα βασιστεί η μοντελοποίηση με **UML** έχει ως εξής. Ένας πελάτης επισκέπτεται το ηλεκτρονικό κατάστημα και είτε αγοράζει ένα προϊόν είτε απλά περιηγείται στο **Portal**. Ο χρήστης επιλέγει Τμήμα και έπειτα τη Κατηγορία και τη Φίρμα, ώστε να του παρουσιαστεί μια λίστα συγκεκριμένων προϊόντων από τα οποία να επιλέξει. Για παράδειγμα, θα μπορούσε νε επιλέξει τμήμα **"**Εικόνα-Ήχος**"**, κατηγορία **"**Τηλεοράσεις**"** και φίρμα **"Sony".** Θα του εμφανιστεί μια λίστα προϊόντων που εμπίπτουν στα κριτήρια αυτά

Ο πελάτης μπορεί να επιλέξει το επιθυμητό προϊόν για αγορά. Η διαδικασία αυτή μπορεί να επαναληφθεί για περισσότερα προϊόντα. Όταν τελειώσει την επιλογή προϊόντων, ο πελάτης μπορεί να ελέγξει το καλάθι αγορών του. Αν θέλει να μετατρέψει τα προϊόντα που περιέχει μπορεί να το κάνει σε αυτό το σημείο.

Για την τελική πληρωμή, πρέπει να συνδεθεί στο **portal.** Αν το επισκέπτεται για πρώτη φορά πρέπει να εγγραφεί, ενώ αν είναι ήδη εγγεγραμμένος χρήστης πρέπει απλά να προχωρήσει στη σελίδα **login.**

Το τελικό καλάθι προϊόντων υποβάλλεται για πληρωμή και ζητούνται λεπτομέρειες της κάρτας συναλλαγής και στοιχεία διεύθυνσης **(**αν απαιτείται να γίνει αποστολή των προϊόντων**)**, τα οποία πρέπει να επιβεβαιώσει ο πελάτης. Τελικά, ο πελάτης λαμβάνει μια ενημέρωση που περιλαμβάνει τη ταυτότητα της συναλλαγής και απόδειξη.

Υπάρχουν οι εξής παραδοχές:

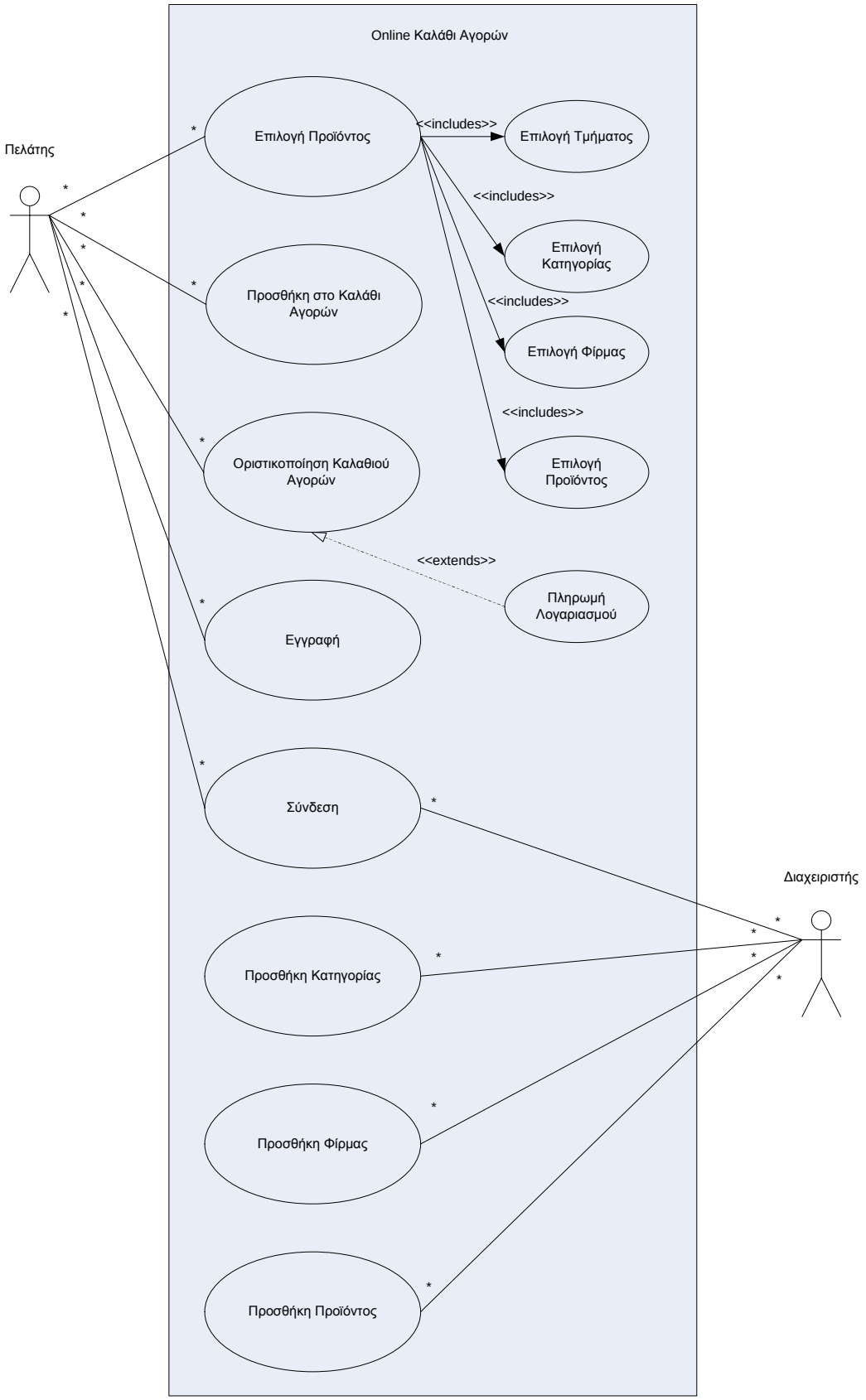
- Το νόμισμα που χρησιμοποιεί το **Portal** είναι ένα, το **€.**
- Υπάρχουν τμήματα κατηγορίες και φίρμες και τα προϊόντα δεν εμπίπτουν αναγκαστικά σε ένα μόνο από αυτά.
- Οι πελάτες πρέπει να ταυτοποιούνται πριν ολοκληρωθεί η οικονομική συναλλαγή.
- Οι **actors** που υπάρχουν είναι ο Πελάτης και ο Διαχειριστής.

Κωδικοποιώντας τα παραπάνω, οι λειτουργικότητα του συστήματος υπό μοντελοποίηση έχει ως εξής:

- **Portal**
  - Επιτρέπει στο πελάτη να βλέπει κατηγορίες, μάρκες και προϊόντα που ανήκουν σε αυτά
  - Επιτρέπει στο πελάτη να αγοράζει προϊόντα ηλεκτρονικά
  - Εκδίδει λογαριασμό και ταυτότητα συναλλαγής που μπορεί να χρησιμοποιηθεί για να παρακολουθείται η διαδικασία παράδοσης των προϊόντων
- Πελάτης
  - Όταν εισέρχεται στο **portal** μπορεί να δει τα προϊόντα από τις διάφορες κατηγορίες
  - Ταυτοποίηση των χρηστών γίνεται για τη πληρωμή των προϊόντων του καλαθιού
  - Ο πελάτης μπορεί να περιηγηθεί στ**ο** ηλεκτρονικό κατάστημα, να δει τα προϊόντα και να τοποθετήσει κάποια από αυτά στο καλάθι αγορών
  - Μπορεί ανά πάσα στιγμή να αλλάξει τα αντικείμενα που βρίσκονται εντός του καλαθιού, είτε προσθέτοντας νέα είτε αφαιρώντας κάποια από τα υπάρχοντα. Έπειτα μπορεί να οριστικοποιήσει τη το καλάθι αγορών και να προχωρήσει στην ολοκλήρωση της συναλλαγής, δίνοντας τα απαραίτητα στοιχεία.
- Διαχειριστής
  - Ενημερώνει τα τμήματα, κατηγορίες και προϊόντα
  - Επιβεβαιώνει τις λεπτομέρειες της κάρτας συναλλαγής του πελάτη και καταγράφει τις λεπτομέρειες της μεταφοράς των προϊόντων.

### 4.1.1 Διάγραμμα Περίπτωσης Χρήσης

Λαμβάνοντας υπόψη το σενάριο που αναπτύχθηκε πιο πάνω, το διάγραμμα Περιπτώσεων Χρήσης θα είναι όπως παρακάτω. Όπως φαίνεται υπάρχουν δύο **actors:** Ο Πελάτης που μπορεί να εγγραφεί και να συνδεθεί στο **portal** του ηλεκτρονικού καταστήματος, να επιλέξει ένα προϊόν, να το προσθέσει στο καλάθι αγορών και να το οριστικοποιήσει. Ο Διαχειριστής που μπορεί να συνδέεται στο **portal**, να προσθέτει Κατηγορίες, Φίρμες και Προϊόντα.




**Εικόνα 54 Διάγραμμα Περιπτώσεων Χρήσης για το Online Καλάθι Αγορών**

Επιπλέον, για να μπορέσει ο Πελάτης να επιλέξει προϊόν, πρέπει να συμπεριληφθούν στην Επιλογή Προϊόντος, τα Επιλογή Τμήματος, Κατηγορίας, Φίρμας και Προϊόντος. Πρακτικά, αυτό σημαίνει ότι όταν ο Πελάτης επιλέξει Προϊόν**/**τα θα πρέπει στη επιλογή του καθενός να έχει συμπεριληφθεί και η επιλογή του τμήματος, της κατηγορίας, της φίρμας και φυσικά του τελικού προϊόντος.

Αντιθέτως, η Πληρωμή Λογαριασμού επεκτείνει την Οριστικοποίηση Καλαθιού Αγορών, δηλαδή η οριστικοποίηση μπορεί να συμπληρωθεί από τη πληρωμή του αντίστοιχου κόστους. Πρακτικά αυτό σημαίνει ότι ο Πελάτης μπορεί να οριστικοποιήσει το καλάθι του χωρίς να προχωρήσει σε αγορά, κάνοντας το πιθανόν σε κάποια μεταγενέστερη στιγμή, ή να προχωρήσει και στη τελική αγορά οπότε σε αυτή τη περίπτωση η οριστικοποίηση του καλαθιού συνοδεύεται και από αίτημα πληρωμής.

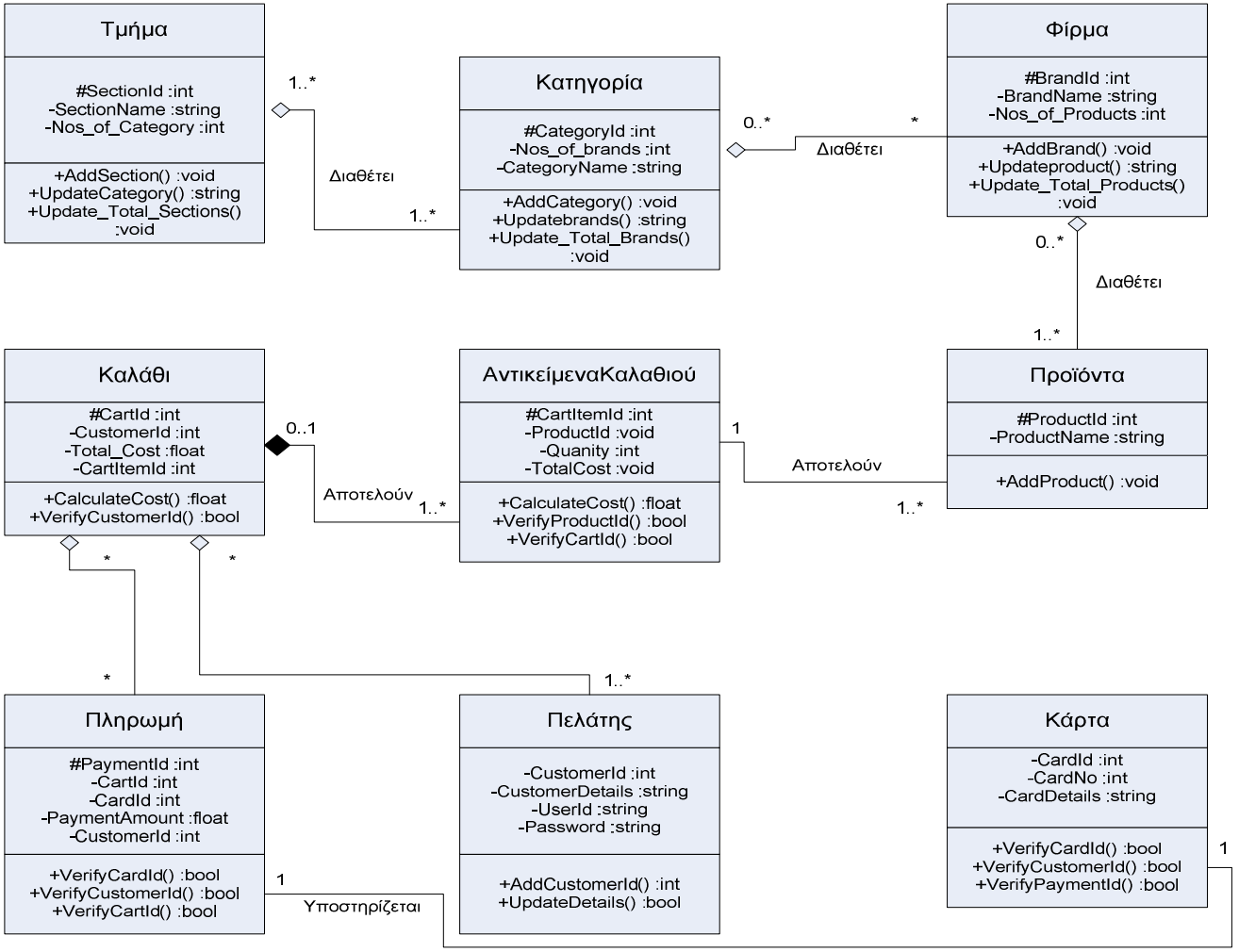
Τέλος, στο πιο πάνω διάγραμμα οι συσχετίσεις είναι γενικά Πολλές προς Πολλά, δηλαδή για παράδειγμα κανένας ή πολλοί Πελάτες μπορούν να πραγματοποιήσουν πολλές Εγγραφές στο **portal**.

### 4.1.2 Διάγραμμα Κλάσεων

Το διάγραμμα κλάσεων αποτελείται από **9** κλάσεις. Οι **3** από αυτές εκφράζουν τα αντικείμενα των Τμημάτων, Κατηγοριών και Φιρμών, στα οποία ανήκει κάποιο προϊόν. Η τέταρτη κλάση είναι αυτή των Προϊόντων. Ένα αντικείμενο της κλάσης Προϊόντα ανήκει σε κάποια Φίρμα, και μπορεί να ανήκει σε μια ή περισσότερες Κατηγορίες και Τμήματα.

Όπως φαίνεται και στο διάγραμμα, κάθε κλάση περιλαμβάνει ιδιότητες και λειτουργίες. Έτσι για παράδειγμα, ένα Τμήμα, έχει ως ιδιότητες, τη ταυτότητα **SectionId** που είναι ακέραιος, το όνομα **SectionName** που είναι αλφαριθμητικό και των αριθμό κατηγοριών **Nos_of_Categories** που περιλαμβάνει και είναι ακέραιος αριθμός. Επιπλέον, διαθέτει λειτουργίες για προσθήκη τμήματος **AddSection**, για ενημέρωση κατηγοριών **UpdateCategory** και ενημέρωση του συνολικού αριθμού τμημάτων **Update_Total_Sections.** Όμοια χαρακτηριστικά ισχύουν και για τις κλάσεις των κατηγοριών και των φιρμών.

Ένα σημαντικό σημείο του διαγράμματος είναι η συσσωμάτωση των κλάσεων Πληρωμή και Πελάτης για να υποστηριχθεί το Καλάθι, το οποίο έχει ως ιδιότητες του πελάτη **(**π.χ. **CustomerId)** και ιδιότητες της κλάσης Πληρωμή **(**π.χ. **CardId)** που με τη σειρά της τις λαμβάνει από τη κλάση Κάρτα **(**π.χ. **CardId).**

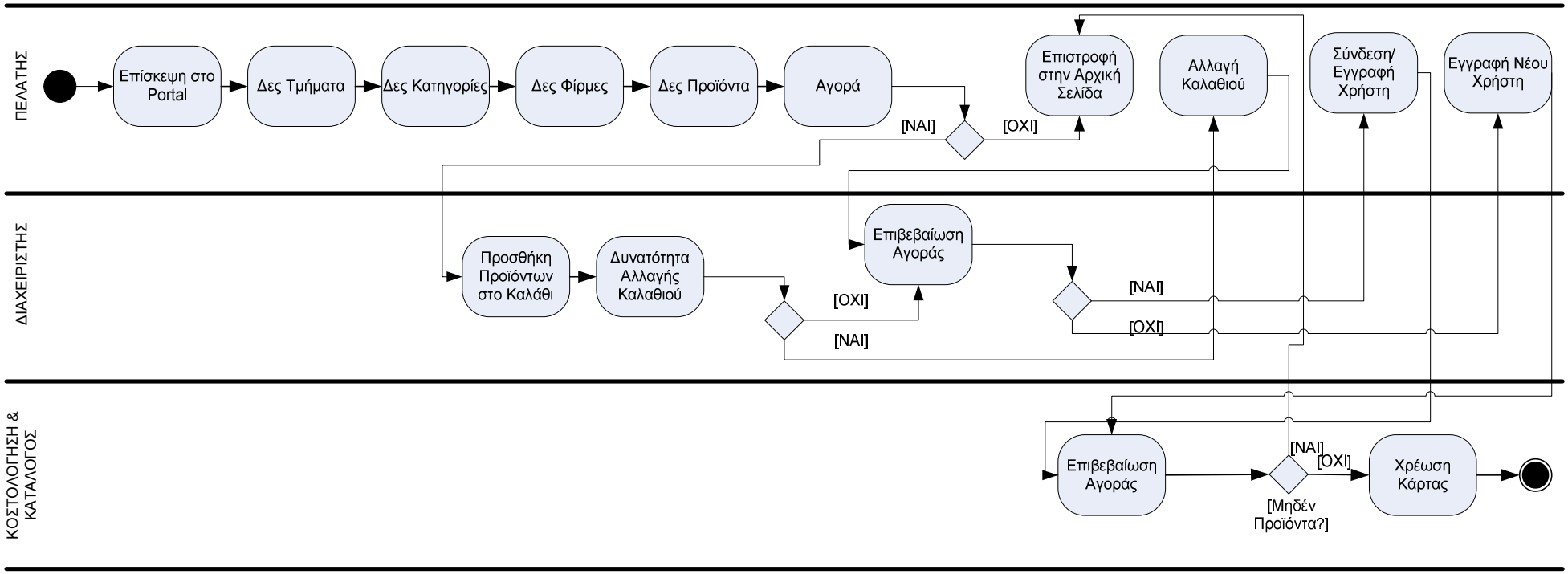



**Εικόνα 55 Διάγραμμα Κλάσεων για το ηλεκτρονικό κατάστημα**

---

## 4.1.3 Διάγραμμα Δραστηριοτήτων

Ακολουθεί το διάγραμμα δραστηριοτήτων.

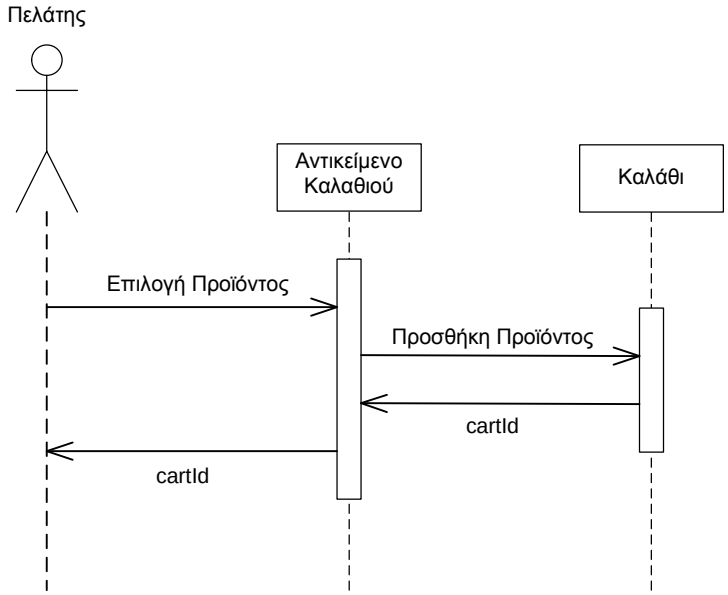



**Εικόνα 56 Διάγραμμα Δραστηριοτήτων για το ηλεκτρονικό κατάστημα**

## 4.1.4 Διαγράμματα Αλληλουχίας

Σε αυτό τι σημείο θα παρουσιαστούν τα διαγράμματα αλληλουχίας για μια σειρά περιπτώσεων χρήσης. Συγκεκριμένα παρουσιάζονται τα διαγράμματα για τις περιπτώσεις χρήσης:

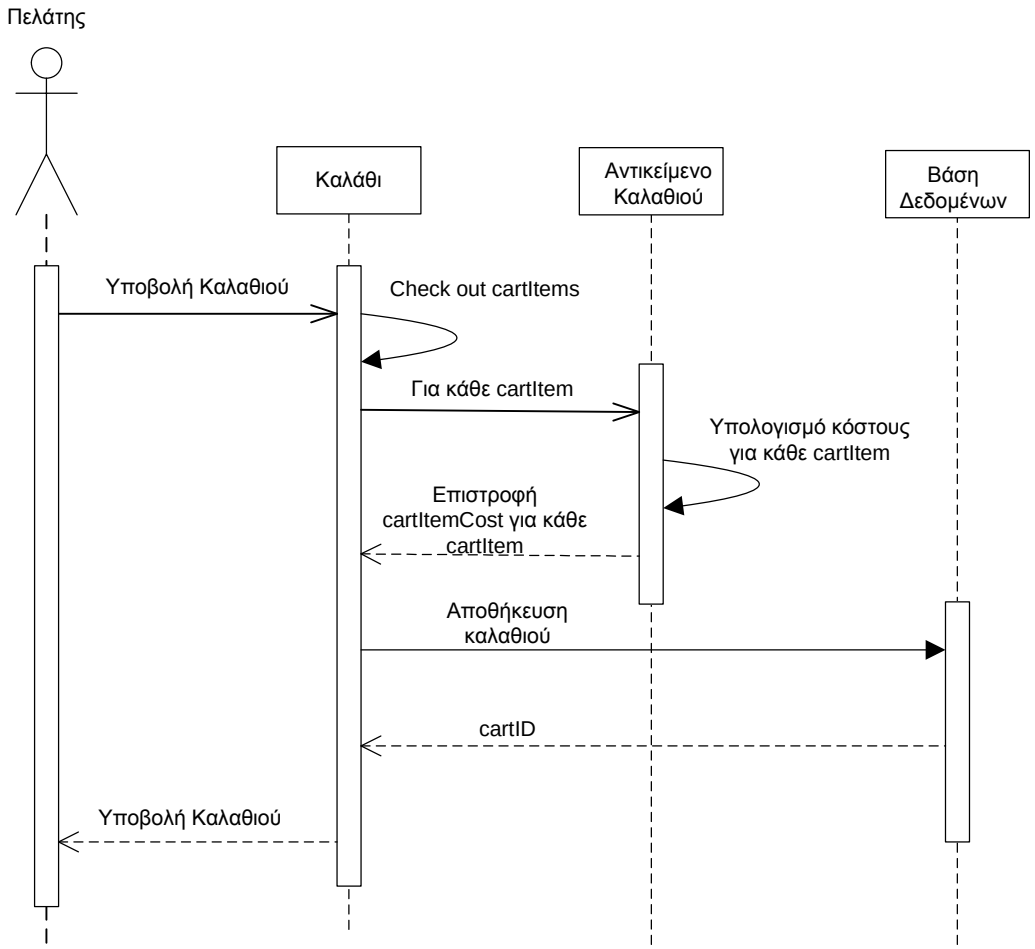
- Προσθήκη στο Καλάθι,
- Υποβολή Καλαθιού
- Εγγραφή Χρήστη
- Σύνδεση Χρήστη
- Προσθήκη νέου Τμήματος
- Προσθήκη νέας Κατηγορίας
- Προσθήκη νέας Φίρμας
- Προσθήκη νέου Προϊόντος




**Εικόνα 57 Διάγραμμα Αλληλουχίας για τη περίπτωση Προσθήκη Προϊόντος στο Καλάθι**

Όπως φαίνεται από το παραπάνω διάγραμμα, η προσθήκη ενός προϊόντος στο καλάθι αγορών προϋποθέτει την επιλογή του προϊόντος από τον Πελάτη. Σε αυτό το σημείο πρέπει να υπενθυμιστεί ότι η επιλογή προϊόντος, όπως άλλωστε φάνηκε και στο διάγραμμα περιπτώσεων χρήσης, συνεπάγεται επιπλέον και την συμπερίληψη του τμήματος, της κατηγορίας και της φίρμας του προϊόντος.

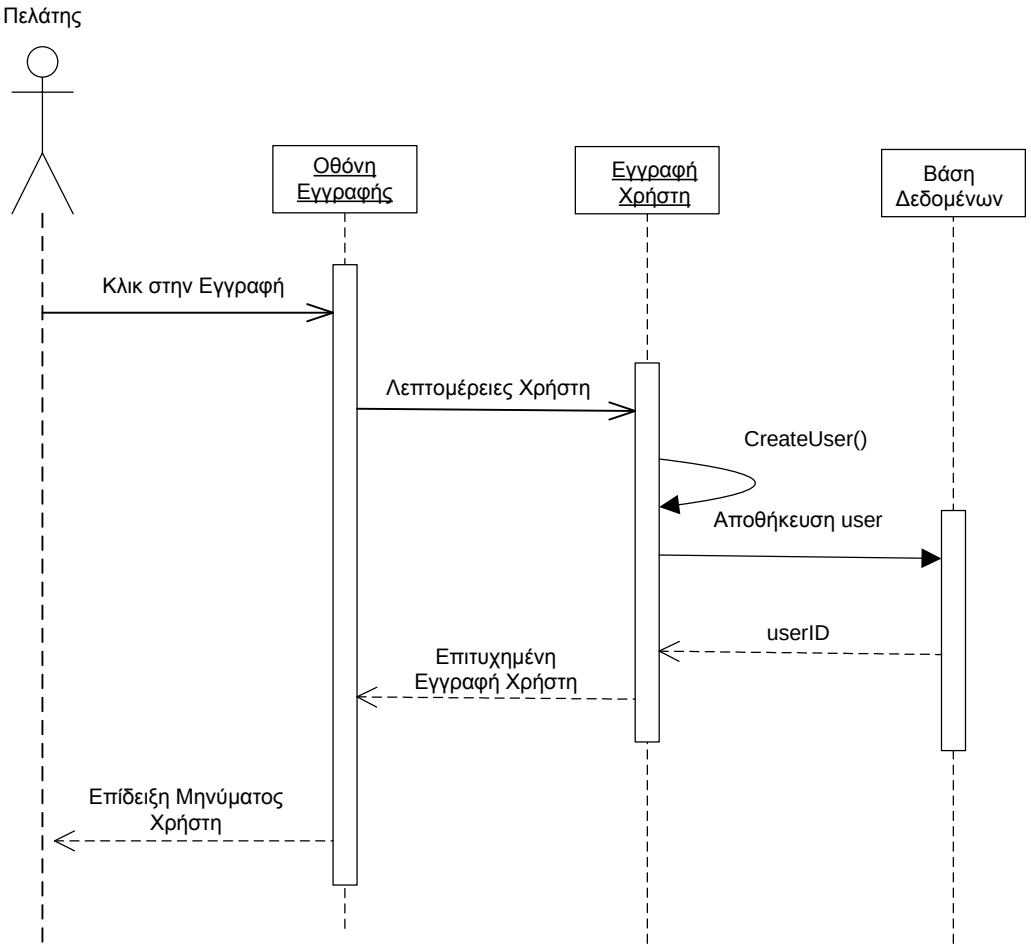
Μόλις το προϊόν προστεθεί στο καλάθι, τότε παράγεται ένα μοναδικό **cartId** που χαρακτηρίζει το καλάθι με το συγκεκριμένο προϊόν. Κάθε φορά που εισάγεται ένα νέο προϊόν στο καλάθι, δημιουργείται ένα νέο **cartId**, το οποίο τελικά χρησιμοποιείται κατά την πληρωμή των προϊόντων.




**Εικόνα 58 Διάγραμμα Αλληλουχίας για τη περίπτωση Υποβολής Καλαθιού**

Το παραπάνω διάγραμμα παρουσιάζει τη περίπτωση που ο χρήστης έχει τοποθετήσει προϊόντα στο καλάθι του και θέλει να το υποβάλει ώστε να ολοκληρώσει την αγορά. Αυτό που πρέπει ουσιαστικά να γίνει είναι να υπολογιστεί το κόστος των προϊόντων που συμπεριλαμβάνονται στο καλάθι. Για να πραγματοποιηθεί αυτός ο υπολογισμός απαιτείται η να εξεταστεί κάθε αντικείμενο ξεχωριστά και να συνυπολογιστεί το κόστος του, ώστε να προκύψει το συνολικό κόστος του καλαθιού.

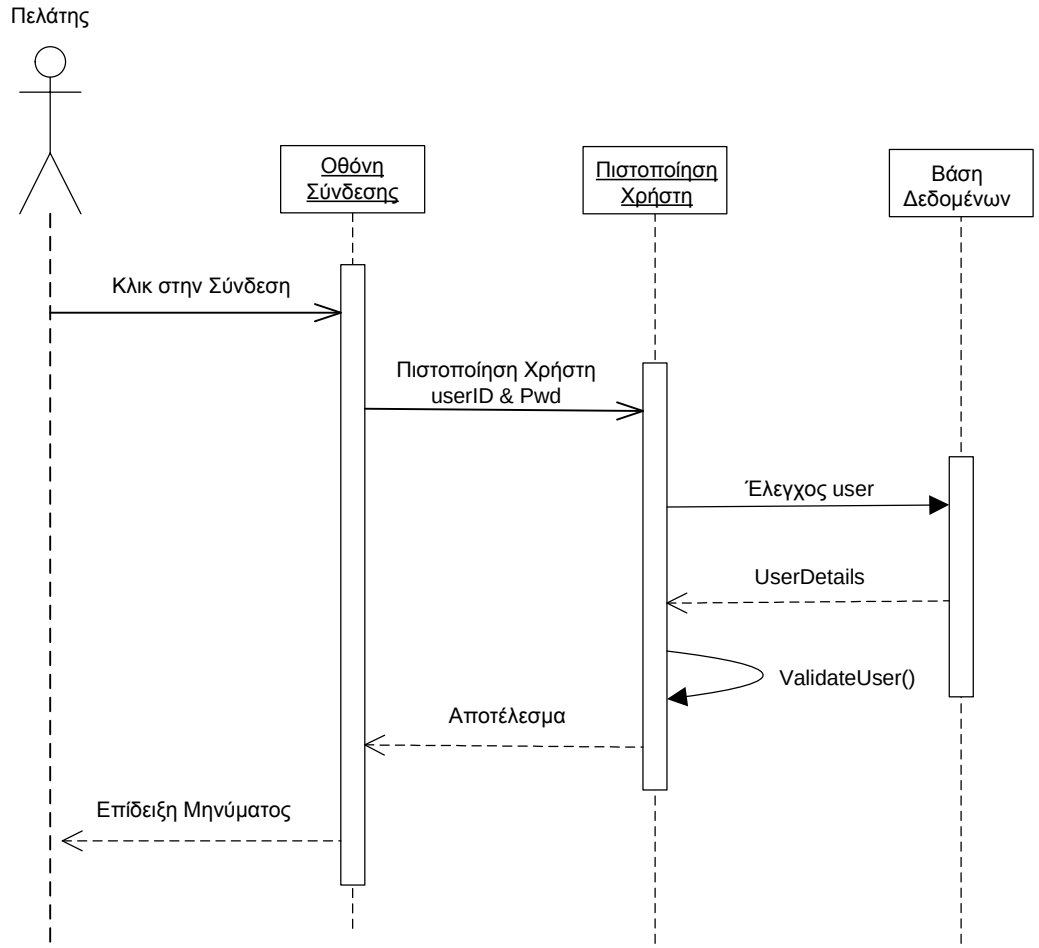
Το μήνυμα προς τη βάση δεδομένων απαιτείται για να ενημερωθεί ως προς τη πώληση των συγκεκριμένων προϊόντων και την ενημέρωση της με τις νέες πλέον διαθέσιμες ποσότητες.




**Εικόνα 59 Διάγραμμα Αλληλουχίας για τη περίπτωση Εγγραφής Χρήστη**

Παραπάνω παρουσιάζεται η ακολουθία της εγγραφής χρήστη στο **Portal** του ηλεκτρονικού καταστήματος. Αφού πατήσει στο κουμπί Εγγραφή, εμφανίζεται η αντίστοιχη σελίδα στην οποία πρέπει να εισάγει τις απαιτούμενες λεπτομέρειες. Τότε καλείται η μέθοδος **CreateUser()** για να δημιουργηθεί ο χρήστης και αποθηκεύονται τα στοιχεία του νέου χρήστη στη βάση δεδομένων των χρηστών.

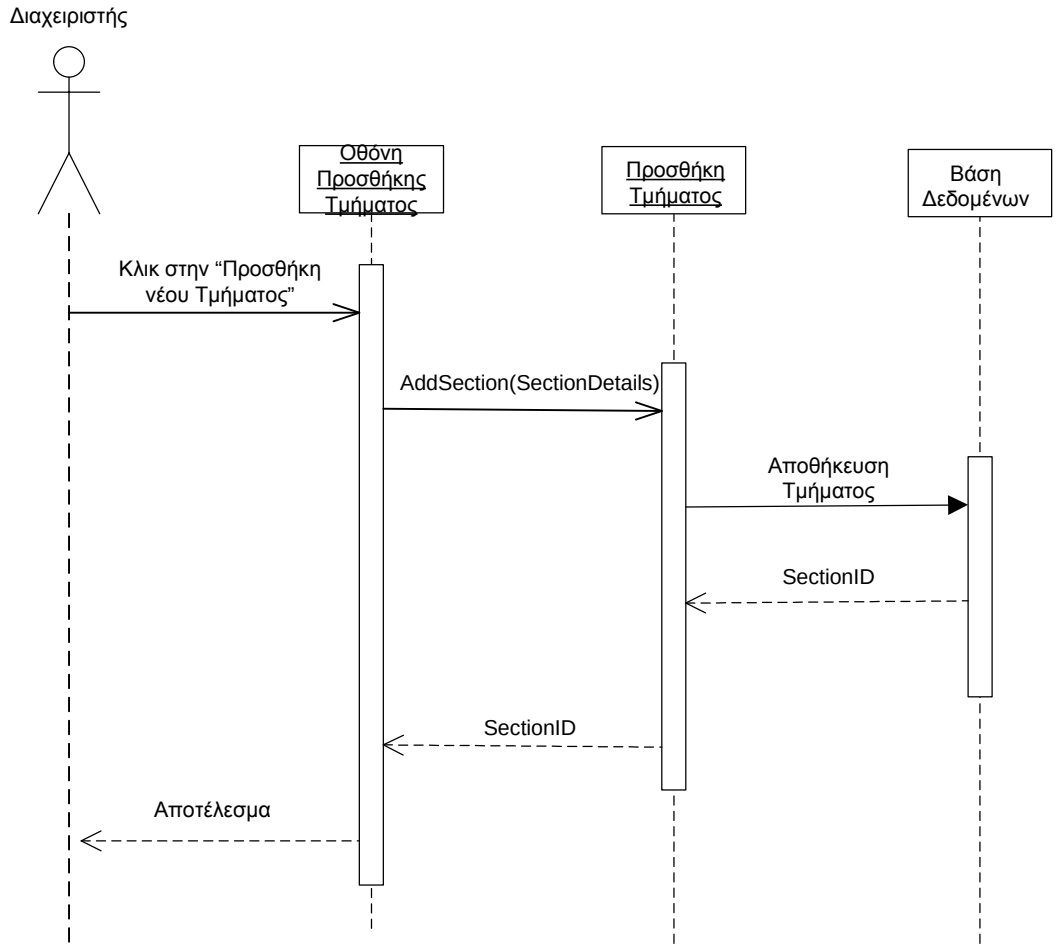
Από τη βάση δεδομένων προκύπτει ένα **userID** που θα τον συνοδεύει πάντα και δημιουργούνται και τα σχετικά μηνύματα επιτυχούς εγγραφής στο σύστημα. Πλέον, ο χρήστης αποτελεί Πελάτη του συστήματος και από εδώ και πέρα θα ακολουθείται η ακολουθία σύνδεσης πελατών, που παρουσιάζεται στο επόμενο διάγραμμα.




**Εικόνα 60 Διάγραμμα Αλληλουχίας για τη περίπτωση Σύνδεσης Εγγεγραμμένου Πελάτη**

Εδώ πλέον και σε αντίθεση με το προηγούμενο διάγραμμα, πραγματοποιείται πιστοποίηση του χρήστη με βάση το **UserID** και το κωδικό του, εφόσον πρόκειται για εγγεγραμμένο χρήστη. Ο έλεγχος του χρήστη πραγματοποιείται με κλήση προς τη βάση δεδομένων χρηστών, η οποία επιστρέφει μήνυμα με τα στοιχεία του χρήστη για να γίνε η πιστοποίησή του. Το αποτέλεσμα της διαδικασίας πιστοποίησης εμφανίζεται στην οθόνη σύνδεσης και επιδεικνύεται και το σχετικό μήνυμα.

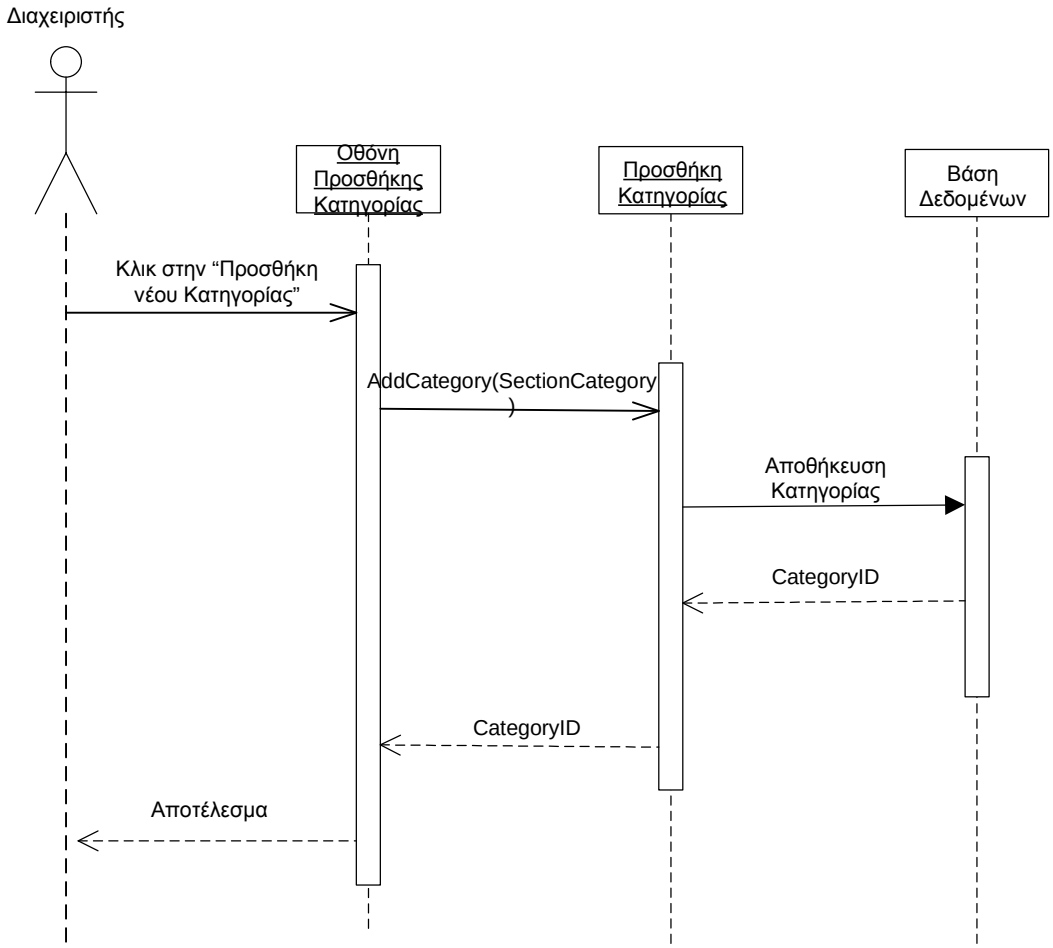
Ο δεύτερος εκ των **actors** του συστήματος είναι ο Διαχειριστής ο οποίος είναι υπεύθυνος για την ενημέρωση των τμημάτων, των κατηγοριών, των φιρμών και των προϊόντων του ηλεκτρονικού καταστήματος.



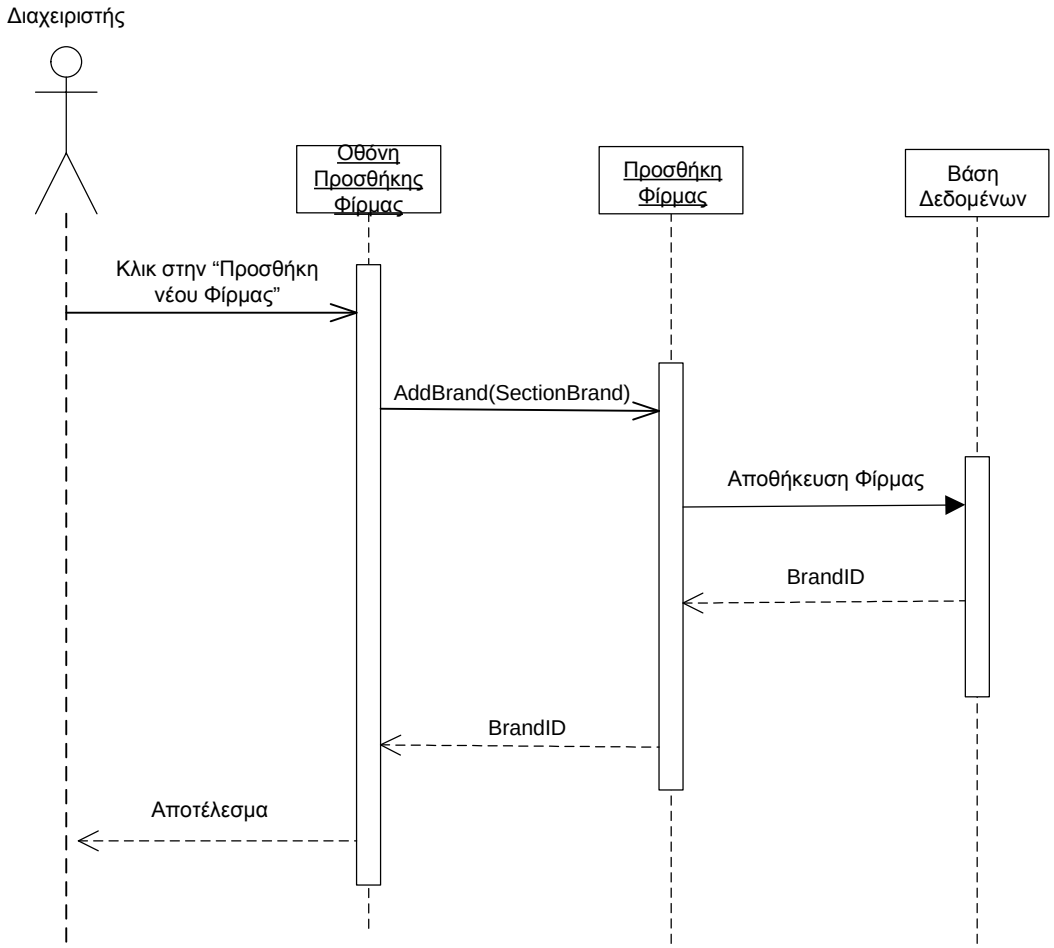

**Εικόνα 61 Διάγραμμα Αλληλουχίας για τη περίπτωση εισαγωγής νέου Τμήματος**

Παραπάνω παρουσιάζεται η αλληλουχία ενημέρωσης του συστήματος με ένα νέο τμήμα προϊόντων. Ο Διαχειριστής μέσω της κατάλληλης οθόνης διαχείρισης καλεί τη μέθοδο **AddSection** για να προσθέσει το νέο τμήμα προϊόντων, παρέχοντας στη μέθοδο τα απαραίτητα στοιχεία του τμήματος. Το τμήμα δημιουργείται και ενημερώνεται η σχετική βάση δεδομένων, η οποία με τη σειρά της παράγει ένα **SectionID** για το νέο τμήμα που θα το συνοδεύει πάντα. Τέλος η οθόνη δημιουργίας νέου τμήματος ενημερώνει τον διαχειριστή για το αποτέλεσμα.

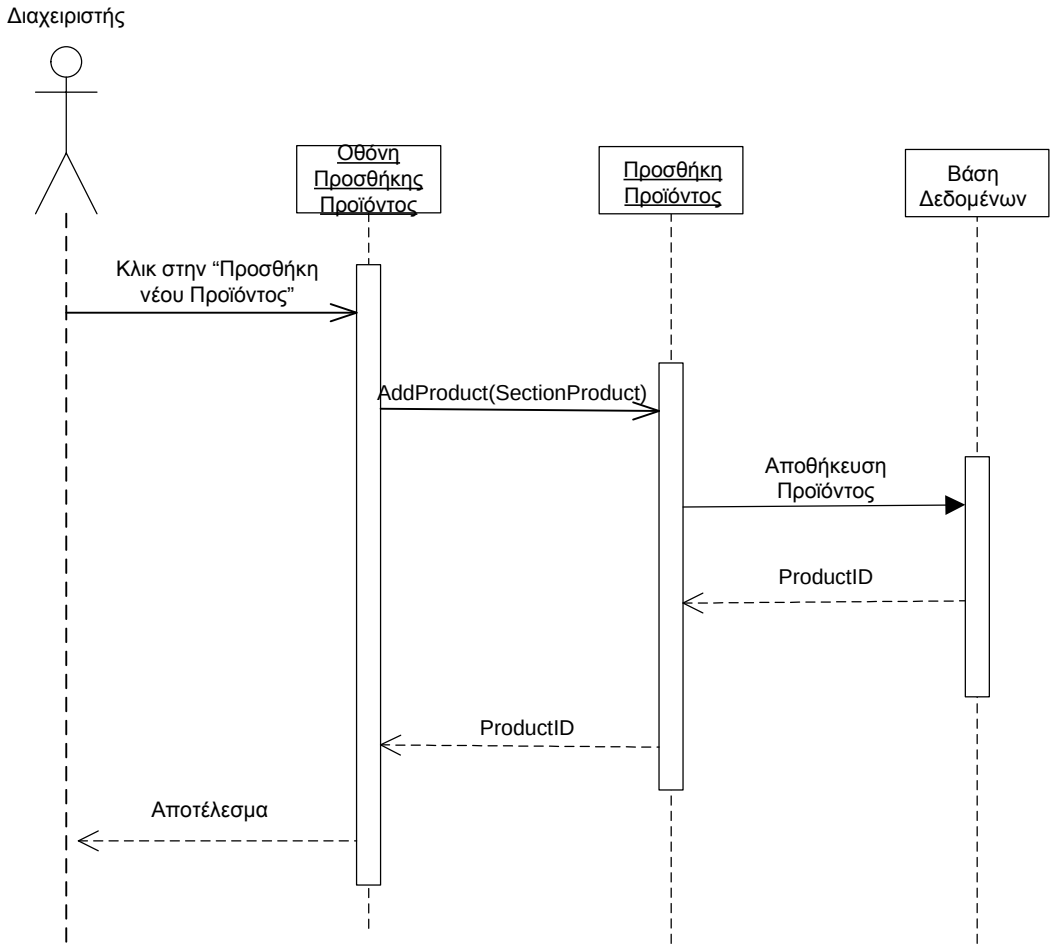
Ομοίως δημιουργούνται και νέες κατηγορίες, φίρμες και προϊόντα. Τα τρία αυτά διαγράμματα παρατίθενται πιο κάτω. Να σημειωθεί ότι δεν υπάρχει μονοσήμαντη σχέση μεταξύ των τμημάτων, κατηγοριών, φιρμών, και προϊόντων. Δηλαδή ένα προϊόν μπορεί να εμπίπτει σε περισσότερα από ένα τμήματα ή σε περισσότερες της μιας κατηγορίες. Αυτό καθορίζεται ουσιαστικά ως παράμετρος της μεθόδου **AddCategory**, **AddBrand**, **AddProduct** κ.λ.π ανά περίπτωση.




**Εικόνα 62 Διάγραμμα Αλληλουχίας για τη περίπτωση εισαγωγής νέας Κατηγορίας**



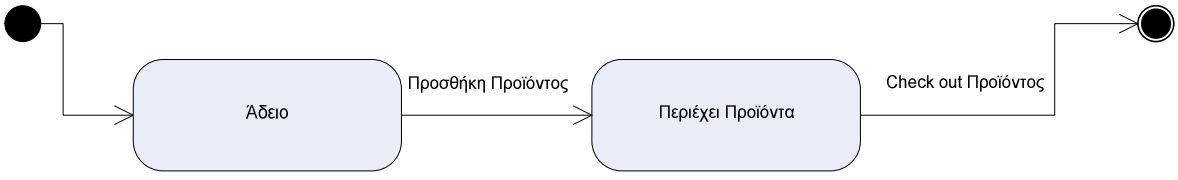

**Εικόνα 63 Διάγραμμα Αλληλουχίας για τη περίπτωση εισαγωγής νέας Φίρμας**




**Εικόνα 64** **Διάγραμμα Αλληλουχίας για τη περίπτωση εισαγωγής νέου Προϊόντος**

## 4.1.5 Διάγραμμα Καταστάσεων

Το διάγραμμα καταστάσεων δείχνει τη μετάβαση μεταξύ καταστάσεων ενός αντικειμένου της κλάσης Καλάθι. Ουσιαστικά ένα Καλάθι μπορεί είτε να είναι άδειο είτε να περιέχει προϊόντα.




**Εικόνα 65** **Διάγραμμα Καταστάσεων για Καλάθι Αγορών**

Με το συμβάν Προσθήκη Προϊόντος το Καλάθι μεταβαίνει από τη κατάσταση Άδειο στη Κατάσταση Περιέχει Προϊόντα**,** ενώ με το συμβάν **Checkout** Προϊόντος το καλάθι επανέρχεται στη αρχική του κατάσταση**.** Ως **check out** μπορεί να είναι η αφαίρεση των προϊόντων από τον ίδιο το χρήστη επειδή π.χ**.** άλλαξε γνώμη ή η αγορά των προϊόντων**.**

# Κεφάλαιο 5: Εργαλεία ανάπτυξης λογισμικού CASE

## *5.1 Τι είναι τα εργαλεία CASE*

Ως **Computer Aided Software Engineering (CASE)** ονομάζεται η διαδικασία της επιστημονικής εφαρμογής μιας σειράς εργαλείων και μεθόδων με χρήση υπολογιστή για τη παραγωγή ενός συστήματος λογισμικού υψηλής ποιότητας, χωρίς σφάλματα και με υψηλή διατηρησιμότητα. **[19]**

Βασικό χαρακτηριστικό των εργαλείων **CASE** είναι ότι επιτρέπουν στους σχεδιαστές, προγραμματιστές, δοκιμαστές και **manager** να μοιράζονται μια κοινή άποψη για το στάδιο εξέλιξης και ανάπτυξης ενός συστήματος κατασκευής λογισμικού.

Η διαδικασία **CASE** προέρχεται από τη δεκαετία του **1970**, όταν οι εταιρίες υπολογιστών άρχισαν να δανείζονται ιδέες από τις διαδικασίες κατασκευής υλικού **(hardware)** και να τις εφαρμόζουν στην ανάπτυξη λογισμικού **(software)**, κάτι που γενικά έχει θεωρηθεί ως ανεπαρκής διαδικασία. Τα εργαλεία **CASE** αρχικά υποστήριζαν το δομημένο προγραμματισμό και άλλες παρόμοιες μεθόδους οργανωμένης ανάπτυξης λογισμικού, αλλά πλέον, συμπεριλαμβάνουν οπτικά εργαλεία προγραμματισμού καθώς και τις διαδικασίες αντικειμενοστραφούς προγραμματισμού.

Τα πρώτα προϊόντα **CASE** δεν έφεραν τα αναμενόμενα **(**και υποσχόμενα**)** αποτελέσματα παραγωγικότητας. Οι κυριότεροι λόγοι που συνέβη αυτό είναι:

⇒ Το μεγάλο μέγεθος των συστημάτων λογισμικού.

Τα μεγάλα συστήματα γενικά αντιμετωπίζουν προβλήματα διαχείρισης σημαντικής πολυπλοκότητας, τόσο όσον αφορά το υπό ανάπτυξη προϊόν, όσο επίσης και την ίδια διαδικασία ανάπτυξης. Έχει επισημανθεί ότι η τεχνολογία **CASE** μπορεί να παρέχει κάποια βοήθεια, αλλά δεν μπορεί να λύσει τα προβλήματα αυτά, διότι, πολύ απλά, η λύση δεν είναι ακόμη γνωστή.

⇒ Η ασυμβατότητα εργαλείων.

Τα υπάρχοντα προϊόντα **CASE** αποτελούν **«**νήσους αυτοματοποίησης**»** διαφόρων εργασιών, άλλων σε μεγαλύτερη και άλλων σε μικρότερη έκταση. Η συνεργασία μεταξύ αυτών των διαφορετικών προϊόντων δεν είναι εύκολη και συνήθως είναι περιορισμένη. Το γεγονός αυτό περιορίζει την ευρεία εφαρμογή της τεχνολογίας.

⇒ Η ελλιπής εκπαίδευση και διαδικασία προσαρμογής.

Οι χρήστες των εργαλείων **CASE**, συχνά υποτιμούν το ρόλο και τη σημασία της εκπαίδευσης και καθώς και της προσαρμογής, εργασιών δηλαδή που

είναι ιδιαίτερα ουσιώδεις για την αποτελεσματική εκμετάλλευση των εργαλείων. Όταν μάλιστα αυτές παραλείπονται σε μεγάλο βαθμό, η χρήση της τεχνολογίας **CASE** μπορεί να έχει αντίστροφα αποτελέσματα.

Σήμερα, μπορούμε να καταγράψουμε τρία γενικά επίπεδα της τεχνολογίας των εργαλείων **CASE:**

⇒ Επίπεδο παραγωγής λογισμικού.

Εδώ συμπεριλαμβάνονται οι εργασίες που εκτελούνται κατά την ανάπτυξη λογισμικού, όπως ο καθορισμός των απαιτήσεων, η ανάλυση, η ανάπτυξη του κώδικα και ο έλεγχος. Τα εργαλεία αυτά είναι από τα πρώτα που εμφανίστηκαν και συνεπώς είναι, σήμερα, από τα πιο ώριμα εργαλεία **CASE.**

⇒ Επίπεδο διαχείρισης.

Στο επίπεδο αυτό κατατάσσεται η μοντελοποίηση και διαχείριση των διαφόρων φάσεων της ανάπτυξης λογισμικού. Αποτελεί, δηλαδή, υπερσύνολο της προηγούμενης κατηγορίας και χρησιμοποιεί υπηρεσίες που παρέχονται απ' αυτή για την υποστήριξη συγκεκριμένων εργασιών μέσα σε κάποια φάση της ανάπτυξης.

⇒ Επίπεδο M**eta–CASE.**

Στο επίπεδο αυτό ανήκουν εργαλεία τα οποία χρησιμοποιούνται για τη γέννηση **(**δημιουργία**)** περιβαλλόντων **CASE** των δυο παραπάνω επιπέδων. Ένας μικρός αριθμός τέτοιων εργαλείων είναι διαθέσιμος σήμερα στην αγορά. Η ρευστότητα των πραγμάτων στις προσεγγίσεις αλλά και στα περιβάλλοντα ανάπτυξης λογισμικού καθιστά τέτοιες ιδέες δύσκολα αποδεκτές από τους κατασκευαστές λογισμικού.

## *5.2 Ταξινόμηση των εργαλείων*

Ο **Alfonso Fuggetta** ταξινόμησε τις διαδικασίες **CASE** σε **3** κατηγορίες**[20]:**

1. Εργαλεία **(Tools).** Υποστηρίζουν μόνο συγκεκριμένες εργασίες **(tasks)** κατά τη διαδικασία ανάπτυξης λογισμικού.
2. Πάγκοι εργασίας **(Workbenches).** Υποστηρίζουν μόνο μία ή μερικές δραστηριότητες **(activities).**
3. Περιβάλλοντα **(Environments).** Υποστηρίζουν ένα ευρύ τμήμα της διαδικασίας ανάπτυξης λογισμικού.

Τα **Workbenches** και τα **Environments** εμπορικά κυκλοφορούν ως σουίτες εργαλείων. Συνεπώς, τα εργαλεία μπορούν να εμφανίζονται ως αυτόνομα προϊόντα ή ως τμήματα των **Workbenches** ή των **Environments.**

Τα εργαλεία **CASE** είναι μια κατηγορία λογισμικού που αυτοματοποιεί πολλές από τις δραστηριότητες που εμπλέκονται σε διάφορες φάσεις του κύκλου ζωής. Για παράδειγμα, κατά τον καθορισμό των λειτουργικών απαιτήσεων της προτεινόμενης εφαρμογής, τα εργαλεία μπορούν να χρησιμοποιηθούν για την ανάπτυξη γραφικών μοντέλων οθονών εφαρμογών, που βοηθούν τους τελικούς χρήστες να αντιληφθούν πώς θα είναι η τελική μορφή μιας εφαρμογής μετά από την ανάπτυξή της.

Στη συνέχεια, οι σχεδιαστές του συστήματος μπορούν να χρησιμοποιήσουν τα αυτοματοποιημένα εργαλεία σχεδιασμού για τη μετατροπή των προτυποποιημένων λειτουργικών απαιτήσεων σε λεπτομερή πλάνα σχεδιασμού. Οι προγραμματιστές μπορούν να χρησιμοποιήσουν στη συνέχεια αυτοματοποιημένες γεννήτριες κώδικα, για την μετατροπή των πλάνων σχεδιασμού σε κώδικα. Τα αυτοματοποιημένα εργαλεία μπορούν να χρησιμοποιηθούν προσθετικά, ή μεμονωμένα. Για παράδειγμα, τα εργαλεία προτυποποίησης θα μπορούσαν να χρησιμοποιηθούν για να καθορίσουν τις απαιτήσεις εφαρμογής που παρέχονται αργότερα στους τεχνικούς οι οποίοι μετατρέπουν τις απαιτήσεις σε λεπτομερή σχέδια με τον παραδοσιακό τρόπο, χρησιμοποιώντας διαγράμματα και κείμενα, χωρίς τη βοήθεια αυτοματοποιημένων λογισμικό σχεδιασμού.

Επιπλέον, τα εργαλεία **CASE** ταξινομούνται και με βάση την λειτουργία που επιτελούν. Πιο συγκεκριμένα διακρίνουμε:

1. Εργαλεία διαχείρισης, ελέγχου και συντήρησης του έργου.

   Συμπεριλαμβάνονται εργαλεία σχεδιασμού του έργου **(Project Planning tools)**, εργαλεία μοντελοποίησης και διαχείρισης των επιχειρηματικών διεργασιών **(Process Modeling and Management tools)**, εργαλεία ανάλυσης κινδύνου **(Risk Analysis tools)**, εργαλεία διοίκησης έργου **(Project Management tools)**, εργαλεία μετρικών λογισμικού **(Metrics tools)**, εργαλεία διασφάλισης ποιότητας, Εργαλεία διαχείρισης βάσεων δεδομένων λογισμικού **(Software Database Management tools)**, εργαλεία διοίκησης σχηματισμών λογισμικού **(Software Configuration Management tools)**. εργαλεία ενοποίησης και ελέγχου **(Integration and Testing tools)**, εργαλεία διαχείρισης ελέγχου **(Test Management tools)**, εργαλεία ελέγχου συστημάτων πελάτη – εξυπηρετητή **(Client – Server Testing tools)**, εργαλεία ανακατασκευής λογισμικού **(Re–engineering tools)**.

2. Εργαλεία προδιαγραφής απαιτήσεων, ανάλυσης και σχεδίασης λογισμικού.

   Συμπεριλαμβάνονται εργαλεία διαχείρισης προδιαγραφών **(Requirements Management tools**, εργαλεία ανάλυσης και σχεδίασης **(Analysis and Design tools)**

3. Εργαλεία κατασκευής και γέννησης προγραμμάτων.

   Εργαλεία προγραμματισμού **(Programming tools)** και εργαλεία σχεδίασης και ανάπτυξης διεπαφών **(Interface Design and Development tools)**.

4. Εργαλεία ανάλυσης και ελέγχου του πηγαίου κώδικα.

Εργαλεία στατικής ανάλυσης **(Static Analysis tools)** και εργαλεία δυναμικής ανάλυσης **(Dynamic Analysis tools).**

**5.** Εργαλεία προσομοίωσης και μοντελοποίησης**.**

Εργαλεία προσομοίωσης **(Simulation tools)** και εργαλεία προτυποποίησης **(Prototyping tools).**

**6.** Εργαλεία τεκμηρίωσης **(Documentation tools).**

## *5.3 Λειτουργίες εργαλείων CASE UML*

Τα αντίστοιχα εργαλεία **CASE** για τη **UML** δύναται να παρέχουν πληθώρα λειτουργιών που ουσιαστικά τα κατατάσσουν και σε κάποιες από τις παραπάνω γενικότερες κατηγορίες **CASE** εργαλείων**.**

Σε κάθε περίπτωση ένα πλήρες **CASE** εργαλείο για τη **UML** Θα πρέπει να διαθέτει λειτουργίες όπως**:**

⇒ Να επιτρέπει το σχεδιασμό διαγραμμάτων **UML** με τήρηση όλων των κανόνων σχετικά με τα συστατικά των μοντέλων**.**

⇒ Κοινή αποθήκευση δεδομένων με την έννοια της ύπαρξης ενός κοινού **repository** για τμήματα πληροφορίας**,** έτσι ώστε αλλάζοντας για παράδειγμα το όνομα μιας κλάσης σε ένα διάγραμμα**,** αυτομάτως να πραγματοποιείται η αντίστοιχη αλλαγή σε όλα τα υπόλοιπα διαγράμματα**.**

⇒ Εύκολη πλοήγηση και υποστήριξη εισαγωγής ***/*** εξαγωγής στοιχείων των διαγραμμάτων και μεταξύ αυτών**.**

⇒ Ταυτόχρονη υποστήριξη πολλών χρηστών ώστε να μπορούν ταυτόχρονα να επεξεργάζονται ένα μοντέλο**,** χωρίς η εργασία του ενός να επηρεάζει την εργασία του άλλου**.**

⇒ Παραγωγή κώδικα από το διάγραμμα**.** Τουλάχιστον η παραγωγή του σκελετού του κώδικα ώστε να είναι δυνατή η λεπτομερής συμπλήρωση του κατά τη διάρκεια της υλοποίησης**.**

⇒ Αντίστροφη ανάπτυξη**,** δηλαδή από το κώδικα να μπορεί να εξάγει μοντέλα**.**

⇒ Η συμβατότητα με άλλα εργαλεία και περιβάλλοντα**,** όπως οι μεταγλωττιστές **(compilers)** και οι συντάκτες κώδικα **(code editors).**

⇒ Διαλειτουργικότητα των παραγόμενων μοντέλων ώστε να μπορούν να εξαχθούν και να εισαχθούν σε άλλα εργαλεία μοντελοποίησης **&** ανάπτυξης**.**

## *5.4 Πλεονεκτήματα & Μειονεκτήματα*

Τα εργαλεία **CASE**, αν και υποσχέθηκαν πολλά, στην αρχή τουλάχιστον δεν είχαν τα αναμενόμενα αποτελέσματα. Ωστόσο καθώς η κοινότητα του λογισμικού ωριμάζει, τα πράγματα γίνονται ολοένα και καλύτερα. Η σωστή εφαρμογή των εργαλείων **CASE** μπορεί να οδηγήσει σε μείωση του κόστους και του χρόνου ανάπτυξης του λογισμικού, καθώς και στη βελτίωση της ποιότητάς του. Επίσης, μπορεί να επιφέρει τον επαναπροσδιορισμό κάποιων εργασιών ανάπτυξης λογισμικού, έτσι ώστε αυτές να είναι περισσότερο αποτελεσματικές.

Πιο συγκεκριμένα, μερικά από τα πλεονεκτήματα της χρήσης των εργαλείων **CASE** περιγράφονται στις επόμενες παραγράφους.

Ως γνωστόν τα σημερινά λογισμικά, ιδιαίτερα τα εμπορικά προϊόντα, χαρακτηρίζονται από μεγάλο βαθμό πολυπλοκότητας και προκύπτουν ως αποτέλεσμα της συνεργασίας μιας ολόκληρης ομάδας προγραμματιστών που ο καθένας αναλαμβάνει τμήμα της συνολικής εργασίας. Τα εργαλεία **CASE** έχουν τη δυνατότητα για να βελτιώνουν την επικοινωνία και συνεργασία μεταξύ των μελών της ομάδας ανάπτυξης λογισμικού, καθώς έχουν σχεδιαστεί ως συνεργατικά εργαλεία.

Επιπλέον, με τη χρήση των εργαλείων **CASE** διευκολύνεται εξαιρετικά και ο έλεγχος ποιότητας των μοντέλων που δημιουργούνται καθώς εξασφαλίζεται η συνοχή, η πληρότητα και η συμμόρφωση με τα πρότυπα ανάπτυξης λογισμικού και συγκεκριμένα με τα πρότυπα της **UML 2.0**.

Ως γνωστόν, μικρά σφάλματα στη λογική των προγραμμάτων μπορεί να έχουν τεράστιες συνέπειες για τον τελικό χρήστη. Με τα εργαλεία **CASE** ένα σημαντικό μέρος της επίλυσης των προβλημάτων της ανάπτυξης εφαρμογών και συντήρησης πραγματοποιείται πριν καν την πραγματική υλοποίηση των προγραμμάτων ή κατά τα πρώτα βήματα υλοποίησης τους.

Με τα εργαλεία **CASE** μεταβάλλεται σημαντικά και ο χρόνος που καταναλώνεται σε κάθε φάση του κύκλου ζωής ανάπτυξης του λογισμικού. Μεγάλο μέρος του κώδικα δημιουργείται πλέον αυτόματα, με λεπτομερείς προδιαγραφές, ενώ και λόγω της αντικειμενοστραφούς λογικής προγραμματισμού που έχει καθιερωθεί, μεγάλα τμήματα του κώδικα επαναχρησιμοποιούνται. Αυτά συνεπάγονται μείωση του χρόνου παραγωγής και των αντίστοιχων πότων που απαιτείται να δαπανηθούν.

Βέβαια στον αντίποδα, υπάρχουν και κάποια μειονεκτήματα σε σχέση με τη χρήση των εργαλείων **CASE**, όπως άλλωστε συμβαίνει και σε κάθε περίπτωση χρήσης ενός εξειδικευμένου εργαλείου. Τα σημαντικότερα από αυτά είναι τα παρακάτω.

Ένα από τα μειονεκτήματα είναι ότι τα εργαλεία **CASE** δεν εμποδίζουν κατ' ανάγκην τους ανθρώπους να κάνουν κακά σχέδια. Το τελικό αποτέλεσμα ενός εργαλείου **CASE** μπορεί να φαίνεται ωραία, αλλά αν έχει σχεδιαστεί ανεπαρκώς, τότε ανεπαρκές θα είναι και το μοντέλο και το λογισμικό.

Ένα άλλο μειονέκτημα έχει να κάνει με την εκπαίδευση. Εάν ένα πρόσωπο είναι νέο στη χρήση εργαλείων **CASE**, μπορεί να χρειαστεί πολύς χρόνο για να αντιληφθεί το

τρόπο λειτουργίας τους. Επιπλέον, πιθανότατα θα χρειαστεί και καθοδήγηση από άλλα άτομα που χρησιμοποίησαν το εργαλείο πριν. Αυτό σημαίνει ότι απαιτείται κάποιος χρόνος και κόστος εκπαίδευσης.

Επιπλέον, η ενσωμάτωση ενός τέτοιου εργαλείου στην υφιστάμενη παραγωγική διαδικασία μιας εταιρείας δεν είναι κάτι το αμελητέο. Πολλές φορές απαιτείται παραμετροποίηση άλλων συστημάτων και προσαρμογή των διαδικασιών που ακολουθούνται.

Τέλος, ένα άλλο μειονέκτημα είναι το κόστος χρήσης αυτών των εργαλείων. Το κόστος του διαφέρει αναλόγως της λειτουργικότητας τους. Εργαλεία που απλά χρησιμοποιούνται για τη σχεδίαση μοντέλων, δεν έχουν ή έχουν μηδαμινό κόστος. Όμως εργαλεία τα οποία εξάγουν κώδικα και γενικά ενσωματώνουν πολλές από τις λειτουργίες της προηγούμενης παραγράφου, μπορεί να κοστίζουν ένα σεβαστό πόσό.

Σε κάθε περίπτωση πάντως, οι δυνατότητες των εργαλείων **CASE** συνεχώς βελτιώνονται κάτι που οδηγεί σχεδόν νομοτελειακά στην υιοθέτηση τους, ειδικά σε περιπτώσεις μεγάλων οίκων κατασκευής λογισμικού.

## *5.4 CASE εργαλεία για εξαγωγή κώδικα από τη UML*

Σε αυτό το τμήμα επιχειρείται μια καταγραφή των πιο ευρέως διαδεδομένων εργαλείων **CASE** που υποστηρίζουν τη **UML 2.0** και έπειτα, καθώς και η παρουσίαση των πιο σημαντικών στοιχείων τους.

### 5.4.1 Rational Software Architect

Το εργαλείο **Rational Software Modeler** της **IBM,**δεν είναι το πιο ευρέως χρησιμοποιούμενο εργαλείο, αλλά έχει μια μακρά ιστορία στη μοντελοποίηση, που ξεκινάει στις αρχές του **1990** με το έργο των "πατεράδων" της **UML, Grady Booch, James Rumbaugh** και **Ivar Jacobson.** Όπως έχει αναφερθεί και στο πρώτο κεφάλαιο, οι τρεις τους συνδύασαν ανταγωνιζόμενες προσεγγίσεις μοντελοποίησης για να σχηματίσουν αυτό που τελικά ονομάστηκε Ενοποιημένη Γλώσσα Μοντελοποίησης **UML.**

Το πρώτο εργαλείο για οπτική μοντελοποίηση της **Rational Software,** ήταν το **Rational Rose,** ένα αυτόνομο εργαλείο μοντελοποίησης που ενσωματώνει στ**ο** επίπεδο **API (Application Programming Interface -** Διεπαφή προγραμματισμού εφαρμογών**) IDEs (Integrated Development Environments -** Ολοκληρωμένα Περιβάλλοντα Ανάπτυξης**)** άλλων κατασκευαστών, για να υποστηρίξει μια ποικιλία γλωσσών προγραμματισμού και τεχνολογιών υλοποίησης.

Ενώ το **Rational Rose** ήταν ένα σημαντικό βήμα στην υιοθέτηση της **MDD (Model-Driven Development –** "μοντέλοστραφούς" ανάπτυξης**)** ανάπτυξης, από τους κατασκευαστές λογισμικού, διαπιστώθηκε ότι μόνο ένας μικρός αριθμός

προγραμματιστών χρησιμοποιούσαν τη μοντελοποίηση σε τακτική βάση. Ένα βασικό πρόβλημα που είχε εντοπιστεί ήταν ότι οι προγραμματιστές δεν ήθελαν να διαφοροποιηθούν από το **IDE** που χρησιμοποιούσαν καθημερινά. Ήθελαν μεν την οπτική μοντελοποίηση, αλλά ενσωματωμένη - όχι με τα **IDE**, αλλά καλύτερα εντός των **IDE**.

**Η Rational** ανταποκρίθηκε στην ανάγκη αυτή και το **2002** με το λογισμικό **IBM Rational XDE**, παρείχε ένα περιβάλλον ανάπτυξης για τις τότε αναδυόμενες τεχνολογίες προγραμματισμού: **Java** και **Microsoft**.NET. Το **IBM Rational XDE** χαρακτηρίστηκε ως η επόμενη γενιά του **Rational Rose** - όχι μια νέα έκδοσή του (εξ ου και η αλλαγή του ονόματος), και όχι κατ 'ανάγκη ένας αντικαταστάτης για το **Rose** (καθώς το **IBM Rational XDE** σκόπιμα περιορίστηκε να υποστηρίζει μόνο ένα επιλεγμένο αριθμός **IDE** και τεχνολογιών υλοποίησης). Ωστόσο, στη πορεία προστέθηκαν εργαλεία και δυνατότητες οπότε η **Rational** έφτασε στα όρια αυτού του είδους ενσωμάτωσης εργαλείων.

Για την επόμενη γενιά προϊόντων **MDD**, οι λειτουργίες χτίστηκαν πάνω στην πλατφόρμα **Eclipse** [21] για να σχηματίσουν ένα πιο ολοκληρωμένο εργαλείο. Τα **IBM Rational Software Architect** [22], **IBM Rational Software Modeler** και **IBM Rational Systems Developer**, είναι το αποτέλεσμα αυτών των αλλαγών, που θέσανε και νέες βάσεις για τα εργαλεία μοντελοποίησης, σχεδιασμού και ανάπτυξης, προσδίδοντας τους πιο ολοκληρωμένα χαρακτηριστικά.

Ακολουθούν τα χαρακτηριστικά του εργαλείου με τις γλώσσες στις οποίες εξάγεται κώδικας από διάγραμμα **UML** και τις γλώσσες από τις οποίες μπορεί να εξαχθεί διάγραμμα **UML**.

**Πίνακας 6 Χαρακτηριστικά Rational Software Architect**

| Όνομα | UML è Γλώσσα | Γλώσσα è UML | Πλατφόρμα | Άδεια χρήσης |
|---|---|---|---|---|
| **Rational Software Architect** | **Java, C#, C++, EJB, WSDL, XSD, CORBA IDL, SQL** | **Java, C++,.NET** | **Eclipse** | Εμπορική |

## 5.4.2 Enterprise Architect

Το **Enterprise Architect** της **Sparx Systems** [23], είναι ένα εργαλείο οπτικής μοντελοποίησης που υποστηρίζει το σχεδιασμό και την κατασκευή συστημάτων λογισμικού, τη μοντελοποίηση επιχειρηματικών διαδικασιών και τη μοντελοποίηση βιομηχανικών διαδικασιών. Χρησιμοποιείται από επιχειρήσεις και οργανισμούς, όχι μόνο για την απεικόνιση της αρχιτεκτονικής των συστημάτων τους, αλλά και για την επεξεργασία των παραγόμενων μοντέλων σε όλο τον κύκλο ζωής της ανάπτυξής του συστήματος.

Γενικά υποστηρίζει διαχείριση απαιτήσεων, μοντελοποίηση και ανάλυση επιχειρηματικών διαδικασιών, προσομοίωση, ανάπτυξη συστήματος, διαχείριση δοκιμών, ανάλυση οπτικής εκτέλεσης, μοντελοποίηση δεδομένων, διαχείριση έργων, διαχείριση αλλαγών, συνεργατική ανάπτυξη και συνεργασία, αρχιτεκτονικές προσανατολισμού σε υπηρεσίες, ενσωμάτωση με άλλα εργαλεία, **reporting.**

Αρχικά κυκλοφόρησε το **2000** καλύπτοντας την έκδοση **UML 1.1** και παροδικά αναπτύχθηκε και εμπλουτίστηκε, καλύπτοντας πλέον την έκδοση **UML 2.4.1.**

**Πίνακας 7 Χαρακτηριστικά Enterprise Architect**

| Όνομα | UML è Γλώσσα | Γλώσσα è UML | Πλατφόρμα | Άδεια χρήσης |
|---|---|---|---|---|
| **Enterprise Architect** | **ActionScript, C, C#, C++, Delphi, Java, PHP, Python, Visual Basic, Visual Basic .NET, DDL, EJB, XML Schema, Ada, VHDL, Verilog, WSDL, BPEL, Corba IDL** | **ActionScript, C, C#, C++, Delphi, Java, PHP, Python, Visual Basic, Visual Basic .NET, DDL, XML Schema, WSDL** | **Eclipse, Visual Studio, TcSE** | Εμπορική |

## 5.4.3 Magic Draw UML

Το **MagicDraw [24]** είναι ένα εργαλείο μοντελοποίησης **UML, SysML (Systems Modeling Language), BPMN (Business Process Model and Notation),** και **UPDM (Unified Profile for DoDAF/MODAF)** που υποστηρίζει συνεργατική ανάπτυξη. Σχεδιασμένο για επιχειρηματικούς αναλυτές, αναλυτές λογισμικού, προγραμματιστές και μηχανικούς εξασφάλισης ποιότητας, αυτό το δυναμικό και ευέλικτο εργαλείο διευκολύνει την ανάλυση και το σχεδιασμό των αντικειμενοστραφών συστημάτων και βάσεων δεδομένων.

Κυκλοφόρησε το **1998** από την εταιρεία **No Magic,** με υποστήριξη για **UML 1.1** και πλέον υποστηρίζει **UML 2.3.**

**Πίνακας 8 Χαρακτηριστικά Magic Draw UML**

| Όνομα | UML è Γλώσσα | Γλώσσα è UML | Πλατφόρμα | Άδεια χρήσης |
|---|---|---|---|---|
| **Magic Draw** | **Java, C++, C#,** | **Java, C++, C#,** | **Eclipse, EMF,** | Εμπορική |

| | | | | |
|---|---|---|---|---|
| **UML** | **CIL, CORBA IDL, DDL, EJB, XML Schema, WSDL** | **CIL, CORBA IDL, DDL, EJB, XML Schema, WSDL** | **NetBeans** | |

### 5.4.4 Modelio

Το **Modelio** [25] είναι ένα εργαλείο ανοιχτού κώδικα που αναπτύχθηκε από την **Modeliosoft 2009** και πλέον υποστηρίζει τα πρότυπα των **UML 2.0** και το **BPMN 2.0.** Το **Modelio** είναι κυρίως ένα περιβάλλον μοντελοποίησης, που υποστηρίζει μια ευρεία γκάμα μοντέλων και διαγραμμάτων, και παρέχει υποστήριξη στη δημιουργία διαγραμμάτων καθώς και έλεγχο ορθότητας. Στα πλαίσια της **UML 2.0** υποστηρίζει τα κυριότερα διαγράμματα όπως κλάσεων, πακέτων, αντικειμένων, περιπτώσεων Χρήσης, αλληλουχίας, δραστηριοτήτων, επικοινωνίας, καταστάσεων και σύνθετης δομής.

Επιπλέον, χαρακτηρίζεται από τη χρηστικότητα του, ενσωματώνοντας λειτουργίες για τη δημιουργία στοιχείων με απλό **drag & drop** σε διαγράμματα ή μοντέλα, ή τη χρήση **wizards** για τη δημιουργία πιο σύνθετων στοιχείων. Στο παρακάτω πίνακα παρουσιάζονται τα σημαντικότερα χαρακτηριστικά του.

**Πίνακας 9 Χαρακτηριστικά Modelio**

| Όνομα | UML è Γλώσσα | Γλώσσα è UML | Πλατφόρμα | Άδεια χρήσης |
|---|---|---|---|---|
| **Modelio** | **Java, C++, C#, XSD, WSDL, SQL** | **Java, C++, C#** | **Eclipse, EMF** | Ελεύθερη υπό **GPL v3** |

### 5.4.5 Visual Paradigm

Το **Visual Paradigm** για **UML (VP-UML)** [26] είναι ένα **UML CASE** εργαλείο που υποστηρίζει **UML 2.0, SysML** και **BPMN,** πρότυπα που καθορίζονται από το **OMG.** Εκτός από την υποστήριξη μοντελοποίησης, παρέχει και δυνατότητες **reporting** και παραγωγής κώδικα από τα διαγράμματα, ενώ υποστηρίζει και την συνεργατική ανάπτυξη λογισμικού καθώς διαθέτει **κ**εντρικό αποθετήριο στοιχείων.

**Πίνακας 10 Χαρακτηριστικά Visual Paradigm for UML**

| Όνομα | UML è Γλώσσα | Γλώσσα è UML | Πλατφόρμα | Άδεια χρήσης |
|---|---|---|---|---|
| **Visual Paradigm for** | **Java, C#, C++, PHP, Ada, Action** | **Java, C# (binary), C++,** | **Eclipse, NetBeans** και | Εμπορική & Ελεύθερη με |

| | | | | |
|---|---|---|---|---|
| **UML** | **Script (**όλα στην εμπορική έκδοση**)** | **PHP (**όλα στην εμπορική έκδοση**)** | **IntelliJ** | λιγότερα χαρακτηριστικά |

Το εργαλείο **VP-UML** πρωτοεμφανίστηκε το **2002.**

### 5.4.6 BOUML

**Το BOUML [27]** είναι ένα δωρεάν λογισμικό σχεδίασης **UML** διαγραμμάτων**.** Έχει δημιουργηθεί σε **C++** και **Qt,** και διατίθεται ελεύθερα σύμφωνα με την άδεια χρήσης **GPL.** Είναι πολύγλωσσο**,** υποστηρίζει παραγωγή κώδικα από διαγράμματα και το αντίστροφο**.** Επιπλέον**,** είναι **cross platform** και τρέχει σε περιβάλλον **Linux / Unix / Solaris, Mac OS X** και **Windows** χάρη στο **Qt.**

Η υποστήριξη του **BOUML** σταμάτησε μεταξύ **2010** και Ιανουάριο **2012,** αλλά πολύ σύντομα θα διατεθεί η έκδοση **5** που όμως θα είναι εμπορική**.**

**Πίνακας 11 Χαρακτηριστικά BOUML**

| Όνομα | UML è Γλώσσα | Γλώσσα è UML | Πλατφόρμα | Άδεια χρήσης |
|---|---|---|---|---|
| **BOUML** | **C++, Java, PHP, IDL, Python** | **C++, Java, PHP** | **Eclipse, NetBeans** και **IntelliJ** | Ελεύθερη υπό **GPL** προς το παρόν |

### 5.4.7 StarUML

Το **StarUML [28]** ήταν ένα **UML** εργαλείο ανοικτού κώδικα**,** σύμφωνα με την τροποποιημένη άδεια **GNU GPL.** Για κάποιο χρονικό διάστημα σταμάτησε η εξέλιξη του**,** αλλά αναβίωσε για λίγο με τη μετακίνηση του του από **Delphi** σε **Java/Eclipse** αλλά και πάλι σταμάτησε**.** Ωστόσο**,** η κοινότητα είναι ακόμα ενεργή**.** Ο αρχικός δεδηλωμένος στόχος του **StarUML** ήταν να υποκαταστήσει μεγαλύτερες εμπορικές εφαρμογές όπως το **Rational Rose.**

Το **StarUML** υποστηρίζει περισσότερους από τους τύπους διαγραμμάτων της **UML 2.0.** Δεν υποστηρίζει διαγράμματα αντικειμένων**,** πακέτων**,** χρονισμού και διαγράμματα επισκόπησης αλληλεπίδρασης **(**αν και τα δύο πρώτα μπορούν να παρασταθούν μέσω κατάλληλης διαμόρφωσης του διαγράμματος κλάσης**).**

**Πίνακας 12 Χαρακτηριστικά StarUML**

| Όνομα | UML è Γλώσσα | Γλώσσα è UML | Πλατφόρμα | Άδεια χρήσης |
|---|---|---|---|---|
| **StarUML** | **Java,C#,C++** | **Java, C++, C#** | **C#** | Τροποποιημένη, **GPL** |

## 5.4.8 Power Designer

Το **PowerDesigner** [29] είναι ένα συνεργατικό εργαλείο μοντελοποίησης της **Sybase.** Τρέχει σε περιβάλλον **Microsoft Windows** ως εγγενής εφαρμογή αλλά και σε περιβάλλον **Eclipse** μέσω **plugin.** Το **PowerDesigner** υποστηρίζει **"**μοντέλοστραφή**"** σχεδιασμό αρχιτεκτονικής λογισμικού και χρησιμοποιεί το δικό του **format** αρχείων**,** το **.PDM.**

Είναι από τα ακριβά πακέτα καθώς διατίθεται με τιμή από **$3.000** έως **$7.500** δολάρια ανά άδεια χρήσης**.** Το **PowerDesigner** από τα πιο παλιά πακέτα αφού ξεκίνησε το **1989** με το όνομα **AMC * Designor** στη Γαλλία και **S-Designor** διεθνώς**,** από την **SDP.** Αρχικά το προϊόν αναπτύχθηκε για το σχεδιασμό βάσεων δεδομένων **Oracle,** αλλά πολύ γρήγορα εξελίχθηκε για την υποστήριξη όλων των μεγάλων πακέτων σχεσιακών βάσεων δεδομένων**. H Powersoft** εξαγόρασε την **SDP** το **1995,** ενώ νωρίτερα το **1994** η **Sybase** εξαγόρασε τη **Powersoft.** Λίγο μετά την εξαγορά**,** το προϊόν μετονομάστηκε με την σημερινή του ονομασία**.**

**Πίνακας 13 Χαρακτηριστικά PowerDesigner**

| Όνομα | UML è Γλώσσα | Γλώσσα è UML | Πλατφόρμα | Άδεια χρήσης |
|---|---|---|---|---|
| **PowerDesigner** | **Java, C#, VB .NET** | | **Eclipse (**ως **Plugin)** | Εμπορική |

## 5.4.9 Software Ideas Modeler

Το **Software Ideas Modeler** [30] υποστηρίζει και τα **14** είδη διαγραμμάτων που ορίζονται στο **UML 2.2.** Υποστηρίζει επίσης **ERD (Entity Relationship Diagrams)** διαγράμματα**,** διαγράμματα ροής και ροής δεδομένων**.** Το εργαλείο γράφτηκε σε **C #.** Θεωρείται από τα πιο **"**ελαφριά**"** εργαλεία και πλέον βρίσκεται στην έκδοση **5.02.**

**Πίνακας 14 Χαρακτηριστικά Software Ideas Modeler**

| Όνομα | UML è Γλώσσα | Γλώσσα è UML | Πλατφόρμα | Άδεια χρήσης |
|---|---|---|---|---|
| **Software Ideas Modeler** | **C++, C#, Java, PHP, Python, Ruby, SQL DDL, VB.NET, VB6** | **C#, VB.NET** | | Εμπορική ή ελεύθερη για μη εμπορική χρήση |

## 5.4.10 Astah*

Το **astah*** **[31]** που παλαιότερα ήταν γνωστό ως **JUDE,** είναι ένα εργαλείο μοντελοποίησης που δημιουργήθηκε από την ιαπωνική εταιρεία **Change Vision.** Το **JUDE** βραβεύτηκε ως το **"**Προϊόν Λογισμικού της Χρονιάς **2006"** της Ιαπωνίας**.** Υποστηρίζει και τα **14** είδη διαγραμμάτων της **UML 2.0** και διατίθεται σε διάφορες εκδόσεις με την **professional** να είναι εμπορική**.**

**Πίνακας 15 Χαρακτηριστικά Astah***

| Όνομα | UML è Γλώσσα | Γλώσσα è UML | Πλατφόρμα | Άδεια χρήσης |
|---|---|---|---|---|
| **Astah*** | **Java, C++, C#** | **Java, C++, C#** | | Εμπορική ή ελεύθερη για δοκιμή |

# Επίλογος

Είναι βέβαιο ότι η **UML** αποτελεί απαραίτητο εφόδιο για κάθε επαγγελματία της πληροφορικής, είτε πρόκειται για αναλυτή, είτε για σχεδιαστή, είτε για προγραμματιστή. Όμως είναι γενικότερα αποδεκτό ότι η γλώσσα μοντελοποίησης **UML** δεν περιορίζει τη χρήση της μόνο στο κλάδο της πληροφορικής, αλλά αντιθέτως έχει εφαρμογή και στη μοντελοποίηση άλλων συστημάτων όπως για παράδειγμα επιχειρηματικών διαδικασιών. Εξάλλου, η δομή της γλώσσας είναι τέτοια ώστε να μην εξαρτάται από την διαδικασία ανάπτυξης του συστήματος.

**Η UML** διαθέτει μια σήμανση απλή, κάτι που την κάνει κατανοητή, ενώ ταυτόχρονα παραμένει επεκτάσιμη και ευέλικτη, αλλά κυριότερα αποτελεί μια γλώσσα πρότυπο, που ελέγχεται από το **OMG**, ένα **group** ανοικτών προτύπων που υποστηρίζεται από πολυάριθμους ακαδημαϊκούς και κατασκευαστές. Το **OMG** εγγυάται τη "μεταφερσιμότητα" της **UML** και τη συμβατότητά της ανεξαρτήτως πλατφόρμας, χωρίς να εξαρτάται συγκεκριμένα από κάποιο εμπορικό προϊόν.

Οι δεκατέσσερις τύποι διαγραμμάτων μοντελοποίησης που διαθέτει, την καθιστούν ένα πολύτιμο εργαλείο καθώς οι δυνατότητες που προσφέρονται είναι τεράστιες. Δεν είναι τυχαίο άλλωστε ότι στις περισσότερες περιπτώσεις ένας μικρός αριθμός διαγραμμάτων από το σύνολο τους χρησιμοποιείται για να μοντελοποιηθεί ένα περίπλοκο σύστημα. Εξάλλου, η σημαντικότερη αξία της **UML** έγκειται στο γεγονός ότι έχει τη δυνατότητα να απλοποιεί την αντίληψη της λειτουργικότητας ενός συστήματος χωρίς να προσθέτει σύνθετες λεπτομέρειες που αποπροσανατολίζουν τον παρατηρητή.

Μεγάλο της πλεονέκτημα της **UML**, είναι επίσης ότι υποστηρίζεται από εργαλεία **Computer-Aided Software Engineering (CASE)** που επιτρέπουν την αυτόματη δημιουργία κώδικα βάσει των διαγραμμάτων. Ο σχεδιαστής το μόνο που έχει να κάνει είναι να σχεδιάσει με λεπτομέρεια τα διαγράμματα και τα εργαλεία **CASE** θα δημιουργήσουν τον αντίστοιχο κώδικα.

Αυτή τη στιγμή η **UML** βρίσκεται στην έκδοση **2.4** και αποτελεί, πλέον την πιο διαδεδομένη γλώσσα μοντελοποίησης για την ανάπτυξη λογισμικού, καθώς χρησιμοποιείται από τους αναλυτές, τους σχεδιαστές και τους προγραμματιστές. Η **UML** δίνει στον κάθε συμμετέχοντα, από τον αναλυτή επιχειρήσεων μέχρι τον προγραμματιστή, ένα κοινό λεξιλόγιο για την επικοινωνία τους σχετικά με το σχεδιασμό λογισμικού και αυτός είναι και ο κύριος λόγος που η χρήση της **UML** έχει επικρατήσει σε κάθε φάση του κύκλου ζωής ενός συστήματος.

# Αναφορές


**[1] http://www.uml-forum.com/FAQ.htm**

**[2] Grady Booch James Rumbaugh Ivar Jacobson, "The Unified Modeling Language User Guide", Addison-Wesley, 1999.**

**[3] Roel Wuyts, "UML: History and Overview", INFO025: Analyse et Méthodologie Informatiques, Université Libre de Bruxelles (ULB), 2005.**

**[4] "OMG Formally Released Versions of UML", http://www.omg.org/spec/UML/**

**[5] "UML Superstructure Specification" Version 2.2. OMG, February 2009, http://www.omg.org/spec/UML/2.2/**

**[6] Alexander Backlund, (2000) "The definition of system", Kybernetes, Vol. 29 Iss: 4, pp.444 - 451**

**[6] "The Entity-Relationship Model: Toward a Unified View of Data" Peter Pin-shan Chen, 1976**

**[7] Petri Nets World, http://www.informatik.uni-hamburg.de/TGI/PetriNets/index.html**

**[8] http://www.sysml.org/**

**[9] Event-driven process chain, From Wikipedia, http://en.wikipedia.org/wiki/Event-driven_process_chain**

**[10] http://www.agilemodeling.com/**

**[11] Object Management Group, Business Process Model and Notation http://www.bpmn.org/**

**[12] (http://scottishdevelopers.com/modules/news/article.php?storyid=110)**

**[13] http://www.computerworld.com/s/article/100542/Computerworld_Development_Survey_gives_nod_to_C_?taxonomyId=011**

**[14] "Software Development in Austria: Results of an Empirical Study among Small and Very Small Enterprises" Christian Hofer, 2002**

**[15] http://www.sdtimes.com/content/article.aspx?ArticleID=26637**

**[16] "How UML is used", Brian Dobing, Jeffrey Parsons, Magazine Communications of the ACM, May 2006**

**[17] "Sidebar: Waiting for UML 2.0 News Story by Carol Sliwa, http://www.computerworld.com/s/article/91325/Sidebar_Waiting_for_UML_2.0?taxonomyId=011).**

**[18] John Krogstie, "Advanced Conceptual Modeling Techniques" ER 2002 Workshops Tampere, Finland, October 7-11, 2002, (2003)**

**[19] Kuhn, D.L "Selecting and effectively using a computer aided software engineering tool" Annual Westinghouse computer symposium, Pittsburgh U.S, 1989**

**[20] Alfonso Fuggetta "A classification of CASE technology".** IEEE **Computer journal vol. 26 no. 12, 1993**

http://www.computer.org/portal/web/csdl/doi?doc=abs/mags/co/1993/12/rz025abs.htm

**[21]** Πλατφόρμα **Eclipse** http://en.wikipedia.org/wiki/Eclipse_(software)

**[22] IBM Rational Software Architect**
http://www.ibm.com/developerworks/rational/products/rsa/

**[23]** http://www.sparxsystems.com/

**[24]** https://www.magicdraw.com/

**[25]** http://www.modelio.org/

**[26]** http://www.visual-paradigm.com/product/vpuml/

**[27]** bouml.free.fr

**[28]** http://staruml.sourceforge.net/en/

**[29]** http://www.sybase.com/products/modelingdevelopment/powerdesigner

**[30]** http://www.softwareideas.net/

**[31]** http://astah.net/