“Ανάπτυξη προηγμένου περιβάλλοντος μάθησης για εκπαίδευση από απόσταση στον αντικειμενοστρεφή σχεδιασμό και ανάπτυξη συστημάτων με χρήση της γλώσσας UML ”

Τμήμα Επιχειρηματικού Σχεδιασμού και Πληροφοριακών Συστημάτων

Συντελεστές:
Κάζης Αριστείδης
Λιάλιος Γεώργιος

Εισηγητής Καθηγητής:
Dr. Νικόλαος Σ. Βώρος

ΠΕΡΙΕΧΟΜΕΝΑ

5. ΠΕΡΙΓΡΑΦΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

5.1 ΕΙΣΑΓΩΓΗ ΣΤΗ UML

5.1.1 Προσθέτοντας μια νέα μέθοδο
5.1.2 Πώς αναπτύχθηκε η UML
5.1.3 Κατανοώντας την UML
5.1.3.1 Στοιχεία της UML
5.1.4 Αναγνωρίζοντας τα επί μέρους τμήματα της UML
5.1.4.1 Δομημένα διαγράμματα
5.1.4.1.1 Διαγράμματα αντικειμένων και κλάσεων
5.1.4.1.2 Διαγράμματα ανάπτυξης και συστατικών
5.1.4.2 Διαγράμματα συμπεριφοράς
5.1.4.2.1 Διαγράμματα σεναρίων
5.1.4.2.2 Διαγράμματα δραστηριότητας
5.1.4.2.3 Διαγράμματα ακολουθίας
5.1.4.2.4 Διαγράμματα επικοινωνίας
5.1.4.2.5 Διαγράμματα καταστάσεων
5.1.5 Ο σκοπός των διαγραμμάτων

5.2 ΚΑΤΑΝΟΩΝΤΑΣ ΤΗΝ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΕΦΕΙΑ

5.2.1 Αντικείμενα
5.2.2 Αντικειμενοστρεφείς έννοιες
5.2.2.1 Αφαίρεση
5.2.2.2 Κληρονομικότητα
5.2.2.3 Πολυμορφισμός
5.2.2.4 Ενθυλάκωση
5.2.2.5 Αποστολή μηνύματος
5.2.2.6 Διασυνδέσεις
5.2.2.7 Ενσωμάτωση
5.2.3 Τα οφέλη της αντικειμενοστρέφειας
5.2.4 Περίληψη

5.3 ΔΟΥΛΕΥΟΝΤΑΣ ΜΕ ΤΗΝ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΕΦΕΙΑ

5.3.1 Αναπαράσταση μιας κλάσης
5.3.1.1 Ιδιότητες
5.3.1.2 Λειτουργίες
5.3.1.3 Ιδιότητες, Λειτουργίες και Οπτικοποίηση
5.3.1.4 Υποχρεώσεις και Περιορισμοί
5.3.1.5 Συνημμένες σημειώσεις
5.3.1.6 Κλάσεις – Τι κάνουν και πώς να τις βρείτε
5.3.2 Περίληψη

5.4 ΤΑ ΔΙΑΓΡΑΜΜΑΤΑ ΤΗΣ UML

5.4.1 ΔΙΑΓΡΑΜΜΑΤΑ ΚΛΑΣΕΩΝ Δουλεύοντας με τις σχέσεις

5.4.1.1 Συσχετίσεις

5.4.1.1.1 Περιορισμοί συσχετίσεων

5.4.1.1.2 Κλάσεις Συσχέτισης-Συσχετισμένες Κλάσεις

5.4.1.2 Πολλαπλότητα

5.4.1.3 Χαρακτηρισμένες Συσχετίσεις

5.4.1.4 Αυτοπαθείς Συσχετίσεις

5.4.1.5 Κληρονομικότητα και Γενίκευση

5.4.1.5.1 Ανακαλύπτοντας την Κληρονομικότητα

5.4.1.5.2 Αφηρημένες Κλάσεις

5.4.1.6 Εξαρτήσεις

5.4.1.7 Περίληψη

5.4.2 ΔΙΑΓΡΑΜΜΑΤΑ ΣΕΝΑΡΙΩΝ

5.4.2.1 Ορισμός διαγραμμάτων σεναρίου

5.4.2.1.1 Γιατί χρησιμοποιούμε τα διαγράμματα σεναρίων

5.4.2.2 Σημειολογικά στοιχεία διαγραμμάτων σεναρίου

5.4.2.2.1 Σύστημα

5.4.2.2.2 Δράστες

5.4.2.2.3 Υποδείγματα εργασίας

5.4.2.2.4 Συσχετίσεις

5.4.2.2.5 Υποκατηγορίες

5.4.2.3 Κατανοώντας την τεχνική της γενίκευσης

5.4.2.3.1 Συσχετισμός σεναρίου με σενάριο

5.4.2.3.2 Συσχετισμός δράστη με δράστη

5.4.2.4 Σχέσεις ενσωμάτωσης και επέκτασης

5.4.2.4.1 Σχέση ενσωμάτωσης

5.4.2.4.2 Σχέση επέκτασης

5.4.2.5 Μοντελοποίηση διαγραμμάτων σεναρίου

5.4.2.6 Περίληψη

5.4.3 ΔΙΑΓΡΑΜΜΑΤΑ ΑΚΟΛΟΥΘΙΑΣ

5.4.3.1 Τί είναι ένα διάγραμμα ακολουθίας

5.4.3.1.1 Ενεργά αντικείμενα

5.4.3.1.2 Μηνύματα

5.4.3.1.3 Χρόνος

5.4.3.2 Δημιουργία και διαγραφή αντικειμένων

5.4.3.3 Διακλάδωση και εναλλακτικές ροές

5.4.3.4 Μοντελοποίηση διαγραμμάτων ακολουθίας

5.4.3.5 Περίληψη

5.4.4 ΔΙΑΓΡΑΜΜΑΤΑ ΕΠΙΚΟΙΝΩΝΙΑΣ
5.4.4.1 Τι είναι ένα διάγραμμα επικοινωνίας
5.4.4.1.1 Αντικείμενα και ρόλοι
5.4.4.1.2 Ρόλοι συσχέτισης
5.4.4.1.3 Σύνδεσμοι
5.4.4.2 Μηνύματα
5.4.4.2.1 Διαφορετικά είδη μηνυμάτων
5.4.4.2.2 Περισσότερα για τα μηνύματα
5.4.4.2.3 Ακολουθία
5.4.4.2.4 Συνθήκες φύλαξης
5.4.4.2.5 Δημιουργώντας στιγμιότυπα
5.4.4.2.6 Επανάληψη
5.4.4.3 Μοντελοποίηση διαγραμμάτων επικοινωνίας
5.4.4.4 Περίληψη

5.4.5 ΔΙΑΓΡΑΜΜΑΤΑ ΚΑΤΑΣΤΑΣΗΣ
5.4.5.1 Ορισμός των διαγραμμάτων κατάστασης
5.4.5.1.1 Σημειολογία διαγραμμάτων κατάστασης
5.4.5.1.2 Καταστάσεις
5.4.5.1.3 Μεταβάσεις
5.4.5.1.4 Σημεία αποφάσεων
5.4.5.1.5 Συγχρονισμός
5.4.5.2 Πως ορίζονται ενέργειες και γεγονότα για καταστάσεις
5.4.5.2.1 Γεγονότα
5.4.5.2.2 Ενέργειες
5.4.5.3 Κατανοώντας τη χρήση σύνθετων καταστάσεων
5.4.5.3.1 Υποκαταστάσεις
5.4.5.3.2 Σύνθετες καταστάσεις
5.4.5.4 Μοντελοποίηση διαγραμμάτων κατάστασης
5.4.5.5 Περίληψη

5.4.6 ΔΙΑΓΡΑΜΜΑΤΑ ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ
5.4.6.1 Εισαγωγή στα διαγράμματα δραστηριοτήτων
5.4.6.1.1 Γιατί μοντελοποιούμε διαγράμματα δραστηριοτήτων
5.4.6.2 Τα σημειολογικά στοιχεία των διαγραμμάτων δραστηριοτήτων
5.4.6.2.1 Δραστηριότητες: καταστάσεις δράσης
5.4.6.2.2 Καταστάσεις
5.4.6.2.3 Μεταβάσεις
5.4.6.2.4 Συνθέτοντας τα σημειολογικά στοιχεία
5.4.6.3 Χρησιμοποιώντας συνθήκες
5.4.6.3.1 Σημεία φύλαξης
5.4.6.3.2 Σημεία απόφασης
5.4.6.4 Επιπλέον σημειολογίες
5.4.6.4.1 Συμβάντα και διεγέρτες
5.4.6.4.2 Στηλοθέτηση

5.6.3.5 Ποιότητα διεργασιών και προϊόντων
5.6.3.6 Μοντελοποίηση

5.6.4 Η μεθοδολογία GRAPPLE

5.6.5 Η δομή της GRAPPLE (RAD[3])

5.6.5.1 Συγκέντρωση απαιτήσεων

5.6.5.1.1 Ανακαλύψτε τις διεργασίες μιας επιχείρησης
5.6.5.1.2 Εκτέλεση Ανάλυσης Πεδίων Ορισμού
5.6.5.1.3 Συνεργαζόμενα Συστήματα
5.6.5.1.4 Ανακαλύψτε τις Απαιτήσεις του Συστήματος
5.6.5.1.5 Παρουσίαση των αποτελεσμάτων στον πελάτη

5.6.5.2 Ανάλυση

5.6.5.2.1 Καταλαβαίνοντας τη χρήση του συστήματος
5.6.5.2.2 Ανάλυση διαγραμμάτων σεναρίων
5.6.5.2.3 Αποσαφηνίζοντας ένα διάγραμμα κλάσης
5.6.5.2.4 Αναλύστε τις Αλλαγές Καταστάσεων στα Αντικείμενα
5.6.5.2.5 Καθορίστε την αλληλεπίδραση ανάμεσα στα αντικείμενα
5.6.5.2.6 Αναλύστε την Ενσωμάτωση των Συστημάτων Συνεργασίας

5.6.5.3 Σχεδιασμός

5.6.5.3.1 Ανάπτυξη και Αποσαφήνιση των Διαγραμμάτων Αντικειμένων
5.6.5.3.2 Αναπτύξτε Διαγράμματα Συστατικών
5.6.5.3.3 Προσχέδιο για ανάπτυξη
5.6.5.3.4 Σχεδιασμός και Πρωτότυπη Διασύνδεση με το Χρήστη
5.6.5.3.5 Τεστ σχεδιασμού
5.6.5.3.6 Αρχή Τεκμηρίωσης

5.6.5.4 Ανάπτυξη

5.6.5.4.1 Χτίζοντας τον κώδικα
5.6.5.4.2 Δοκιμάζοντας τον κώδικα
5.6.5.4.3 Χτίστε Διασύνδεση με το Χρήστη , Συνδεθείτε στον Κώδικα και Δοκιμάστε
5.6.5.4.4 Ολοκληρωμένη Τεκμηρίωση

5.6.5.5 Επέκταση

5.6.5.5.1 Εφεδρεία και Επανόρθωση
5.6.5.5.2 Εγκατάσταση του Τελικού Συστήματος
5.6.5.5.3 Δοκιμή στο Εγκαταστημένο Σύστημα

5.6.6 Το περιτύλιγμα του GRAPPLE

5.6.7 Περίληψη

5.7 ΠΑΡΑΔΕΙΓΜΑ ΠΕΡΙΓΡΑΦΗΣ ΕΝΟΣ ΣΥΣΤΗΜΑΤΟΣ

5.7.1 Διάγραμμα κλάσης
5.7.2 Διάγραμμα σεναρίου
5.7.3 Διάγραμμα κατάστασης
5.7.4 Διάγραμμα ακολουθίας
5.7.5 Διαγράμματα δραστηριότητας
5.7.6 Διάγραμμα επικοινωνίας
5.7.7 Διάγραμμα συστατικών
5.7.8 Διάγραμμα ανάπτυξης

5.8 ΤΟ ΜΕΛΛΟΝ ΤΗΣ UML

5.8.1 UML 2.0

5.8.1.1 Οι σημαντικότερες αλλαγές στη UML

5.8.1.2 Βαθμός ακρίβειας

5.8.1.3 Βελτιωμένη οργάνωση της γλώσσας

5.8.1.4 Μοντελοποίηση συστημάτων μεγάλης κλίμακας

5.8.1.5 Βελτιστοποίηση των εξειδικεύσεων της γλώσσας

5.8.1.6 Περίληψη

5.8.2 SysML

5.8.2.1 Σχέσεις με άλλα πρότυπα

5.8.2.2 Αρχιτεκτονική

5.8.2.3 Επισημοποίηση της γλώσσας

5.8.2.4 Η μετάβαση από την UML 2.0 στην SysML

1. ΕΙΣΑΓΩΓΗ

Η παρούσα πτυχιακή εργασία έχει ως σκοπό την παρουσίαση της γλώσσας μοντελοποίησης UML (Unified Modeling Language). Απευθύνεται κυρίως σε άτομα με μικρή ή καθόλου γνώση της συγκεκριμένης γλώσσας για τη διευκόλυνση των οποίων αποδίδεται σε ηλεκτρονική μορφή e-learning που επιτρέπει την εύκολη και ταχεία προσπέλαση στα διάφορα τμήματα του κειμένου.

Για το σχεδιασμό της εργασίας χρησιμοποιήθηκε ως οδηγός το πρότυπο ασύγχρονης τηλεκπαίδευσης IMS (Instructional Management System), με βάση το οποίο διαμορφώθηκε η διάρθρωση του κειμένου.

Προκειμένου να αποκτήσει η εργασία εκπαιδευτικό χαρακτήρα και να ανταποκρίνεται σε ένα σύγχρονο περιβάλλον εκμάθησης (e-learning) έγινε χρήση του προγράμματος Microsoft LRN (Learning Resource Interchange). Το LRN παρέχει στους δημιουργούς του περιεχομένου μια πρότυπη μέθοδο για την αναγνώριση, κατανομή, ανανέωση και δημιουργία δικτυακού περιεχομένου και υλικού για εκμάθηση.

Όπως προαναφέρθηκε, η UML είναι μια γλώσσα μοντελοποίησης, κύριο συστατικό της οποίας είναι τα σχεδιαγράμματα αναπαράστασης που εμπεριέχει στη σημειολογία της. Για την μοντελοποίηση αυτών των διαγραμμάτων χρησιμοποιήθηκε το πρόγραμμα Visual Paradigm, με τη χρήση του οποίου διαμορφώθηκαν όλα τα διαγράμματα της εργασίας.

2. ΤΟ ΠΡΟΤΥΠΟ ΑΣΥΓΧΡΟΝΗΣ ΤΗΛΕΚΠΑΙΔΕΥΣΗΣ IMS

Η αποστολή της Παγκόσμιας Κοινοπραξίας Εκμάθησης IMS (IMS Global Learning Consortium) είναι να υποστηρίξει την υιοθέτηση και τη χρήση της τεχνολογίας εκμάθησης σε παγκόσμια κλίμακα. Το IMS είναι μη κερδοσκοπική οργάνωση που περιλαμβάνει περισσότερα από 50 συμβεβλημένα μέλη και θυγατρικές. Αυτά τα μέλη προέρχονται από κάθε τομέα της παγκόσμιας κοινότητας e-learning. Είναι προμηθευτές υλικού και λογισμικού, εκπαιδευτικά όργανα, εκδότες, κυβερνητικές αντιπροσωπείες, ολοκληρωτές συστημάτων, προμηθευτές πολυμέσων και άλλες ομάδες επιχειρήσεων. Η κοινοπραξία παρέχει ένα ουδέτερο φόρουμ, στο οποίο τα μέλη με τα ανταγωνιστικά επιχειρησιακά ενδιαφέροντα και τα διαφορετικά κριτήρια λήψης αποφάσεων συνεργάζονται ώστε να ικανοποιήσουν τις πραγματικές απαιτήσεις για τη διαλειτουργικότητα και την δυνατότητα επαναχρησιμοποίηση.

Το IMS αναπτύσσει και προωθεί την έγκριση ανοικτών τεχνικών προδιαγραφών για τη διαλειτουργική τεχνολογία εκμάθησης. Διάφορες προδιαγραφές IMS έχουν γίνει de facto παγκόσμια πρότυπα για την κατασκευή προϊόντων και υπηρεσιών εκμάθησης. Οι προδιαγραφές IMS και οι σχετικές δημοσιεύσεις τίθενται στη διάθεση του κοινού δωρεάν.

2.1 Υπόβαθρο

Το 1997, το IMS δημιουργήθηκε ως πρόγραμμα στα πλαίσια της εθνικής πρωτοβουλίας υποδομής εκμάθησης EDUCAUSE. Ενώ το IMS ξεκίνησε με μια εστίαση στην τριτοβάθμια εκπαίδευση, οι προδιαγραφές που έχουν δημοσιευτεί μέχρι σήμερα καθώς επίσης και τρέχοντα έργα απευθύνουν τις απαιτήσεις του πλαισίου εκμάθησης.

Το πεδίο για τις προδιαγραφές IMS, που ορίζεται ευρέως ως η "διανεμημένη εκμάθηση" (“distributed learning”), περιλαμβάνει ταυτόχρονα τις σε απευθείας σύνδεση (online) και τις σε μη-απευθείας σύνδεση (offline) ρυθμίσεις, που λαμβάνουν χώρα συγχρονισμένα (σε πραγματικό χρόνο) ή ασύγχρονα. Αυτό σημαίνει ότι τα πλαίσια εκμάθησης που ωφελούνται από τις προδιαγραφές IMS περιλαμβάνουν τα Διαδικτυακά (Internet-specific) περιβάλλοντα (όπως τα βασισμένα στο WEB συστήματα διαχείρισης σειράς μαθημάτων (web based course management systems) καθώς επίσης και τις καταστάσεις εκμάθησης που περιλαμβάνουν ηλεκτρονικούς πόρους σε μη-απευθείας σύνδεση (όπως ένας μαθητευόμενος που έχει πρόσβαση στους πόρους εκμάθησης μέσω ενός CD-ROM). Οι μαθητευόμενοι μπορούν να βρίσκονται σε ένα παραδοσιακό εκπαιδευτικό περιβάλλον (σχολική τάξη, πανεπιστήμιο), σε ένα πρόγραμμα εταιρικής ή κυβερνητικής κατάρτισης, ή στο σπίτι. Παραδείγματος χάριν, η Προδιαγραφή Μετα-δεδομένων των Πόρων Εκμάθησης IMS (www.imsglobal.org/metadata), ωφελεί το μαθητευόμενο που ψάχνει τις πληροφορίες με ένα ενήμερο εργαλείο αναζήτησης μεταδεδομένων και όταν η αναζήτηση είναι βασισμένη σε πόρους του WEB, και όταν ψάχνει μέσω μιας εγκυκλοπαίδειας CD-ROM ή DVD-ROM στον υπολογιστή του στο σπίτι. Υπεύθυνοι για την ανάπτυξη που έχουν εφαρμόσει το IMS έχουν καταστήσει πολύ εύκολο για το άτομο που κάνει

αναζήτηση το να βρίσκει τους πόρους που αυτό θέλει με ένα πολύ αποδοτικό τρόπο, δεδομένου ότι τα μετα-δεδομένα επιτρέπουν στο χρήστη να είναι πολύ περισσότερο συγκεκριμένος στους όρους αναζήτησης που θέτουν.

2.2 Η διαδικασία ανάπτυξης προδιαγραφών IMS

Το IMS καλύπτει ένα ευρύ πεδίο εργασιών. Συλλέγονται οι απαιτήσεις μέσω των συνεδριάσεων, συγκεντρώνονται ομάδες, και άλλες πηγές σε όλη την υδρόγειο για να καθιερώσουν τις κρίσιμες πτυχές της διαλειτουργικότητας στις αγορές εκμάθησης. Με βάση αυτές τις απαιτήσεις, αναπτύσσονται οι προδιαγραφές σχεδίων περιγράφοντας τον τρόπο που το λογισμικό πρέπει να χτιστεί προκειμένου να καλυφθούν οι απαιτήσεις. Σε όλες τις περιπτώσεις, οι προδιαγραφές αναπτύσσονται για να υποστηρίξουν τις διεθνείς ανάγκες. Μόλις οριστικοποιηθούν εσωτερικά οι προδιαγραφές και αποδειχθούν δοκιμασμένες μέσω των δοκιμών διαλειτουργικότητας, που περιλαμβάνουν συνήθως τα συμβαλλόμενα μέλη και τους συμμετέχοντες υπευθύνους δικτύων για την ανάπτυξη, η προδιαγραφή σχεδίων εγκρίνεται τυπικά από τον τεχνικό πίνακα IMS και διατίθεται έπειτα στο κοινό. Οι προδιαγραφές IMS είναι διαθέσιμες στο ευρύ κοινό, ανεξάρτητα από εάν κάποιος είναι ή όχι μέλος του IMS, μόλις το τεχνικό συμβούλιο εγκρίνει τη διάθεσή τους. Υποβάλλονται επίσης στις ομάδες που αναγνωρίζουν τα πρότυπα προκειμένου να συμβάλλουν στην αναγνώριση και την υιοθέτηση μιας παγκόσμιας βάσης τεχνικών προτύπων για τη διανεμημένη εκμάθηση.

2.3 Το πεδίο

Ο στόχος της ομάδας εργασίας εκμάθησης του IMS είναι να εργαστεί προς την καθιέρωση των προδιαγραφών για την περιγραφή των στοιχείων και της δομής οποιασδήποτε μονάδας της εκμάθησης, που συμπεριλαμβάνουν:

- πόρους
- οδηγίες για εκπαιδευτικές δραστηριότητες
- φόρμες για δομημένες αλληλεπιδράσεις
- εννοιολογικά μοντέλα (π.χ., εκμάθηση problem-based)
- διαπίστωση των στόχων και αποτελεσμάτων
- εργαλεία αξιολόγησης και στρατηγικές

Οι προδιαγραφές, που περιγράφουν αυτό το πλαίσιο, οφείλουν:

- Να περιγράφουν και εφαρμόζουν τα διαφορετικά είδη εκμάθησης
- Να καθιστούν δυνατή τη δημιουργία επαναλαμβανόμενων, αποτελεσματικών, και αποδοτικών μονάδων εκμάθησης .
- Να παρέχουν πρόσβαση και δυνατότητα ανταλλαγής στις μονάδες της εκμάθησης
- Να υποστηρίζουν πολλαπλάσια μοντέλα παράδοσης
- Να υποστηρίζουν την επαναχρησιμοποίηση μονάδων εκμάθησης ή των συστατικών στοιχείων τους

- Να υποστηρίζουν την επαναχρησιμοποίηση και τον επαναπροσδιορισμό του πλαισίου και των συστατικών μιας μονάδας
- Να υποστηρίζουν τη σύνταξη προδιαγραφών και προτύπων
- να είναι πολιτιστικά ανοικτές (διεθνοποίηση)
- να υποστηρίζουν πολλούς μαθητευομένους και ρόλους σε μια δραστηριότητα εκμάθησης με βάση εμπειρίες εκμάθησης συλλογικού χαρακτήρα
- να υποστηρίζουν τη σύνταξη έκθεσης και ανάλυση της αποδοτικότητας του συστήματος

Ο στόχος είναι να επιτραπεί η δημιουργία πολλών ειδών εκπαιδευτικών σχεδίων , χρησιμοποιώντας μια συνεπή σημειολογία, η οποία μπορεί να εφαρμοστεί ομοιόμορφα σε πολλά μαθήματα ή προγράμματα εκμάθησης.

Η παροχή των εννοιολογικών προτύπων για διάφορους τύπους εκμαθήσεων είναι ένας μεγάλος στόχος που απαιτεί την ανάπτυξη μιας ικανότητας χαρτογράφησης θέματος ή μερικές ισοδύναμες και δεν υποστηρίζεται στο παρόν έγγραφο. Ομοίως, τα εργαλεία αξιολόγησης και οι στρατηγικές δεν συμπεριλαμβάνονται ρητά στο παρόν έγγραφο

3. ΔΗΜΙΟΥΡΓΙΑ ΗΛΕΚΤΡΟΝΙΚΟΥ ΠΕΡΙΕΧΟΜΕΝΟΥ ΜΕ ΤΟ LRN

Η Microsoft συνεργάστηκε με τις σημαντικότερες εταιρείες στο χώρο της Ηλεκτρονικής Εκμάθησης (eLearning) για να υποστηρίξουν το Learning Resource Interchange (LRN). Το LRN βοηθάει στην ανταλλαγή εκπαιδευτικού περιεχομένου που παρέχει στους δημιουργούς του περιεχομένου μια πρότυπη μέθοδο για την αναγνώριση, κατανομή, ανανέωση και δημιουργία δικτυακού περιεχομένου και υλικού για εκμάθηση. Το LRN είναι μια εμπορική εφαρμογή του πακέτου περιεχομένου IMS και των προδιαγραφών των Μεταδεδομένων (Metadata). Ο Παγκόσμιος Οργανισμός Εκμάθησης IMS είναι ένας οργανισμός οδηγούμενος απο τη βιομηχανία, με προορισμό να ορίζει και να προτείνει πρότυπα για τον τομέα της Ηλεκτρονικής Εκπαίδευσης. Το LRN υποστηρίζει ακόμα το πρότυπο Ηλεκτρονικής Εκμάθησης ADL SCORM (Advanced Distributed Learning Sharable Content Object Reference Model).

Το Πακέτο Εργαλείων LRN (LRN Toolkit) παρέχει συμπεριλαμβάνει τις προδιαγραφές του IMS και του ADL SCORM, καθώς και εργαλεία για να ξεκινήσει κάποιος άμεσα κάποια συγκεκριμένη μορφή.

Το Πακέτο Εργαλείων LRN παρέχει τα ακόλουθα αντικείμενα:

- Ένα σετ εργαλείων για τη δημιουργία, παρακολούθηση και έλεγχο του περιεχομένου του LRN
- Δείγματα που επιδεικνύουν την ευελιξία του LRN
- Τις Προδιαγραφές του Πακέτου IMS (IMS Content Packaging Specification)
- Τις Προδιαγραφές των Μεταδεδομένων του IMS (IMS Meta-data Specification)
- Τα προφίλ εφαρμογής του SCORM (ADL Sharable Content Object Reference Model (SCORM) Packaging Application Profiles)
- Λεπτομερείς παρουσιάσεις, οδηγούς και τεκμηρίωση του Πακέτου Εργαλείων του LRN.

3.1 Δυσκολίες στην αγορά της ηλεκτρονικής εκμάθησης σήμερα

3.1.1 Έλλειψη Εκπαιδευτικών Προτύπων

Η αγορά Ηλεκτρονικής Εκμάθησης βρίσκεται στα πρώτα στάδια της ανάπτυξής της. Πολλές εταιρείες παρέχουν περιεχόμενα, εργαλεία και λύσεις, αλλά τα πρότυπα για να λειτουργήσουν από κοινού αυτά τα διάφορα μέρη μόλις έχουν αρχίσει να αναδύονται. Αν και η επιτυχία του Διαδικτύου οδηγεί όλο και περισσότερες υπηρεσίες στην υποστήριξη βασικών προτύπων του Διαδικτύου όπως τα HTML and HTTP, αυτά τα πρότυπα από μόνα τους δεν είναι αρκετά για να υποστηρίξουν μια ολοκληρωμένη εκπαιδευτική εμπειρία.

3.1.2 Συστήματα διαχείρισης πολλαπλής εκμάθησης

Εξαιτίας της απουσίας ενός ευρέως αποδεκτού προτύπου ανταλλαγής περιεχομένου , πολλοί παροχείς περιεχομένου παρέχουν το δικό τους σύστημα διαχείρισης εκπαιδευτικού υλικού. Αυτή η στρατηγική επιτρέπει στους παροχείς αυτούς να προσθέσουν προηγμένες λειτουργίες στο περιεχόμενο τους, αλλά δημιουργεί ένα πρόβλημα για τα σχολεία και τα πανεπιστήμια που πρέπει να συντηρούν διαφορετικά συστήματα εκμάθησης έτσι ώστε να μπορούν να χρησιμοποιούν περιεχόμενα από διαφορετικούς παροχείς.

Το πρόβλημα αυτό επιδεινώνεται εξαιτίας του ότι για να αποκομίσουν τα μέγιστα από τις τεχνολογικές τους επενδύσεις τα περισσότερα κολεγιά και πανεπιστήμια θέλουν να συνδέουν τα πληροφοριακά συστήματα των μαθητών με αλλά πληροφοριακά συστήματα , όπως τις πληροφορίες που αφορούν και την οικονομική βοήθεια που τους παρέχεται. Τα συστήματα διαχείρισης πολλαπλής εκμάθησης αυξάνουν το κόστος για τα ιδρύματα, αυτά λόγω της ανάγκης τους για συντήρηση και διαχείριση.

3.1.3 Έλλειψη ενσωματωμένων εργαλείων συγγραφής

Κάθε εργαλείο συγγραφής χρειάζεται να γράφει κώδικα που στοχεύει ανεξάρτητα το κάθε πρότυπο εκπαιδευτικού περιεχομένου. Παρόλα αυτά, η κατάσταση της αγοράς εκπαιδευτικού περιεχομένου σήμερα είναι παρόμοια με την κατάσταση των εργαλείων συγγραφής πολυμέσων πριν από την έναρξη του Διαδικτύου. Τότε, κάθε εργαλείο συγγραφής δημιουργούσε το δικό του περιεχόμενο που συνδεόταν καλύτερα με την οικογένεια προϊόντων της δικιάς του εταιρείας.

Μετά τη δημιουργία ενός ευρέως αποδεκτού προτύπου από την HTML για το περιτύλιγμα του περιεχομένου των πολυμέσων, το marketing και η χρήση εργαλείων συγγραφής πολυμέσων που ενσωματώνονταν στο Διαδίκτυο άνθισαν. Τώρα αποτελεί κοινή επιδίωξη να βρεθούν Διαδικτυακοί τόποι που χρησιμοποιούν όλους τους τύπους πολυμέσων που έχουν παραχθεί με μια σουίτα εργαλείων που ενσωματώνονται απευθείας με τη χρήση της HTML. Παρόμοια, με την ανάδυση ενός ευρέως αποδεκτού προτύπου για την περιγραφή και ανταλλαγή εκπαιδευτικού περιεχομένου, εργαλεία συγγραφής τρίτων εταιρειών μπορούν να υιοθετηθούν για την υποστήριξη αυτού του προτύπου, και μ’ αυτό τον τρόπο επιτρέπουν στους χρήστες να επιλέξουν το καλύτερο εργαλείο για την υποστήριξη των αναγκών τους.

3.1.4 Ο εκπαιδευτικός κύκλος δεν ταιριάζει με τις ανάγκες του σπουδαστή

Οι σημερινοί σπουδαστές έχουν λιγότερο χρόνο να αφιερώσουν στη συνεχόμενη εκπαίδευση. Πολλοί έχουν άλλες δεσμεύσεις, όπως επαγγέλματα πλήρους ή μερικής απασχόλησης και οικογένεια, τις οποίες οφείλουν να υπηρετούν παράλληλα με τα μαθήματά τους. Μπορεί να είναι σχεδόν αδύνατο να αφιερώσουν δυο ημέρες ή μια εβδομάδα σε απερίσπαστο διάβασμα. Προκειμένου ο εκπαιδευτικός κύκλος να υπηρετεί αποτελεσματικά την εκπαίδευση χρειάζεται να λειτουργεί ως μια

συνεχόμενη εμπειρία εκμάθησης αλλά και να χρησιμεύει ως αναφορά για την επίλυση συγκεκριμένων προβλημάτων όποτε αυτά εμφανίζονται.

3.1.5 Έλλειψη συμπληρωματικών υπηρεσιών από τρίτους

Είναι δύσκολο προς το παρόν να δημιουργηθεί μια σειρά μαθημάτων που θα χρησιμοποιεί ένα ευρύ φάσμα υπηρεσιών , όπως εικονικές τάξεις , στιγμιαία μηνύματα , προσομοιώσεις και συζητήσεις . Η ενσωμάτωση τέτοιων υπηρεσιών γίνεται μέσω των συστημάτων διαχείρισης της εκμάθησης. Τα μαθήματα πρέπει να προσαρμοστούν σε κάθε σύστημα για να χρησιμοποιήσουν τις υπηρεσίες.

Αυτό έχει ως αποτέλεσμα να είναι πολύ δύσκολη η ικανοποιητική προσαρμογή υπηρεσιών από τρίτους, εκτός εάν αυτοί σχεδιάσουν τις υπηρεσίες τους αποκλειστικά για το συγκεκριμένο σύστημα

3.2 Πως το LRN αντιμετωπίζει τα προβλήματα αυτά

3.2.1 Το LRN εφαρμόζει ευρέως αποδεκτά πρότυπα ανταλλαγών

Το IMS ορίζει ένα απλό περιτύλιγμα για το εκπαιδευτικό περιεχόμενο που του επιτρέπει να ανταλλάσσεται αποτελεσματικά μεταξύ διαφόρων εκπαιδευτικών συστημάτων. Ένα πακέτο IMS περιλαμβάνει μεταδεδομένα που περιγράφουν το περιεχόμενο, με έναν τίτλο ή μια περιγραφή, την οργάνωση του περιεχομένου και τις πηγές του, όπως τα HTML αρχεία και τα γραφικά.

Εκτός από το να συμπεριλαμβάνει και να ορίζει τις προϋποθέσεις του IMS το SCORM υποστηρίζει την ανταλλαγή πόρων εκπαιδευτικού περιεχομένου. Συγκεκριμένα περιέχει:

- ένα πρωτόκολλο εκτέλεσης του περιεχομένου για να το συνδέει με το εκάστοτε πρόγραμμα.
- ένα περιβάλλον λειτουργίας του περιεχομένου για να επικοινωνεί με τα συστήματα διαχείρισης εκπαιδευτικού υλικού.
- ένα μοντέλο δεδομένων του περιεχομένου που ορίζει τη δομή και τη σημειολογία των δεδομένων που ανταλλάσσονται μεταξύ του LMS και ενός αλλού συστήματος.

3.2.2 Το LRN δεν επηρεάζεται από τη μορφή του περιεχομένου

Ένα αρχείο του LRN παρουσιάζει το περιεχόμενο του με τη χρήση URL (Uniform Resource Locators) και δεν περιέχει καμιά πληροφορία σχετική με τη μορφή του περιεχομένου.

Το LRN στηρίζεται στο γεγονός ότι το μεγαλύτερο μέρος του περιεχομένου μπορεί να φιλοξενηθεί στον παγκόσμιο ιστό.
Για τα τμήματα εκείνα του περιεχομένου που δεν μπορούν να φιλοξενηθούν σε μια ιστοσελίδα χρησιμοποιείται η XML για να διαμορφωθεί μια καινούργια περιγραφή του περιεχομένου αυτού.

3.2.3 Η δομή του LRN είναι επεκτάσημη

Το LRN στηρίζεται πάνω στην αρχιτεκτονική της XML που είναι επεκτάσιμη. Καινούργιες πληροφορίες μπορούν να τοποθετηθούν σε οποιοδήποτε τμήμα ενός εγγράφου του LRN. Νέα μεταδεδομένα, νέοι τύποι οργάνωσης, νέοι τύποι πηγών μπορούν όλα να προσδιοριστούν.

3.2.4 Η LRN ορίζει τους εκπαιδευτικούς πόρους και όχι τα μαθήματα

Ένας εκπαιδευτικός πόρος είναι οποιαδήποτε συλλογή πόρων που περιγράφει μια έννοια που μπορεί να διδαχτεί ανεξάρτητα από άλλες έννοιες. Μπορεί να περιέχει άλλους εκπαιδευτικούς πόρους έτσι ώστε να επιτρέπει μεγαλύτερες έννοιες να δημιουργούνται από τον συνδυασμό άλλων μικρότερων. Ένας εκπαιδευτικός πόρος σε ένα LRN XML έγγραφο αντιπροσωπεύεται από το αρχείο <manifest>.

3.3 Νέα σενάρια που πραγματοποιούνται από το LRN

3.3.1 Δημιουργία εγγραφών με την χρήση καθιερωμένων εργαλείων

Επειδή η LRN επιτρέπει την προσθήκη κάθε είδους αρχείου μπορεί να δημιουργηθεί εκπαιδευτικό περιεχόμενο με καθιερωμένα εργαλεία όπως είναι το Microsoft Word ή το Power Point και να ενσωματωθεί σε ένα ήδη υπάρχον εκπαιδευτικό πρόγραμμα. Άρα με το LRN δεν είναι πλέον αναγκαίο να ταιριάζουν τα εργαλεία συγγραφής που έχει γραφτεί το εκπαιδευτικό περιεχόμενο.

3.3.2 Επέκταση μιας σειράς μαθημάτων με επιχειρησιακές πληροφορίες

Επειδή μπορεί να προστεθεί καινούργιο υλικό σε υπάρχοντα μαθήματα μπορεί κάποιος για παράδειγμα να προσθέσει σε ένα μάθημα για το Outlook συνδέσμους σε αρχεία του Word που περιγράφουν την πολιτική μιας εταιρείας για τη χρήση του ηλεκτρονικού ταχυδρομείου.

3.3.3 Επικοινωνία εν ώρα λειτουργίας

Η υποστήριξη του SCORM επιτρέπει στους δημιουργούς εκπαιδευτικού περιεχομένου να τοποθετήσουν διαδραστικα προγράμματα μέσα στο περιεχόμενο γνωρίζοντας ότι θα υποστηρίζεται από διαφορετικές πλατφόρμες LMS.

3.4 Το πακέτο εργαλείων LRN

Το LRN toolkit περιέχει εργαλεία και παραδείγματα περιεχομένου που βοηθούν στην δημιουργία εκπαιδευτικού περιεχομένου και μεταδεδομενων. Το LRN toolkit επικεντρώνεται στη δημιουργία , επαλήθευση και εμφάνιση των πακέτων του LRN.

Τα διαθέσιμα εργαλεία μέσα στο πακέτο είναι :

- Μετατροπέας για το Microsoft Office : το πακετο εργαλειων παρεχει προσθετα πακετα για το Microsoft PowerPoint και το Microsoft FrontPage που επιτρεπουν τη δημοσιοποιηση εγγραφων του Office σε μορφη LRN.
- Μετατροπέας για το LRN : επιτρεπουν τη μετατροπη αρχειων του LRN σε πιο καινουργιες εκδοσεις.
- Μετατροπέας για τη Microsoft Encarta Class Server : μετατρεπουν ενα αρχειο απο τη μορφη του LRN στη μορφη που γινεται κατανοητη απο τον διακομιστη (server) της Microsoft Encarta.
- Επεξεργαστής LRN: Χρησιμοποιείται για να δημιουργεί και να επεξεργάζεται μανιφέστα LRN. Αυτό το εργαλείο είναι βασισμένο στη Visual Basic.
- Ελεγκτής LRN: Ο ελεγκτής LRN χρησιμοποιείται για να επαληθεύσει πως ένα αρχείο *imsmanifest.xml* περιέχει έγκυρη μορφή της γλώσσας XML.
- Εργαλείο προεπισκόπησης LRN: Το εργαλείο προεπισκόπησης LRN παράγει μια όψη DHTML του περιεχομένου του LRN, προσθέτοντας χαραστηριστικά πλοήγησης και παρέχει ένα πλαίσιο υποστήριξης του SCORM.

4. ΤΟ VISUAL PARADIGM ΓΙΑ ΤΗ UML

Το Visual Paradigm για την UML (εφεξής καλούμενη VP-UML) είναι ένα οπτικό εργαλείο μοντελοποίησης της UML. Αυτό το εργαλείο είναι σχεδιασμένο για ένα ευρύ φάσμα χρηστών, συμπεριλαμβανομένων των μηχανικών λογισμικού, των αναλυτών συστημάτων, των επιχειρησιακών αναλυτών και αρχιτεκτόνων συστημάτων, οι οποίοι ενδιαφέρονται να σχεδιάσουν συστήματα λογισμικού μεγάλης κλίμακας με πιστότητα, μέσω της αντικειμενοστρεφούς προσέγγισης. Το VP-UML υποστηρίζει τα πιό πρόσφατα πρότυπα της Java και της σημειολογίας της UML και παρέχει την πλήρη παραγωγή κώδικα για Java. Οι μεταβάσεις από την ανάλυση στο σχεδιασμό και έπειτα στην εφαρμογή είναι ενσωματωμένες μέσα στο εργαλείο, μειώνοντας έτσι σημαντικά τις προσπάθειες σε όλα τα στάδια του κύκλου ζωής της ανάπτυξης λογισμικού.

4.1 Τι μπορούμε να κάνουμε με το VP-UML

Το VP-UML προσφέρει ένα ευρύ φάσμα λειτουργιών, συμπεριλαμβανομένου ενός συντάκτη διαγραμμάτων UML που υποστηρίζει:

- Διαγράμματα σεναρίων
- Διαγράμματα Κλάσεων
- Διαγράμματα Ακολουθίας
- Διαγράμματα Επικοινωνίας
- Διαγράμματα Κατάστασης
- Διαγράμματα Δραστηριότητας
- Διαγράμματα Συστατικών
- Διαγράμματα Ανάπτυξης
- Ανάλυση Κειμένου
- Ένα όργανο ελέγχου διαγραμμάτων επισκόπησης σε πραγματικό χρόνο για τα σύνθετα διαγράμματα.
- Γνωστές διαδικασίες έκδοσης παρόμοιες με εκείνες που παρέχονται από το περισσότερο σύγχρονο λογισμικό (π.χ. Microsoft Word, Excel κτλ) όπως η αντιγραφή (copy) και η επικόλληση (paste) ενός διαγράμματος σε και από την περιοχή αποκομμάτων (clipboard).

Η UML παρέχει επίσης τα ακόλουθα προηγμένα χαρακτηριστικά γνωρίσματα:

- Παραγωγή κώδικα από τα διαγράμματα στον κώδικα της Java σε πραγματικό χρόνο όταν τα διαγράμματα εκδίδονται, και αντίστροφα
- Αυτόματη παραγωγή αναφορών σε τύπους HTML και PDF
- Ένα ισχυρό εργαλείο εκτύπωσης που επιτρέπει να σχηματοποιήσουμε το σχεδιάγραμμα προεκτύπωσης.
- απεικόνιση “δέντρου” παρέχεται για να υποστηρίξει την ομαδοποίηση και τη διαχείριση των διαγραμμάτων UML για ένα έργο και τους σχετικούς πόρους των βιβλιοθηκών της. Βοηθά στη διαχείριση όλων των αρχείων του έργου

- Ένας Εισαγωγέας Έργων (Project Importer) για την εισαγωγή έργου XMI και έργου Rose.
- Ο Ελεγκτής Μοντέλων επικυρώνει τα στοιχεία του μοντέλου σύμφωνα με τη σύνταξη UML.

4.2 Τα ειδικά χαρακτηριστικά γνωρίσματα της VP-UML

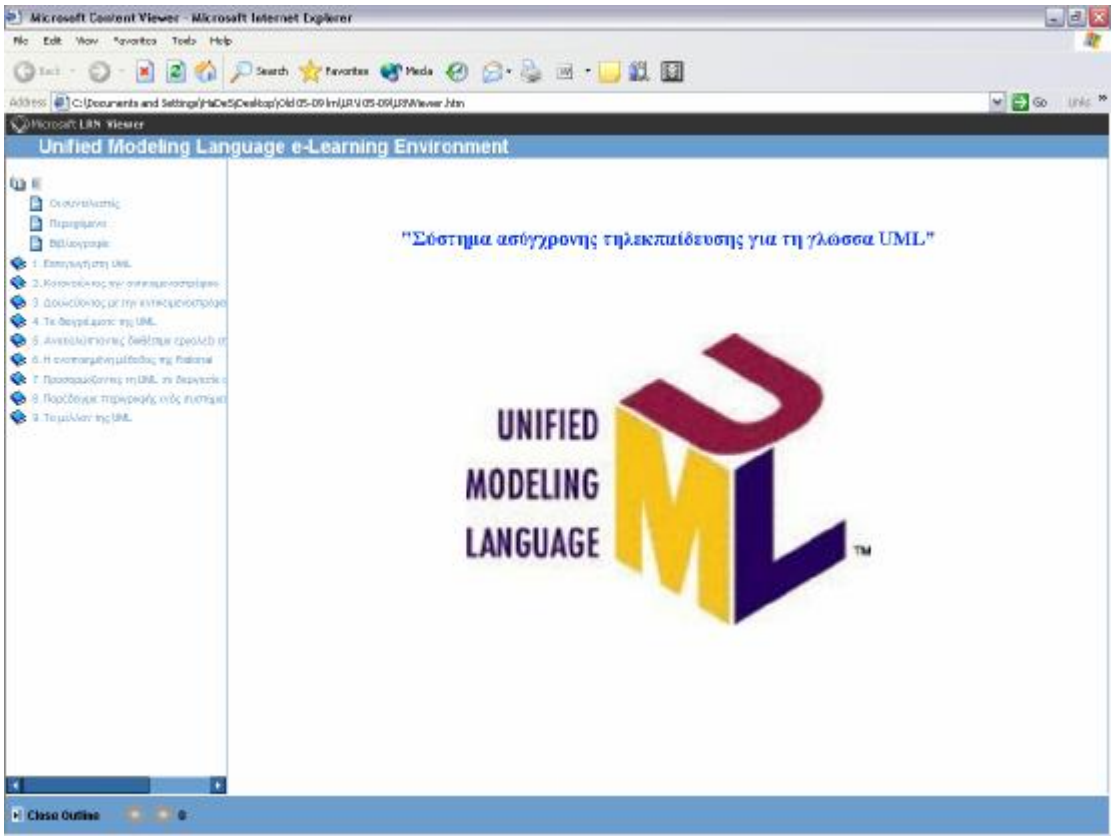
Η VP-UML παρέχει διάφορα ισχυρά χαρακτηριστικά γνωρίσματα που δεν είναι ενσωματωμένα σε πολλά εργαλεία διαθέσιμα σήμερα στην αγορά. Επιπλέον, η VP-UML έχει σχεδιαστεί ώστε να χρησιμοποιείται το εργαλείο με τις ελάχιστες προσπάθειες. Μερικά από αυτά τα χαρακτηριστικά γνωρίσματα είναι:

- Το Κέντρο Εκμάθησης που διδάσκει πώς χρησιμοποιείται διαδραστικά η VP-UML
- Ένα ευκατανόητο περιβάλλον απεικόνισης, που επιτρέπει την αποτελεσματική λήψη ενεργειών
- Ένα πορο-κεντρικό (resource centric) περιβάλλον χρήστη που υιοθετείται για να επιτρέψει να κατασκευαστούν τα διαγράμματα UML με τις ελάχιστες προσπάθειες
- Με το νεοπαγές αυτό περιβάλλον, μόνο οι έγκυροι πόροι σύνταξης ομαδοποιούνται γύρω από μια γραφική οντότητα, αποβάλλοντας συνολικά τις άκυρες διαδικασίες κατά τη διάρκεια της κατασκευής διαγραμμάτων
- Ένα σήμα προειδοποίησης εμφανίζεται όποτε εκτελείται μια άκυρη λειτουργία σύνταξης, ώστε να ελαχιστοποιηθούν τα λάθη στο διάγραμμα.
- Δυνατότητα σύνδεσης ενός διαγράμματος με άλλα διαγράμματα UML χαμηλοτέρου επιπέδου
- Δυνατότητα δημιουργίας αναφορών σε μορφή HTML ή PDF
- Ο οδηγός επικαιροποίησης του προγράμματος καθιστά εύκολη τη διαδικασία αυτή με μερικά απλά βήματα

5. ΠΕΡΙΓΡΑΦΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

Όπως αναφέραμε στην Εισαγωγή, υλοποιήσαμε ένα περιβάλλον ηλεκτρονικής εκμάθησης της UML σύμφωνα με το πρότυπο IMS και με τη χρήση των προγραμμάτων LRN και Visual Paradigm.

Ένα δείγμα του περιβάλλοντος της εφαρμογής που υλοποιήθηκε, παραθέτουμε παρακάτω:



Στη συνέχεια παρουσιάζεται το κείμενο, το οποίο αποτελεί το εκπαιδευτικό υλικό της εργασίας μας.

5.1 ΕΙΣΑΓΩΓΗ ΣΤΗ UML

5.1.1 Προσθέτοντας μια νέα μέθοδο

Στις πρώτες εποχές των υπολογιστών, κάποιοι προγραμματιστές περιορίστηκαν στην βαθιά ανάλυση των δεδομένων προβλημάτων που αντιμετώπιζαν. Εάν κιόλας έκαναν κάποια ανάλυση που κι αυτό γινόταν με προχειρότητα. Συχνά «έγραφαν» προγράμματα απ' την αρχή, δημιουργώντας κώδικα όπως προχωρούσαν. Αν και αυτό προσέθετε μια αύρα ρομαντισμού και τόλμης στη διαδικασία, αποδείχθηκε πως είναι ακατάλληλο για τον σημερινό απαιτητικό επιχειρηματικό κόσμο.

Σήμερα ένα καλά δομημένο σχέδιο είναι απαραίτητο. Ο πελάτης πρέπει να καταλάβει το τι πρόκειται να κάνει μια ομάδα ανάπτυξης (λογισμικού) και να είναι σε θέση να προτείνει αλλαγές στην περίπτωση που η ομάδα δεν έχει καλύψει πλήρως τις σημερινές και μελλοντικές του ανάγκες (ή αν ο πελάτης αλλάξει τις προδιαγραφές του στην πορεία).

Επίσης, η ανάπτυξη είναι τυπικά μια ομαδική προσπάθεια, έτσι κάθε μέλος της ομάδας πρέπει να γνωρίζει σε ποιο σημείο του έργου «ταιριάζει» αυτό που πρόκειται να κάνει (και να ξέρει τι και ποιο είναι το έργο).

Όσο οι ανάγκες γίνονται όλο και πιο πολύπλοκες, τα υπολογιστικά συστήματα πρέπει κι αυτά να γίνουν πιο σύνθετα. Συχνά περιλαμβάνουν διάφορα κομμάτια από software και hardware, δικτυωμένα σε μεγάλες αποστάσεις, συνδεδεμένα σε βάσεις δεδομένων που περιέχουν πληθώρα πληροφοριών. Αν θέλουμε να δημιουργήσουμε επιτυχημένα συστήματα πρέπει να αντιμετωπίσουμε επιτυχώς αυτή την πολυπλοκότητα.

Το ζητούμενο είναι να οργανωθεί η διαδικασία σχεδιασμού με τρόπο ώστε οι αναλυτές, οι πελάτες, οι προγραμματιστές και άλλοι εμπλεκόμενοι στην ανάπτυξη του συστήματος να μπορούν να κατανοήσουν και να συμφωνήσουν πάνω σε κάτι. Η UML παρέχει αυτή τη δυνατότητα.

Όπως δε θα χτίζαμε μια πολύπλοκη κατασκευή όπως ένα κτίριο δίχως πρώτα να κατασκευάσουμε ένα αναλυτικό σχέδιο, έτσι δεν κατασκευάζουμε ένα πολύπλοκο σύστημα για να τοποθετηθεί σ'εκείνο το κτίριο δίχως πρώτα να συνθέσουμε ένα αναλυτικό πλάνο. Το πλάνο πρέπει να είναι τέτοιο ώστε να μπορούμε να το παρουσιάσουμε στον πελάτη, όπως ακριβώς ένας αρχιτέκτονας δείχνει τα σχέδια σε εκείνον που πληρώνει για την κατασκευή του κτιρίου. Αυτό το σχεδιαστικό πλάνο πρέπει να απορρέει από μια προσεκτική ανάλυση των αναγκών του πελάτη.

Οι χρονικοί περιορισμοί που τίθενται στον αναπτυξιακό τομέα αποτελούν ένα άλλο γνώρισμα της ανάπτυξης συστημάτων. Όταν οι προθεσμίες έρχονται η μια μετά την άλλη, ένα «στιβαρό» σχέδιο αποτελεί επιτακτικής αναγκαιότητας.

Οι εταιρικές συγχωνεύσεις είναι ένας ακόμη τομέας της μοντέρνας ζωής απαιτεί «στιβαρό» σχεδιασμό: Όταν μια εταιρεία συγχωνεύεται με μια άλλη, η νέα οργάνωση μπορεί να αλλάξει σημαντικές πτυχές ενός εν ενεργεία αναπτυξιακού έργου (το

εργαλείο εφαρμογής, η γλώσσα προγραμματισμού, και άλλα). Ένα «αδιαπέραστο» σχέδιο έργου θα διευκολύνει την αλλαγή. Εάν ο σχεδιασμός είναι σωστός, μια αλλαγή στην εφαρμογή μπορεί να ενσωματωθεί ομαλά.

Η ανάγκη για σωστό σχεδιασμό επέφερε με τη σειρά της μια ανάγκη για σχεδιαστικές σημειώσεις που οι αναλυτές, οι προγραμματιστές και οι πελάτες θα αποδεχθούν ως πρότυπο (standard) - ακριβώς όπως οι σημειώσεις στα σχηματικά διαγράμματα αποτελούν πρότυπα για τους ηλεκτρονικούς μηχανικούς και οι σημειώσεις στα διαγράμματα Feynman αποτελούν πρότυπα για τους φυσικούς. Η UML επέχει τη θέση αυτών των σημειώσεων.

5.1.2 Πώς αναπτύχθηκε η UML

Η UML είναι πνευματική δημιουργία των Grady Booch, James Rumbaugh, και Ivar Jacobson. Γνωστοί ως " the Three Amigos", εργάστηκαν σε διαφορετικούς οργανισμούς μέσα στη δεκαετία του '80 και στις αρχές του '90, ο καθένας αναπτύσσοντας τη δική του μεθοδολογία για την αντικειμενοστρεφή ανάλυση και σχεδιασμό. Οι μεθοδολογίες τους ξεπέρασαν αυτές των περισσότερων ανταγωνιστών τους. Στα μέσα της δεκαετίας του '90, άρχισαν να δανείζονται ιδέες ο ένας απ'τον άλλον, οπότε αποφάσισαν να εξελίξουν τη δουλειά τους από κοινού.

Το 1994 ο Rumbaugh έγινε μέλος της Rational Software Corporation, όπου ο Booch εργαζόταν ήδη. Ο Jacobson προσχώρησε ένα χρόνο μετά.

Πρώϊμες εκδόσεις της UML άρχισαν να κυκλοφορούν στη διαδικασία λογισμικού, πράγμα που επέφερε σημαντικές αλλαγές. Επειδή πολλοί οργανισμοί ένιωσαν πως η UML θα υπηρετούσε τους στρατηγικούς τους στόχους, μια κοινοπραξία άρχισε να ανθίζει. Τα μέλη συμπεριλάμβαναν τις DEC, Hewlett-Packard, Intellicorp, Microsoft, Oracle, Texas Instruments, Rational, και άλλες εταιρείες. Το 1997 η κοινοπραξία. παρήγαγε την έκδοση 1.0 της UML και την κατέθεσε στο Object Management Group (OMG) ανταποκρινόμενο στην επιθυμία του OMG για μια standard γλώσσα μοντελοποίησης.

Η κοινοπραξία επεκτάθηκε, ανέπτυξε την έκδοση 1.1 και την κατέθεσε στον OMG, που την υιοθέτησε στο τέλος του 1997. Ο OMG ανέλαβε την συντήρηση της UML και ανέπτυξε δυο ακόμα εκδόσεις το 1998. Η UML αποτελεί ένα de-facto standard στη βιομηχανία λογισμικού, και συνεχίζει να εξελίσσεται. Οι εκδόσεις 1.3, 1.4 και 1.5 έχουν ήδη υλοποιηθεί και ο OMG πρόσφατα ανέπτυξε και την έκδοση 2.0. Οι προηγούμενες εκδόσεις 1.x, έχουν αποτελέσει τη βάση των περισσότερων μοντέλων και βιβλίων μοντέλων.

5.1.3 Κατανοώντας την UML

Η UML δεν είναι τίποτα άλλο από μια γλώσσα . Δεν είναι ένας τρόπος για να σχεδιάσεις ένα σύστημα αλλά ένας τρόπος για να μοντελοποίησεις ένα σύστημα . Για να χρησιμοποιήσεις την UML χρειάζεται να εφαρμοστεί μια μέθοδος . Υπάρχουν

πολλές μέθοδοι που έχουν δημιουργηθεί γι' αυτό το λόγο με πιο γνωστή και πιθανότατα και πρώτη που ασχολήθηκε με την UML την Rational Unified Process .
Η UML είναι μια σημειογραφική και σημειολογική γλώσσα που μπορεί να εφαρμοστεί σε κάθε μορφής ανάπτυξη λογισμικού και όχι μόνο. Η UML χρησιμοποιεί διαφόρους τύπους σχεδιαγραμμάτων που μπορούν να χρησιμοποιηθούν για τη μοντελοποίηση αντικειμενοστρεφών συστημάτων λογισμικού.

5.1.3.1 Στοιχεία της UML

Η UML αποτελείται από έναν αριθμό γραφικών στοιχείων που συνδυάζονται για να σχηματίσουν διαγράμματα. Επειδή η UML είναι γλώσσα, έχει κανόνες για να συνδυάζει αυτά τα στοιχεία. Ίσως είναι προτιμότερο να μη σταθούμε σ' αυτά τα στοιχεία και τους κανόνες αλλά να μελετήσουμε κατ' ευθείαν τα διαγράμματα γιατί αυτά είναι που χρησιμοποιούμε για να κάνουμε την ανάλυση συστημάτων.

Ο σκοπός αυτών των διαγραμμάτων είναι να παρουσιάσουν διάφορες όψεις του συστήματος. Αυτό το σύνολο των πολλαπλών όψεων λέγεται μοντέλο. Ένα μοντέλο συστήματος UML είναι κάτι σαν ένα μοντέλο ενός κτιρίου μαζί με την άποψη ενός καλλιτέχνη για το κτίριο. Είναι σημαντικό να σημειώσουμε πως ένα μοντέλο UML περιγράφει τι πρέπει να κάνει ένα σύστημα, όχι όμως και το πώς θα υλοποιηθεί.

5.1.4 Αναγνωρίζοντας τα επί μέρους τμήματα της UML

5.1.4.1 Δομημένα διαγράμματα

5.1.4.1.1 Διαγράμματα αντικειμένων και κλάσεων

Τα διαγράμματα κλάσης χρησιμοποιούνται για να δείξουν τα διαφορετικά τμήματα του σχεδίου, τις σχέσεις μεταξύ των τμημάτων και σε ποια υποσυστήματα ανήκουν. Τα διαγράμματα κλάσης περιλαμβάνουν και λειτουργίες και χαρακτηριστικά καθώς και πολλούς τύπους ρόλων και συσχετίσεων.
Ένα διάγραμμα αντικειμένων είναι σε πολλά σημεία όμοιο με ένα διάγραμμα κλάσης με την διαφορά ότι στην θέση των κλάσεων έχουμε αντικείμενα που είναι στιγμιότυπα των κλάσεων. Αυτά τα διαγράμματα έχουν να κάνουν συνήθως με σχεδιασμό με την χρήση παραδειγμάτων. Με άλλα λόγια, τα αντικείμενα έχουν να κάνουν με πιο συγκεκριμένα θέματα σε αντίθεση με τις κλάσεις που είναι πιο γενικές.
Ως βάσεις για κάθε αντικειμενοστρεφές σύστημα, οι κλάσεις και τα αντικείμενα παρέχουν εύκολη διαχείριση των πληροφοριών τους. Οι κλάσεις χρησιμοποιούνται για την μοντελοποίηση στα αρχικά στάδια της φάσης επεξεργασίας αλλά και για το χτίσιμο περιπλοκών τμημάτων του λογισμικού μοντέλου σε μεταγενέστερα στάδια της ίδιας φάσης επεξεργασίας .

5.1.4.1.2 Διαγράμματα ανάπτυξης και συστατικών

Ένα διάγραμμα ανάπτυξης δείχνει που θα καταλήξουν τα συστατικά του συστήματος αφού εγκατασταθούν στο σύστημα και πως θα αλληλεπιδράσουν με αυτό. Ενώ το διάγραμμα συστατικών δείχνει πώς τα συστατικά ενός συστήματος αλληλεπιδρούν μεταξύ τους . Ένα τέτοιο διάγραμμα παρουσιάζει τις συσχετίσεις μεταξύ βασικών αρχείων και κλάσεων καθώς και των συστατικών στα οποία ανήκουν . Στην δεύτερη έκδοση της UML τα διαγράμματα συστατικών έχουν ενταχθεί στην κατηγορία των βασικών διαγραμμάτων, πράγμα που δείχνει πόσο σημαντικά θεωρούνται στην μοντελοποίηση σήμερα .

5.1.4.2 Διαγράμματα συμπεριφοράς

5.1.4.2.1 Διαγράμματα σεναρίων

Τα διαγράμματα σεναρίων περιέχουν ρόλους και τις σχέσεις μεταξύ αυτών των ρολών .Τα διαγράμματα σεναρίων είναι η αρχή της φάσης της ανάλυσης για το σχεδιασμό ενός συστήματος . Αρχική επινόηση του Invar Jacobson , οι σχέσεις είναι η βάση των διαγραμμάτων σεναρίων . Ενώνονται με τις συσχετίσεις και καταλήγουν στους ρόλους με σκοπό να δείξουν τη συνολική δομή και διαθεσιμότητα του συστήματος στους μη ειδικούς αναγνώστες του μοντέλου καθώς και στους χρηστές .

5.1.4.2.2 Διαγράμματα δραστηριότητας

Τα διαγράμματα δραστηριοτήτων χρησιμοποιούνται για να αναλύσουμε τη συμπεριφορά μέσα σε περίπλοκες σχέσεις και να δείξουμε και την αλληλεπίδραση με άλλες σχέσεις. Τα διαγράμματα δραστηριότητας έχουν πολλές ομοιότητες με τα διαγράμματα σεναρίων όσο αφορά το ότι δείχνουν την ροή των πληροφοριών .

5.1.4.2.3 Διαγράμματα ακολουθίας

Τα διαγράμματα ακολουθίας χρησιμοποιούνται για να δείξουν την επίδραση μεταξύ των ρολών ενός συστήματος και των αντικειμένων του συστήματος . Μηνύματα στέλνονται από τους ρόλους στα αντικείμενα και από αντικείμενα σε αντικείμενα και τέλος από αντικείμενα πάλι πίσω στους ρόλους έτσι ώστε να φαίνεται η ροή του ελέγχου μέσα σε ένα σύστημα . Τα διαγράμματα ακολουθίας επαληθεύουν τα σενάρια με το να δείχνουν με ποιό τρόπο το κάθε σενάριο λειτουργεί στο σύστημα.

5.1.4.2.4 Διαγράμματα επικοινωνίας

Τα διαγράμματα επικοινωνίας φέρνουν τα διαγράμματα κλάσης στο επόμενο επίπεδο. Περιγράγουν τις επιδράσεις και τις συσχετίσεις μεταξύ των αντικειμένων που δημιουργηθήκαν σε προηγούμενες φάσεις τις μοντελοποίησης του συστήματος . Αυτά τα διαγράμματα μπορούν να χρησιμοποιηθούν και για να μοντελοποιήσουν και μηνύματα μεταξύ διαφορετικών αντικειμένων .

5.1.4.2.5 Διαγράμματα καταστάσεων

Ένα διάγραμμα κατάστασης χρησιμοποιείται για να μοντελοποίησει τη συμπεριφορά των υποσυστημάτων , να μοντελοποίησει τις σχέσεις των κλάσεων και της επαφής του συστήματος. Τα διαγράμματα καταστάσεων είναι μια εξαίρετη οπτική απεικόνισης της ροής μιας εφαρμογής .

5.1.5 Ο σκοπός των διαγραμμάτων

Τα διαγράμματα της UML καθιστούν εφικτό να εξεταστεί ένα σύστημα από διάφορες σκοπιές. Είναι σημαντικό να σημειώσουμε ότι δεν είναι απαραίτητο να εμφανίζονται όλα τα διαγράμματα σε κάθε μοντέλο UML. Τα περισσότερα μοντέλα UML στην πραγματικότητα περιέχουν ένα υποσύνολο των διαγραμμάτων που προαναφέραμε.

Η σημαντικότητα του να παρουσιαστούν διάφορες όψεις του συστήματος έγκειται στο εξής: Τυπικά, ένα σύστημα έχει έναν αριθμό από διαφορετικά άτομα που έχουν διαφορετικά ενδιαφέροντα για διαφορετικά μέρη του συστήματος. Αν για παράδειγμα σχεδιάζουμε τη μηχανή ενός αυτοκινήτου έχουμε τη μία όψη του συστήματος, εάν γράφουμε το εγχειρίδιο χρήσης της συγκεκριμένης μηχανής έχουμε μια άλλη όψη του συστήματος. Και τέλος, εάν σχεδιάζουμε το αυτοκίνητο στο σύνολό του βλέπουμε το σύστημα από μια τελείως διαφορετική σκοπιά, σε σχέση με κάποιον που απλά θα χρησιμοποιήσει το αυτοκίνητο.

Ο συνειδητός σχεδιασμός ενός συστήματος αφορά όλες τις πιθανές όψεις, και κάθε διάγραμμα της UML προσπαθεί να δώσει έναν τρόπο για να περιγραφεί μια συγκεκριμένη όψη του. Ο τελικός σκοπός είναι να επικοινωνήσουν καθαρά μεταξύ τους όλοι οι ενδιαφερόμενοι του συστήματος.

5.2 ΚΑΤΑΝΟΩΝΤΑΣ ΤΗΝ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΕΦΕΙΑ

Η αντικειμενοστρέφεια έχει εισβάλει στοn κόσμο του λογισμικού ως καταιγίδα, πράγμα εύλογο. Ως τρόπος για τη δημιουργία προγραμμάτων, έχει αρκετά πλεονεκτήματα. Ενθαρρύνει μια προσέγγιση στην ανάπτυξη του λογισμικού βασισμένη σε συστατικά στοιχεία ώστε να συντεθεί πρώτα ένα σύστημα με βάση ένα σύνολο κλάσεων. Μετά μπορεί να αναπτυχθεί το σύστημα προσθέτοντας δυνατότητες στα συστατικά στοιχεία που έχουν ήδη κατασκευαστεί ή προσθέτοντας νέα συστατικά στοιχεία. Τέλος, μπορούν να ξαναχρησιμοποιηθούν οι κλάσεις που δημιουργήθηκαν κατά την κατασκευή του νέου συστήματος, μειώνοντας σημαντικά το χρόνο ανάπτυξης του συστήματος.

Αυτό που συνεισφέρει η UML είναι ότι μας επιτρέπει να κατασκευάζουμε εύκολα προς χρήση και κατανόηση μοντέλα αντικειμένων. Οι προγραμματιστές μπορούν να δημιουργήσουν αυτά τα αντικείμενα στο λογισμικό.

Η αντικειμενοστρέφεια είναι ένας τρόπος σκέψης – που εξαρτάται από λίγες βασικές αρχές. Σε αυτό το κεφάλαιο θα εξετάσουμε αυτές τις αρχές. Θα ανακαλύψουμε τι κάνει τα αντικείμενα να «λειτουργούν» και πώς να τα χρησιμοποιούμε στην ανάλυση και στο σχεδιασμό.

5.2.1 Αντικείμενα

Τα αντικείμενα (objects), συγκεκριμένα ή και όχι, είναι παντού γύρω μας. Αποτελούν τον κόσμο μας. Το μοντέρνο λογισμικό προσομοιώνει τον κόσμο – ή ένα μικρό κομμάτι του – έτσι τα προγράμματα μιμούνται συνήθως τα αντικείμενα στον κόσμο. Αν καταλάβουμε μερικές βασικές έννοιες, θα κατανοήσουμε τι πρέπει να εισέλθει στις αναπαραστάσεις του λογισμικού τους και αν το λογισμικό είναι αντικειμενοστρεφές ή όχι. Οι αντικειμενοστρεφείς έννοιες μπορούν να ωφελήσουν τους προγραμματιστές παρέχοντάς τους διορατικότητα για να μοντελοποιήσουν στον τομέα ειδίκευσής τους.

Πρώτον και κύριον, ένα αντικείμενο είναι στιγμιότυπο (instance) μιας κλάσης (μία κατηγορία). Όλοι υπαγόμαστε στην κλάση Άτομο. Ένα αντικείμενο έχει μια δομή. Αυτό σημαίνει ότι έχει ιδιότητες και συμπεριφορά. Η συμπεριφορά ενός αντικειμένου απορρέει από τις λειτουργίες που εκτελεί. Οι ιδιότητες και οι λειτουργίες μαζί ονομάζονται χαρακτηριστικά γνωρίσματα.

Ως αντικείμενα στην κλάση Άτομο, έχουμε αυτές τις ιδιότητες: ύψος, βάρος και ηλικία. (Μπορούμε να φανταστούμε πολλά άλλα). Καθένας από εμάς είναι ξεχωριστός χάρη στις συγκεκριμένες αξίες που έχει ο καθένας μας για αυτές τις ιδιότητες. Εκτελούμε και τις ακόλουθες λειτουργίες: τρώμε, κοιμόμαστε, διαβάζουμε, γράφουμε, μιλάμε, πάμε στη δουλειά και άλλα πολλά – ή, στη γλώσσα των αντικειμένων, τρώμε( ), κοιμόμαστε( ), διαβάζουμε( ), γράφουμε( ), μιλάμε( ) και πάμεΣτηΔουλειά( ) . Αν επρόκειτο να δημιουργήσουμε ένα σύστημα που περιέχει πληροφορίες για τους ανθρώπους – ένα σύστημα κατάστασης μισθοδοσίας ας πούμε ή ένα σύστημα για το Τμήμα Ανθρώπινου Δυναμικού – θα περιλαμβάναμε πιθανόν

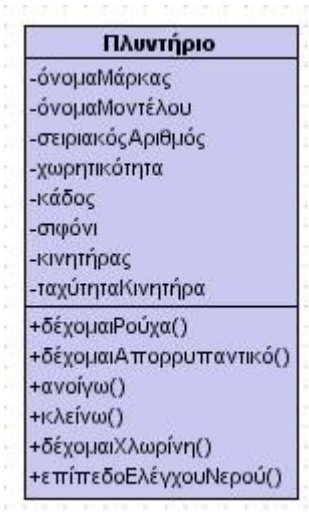
μερικές από αυτές τις ιδιότητες και μερικές από αυτές τις λειτουργίες στο λογισμικό μας.

Στον κόσμο της αντικειμενοστρέφειας, μία κλάση εξυπηρετεί έναν άλλο σκοπό πέρα από την κατηγοριοποίηση. Η κλάση είναι ένα πρότυπο για να κάνουμε αντικείμενα – όπως μια μήτρα για μπισκότα που χρησιμοποιούμε για να χωρίσουμε τα μπισκότα.
Ας πάρουμε για παράδειγμα ένα πλυντήριο. Αν καθορίσουμε ότι η κλάση πλυντήριο (WashingMachine) έχει τις ιδιότητες όνομαΜάρκας, όνομαΜοντέλου, σειριακόςΑριθμός και χωρητικότητα – μαζί με τις λειτουργίες δέχομαιΡούχα **( ),** δέχομαιΑπορρυπαντικό **( ),** ανοίγω **( ),** και κλείνω **( )** – έχουμε ένα μηχανισμό για να προστεθούν νέα στιγμιότυπα στην κλάση πλυντήριο. Αυτό σημαίνει ότι μπορείτε να δημιουργήσετε νέα αντικείμενα βασισμένα σε αυτήν την κλάση.

Αυτό είναι ιδιαιτέρως σημαντικό για την ανάπτυξη του αντικειμενοστρεφούς λογισμικού.

Σχήμα 5.2.1-1: Το εικονίδιο της κλάσης στη UML

Ας θυμηθούμε ότι ο σκοπός της αντικειμενοστρέφειας είναι να αναπτύξει λογισμικό που αντικατοπτρίζει (δηλαδή, μοντελοποιεί) ένα συγκεκριμένο κομμάτι του κόσμου. Όσο περισσότερες ιδιότητες και συμπεριφορές λάβουμε υπόψη, τόσο το μοντέλο σας θα είναι εναρμονισμένο με την πραγματικότητα. Στο παράδειγμα του πλυντηρίου θα έχουμε ένα δυνάμει πιο ακριβές μοντέλο αν συμπεριλάβουμε και τις ιδιότητες: κάδος, σιφόνι, κινητήρας και ταχύτηταΚινητήρα. Μπορούμε επίσης να αυξήσουμε την ακρίβεια του μοντέλου αν συμπεριλάβουμε λειτουργίες όπως δέχομαιΧλωρίνη ( ) και επίπεδοΕλέγχουΝερού ( ).



Σχήμα 5.2.1-2: Προσθέτοντας επιπλέον λειτουργίες

5.2.2 Αντικειμενοστρεφείς έννοιες

Η αντικειμενοστρέφεια πάει πιο πέρα από το να μοντελοποιεί απλώς ιδιότητες και συμπεριφορές. Αυτές οι πτυχές ονομάζονται *αφαίρεση*, *κληρονομικότητα*, *πολυμορφισμός* και *ενθυλάκωση*. Τρία άλλα σημαντικά μέρη της αντικειμενοστρέφειας είναι η *αποστολή μηνύματος*, *η διασύνδεση* και *η συνάθροιση*. Ας εξετάσουμε την καθεμιά έννοια ξεχωριστά.

5.2.2.1 Αφαίρεση

Αφαίρεση (abstraction) σημαίνει απλώς να αφαιρέσουμε τις ιδιότητες και τις λειτουργίες ενός αντικειμένου μέχρι να απομείνουν μόνο αυτές που χρειάζονται. Τι σημαίνει «μόνο αυτές που χρειάζονται»;

Διαφορετικοί τύποι προβλημάτων απαιτούν διαφορετική ποσότητα πληροφοριών, ακόμη κι αν αυτά τα προβλήματα βρίσκονται στην ίδια γενική περιοχή. Στο δεύτερο βήμα της κατασκευής μιας κλάσης ενός πλυντηρίου, προέκυψαν περισσότερες ιδιότητες και λειτουργίες από το πρώτο βήμα. Άξιζε τον κόπο;

Αν είμαστε μέλη μιας ομάδας ανάπτυξης που εν τέλει πρόκειται να δημιουργήσει ένα πρόγραμμα υπολογιστή το οποίο εξομοιώνει πραγματικά πώς ένα πλυντήριο κάνει ό,τι κάνει, σίγουρα αξίζει τον κόπο. Ένα τέτοιο πρόγραμμα (που ίσως είναι χρήσιμο για σχεδιαστές μηχανικούς που κατασκευάζουν ένα πλυντήριο) πρέπει να αποτελείται από αρκετά στοιχεία ώστε να γίνονται προβλέψεις ακριβείας ως προς τι θα συμβεί όταν κατασκευαστεί το πλυντήριο, σε πλήρη λειτουργία και ενώ πλένει ρούχα. Για αυτού του είδους το πρόγραμμα, μάλιστα, μπορείτε να αφαιρέσετε το γνώρισμα σειριακόςΑριθμός γιατί πιθανόν δε θα είναι ιδιαίτερα χρήσιμος.

Κι αν, από την άλλη, πρόκειται να δημιουργήσουμε ένα λογισμικό για να εντοπίσουμε τις δοσοληψίες σε μια πλύση που έχει ένας μεγάλος αριθμός πλυντηρίων; Σε αυτό το πρόγραμμα δε θα χρειαστούμε μάλλον τις λεπτομερείς ιδιότητες και τις λειτουργίες που αναφέρθηκαν στην προηγούμενη ενότητα. Ίσως, παρόλ' αυτά, χρειαστεί να συμπεριλάβουμε το σειριακόΑριθμό κάθε πλυντηρίου-αντικειμένου.

Σε κάθε περίπτωση, ό,τι απομείνει, αφού πάρουμε τις αποφάσεις μας ως προς τι θα συμπεριλάβουμε και τι όχι, είναι η αφαίρεση στην περίπτωση ενός πλυντηρίου.
Μερικές αυθεντίες υποστηρίζουν ότι η αφαίρεση – δηλαδή το να ξέρει κανείς τι να συμπεριλάβει σε ένα μοντέλο και τι να αφαιρέσει – είναι η πιο κρίσιμη επιδεξιότητα ενός μοντελιστή.

5.2.2.2 Κληρονομικότητα

Πλυντήρια, ψυγεία, φούρνοι μικροκυμάτων, τοστιέρες, πλυντήρια πιάτων, ραδιόφωνα, μηχανές για βάφλες, μπλέντερ και σίδερα είναι όλα συσκευές. Στον κόσμο της αντικειμενοστρέφειας, θα λέγαμε ότι καθένα από αυτά είναι υποκατηγορία

της κλάσης Συσκευές. Ένας άλλος τρόπος για να πούμε ότι οι Συσκευές είναι υπερκατηγορία όλων των άλλων.

Η κλάση Συσκευές έχει τις ιδιότητες ανοιχτόςΚλειστόςΔιακόπτης και ηλεκτρικόΚαλώδιο και τις λειτουργίες ανοίγω **( )** κλείνω **( )**. Συνεπώς, αν ξέρετε ότι κάτι είναι συσκευή, τότε ξέρετε αμέσως ότι έχει τις ιδιότητες και τις λειτουργίες της κλάσης Συσκευές.

Η αντικειμενοστρέφεια αναφέρεται σε αυτή τη σχέση ως κληρονομικότητα (inheritance). Κάθε υποκλάση των Συσκευών (πλυντήριο, ψυγείο, μπλέντερ και ούτω καθεξής) κληρονομεί τα χαρακτηριστικά γνωρίσματα των συσκευών. Είναι σημαντικό να αναφέρουμε ότι κάθε υποκατηγορία προσθέτει τις δικές της ιδιότητες και λειτουργίες.
Η κληρονομικότητα δε χρειάζεται να σταματήσει εδώ. Η συσκευή για παράδειγμα είναι μια υποκατηγορία της κλάσης ΑντικείμεναΝοικοκυριού. Το Έπιπλο είναι μια άλλη υπερκατηγορία του ΑντικείμεναΝοικοκυριού. Το Έπιπλο έχει βεβαίως τις δικές του υποκατηγορίες.

5.2.2.3 Πολυμορφισμός

Μερικές φορές μια λειτουργία έχει το ίδιο όνομα σε διαφορετικές κλάσεις. Για παράδειγμα, μπορείτε να ανοίξετε μια πόρτα, μπορείτε να ανοίξετε ένα παράθυρο και μπορείτε να ανοίξετε μια εφημερίδα, ένα δώρο, έναν τραπεζικό λογαριασμό ή μια συζήτηση. Σε κάθε περίπτωση, εκτελείτε μια διαφορετική λειτουργία. Στην αντικειμενοστρέφεια, η κάθε κλάση ξέρει πώς αυτή η λειτουργία πρέπει να λάβει χώρα. Αυτό ονομάζεται πολυμορφισμός (polymorphism).

Με μια πρώτη ματιά, θα φαινόταν ότι αυτή η έννοια είναι πιο σημαντική για την ανάπτυξη του λογισμικού παρά για τη μοντελοποίηση. Ούτως ή άλλως, στην ανάπτυξη λογισμικού πρέπει κανείς να φτιάξει το λογισμικό που εφαρμόζει αυτές τις μεθόδους σε προγράμματα κομπιούτερ και πρέπει να έχει επίγνωση σημαντικών διαφορών ανάμεσα στις λειτουργίες με το ίδιο όνομα. Και μπορεί, επίσης, να κατασκευάσει κλάσεις λογισμικού που «ξέρουν» τι πρέπει να κάνουν.

Αλλά ο πολυμορφισμός είναι σημαντικός και για τη μοντελοποίηση γιατί επιτρέπει στο μοντελιστή να μιλήσει στον πελάτη (που είναι εξοικειωμένος με το κομμάτι του κόσμου προς μοντελοποίηση) με τα λόγια και την ορολογία του ίδιου του πελάτη. Μερικές φορές αυτή η ορολογία οδηγεί από φυσικού σε λέξεις λειτουργίας (όπως «ανοίγω») που μπορούν να έχουν παραπάνω από μία σημασία. Ο πολυμορφισμός δίνει τη δυνατότητα στο μοντελιστή να διατηρήσει την ορολογία χωρίς να χρειαστεί να εφευρίσκει τεχνητές λέξεις για να διατηρήσει μια περιττή (και αφύσικη) μοναδικότητα των όρων.

5.2.2.4 Ενθυλάκωση

Σε μια διαφήμιση στην τηλεόραση που προβλήθηκε πριν από μερικά χρόνια, δύο άνθρωποι συζητούν πόσα λεφτά θα γλιτώσουν εάν πληκτρολογήσουν ένα εφταψήφιο πρόθεμα πριν κάνουν ένα μακρινό τηλεφώνημα.
Ένας από αυτούς ρωτάει με δυσπιστία, «Πώς δουλεύει αυτό»;
Ο άλλος απαντάει, «Πώς πετάγεται το ποπ κορν; Ποιος νοιάζεται»;

Αυτή είναι η ουσία της ενθυλάκωσης (encapsulation). Όταν ένα αντικείμενο προχωρά στις λειτουργίες του, αυτές οι λειτουργίες παραμένουν κρυφές. Όταν οι περισσότεροι άνθρωποι παρακολουθούν ένα σώου στην τηλεόραση, συνήθως δεν ξέρουν ή δεν ενδιαφέρονται για τα πολύπλοκα ηλεκτρονικά εξαρτήματα που βρίσκονται πίσω από την οθόνη της τηλεόρασης και όλες τις λειτουργίες που πρέπει να συμβούν για να μπει χρώμα στην εικόνα της οθόνης. Η τηλεόραση κάνει ό,τι κάνει και κρύβει τις διαδικασίες από εμάς. Οι περισσότερες άλλες συσκευές λειτουργούν επίσης με τον ίδιο τρόπο.

Γιατί είναι αυτό σημαντικό; Στον κόσμο του λογισμικού, η ενθυλάκωση βοηθάει στη μείωση της προδιάθεσης να συμβούν άσχημα πράγματα. Σε ένα σύστημα που περιλαμβάνει αντικείμενα, τα αντικείμενα αλληλοεξαρτώνται με διάφορους τρόπους. Εάν ένα από αυτά τύχει να πάθει κάποια βλάβη και πρέπει οι μηχανικοί λογισμικού να το αλλάξουν με κάποιο τρόπο, κρύβοντας τις λειτουργίες του από τα άλλα αντικείμενα σημαίνει ότι πιθανόν δε θα είναι απαραίτητο να αλλάξουν αυτά τα άλλα αντικείμενα.

Περνώντας από το λογισμικό στην πραγματικότητα, βλέπουμε πόσο σημαντική είναι η ενθυλάκωση για τα αντικείμενα με τα οποία δουλεύουμε. Το μόνιτορ του υπολογιστή, κατά μία έννοια, κρύβει τις λειτουργίες του από το CPU του υπολογιστή. Όταν κάτι πάει στραβά με το μόνιτορ μας, είτε το φτιάχνουμε είτε το αντικαθιστουμε. Το πιο πιθανόν είναι ότι δε θα φτιάξουμε ούτε θα αντικαταστήσουμε και το CPU.

Αφού μιλάμε για αυτό το θέμα, ιδού μία συγγενής έννοια. Αφού ενθυλάκωση σημαίνει ότι το αντικείμενο κρύβει τι κάνει από τα άλλα αντικείμενα και από τον έξω κόσμο, η ενθυλάκωση ονομάζεται επίσης απόκρυψη πληροφοριών. Αλλά ένα αντικείμενο σίγουρα πρέπει να παρουσιάσει ένα «πρόσωπο» στον έξω κόσμο ώστε να μπορούμε να μυηθούμε σε αυτές τις λειτουργίες. Η τηλεόραση έχει, για παράδειγμα, κουμπιά είτε πάνω στην τηλεόραση είτε σε ένα τηλεκοντρόλ. Ένα πλυντήριο έχει δείκτες που σας δίνουν τη δυνατότητα να ρυθμίσετε τη θερμοκρασία και τη στάθμη του νερού. Τα κουμπιά της τηλεόρασης και οι δείκτες του πλυντηρίου ονομάζονται διεπαφές.

5.2.2.5 Αποστολή μηνύματος

Σε ένα σύστημα τα αντικείμενα συνεργάζονται. Αυτό γίνεται στέλνοντας μηνύματα το ένα στο άλλο. Ένα αντικείμενο στέλνει σε ένα άλλο ένα μήνυμα – ένα αίτημα να εκτελέσει μια λειτουργία ενώ το αντικείμενο που δέχεται το μήνυμα εκτελεί αυτή τη λειτουργία.

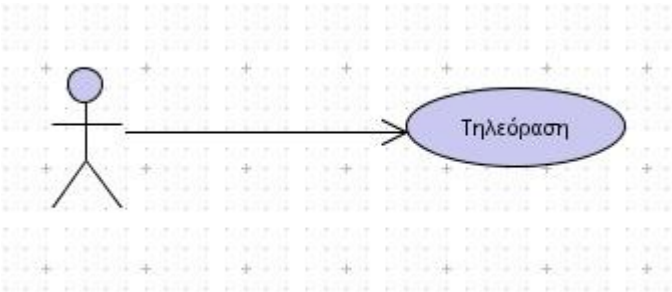
Η τηλεόραση και το τηλεκοντρόλ είναι ένα παράδειγμα. Όταν θέλουμε να παρακολουθήσουμε ένα σώου στην τηλεόραση, ψάχνουμε το τηλεκοντρόλ, καθόμαστε αναπαυτικά στην αγαπημένη μας καρέκλα και πατάμε το κουμπί On. Τι συμβαίνει; Το αντικείμενο-τηλεκοντρόλ στέλνει ένα μήνυμα (στην κυριολεξία!) στο αντικείμενο-τηλεόραση να ανοίξει μόνο του. Το αντικείμενο-τηλεόραση λαμβάνει το μήνυμα, ξέρει πώς να εκτελέσει τη λειτουργία «ανοίγω» και ανοίγει μόνο του. Όταν θέλουμε να παρακολουθήσουμε ένα διαφορετικό κανάλι, πατάμε το κατάλληλο κουμπί στο τηλεκοντρόλ και το αντικείμενο-τηλεκοντρόλ στέλνει ένα διαφορετικό μήνυμα – «άλλαξε το κανάλι» – στο αντικείμενο-τηλεόραση. Το τηλεκοντρόλ μπορεί να επικοινωνήσει με την τηλεόραση μέσω άλλων μηνυμάτων, για να αλλάξει την ένταση του ήχου, να κλείσει τον ήχο, να εγκαταστήσει διαφορετική λεζάντα.

Ας επιστρέψουμε στις διεπαφές για λίγο. Τα περισσότερα πράγματα που κάναμε με το τηλεκοντρόλ, μπορούμε επίσης να τα κάνουμε αν σηκωθούμε από την καρέκλα, πάμε στην τηλεόραση και πατήσουμε τα κουμπιά στην τηλεόραση. Η διεπαφή που η τηλεόραση μας παρουσιάζει (τα κουμπιά) δεν είναι προφανώς η ίδια διεπαφή που παρουσιάζει στο τηλεκοντρόλ (ένας υπέρυθρος δέκτης).

5.2.2.6 Συσχετίσεις

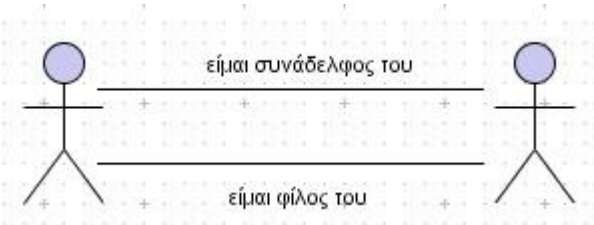
Άλλο ένα κοινό συμβάν είναι ότι τα αντικείμενα σχετίζονται το ένα με το άλλο κατά κάποιο τρόπο. Για παράδειγμα, όταν ανοίγουμε την τηλεόραση, σε αντικειμενοστρεφείς όρους, βρισκόμαστε σε συσχέτιση (association) με την τηλεόραση.

Η συσχέτιση «ανοίγω» είναι μιας κατευθύνσεως, όπως στο σχέδιο. Δηλαδή, ανοίγουμε την τηλεόραση. Άλλες συσχετίσεις όπως «παντρεύομαι» είναι δύο κατευθύνσεων.



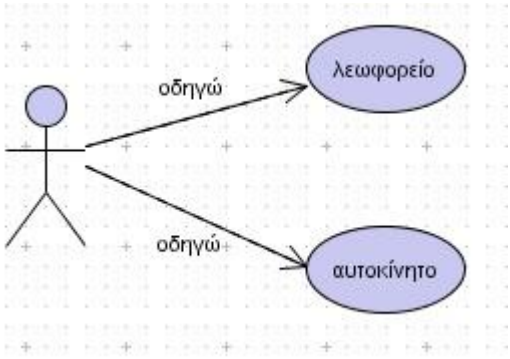
Σχήμα 5.2.2.6-1: Συσχέτιση αντικειμένων

Μερικές φορές ένα αντικείμενο μπορεί να συνδέεται με ένα άλλο με περισσότερους από έναν τρόπο. Αν εσείς και ο συνάδελφός σας είστε φίλοι, αυτό αποτελεί ένα παράδειγμα. Ανήκετε και στη συσχέτιση «είμαι φίλος του» και στη συσχέτιση «είμαι συνάδελφος του», όπως δείχνει το σχέδιο.



Σχήμα 5.2.2.6-2: Συσχέτιση αντικειμένων με περισσότερους τρόπους

Μία κλάση μπορεί να συνδεθεί με περισσότερες από μία κλάσεις. Ένα άτομο μπορεί να οδηγήσει ένα Ι.Χ. αυτοκίνητο και το ίδιο άτομο μπορεί να οδηγήσει και λεωφορείο.



Σχήμα 5.2.2.6-3: Μια κλάση μπορεί να συσχετίζεται με περισσότερες από μία κλάσεις

Η πολλαπλότητα είναι σημαντική πτυχή των συσχετίσεων ανάμεσα στα αντικείμενα. Φανερώνει τον αριθμό των κλάσεων ανάμεσα στα αντικείμενα που σχετίζονται με ένα μοναδικό αντικείμενο της κλάσης συσχέτισης. Για παράδειγμα, σε ένα τυπικό μάθημα στο κολέγιο, κάθε μάθημα διδάσκεται από έναν μόνο διδάσκοντα. Το μάθημα και ο διδάσκων βρίσκονται σε συσχέτιση ένα προς ένα. Σε ένα σεμινάριο, πολλοί διδάσκοντες ίσως διδάξουν το μάθημα στη διάρκεια του εξαμήνου. Σε αυτήν την περίπτωση, το μάθημα και ο διδάσκοντας βρίσκονται σε συσχέτιση ένα προς πολλά.

Μπορούμε να βρούμε πολλά είδη πολλαπλότητας αν ψάξουμε πολύ. Ένα ποδήλατο έχει δύο ρόδες (ένα προς δύο), ένα τρίκυκλο τρεις και ένα 18-τροχο 18.

5.2.2.7 Ενσωμάτωση

Ας δούμε το σύστημα του υπολογιστή μας. Αποτελείται από ένα κουτί, ένα πληκτρολόγιο, ένα ποντίκι, ένα μόνιτορ, ένα CD-Rom, έναν ή περισσότερους σκληρούς δίσκους, ένα μόντεμ, μία υποδοχή για δισκέτα, έναν εκτυπωτή και πιθανόν μερικά ηχεία. Μέσα στο κουτί, μαζί με τα προαναφερθέντα, έχουμε ένα CPU, μία κάρτα γραφικών, μία κάρτα ήχου και κάποια άλλα στοιχεία με τα οποία αδιαμφισβήτητα δε θα μπορούσαμε να ζήσουμε.

Ο υπολογιστής σας είναι μια ενσωμάτωση (aggregation), μια άλλη μορφή διασύνδεσης ανάμεσα στα αντικείμενα. Όπως πολλά πράγματα που αξίζει να έχουμε, ο υπολογιστής είναι κατασκευασμένος από ένα σωρό συστατικά διαφορετικού τύπου. Μπορούμε πιθανόν να σκεφτούμε ένα μεγάλο αριθμό άλλων ενσωματώσεων.

Μία μορφή ενσωμάτωσης αφορά μια στενή σχέση ανάμεσα σε ένα ενσωματωμένο αντικείμενο και τα αντικείμενα-συστατικά του. Αυτό ονομάζεται σύνθεση. Το κλειδί για τη σύνθεση είναι ότι το συστατικό είναι χρήσιμο μόνο μέσα στα πλαίσια του σύνθετου αντικειμένου. Για παράδειγμα, ένα πουκάμισο είναι μια σύνθεση ενός σώματος, ενός κολάρου, μανικιών, κουμπιών, κουμπότρυπων και μανσετών. Ξεφορτωθείτε το πουκάμισο και το κολάρο είναι άχρηστο.

Μερικές φορές το συστατικό στοιχείο σε μια σύνθεση δεν κρατάει όσο η ίδια η σύνθεση. Τα φύλλα σε ένα δέντρο μπορεί να «πεθάνουν» πριν το δέντρο. Αν καταστρέψουμε το δέντρο, τότε και τα φύλλα πεθαίνουν.

Η ενσωμάτωση και η σύνθεση είναι σημαντικές γιατί αντικατοπτρίζουν εξαιρετικά κοινά γεγονότα κι έτσι μας βοηθούν να δημιουργήσουμε μοντέλα που πλησιάζουν στην πραγματικότητα.

5.2.3 Τα οφέλη της αντικειμενοστρέφειας

Τα αντικείμενα και η διασύνδεσή τους αποτελούν τη ραχοκοκαλιά των λειτουργικών συστημάτων. Για να μοντελοποιήσουμε αυτά τα συστήματα, πρέπει να καταλάβουμε τι είναι αυτές οι διασυνδέσεις. Αν γνωρίζουμε τους πιθανούς τύπους διασύνδεσης, θα είμαστε καλά εφοδιασμένοι με τα απαραίτητα τεχνάσματα που θα χρειαστούμε όταν μιλάμε με πελάτες για τις ανάγκες τους, όταν θα συγκεντρώνουμε τα αιτήματά τους και δημιουργούμε μοντέλα των συστημάτων που τους βοηθούν να έρθουν αντιμέτωποι με τις επαγγελματικές τους προκλήσεις.

Είναι σημαντικό να χρησιμοποιούμε τις έννοιες της αντικειμενοστρέφειας για να μας βοηθήσει να καταλάβουμε τις γνώσεις του πελάτη (το πεδίο ορισμού του ή της) και να γίνονται οι διευκρινίσεις στον πελάτη με όρους που καταλαβαίνει.
Εδώ είναι που επεμβαίνει η UML. Στα επόμενα κεφάλαια, θα μάθουμε πώς να χρησιμοποιούμε τη UML για να οπτικοποιήσουμε τις έννοιες που μάθαμε σε αυτό το κεφάλαιο.

5.2.4 Περίληψη

Η αντικειμενοστρέφεια είναι ένας τρόπος σκέψης που εξαρτάται από ορισμένες αρχές. Ένα αντικείμενο είναι ένα στιγμιότυπο μιας κλάσης. Μια κλάση είναι μια γενική κατηγορία αντικειμένων που έχουν τις ίδιες ιδιότητες και λειτουργίες. Όταν δημιουργούμε ένα αντικείμενο, η προβληματική περιοχή στην οποία δουλεύουμε καθορίζει πόσες ιδιότητες και πόσες λειτουργίες πρέπει να λάβουμε υπόψη.

Η κληρονομικότητα είναι μια σημαντική πτυχή της αντικειμενοστρέφειας: ένα αντικείμενο κληρονομεί τις ιδιότητες και λειτουργίες της κλάσης του. Ένα αντικείμενο μπορεί επίσης να κληρονομήσει ιδιότητες και λειτουργίες από μια άλλη κλάση.

Ο πολυμορφισμός είναι μια άλλη σημαντική πτυχή. Καθορίζει ότι μια λειτουργία μπορεί να έχει το ίδιο όνομα σε διαφορετικές κλάσεις και κάθε κλάση θα εκτελέσει τη λειτουργία με διαφορετικό τρόπο.
Τα αντικείμενα κρύβουν την εκτέλεση των λειτουργιών τους από άλλα αντικείμενα και από τον έξω κόσμο. Κάθε αντικείμενο παρουσιάζει μια διεπαφή ώστε τα άλλα αντικείμενα (και οι άνθρωποι) να μπορούν να το κάνουν να εκτελέσει τις λειτουργίες του.

Τα αντικείμενα συνεργάζονται στέλνοντας μηνύματα το ένα στο άλλο. Τα μηνύματα είναι αιτήματα για την εκτέλεση λειτουργιών.
Τα αντικείμενα είναι τυπικά συνδεδεμένα μεταξύ τους. Η διασύνδεση μπορεί να πάρει διαφορετικές μορφές. Ένα αντικείμενο σε μια κλάση μπορεί να συνδεθεί με έναν αριθμό αντικειμένων σε μια άλλη.

Η συνάθροιση είναι ένα είδος διασύνδεσης. Ένα συναθροιζόμενο αντικείμενο αποτελείται από σύνθετα αντικείμενα. Η σύνθεση είναι ένα ειδικό είδος συνάθροισης. Σε ένα σύνθετο αντικείμενο τα συστατικά στοιχεία υπάρχουν μόνο ως μέρη της σύνθεσης.

5.3 ΔΟΥΛΕΥΟΝΤΑΣ ΜΕ ΤΗΝ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΕΦΕΙΑ

Τώρα είναι ώρα να βάλουμε μαζί τη UML και τις αντικειμενοστρεφείς έννοιες που μάθαμε στο προηγούμενο Κεφάλαιο. Σε αυτό το κεφάλαιο θα σταθεροποιήσουμε τις γνώσεις μας για την αντικειμενοστρέφεια ενώ θα μαθαίνουμε όλο και περισσότερα για τη UML.

5.3.1 Αναπαράσταση μιας κλάσης

Όπως αναφέραμε στο πρώτο Κεφάλαιο, ένα ορθογώνιο είναι το εικονίδιο που αντιπροσωπεύει μια κλάση στη UML. Θυμηθείτε ότι το όνομα της κλάσης είναι, από σύμβαση, μια λέξη με ένα αρχικό κεφαλαίο γράμμα. Εμφανίζεται κοντά στην κορυφή του ορθογωνίου. Αν η κλάση μας αποτελείται από δύο λέξεις, ενώνουμε τις δύο λέξεις και γράφουμε με κεφαλαίο το πρώτο γράμμα της δεύτερης λέξης.

Άλλο ένα δομικό στοιχείο της UML, το πακέτο, μπορεί να παίξει ένα σημαντικό ρόλο στην ονομασία της κλάσης. Ένα πακέτο είναι ο τρόπος που η UML οργανώνει τα στοιχεία ενός διαγράμματος. Όπως ίσως θυμάστε, η UML αντιπροσωπεύει ένα πακέτο ως ένα στηλοθετημένος φάκελος. Το όνομα του πακέτου είναι ένα αλφαριθμητικό κείμενο.

Σχήμα 5.3.1-1: Πάνω: Το εικονίδιο μιας κλάσης
Κάτω: Ένα πακέτο

Αν η κλάση πλυντήριο είναι μέρος ενός πακέτου που ονομάζεται Νοικοκυριό, μπορείτε να του δώσετε το όνομα Νοικοκυριό :: πλυντήριο. Οι διπλές άνω κάτω τελείες χωρίζουν το όνομα του πακέτου στα αριστερά από το όνομα της κλάσης στα δεξιά. Αυτό το όνομα κλάσης ονομάζεται διαδρομή (βλ. *Σχήμα 5.3.1-2*)

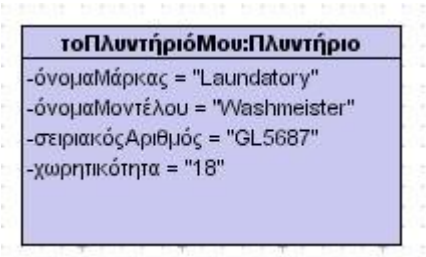
Σχήμα 5.3.1-2: Μια κλάση με μια διαδρομή

5.3.1.1 Ιδιότητες

Μια ιδιότητα (attribute) είναι χαρακτηριστικο μιας κλάσης. Περιγράφει μια αλληλουχία τιμών που η ιδιότητα μπορεί να συγκρατεί στα αντικείμενα (δηλαδή στα στιγμιότυπα) αυτής της κλάσης. Μια κλάση μπορεί να έχει μηδέν ή περισσότερες ιδιότητες. Από σύμβαση, το όνομα του ιδιότητας που αποτελείται από μία λέξη γράφεται με μικρά γράμματα. Αν το όνομα αποτελείται από περισσότερες από μία λέξεις, τότε οι λέξεις ενώνονται και κάθε μία λέξη εκτός της πρώτης ξεκινά με κεφαλαίο γράμμα. Η λίστα των ονομάτων των ιδιοτήτων ξεκινάει κάτω από μια γραμμή που τα χωρίζει από το όνομα της κλάσης, όπως δείχνει το *Σχήμα 5.3.1.1-1* .

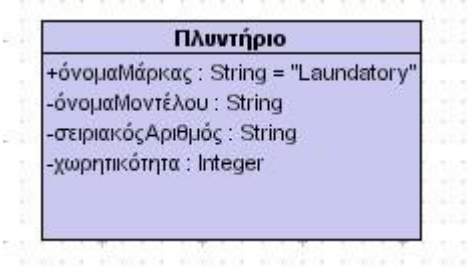
Σχήμα 5.3.1.1-1: Μια κλάση και οι ιδιότητές της

Κάθε αντικείμενο της κλάσης έχει συγκεκριμένη τιμή για κάθε ιδιότητα. Το σχέδιο παρουσιάζει ένα παράδειγμα. Προσέξτε ότι το όνομα του αντικειμένου ξεκινάει με μικρό γράμμα, προηγείται μιας άνω και κάτω τελείας που προηγείται του ονόματος της κλάσης, και όλο το όνομα είναι υπογραμμισμένο.



Σχήμα 5.3.1.1-2: Ένα αντικείμενο έχει συγκεκριμένες τιμές για καθεμία από τις ιδιότητές του

Η UML μας δίνει την επιλογή να υποδείξουμε επιπλέον πληροφορίες στις ιδιότητες . Στο εικονίδιο για την κλάση, μπορούμε να συγκεκριμενοποιήσουμε ένα τύπο για την αξία κάθε ιδιότητας. Πιθανοί τύποι περιλαμβάνουν αλφαρίθμους, κινητούς δεκαδικούς, ακέραιους και λογική μεταβλητή (και άλλοι αριθμοί). Για να υποδείξουμε ένα τύπο, χρησιμοποιούμε την άνω και κάτω τελεία για να χωρίσουμε το όνομα του ιδιοχαρακτηριστικού από τον τύπο. Μπορούμε επίσης να υποδείξουμε μια προεπιλεγμένη τιμή για μια ιδιότητα. Το *Σχήμα 5.3.1.1-3* δείχνει αυτούς τους τρόπους για να καθορίσουμε ιδιότητες.



Σχήμα 5.3.1.1-3: Μια ιδιότητα μπορεί να εμφανίζει τον τύπο της καθώς και μια προκαθορισμένη τιμή

Ονομάζοντας τις τιμές **à** Ένας αριθμητικός τύπος είναι ένας τύπος δεδομένων που καθορίζεται από μια λίστα ονοματοποιημένων τιμών. Οι λογικές μεταβλητές, για παράδειγμα, είναι ένας αριθμητικός τύπος γιατί περιλαμβάνει τιμές «σωστές» και «λαθεμένες». Μπορούμε να καθορίσουμε τους δικούς μας αριθμητικούς τύπους όπως Κατάσταση, που περιλαμβάνει τιμές όπως «στερεός», «υγρός» και «αέριος».

5.3.1.2 Λειτουργίες

Μία λειτουργία (Operation) είναι κάτι που μπορεί να κάνει μια κλάση κι εκ τούτου κάτι που εσείς (ή μια άλλη κλάση) μπορεί να ζητήσει από μια άλλη κλάση να κάνει. Όπως ένα ιδιοχαρακτηριστικό όνομα, το όνομα μιας λειτουργίας γράφεται με μικρό αν αποτελείται από μία μόνο λέξη. Αν το όνομα αποτελείται από περισσότερες από μία λέξη, ενώστε τις λέξεις και ξεκινούν όλες τις λέξεις μετά την πρώτη με κεφαλαίο. Η λίστα των λειτουργιών ξεκινάει με μια γραμμή που χωρίζει τις λειτουργίες από τις ιδιότητες, όπως φαίνεται στο σχέδιο *Σχήμα 5.3.1.2-1.*

Σχήμα 5.3.1.2-1: Οι λειτουργίες μιας κλάσης εμφανίζονται κάτω από τις ιδιότητές της.

Ακριβώς όπως υποδεικνύουμε επιπλέον πληροφορίες στις ιδιότητες, μπορούμε να υποδείξουμε επιπλέον χαρακτηριστικά στις λειτουργίες. Στις παρενθέσεις που ακολουθούν το όνομα της λειτουργίας, μπορούμε να δείξουμε την παράμετρο στην οποία επηρεάζει τη λειτουργία, μαζί με τον τύπο της παραμέτρου. Ένα είδος παραμέτρου, η συνάρτηση, επιστρέφει μία τιμή αφού κάνει τη δουλειά της. Για μια συνάρτηση, μπορούμε να δείξουμε την τιμή που επιστρέφει και τον τύπο της τιμής. Αυτές οι πληροφορίες για τη λειτουργία ονομάζονται υπογραφή της λειτουργίας. Το *Σχήμα 5.3.1.2-2* δείχνει διάφορους τρόπους για να αναπαραστήσετε την υπογραφή. Οι δύο πρώτες λειτουργίες δείχνουν τον τύπο της παραμέτρου. Η τρίτη και η τέταρτη δείχνουν τον τύπο της επιστρεφόμενης τιμής.

Σχήμα 5.3.1.2-2: Υπογραφές για τις λειτουργίες

5.3.1.3 Ιδιότητες, Λειτουργίες και Οπτικοποίηση

Ασχολούμαστε με κλάσεις σε απομόνωση και δείχνουμε όλες τις ιδιότητες και τις λειτουργίες μίας κλάσης. Στην πράξη, παρόλ' αυτά, θα δείξουμε περισσότερες από μία κλάσεις τη στιγμή. Όταν το κάνουμε αυτό, είναι τυπικά περιττό να εκθέτουμε όλες τις ιδιότητες και τις λειτουργίες. Αν το κάνεουμε αυτό, τότε το διάγραμμα θα είναι πιθανόν «φορτωμένο». Αντί αυτού, μπορούμε απλά να δείξουμε το όνομα της κλάσης και να αφήσετε είτε την περιοχή των ιδιοχαρακτηριστικών είτε των λειτουργιών κενή (ή να αφήσουμε και τις δύο κενές), όπως φαίνεται παρακάτω.

Σχήμα 5.3.1.3-1: Στην πράξη εν δείχνονται πάντοτε όλες οι ιδιότητες και λειτουργίες μιας κλάσης

Μερικές φορές ίσως είναι χρήσιμο να δείξουμε μερικές (αλλά όχι όλα) από τις ιδιότητες και τις λειτουργίες. Για να υποδείξουμε ότι έχουμε δείξει μερικά από αυτά, μετά τη λίστα ακολουθούν τρεις τελείες «…». Αυτό ονομάζεται έλλειψη και η παράλειψη ενός ή όλων των ιδιοχαρακτηριστικών ή των λειτουργιών ονομάζεται έκθλιψη μιας κλάσης. Το σχέδιο *Σχήμα 5.3.1.3-2* δείχνει τη χρήση της έλλειψης.

Σχήμα 5.3.1.3-2: Η χρήση της έλλειψης

Αν έχουμε μια μεγάλη λίστα από ιδιότητες ή λειτουργίες, μπορούμε να χρησιμοποιήσουμε μια λέξη-κλειδί για να τα οργανώσουμε με τέτοιο τρόπο ώστε να

κάνουμε τη λίστα κατανοητή. Μία λέξη-κλειδί περιβάλλεται από δύο ζευγάρια εισαγωγικών που ονομάζονται ***guillemets.*** (Σημ: είναι με απλά λόγια τα ελληνικά εισαγωγικά). Για μια λίστα ιδιοχαρακτηριστικών, μπορούμε να χρησιμοποιήσουμε μια λέξη-κλειδί ως τίτλο για ένα υποσύνολο ιδιοχαρακτηριστικών.

5.3.1.4 Υποχρεώσεις και Περιορισμοί

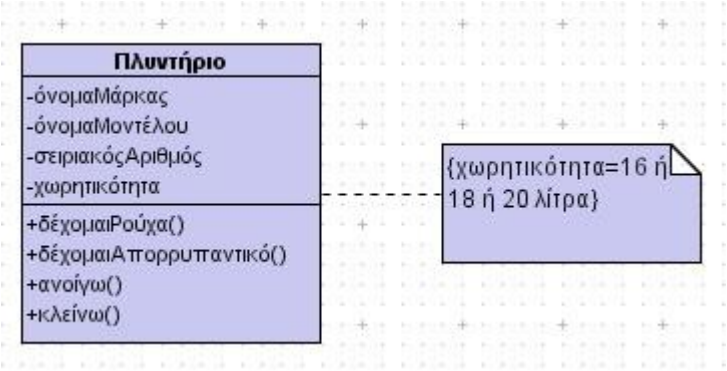
Το εικονίδιο μιας κλάσης μας βοηθάει να προσδιορίσουμε άλλο ένα τύπο πληροφοριών για μια κλάση. Στην περιοχή κάτω από τη λίστα λειτουργιών, μπορούμενα δείξουμε τις υποχρεώσεις της κλάσης. Η υποχρέωση είναι μια περιγραφή του τι πρέπει να κάνει μια κλάση – δηλαδή τι προσπαθούν να καταφέρουν τις ιδιότητες και οι λειτουργίες. Ένα πλυντήριο, για παράδειγμα, έχει την υποχρέωση να παίρνει μέσα τα βρώμικα ρούχα και να βγάζει καθαρά ρούχα.

Στο εικονίδιο προσδιορίζουμε τις υποχρεώσεις σε μια περιοχή κάτω από την περιοχή που περιλαμβάνει τις λειτουργίες (βλ. *Σχήμα 5.3.1.4-1*).

Σχήμα 5.3.1.4-1: Μπορούν να προστεθούν οι υποχρεώσεις της κλάσης κάτω από την περιοχή των λειτουργιών της

Η ιδέα εδώ είναι να συμπεριλάβουμε αρκετές πληροφορίες για μία κλάση με ένα σαφή τρόπο. Το να προσδιορίσουμε τις υποχρεώσεις της κλάσης είναι ένας ανεπίσημος τρόπος να εξαλείψουμε την ασάφεια.

Ένας περισσότερο επίσημος τρόπος είναι να προσθέσουμε έναν περιορισμό, ένα κείμενο ελεύθερο από μορφή που συμπεριλαμβάνουμε ανάμεσα σε { }. Το κείμενο στις αγκύλες προσδιορίζει ένα ή περισσότερους κανόνες που ακολουθεί η κλάση. Για παράδειγμα, ας υποθέσουμε ότι στην κλάση Πλυντήριο θέλαμε να προσδιορίσουμε ότι η χωρητικότητα μιας πλύσης μπορεί να είναι μόνο 16, 18 ή 20 λίμπρες (και ως εξής περιορίζει το ιδιοχαρακτηριστικό της χωρητικότητας της κλάσης Πλυντήριο). Θα γράφαμε **{**χωρητικότητα **= 16** ή **18** ή **20** λίτρα**}** δίπλα στο εικονίδιο της κλάσης Πλυντήριο. Το *Σχήμα 5.3.1.4-2* δείχνει πώς να το κάνουμε.

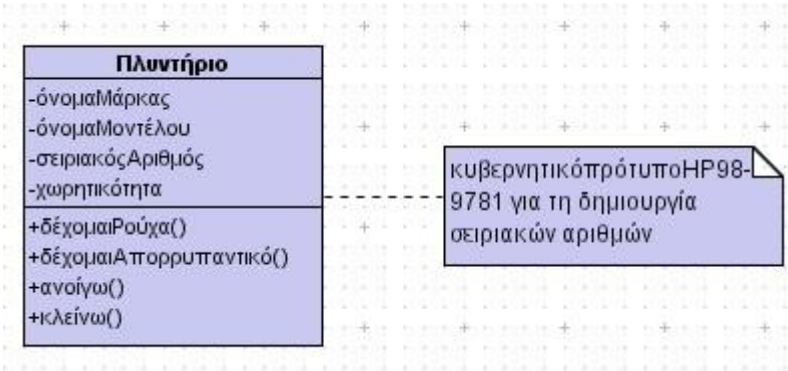


Σχήμα 5.3.1.4-2: Η συνθήκη στα άγγιστρα περιορίζει την ιδιότητα «χωρητικότητα» να είναι μια από τις 3 τιμές

Περισσότερα για τους περιορισμούς **à** Η UML δουλεύει με έναν άλλο – κι ακόμη πιο επίσημο – τρόπο πρόσθεσης περιορισμών που κάνουν τους ορισμούς πιο φανερούς. Πρόκειται για μια ολόκληρη γλώσσα που ονομάζεται Object Constraint Language (OCL). Ένα προχωρημένο και μερικές φορές χρήσιμο εργαλείο, η OCL έχει τους δικούς της κανόνες, όρους και χειριστές. Η ιστοσελίδα του Object Management Group (www.omg.org) παρέχει κείμενα για την OCL.

5.3.1.5 Συνημμένες σημειώσεις

Πέρα από τις ιδιότητες, τις λειτουργίες, τις υποχρεώσεις και τους περιορισμούς, μπορείτε να προσθέσετε κι άλλες πληροφορίες σε μία κλάση υπό τη μορφή σημειώσεων συνημμένων σε μια κλάση.



Σχήμα 5.3.1.5-1: Μια συνημμένη σημείωση παρέχει περεταίρω πληροφορίες για την κλάση

Μία σημείωση μπορεί να περιλαμβάνει επίσης ένα κείμενο γραφικών.

5.3.1.6 Κλάσεις – Τι κάνουν και πώς να τις βρείτε

Οι κλάσεις είναι το λεξιλόγιο και η ορολογία ενός φάσματος γνώσεων. Ενώ μιλάμε με πελάτες, αναλύουμε το φάσμα των γνώσεών τους και σχεδιάζουμε συστήματα υπολογιστών που λύνουν τα προβλήματα σε αυτό το φάσμα, μαθαίνουμε την ορολογία και μοντελοποιούμε τους όρους σαν κλάσεις της UML.

Στις συζητήσεις μας με τους πελάτες, να είμαστε σε ετοιμότητα για ουσιαστικά που χρησιμοποιούν για να περιγράψουν οντότητες στην επιχείρησή τους. Αυτά τα ουσιαστικά θα γίνουν οι κλάσεις του μοντέλου μας. Ξεχωρίζουμε επίσης ρήματα που θα ακούσουμε γιατί αυτά θα αποτελέσουν τις λειτουργίες σε αυτές τις κλάσεις. Οι ιδιότητες θα προκύψουν ως ουσιαστικά που σχετίζονται με τα ουσιαστικά της κλάσης. Αφού έχουμε τη λίστα με τον πυρήνα των κλάσεων, ρωτούμε τον πελάτη τι υποτίθεται ότι πρέπει να κάνει η κάθε κλάση μέσα στην επιχείρηση. Οι απαντήσεις του θα σας φανερώσουν τις υποχρεώσεις των κλάσεων.

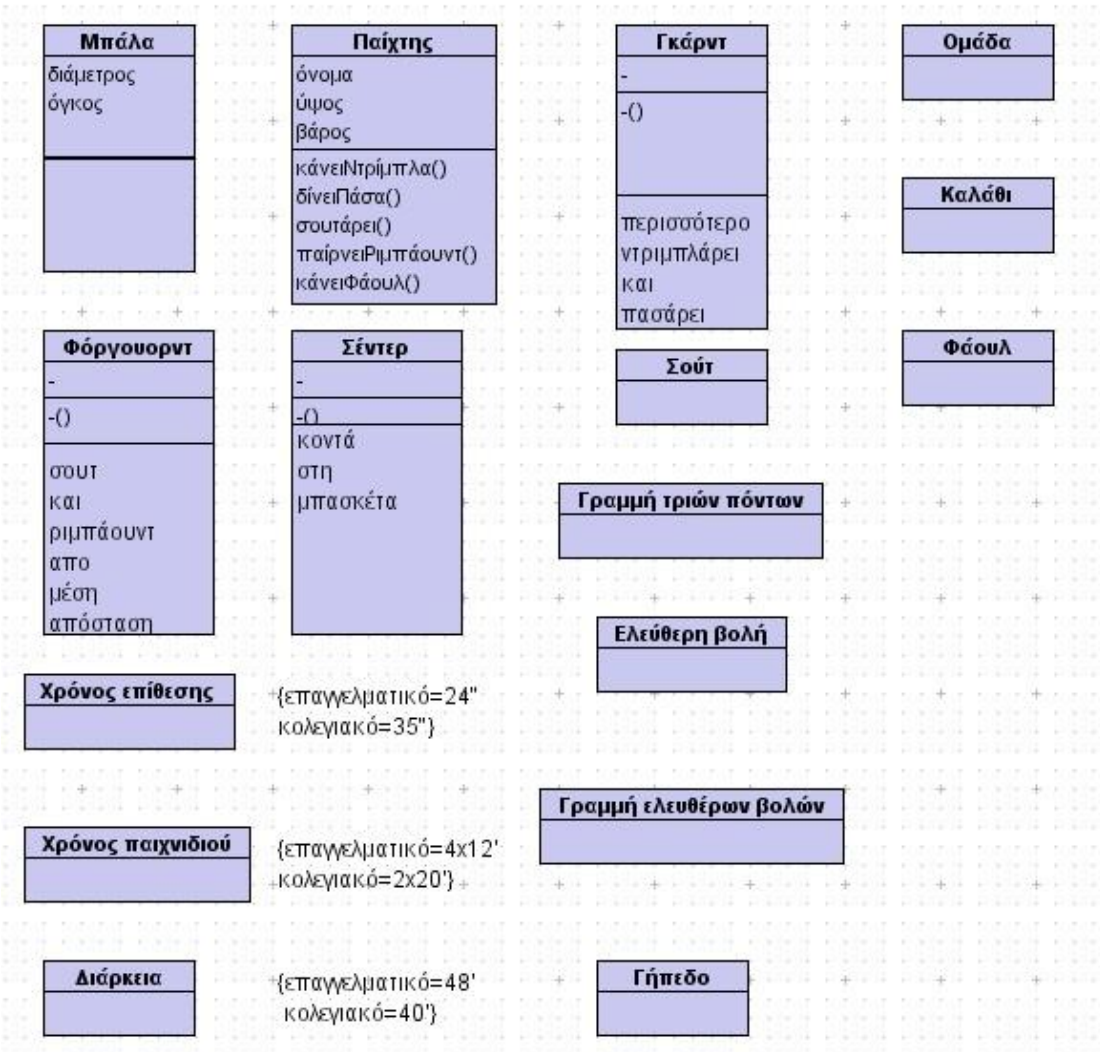
Ας υποθέσουμε ότι είστε ένας αναλυτής που κατασκευάζει ένα μοντέλο για ένα παιχνίδι μπάσκετ και παίρνετε συνέντευξη από τον προπονητή για να καταλάβουμε το παιχνίδι. Η συζήτηση ίσως εξελιχθεί κάπως έτσι:

Αναλυτής: «Τί είναι το μπάσκετ τελικά»;
Προπονητής: «Ο σκοπός του παιχνιδιού είναι να ρίξεις την μπάλα μέσα στην μπασκέτα και να σκοράρεις περισσότερους πόντους από τον αντίπαλό σου. Η κάθε ομάδα έχει πέντε παίκτες: δύο γκαρντ, δύο φόργουορντ και ένα σέντερ. Κάθε ομάδα προχωράει με την μπάλα προς την μπασκέτα με σκοπό να ρίξει τελικά την μπάλα μέσα στην μπασκέτα».
Αναλυτής: «Πώς προχωράει με την μπάλα»;
Προπονητής: «Κάνοντας ντρίμπλες και πασάροντας. Αλλά η ομάδα πρέπει να σουτάρει στην μπασκέτα πριν τη λήξη του χρόνου επίθεσης».
Αναλυτής: «Ο χρόνος επίθεσης»;
Προπονητής: «Ναι. 24 δευτερόλεπτα, δηλαδή, στους επαγγελματίες, 30 δευτερόλεπτα στα διεθνή παιχνίδια και 35 δευτερόλεπτα στο κολέγιο για να σουτάρει κανείς από τη στιγμή που έχει κατοχή της μπάλας».
Αναλυτής: «Πώς λειτουργεί το σκοράρισμα»;
Προπονητής: «Κάθε καλάθι μετράει για δύο πόντους, εκτός αν είναι πίσω από τη γραμμή τριών πόντων. Σε αυτήν την περίπτωση, είναι τρεις πόντοι. Η ελεύθερη βολή είναι ένας πόντος. Η ελεύθερη βολή, επί τη ευκαιρία, είναι η ποινή που επιβάλλεται σε μια ομάδα όταν κάνει φάουλ. Αν ένας παίκτης κάνει φάουλ σε έναν αντίπαλο, το παιχνίδι σταματάει και ο αντίπαλος πηγαίνει να σουτάρει από τη γραμμή ελεύθερης βολής».
Αναλυτής: «Πείτε μου κι άλλα για το τι κάνει κάθε παίκτης».
Προπονητής: «Οι γκαρντ, γενικά, είναι αυτοί που κάνουν περισσότερο ντρίμπλες και πάσες. Είναι συνήθως πιο κοντοί από τους φόργουαρντ και οι φόργουαρντ είναι συνήθως πιο κοντοί από τους σέντερ. Όλοι οι παίχτες υποτίθεται ότι μπορούν να ντριμπλάρουν, να πασάρουν, να σουτάρουν και να παίρνουν ριμπάουντ. Οι φόργουαρντ είναι αυτοί που παίρνουν τα περισσότερα ριμπάουντ και κάνουν σουτ από τη μέση της ρακέτας, ενώ ο σέντερ κάθεται κοντά στην μπασκέτα και σουτάρει από κοντινή απόσταση».
Αναλυτής: «Και τι γίνεται με τις διαστάσεις του γηπέδου; Και, επί τη ευκαιρία, πόσο διαρκεί ένα παιχνίδι»;
Προπονητής: «Στα διεθνή παιχνίδια, το γήπεδο είναι 28 μέτρα σε μήκος και 15 μέτρα σε πλάτος. Η μπασκέτα βρίσκεται στα 10 πόδια από το δάπεδο. Στους επαγγελματίες, ένα παιχνίδι διαρκεί 48 λεπτά, χωρισμένο σε τέσσερα 12λεπτα. Στο κολέγιο και στα διεθνή παιχνίδια, διαρκεί 40 λεπτά χωρισμένο σε δύο εικοσάλεπτα. Το χρονόμετρο σε ένα παιχνίδι παρακολουθεί πόση ώρα απομένει».

Αυτό θα μπορούσε να συνεχιστεί αλλά ας σταματήσουμε και ας δούμε πού βρισκόμαστε. Αυτά είναι τα ουσιαστικά που «ξεσκεπάσατε»: μπάλα, μπασκέτα, ομάδα, παίκτες, γκαρντ, φόργουαρντ, σέντερ, σουτ, χρόνος επίθεσης, γραμμή τριών πόντων, ελεύθερη βολή, φάουλ γραμμή ελεύθερης βολής, γήπεδο και χρονόμετρο.
Αυτά είναι τα ρήματα: σουτάρω, προχωρώ, ντριμπλάρω, πασάρω, κάνω φάουλ και παίρνω ριμπάουντ. Έχετε και κάποιες επιπλέον πληροφορίες για μερικά από τα ουσιαστικά – όπως τα σχετικά ύψη των παικτών ανάλογα με τη θέση που παίζουν, τις διαστάσεις του γηπέδου, τη συνολική ποσότητα του χρόνου σε ένα χρονόμετρο και τη διάρκεια του παιχνιδιού.

Τέλος, η ίδια η γνώση σας και η κοινή λογική μπορεί να «μπει στο παιχνίδι» παράγοντας μερικές δικές σας ιδιότητες. Ξέρετε, για παράδειγμα, ότι η μπάλα έχει ιδιότητες όπως όγκος και διάμετρος.

Χρησιμοποιώντας αυτές τις πληροφορίες, μπορούμε να φτιάξουμε ένα διάγραμμα, όπως αυτό στο *Σχήμα 5.3.1.6-1*. Δείχνει τις κλάσεις και παρέχει μερικές ιδιότητες, λειτουργίες και περιορισμούς. Το διάγραμμα δείχνει, επίσης, τις υποχρεώσεις. Μπορούμε να χρησιμοποιήσουμε το διάγραμμα ως βάση για περαιτέρω συζητήσεις με τον προπονητή, ώστε να αποκαλύψει κι άλλες πληροφορίες.



Σχήμα 5.3.1.6-1: Ένα αρχικό διάγραμμα κλάσης για τη μοντελοποίηση του παιχνιδιού «μπάσκετ»

5.3.2 Περίληψη

Το ορθογώνιο είναι το εικονίδιο της UML για να αντιπροσωπεύει μία κλάση. Το όνομα, οι ιδιότητες, οι λειτουργίες και οι υποχρεώσεις της κλάσης χωρούν σε μια περιοχή μέσα στο ορθογώνιο. Μπορείτε να χρησιμοποιήσετε ένα στερεότυπο για να οργανώσετε τις λίστες των ιδιοχαρακτηριστικών και λειτουργιών. Μία κλάση παθαίνει έκθλιψη όταν φανερώνει απλώς ένα υποσύνολο των ιδιοχαρακτηριστικών και των λειτουργιών. Αυτό κάνει ένα διάγραμμα μιας κλάσης λιγότερο «φορτωμένο». Μπορείτε να δείξετε τον τύπο ενός ιδιοχαρακτηριστικού και την αρχική του τιμή και να δείξουμε τις τιμές πάνω στις οποίες μια λειτουργία δουλεύει, όπως και τους τύπους της. Για μια λειτουργία, αυτές οι επιπλέον πληροφορίες ονομάζονται υπογραφή.
Για να περιορίσουμε την ασάφεια στην περιγραφή μιας κλάσης, μπορούμε να προσθέσουμε περιορισμούς. Η UML μας επιτρέπει, επίσης, να πούμε κι άλλα για μια κλάση επισυνάπτοντας σημειώσεις στο ορθογώνιο που την αντιπροσωπεύει.

Οι κλάσεις αντιπροσωπεύουν το λεξικό ενός φάσματος γνώσεων. Οι συζητήσεις με έναν πελάτη ή ειδικό αποκαλύπτουν τα ουσιαστικά που γίνονται κλάσεις σε ένα μοντέλο και τα ρήματα που μπορούν να γίνουν λειτουργίες. Μπορούμε να χρησιμοποιήσουμε το διάγραμμα μιας κλάσης σαν ένα τρόπο να προκαλέσουμε τον πελάτη να μιλήσει περισσότερο για το δικό του ή δικό της φάσμα και να μας αποκαλύψει επιπλέον πληροφορίες.

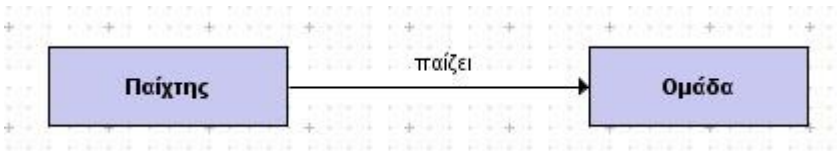
5.4 ΤΑ ΔΙΑΓΡΑΜΜΑΤΑ ΤΗΣ UML

5.4.1 ΔΙΑΓΡΑΜΜΑΤΑ ΚΛΑΣΕΩΝ

5.4.1.1 Συσχετίσεις

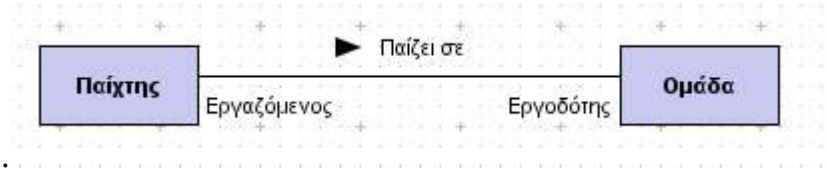
Όταν οι κλάσεις είναι συνδεδεμένες μεταξύ τους νοηματικά, αυτή η σύνδεση ονομάζεται συσχέτιση (association). Ένα παράδειγμα μπορεί να δοθεί από το άθλημα "μπάσκετ".

Ας εξετάσουμε το παράδειγμα της συσχέτισης μεταξύ ενός παίκτη και μιας ομάδας. Μπορούμε να χαρακτηρίσουμε αυτή τη συσχέτιση με τη φράση "ένας παίκτης παίζει σε μια ομάδα". Μοντελοποιούμε τη συσχέτιση ως μια γραμμή που συνδέει τις δυο κλάσεις, με το όνομα της συσχέτισης ("παίζει σε") ακριβώς πάνω από τη γραμμή. Δείχνεται το πώς διαβάζεται η σχέση με ένα γεμάτο τρίγωνο που δείχνει στην ανάλογη κατεύθυνση.



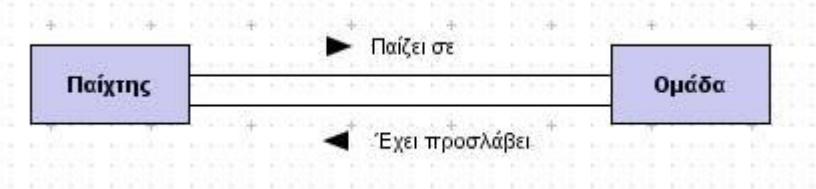
Σχήμα 5.4.1.1-1: Μια συσχέτιση

Όταν μια κλάση συσχετίζεται με μια άλλη, συνήθως η καθεμιά παίζει κάποιο ρόλο στα πλαίσια αυτής της συσχέτισης. Στη συσχέτιση του παίκτη με την ομάδα του, εάν είναι επαγγελματική, η ομάδα είναι εργοδότης και ο παίκτης ο εργαζόμενος.



Σχήμα 5.4.1.1-2: Παράσταση των ρόλων στο διάγραμμα

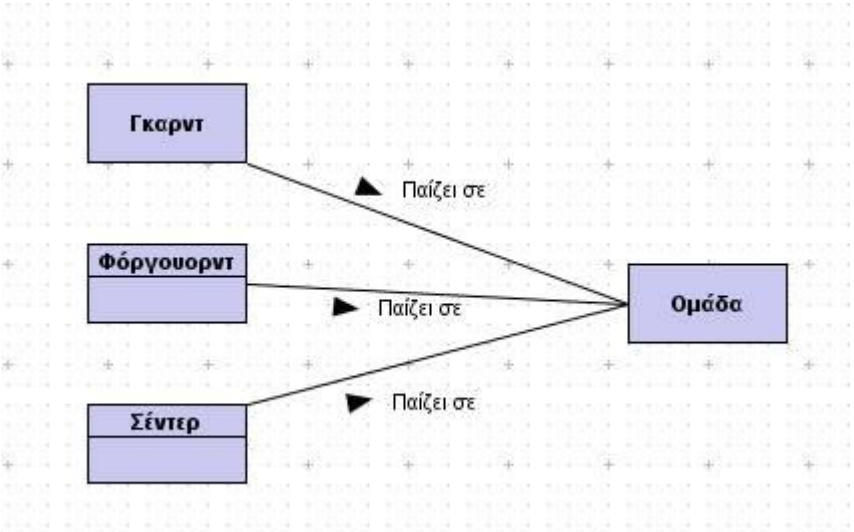
Ένας άλλος τρόπος για να δείξουμε τη συσχέτιση του παίκτη και της ομάδας είναι η ακόλουθη.



Σχήμα 5.4.1.1-3: Δυο συσχετίσεις μεταξύ κλάσεων στο ίδιο διάγραμμα

Οι συσχετίσεις μπορεί να είναι πιο πολύπλοκες απ' ότι μονάχα μια κλάση που συνδέεται με μια άλλη. Πολλαπλές κλάσεις μπορούν να συνδεθούν σε μια άλλη

κλάση. Αν σκεφτούμε τους γκάρντς, φόργουορντς και σέντερς και τις συσχετίσεις τους με την κλάση Ομάδα, θα έχουμε το διάγραμμα που ακολουθεί.



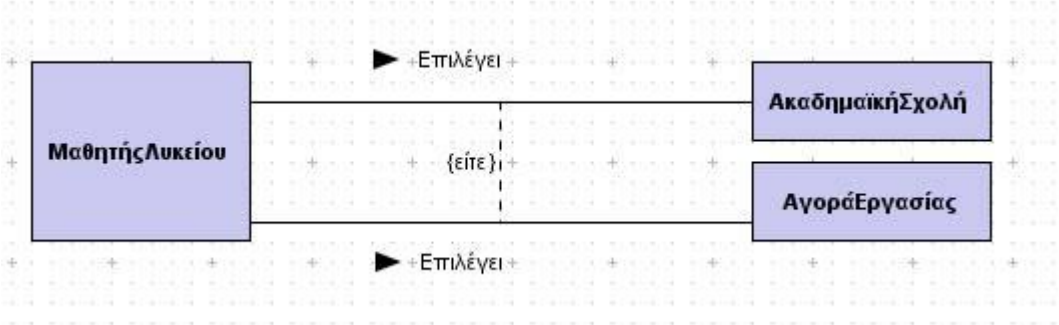
Σχήμα 5.4.1.1-4: Πολλαπλές κλάσεις μπορούν να συσχετιστούν με μια συγκεκριμένη κλάση

5.4.1.1.1 Περιορισμοί συσχετίσεων

Μερικές φορές μια συσχέτιση μεταξύ δυο κλάσεων πρέπει να υπακούει σε κάποιο κανόνα. Καθορίζουμε τον κανόνα βάζοντας ένα περιορισμό (constraint) δίπλα στη γραμμή συσχέτισης. Για παράδειγμα ένας Τραπεζικός εξυπηρετεί ένα πελάτη, αλλά κάθε πελάτης εξυπηρετείται ανάλογα με τη σειρά που έχει στη γραμμή. Αυτό το δείχνουμε βάζοντας τη λέξη "αριθμημένος" μέσα σε άγκιστρα (για να υποδείξουμε τον περιορισμό) δίπλα στη κλάση Πελάτης.

Σχήμα 5.4.1.1.1-1: Περιορισμός σε μια συσχέτιση

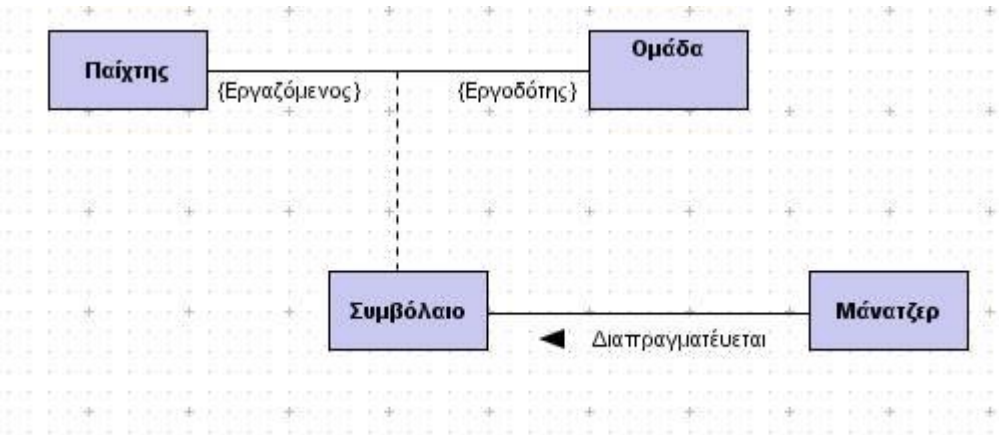
Ένας άλλος τύπος περιορισμού είναι η σχέση "είτε", που παριστάνεται σαν {είτε} σε μια διακεκομμένη γραμμή που συνδέει δυο γραμμές συσχέτισης. Το επόμενο σχήμα μοντελοποιεί ένα μαθητής λυκείου που διαλέγει μια ακαδημαϊκή σχολή είτε την αγορά εργασίας.



Σχήμα 5.4.1.1.1-2: Η σχέση Είτε μεταξύ δυο συσχετίσεων σε ένα περιορισμό

5.4.1.1.2 Κλάσεις Συσχέτισης-Συσχετισμένες Κλάσεις

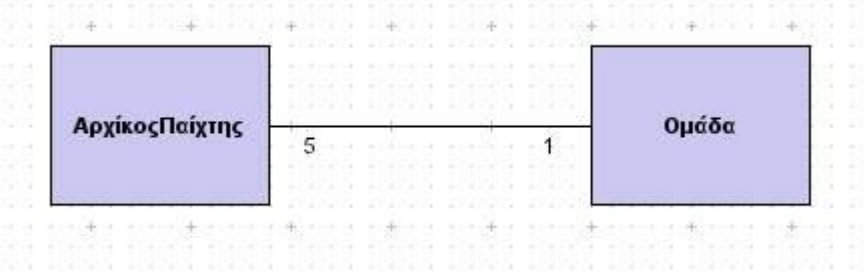
Μια συσχέτιση μπορεί να έχει ιδιότητες και λειτουργίες, ακριβώς όπως μια κλάση. Στην πραγματικότητα, όταν συμβαίνει αυτό, έχουμε μια κλάση συσχέτισης (association class). Μια κλάση συσχέτισης οπτικοποιείται με τον ίδιο τρόπο όπως και μια κανονική κλάση, και χρησιμοποιείται μια διακεκομμένη γραμμή για τη σύνδεσή της με την γραμμή συσχέτισης. Μια κλάση συσχέτισης μπορεί να έχει συσχετίσεις σε άλλες κλάσεις.



Σχήμα 5.4.1.1.2-1: Μια κλάση συσχέτισης μοντελοποιεί τις ιδιότητες και λειτουργίες μιας συσχέτισης. Συνδέεται με την συσχέτιση μέσω μιας διακεκομμένης γραμμής και μπορεί να συσχετισθεί με μια άλλη κλάση

5.4.1.2 Πολλαπλότητα

Η συσχέτιση μέχρι τώρα μεταξύ Παίχτη και Ομάδα υποδηλώνει πως οι δυο κλάσεις έχουν σχέση που ονομάζεται ένα-προς-ένα. Η λογική μας λέει όμως πως αυτό δεν είναι πράγματι αληθές. Μια ομάδα μπάσκετ έχει πέντε παίχτες. Η σχέση *Έχει* πρέπει να το λαμβάνει αυτό υπ' όψιν της. Στην άλλη κατεύθυνση, ένας παίχτης μπορεί να παίξει μόνο σε μια ομάδα, και η σχέση *Παίζει* πρέπει να το λογαριάζει αυτό. Αυτές οι προδιαγραφές είναι παραδείγματα πολλαπλότητας (multiplicity), ο αριθμός των αντικειμένων από μία κλάση που σχετίζονται με ένα αντικείμενο σε μία συσχετισμένη κλάση. Για να αντιπροσωπεύσουμε αυτά τα νούμερα σε ένα διάγραμμα, τα τοποθετούμε δίπλα στη σχετική κλάση, όπως παρακάτω.



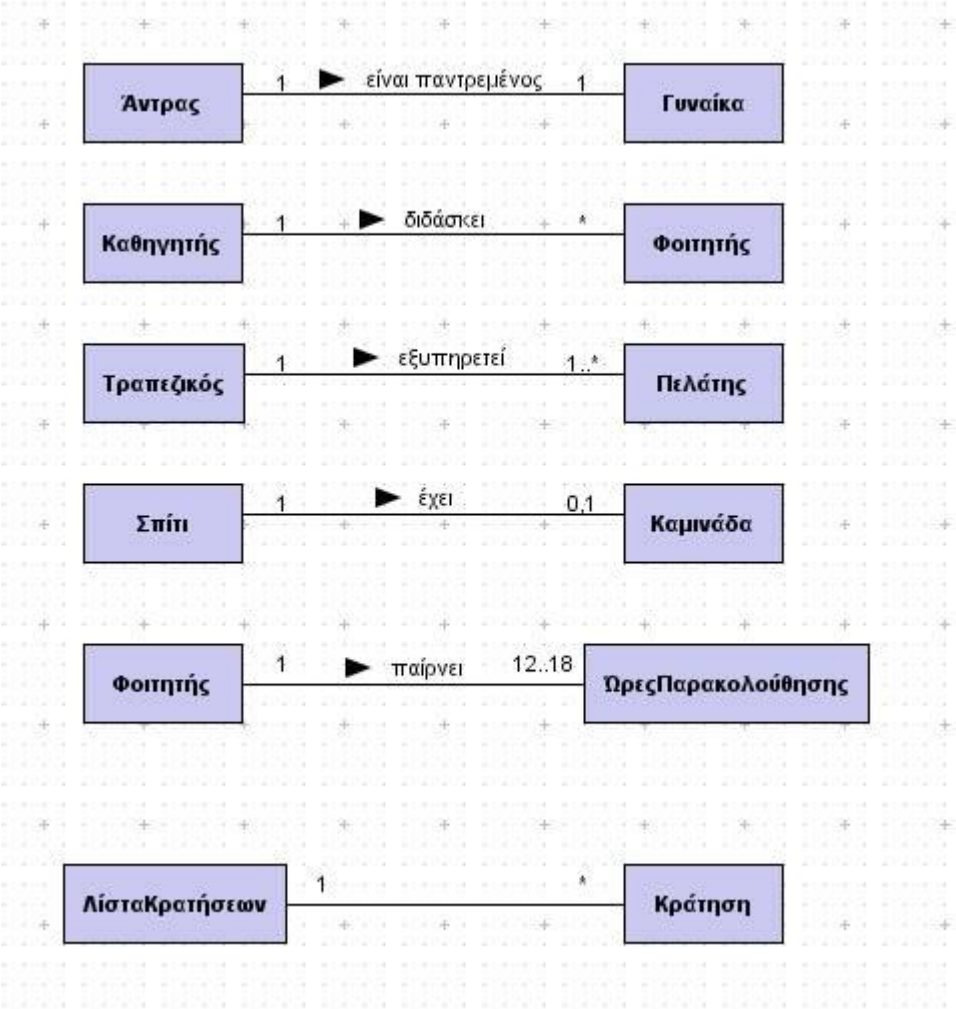
Σχήμα 5.4.1.2-1: Πολλαπλότητα κλάσεων

Η πολλαπλότητα σε αυτό το παράδειγμα ο μοναδικός τύπος. Υπάρχει μια ποικιλία από πολλαπλότητες. Μια κλάση μπορεί να σχετίζεται με μια άλλη με σχέση 1-προς-1, 1-προς-πολλά, 1-προς-"1 και πάνω", 1-προς-"0 ή ένα", 1-προς-"ένα εύρος (πχ. 5-10)", και άλλα.

Η UML χρησιμοποιεί έναν αστερίσκο (*) που παριστάνει το "πολλά" και το "πάνω από". Σε μια περίπτωση το "ή" (or) παριστάνεται με δυο τελείες, όπως το 1..* (ένα ή παραπάνω). Σε μια άλλη το "ή" (or) παριστάνεται με ένα κόμα, όπως το 5,10 ("5 ή 10").

5.4.1.3 Χαρακτηρισμένες Συσχετίσεις

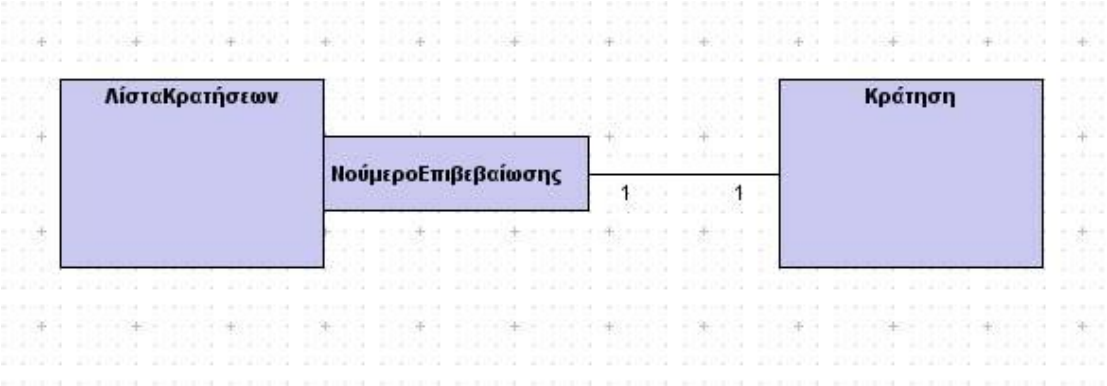
Όταν η πολλαπλότητα μιας συσχέτισης είναι 1-προς-πολλά, εμφανίζεται μια συγκεκριμένη πρόκληση. Όταν ένα συγκεκριμένο αντικείμενο από μια κλάση πρέπει να διαλέξει ένα συγκεκριμένο αντικείμενο από μια άλλη προκειμένου να εκπληρώσει κάποιο ρόλο σε μια συσχέτιση, η πρώτη κλάση πρέπει να βασιστεί σε μια συγκεκριμένη ιδιότητα για να διαλέξει το σωστό αντικείμενο. Αυτή η ιδιότητα συνήθως λέγεται *αναγνωριστικό (identifier)* όπως ένα νούμερο ταυτότητας. Για παράδειγμα, η λίστα κρατήσεων ενός ξενοδοχείου έχει πολλές κρατήσεις, όπως δείχνει το τελευταίο απ'τα παρακάτω σχήματα.



Σχημα 5.4.1.3-1: Πιθανές πολλαπλότητες και πώς αναπαρίστανται στη UML

Όταν κάνουμε μια κράτηση σε ένα ξενοδοχείο, το ξενοδοχείο μας αντιστοιχίζει ένα νούμερο επιβεβαίωσης. Εάν καλέσουμε έπειτα στο ξενοδοχείο για να μάθουμε επιπλέον πληροφορίες για την κράτηση, πρέπει να δώσουμε το νούμερο επιβεβαίωσης για να μας εξυπηρετήσουν.

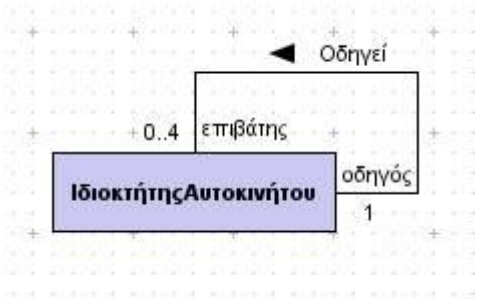
Στη UML, η πληροφορία αυτή ονομάζεται *χαρακτηριστής* ***(qualifier)***. Το σύμβολό του είναι ένα μικρό ορθογώνιο που κολλάει στην κλάση με χαρακτηρισμό “ένα” (1) της συσχέτισης 1-προς-πολλά. Αν και η πολλαπλότητα μεταξύ της ΛίσταΚρατήσεων και της Κράτηση είναι 1-προς-πολλά, η πολλαπλότητα μεταξύ του ΝούμεροΕπιβεβαίωσης και Κράτηση είναι 1-προς-1.



Σχημα 5.4.1.3-2: Η σημειολογία της UML για έναν χαρακτηριστή

5.4.1.4 Αυτοπαθείς Συσχετίσεις

Μερικές φορές, μια κλάση βρίσκεται σε συσχέτιση με τον εαυτό της. Αυτό ονομάζεται αυτοπαθής συσχέτιση (reflexive association) και συμβαίνει όταν μια κλάση έχει αντικείμενα που μπορούν να παίξουν διάφορους ρόλους. Για παράδειγμα, ένας ΙδιοκτήτηςΑυτοκινήτου μπορεί να είναι είτε επιβάτης είτε οδηγός. Στο ρόλο του οδηγού, ένας ΙδιοκτήτηςΑυτοκινήτου είναι οδηγός μηδέν ή περισσότερων ιδιοκτητών αυτοκινήτων που παίζουν το ρόλο των επιβατών. Αυτό φαίνεται στο επόμενο διάγραμμα.

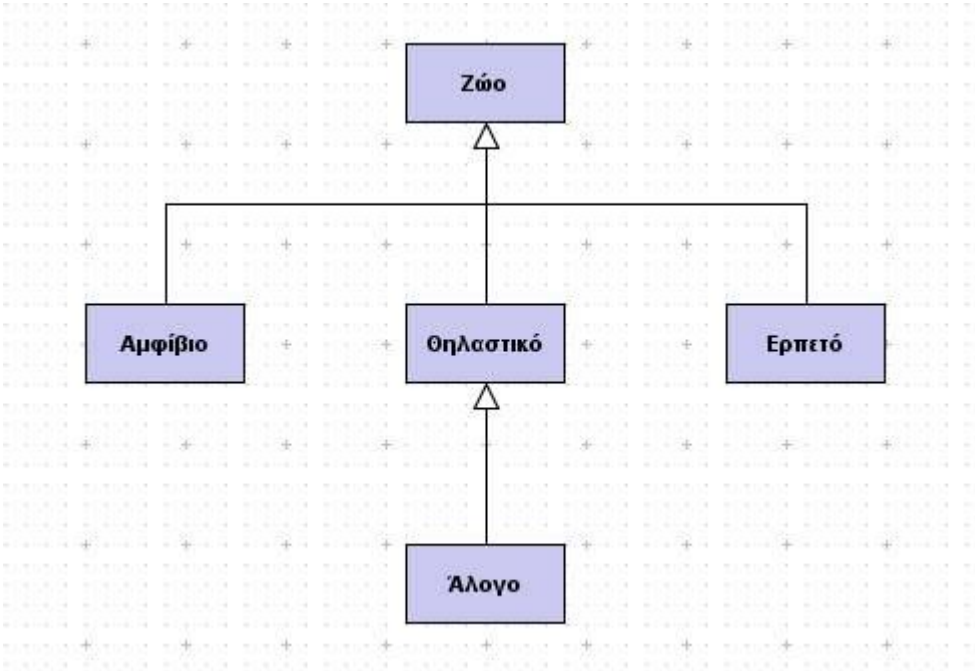


Σχήμα 5.4.1.4-1: Αυτοπαθής συσχέτιση

5.4.1.5 Κληρονομικότητα και Γενίκευση

Ένα σημαντικό χαρακτηριστικό της αντικειμενοστρέφειας είναι ότι εάν ξέρουμε κάτι για μια κατηγορία πραγμάτων, αυτόματα ξέρουμε κάποιες πληροφορίες που μπορούμε να μεταφέρουμε σε άλλες κατηγορίες. Εάν γνωρίζουμε πως κάτι είναι ζώο, θεωρούμε δεδομένο ότι τρώει, κοιμάται κτλ.

Η αντικειμενοστρέφεια το ονομάζει αυτό κληρονομικότητα (inheritance). Η UML αναφέρεται σ' αυτό ως γενίκευση (generalisation). Μια κλάση (η κλάση παιδί ή child class) μπορεί να κληρονομήσει ιδιότητες και λειτουργίες από μια άλλη (την κλάση γονέας ή parent class). Η κλάση γονέας είναι πιο γενική από την κλάση παιδί. Παρακάτω φαίνεται ένα παράδειγμα γενίκευσης.



Σχήμα 5.4.1.5-1: Η ιεραρχία κληρονομικότητας στο ζωϊκό βασίλειο

Οι κλάσεις παιδιά προσθέτουν στις ιδιότητες και λειτουργίες που κληρονομούν. Για παράδειγμα, ένα θηλαστικό έχει μαλλιά και δίνει γάλα, δυο ιδιότητες που δεν τις βρίσκουμε στην κλάση Ζώο. Μια κλάση μπορεί να μην έχει γονείς, στην οποία περίπτωση ονομάζεται βασική κλάση (base or root class). Μια κλάση μπορεί να μην έχει παιδιά, και ονομάζεται κλάση φύλλο (leaf class). Εάν μια κλάση έχει ακριβώς ένα γονέα, έχει μοναδική κληρονομικότητα (single inheritance). Αν μια κλάση έχει περισσότερους από έναν γονείς, τότε έχει πολλαπλή κληρονομικότητα (multiple inheritance).

5.4.1.5.1 Ανακαλύπτοντας την Κληρονομικότητα

Στο παράδειγμα του μπάσκετ που έχει αναλυθεί στο προηγούμενο κεφάλαιο, έχουμε τις κλάσεις Παίχτης, Γκάρντ, Φόργουορντ και Σέντερ. Η κλάση Παίχτης έχει ιδιότητες όπως όνομα, ύψος, βάρος, ταχύτητα και άλμα. Η κλάση έχει λειτουργίες όπως ντριμπλάρει(), πασάρει(), παίρνειΡιμπάουντ(), και σουτάρει(). Οι γκάρντ, φόργουορντ και σέντερ κληρονομούν αυτές τις ιδιότητες και λειτουργίες και προσθέτουν μερικές δικές τους. Ο γκάρντ μπορεί να έχει τις λειτουργίες τρέχειΣτηνΕπίθεση() και κατεβάζειΤηΜπάλα(). Ο σέντερ μπορεί να έχει τη λειτουργία κάρφωμα(). Βασισμένος στα σχόλια του προπονητή για το σχετικό ύψος των παιχτών, ο αναλυτής μπορεί να θέλει να τοποθετήσει περιορισμούς στο ύψος των ατόμων που παίζουν στην κάθε θέση.

Μια άλλη πιθανότητα είναι ότι ο αναλυτής σημειώνει ότι δύο ή περισσότερες κλάσεις έχουν μερικές κοινές ιδιότητες και λειτουργίες. Το μοντέλο του μπάσκετ έχει ένα χρόνοΠαιχνιδιού, που κρατάει πόσος χρόνος μένει για να τελειώσει μια περίοδος, και το χρόνοΕπίθεσης, που κρατάει το χρόνο που απομένει για να γίνει κάποιο σούτ. Συνειδητοποιώντας ότι και τα δύο κρατούνε χρόνο, ο αναλυτής θα μπορούσε να

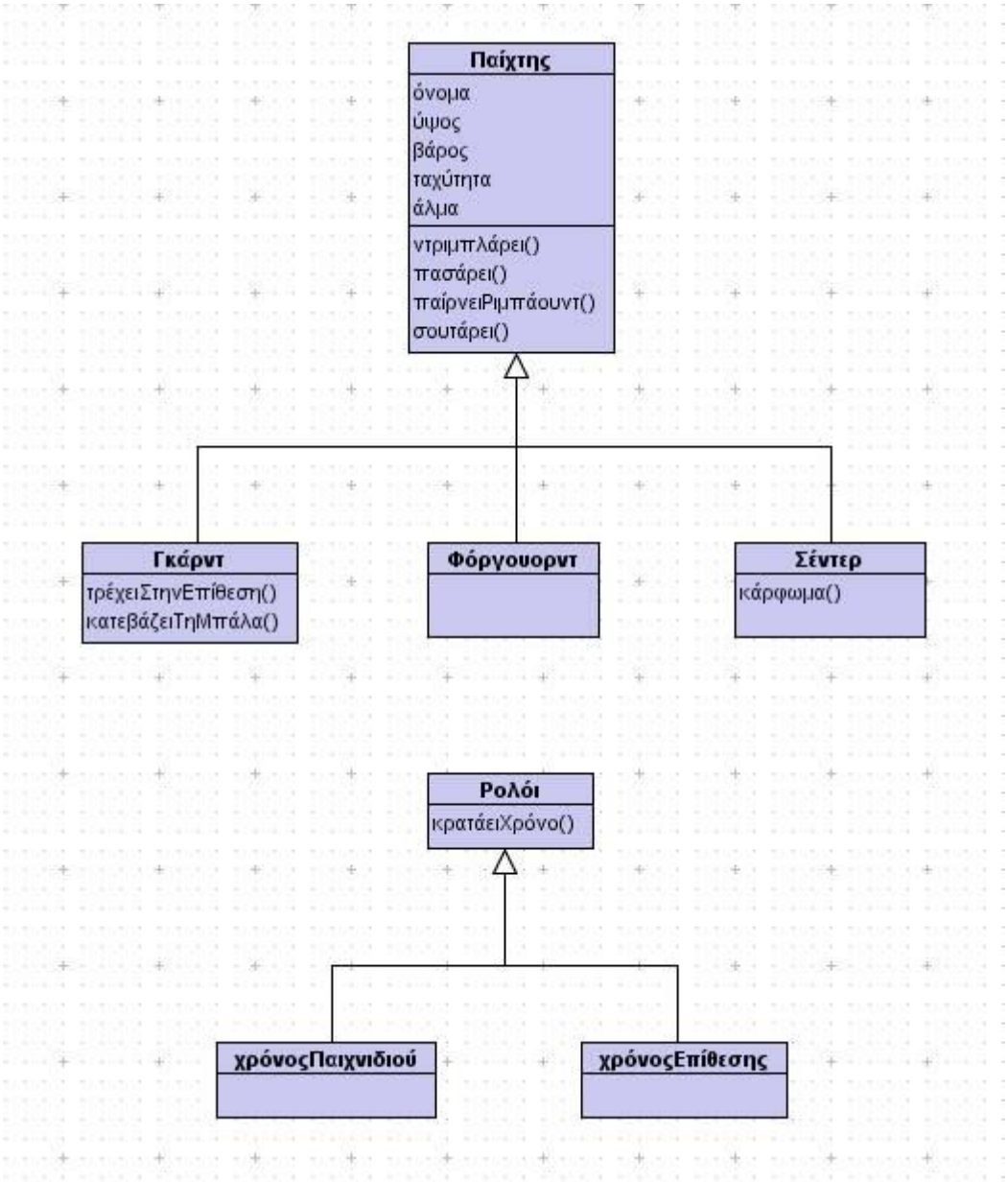
φτιάξει μια κλάση που να ονομάζεται Ρολόι με ιδιότητα το κρατάειΧρόνο() που αμφότερα τα χρόνοςΠαιχνιδιού και χρόνοςΕπίθεσης κληρονομούν.

5.4.1.5.2 Αφηρημένες Κλάσεις

Στο μοντέλο του μπάσκετ, οι δυο κλάσεις που αναφέραμε – Παίχτης και Ρολόι – είναι χρήσιμες διότι υπηρετούν ως γονικές κλάσεις (parent classes) για σημαντικές κλάσεις-παιδιά (child classes). Οι κλάσεις-παιδιά είναι σημαντικές στο μοντέλο γιατί τελικά θα θέλουμε να έχουμε στιγμιότυπα αυτών των κλάσεων. Για να δημιουργήσουμε το μοντέλο θα χρειαστούμε στιγμιότυπα των: Γκαρντ, Φόργουορντ, Σέντερ, χρόνοςΠαιχνιδιού και χρόνοςΕπίθεσης.

Οι κλάσεις Παίχτης και Ρολόι δε θα παρέχουν στιγμιότυπα για το μοντέλο. Ένα αντικείμενο από την κλάση Παίχτης δε θα εξυπηρετούσε σε τίποτα. Το ίδιο ισχύει και για την κλάση Ρολόι.

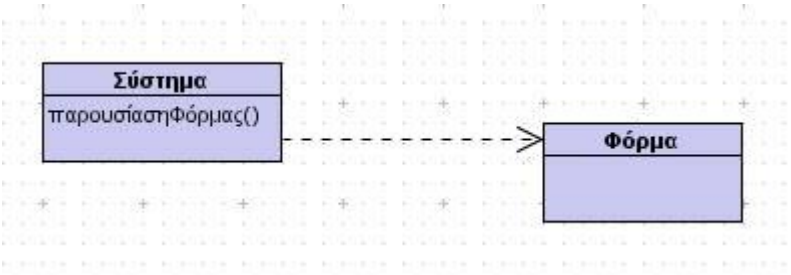
Κλάσεις σαν τις Παίχτης και Ρολόι, που δεν παρέχουν αντικείμενα, λέγονται *αφηρημένες κλάσεις (abstract classes)*. Υποδεικνύουμε μια αφηρημένη κλάση γράφωντας το όνομά της με πλάγια γράμματα.



Σχήμα 5.4.1.5.1-1: Δύο ιεραρχίες κληρονομικότητας στο μοντέλο του μπάσκετ

5.4.1.6 Εξαρτήσεις

Σε μια διαφορετική μορφή σχέσης, μια κλάση χρησιμοποιεί μία άλλη. Αυτό ονομάζεται εξάρτηση (dependency). Η πιο συνηθισμένη χρήση μιας εξάρτησης είναι να δείξει ότι η υπογραφή της λειτουργίας μιας κλάσης χρησιμοποιεί μια άλλη κλάση. Ας υποθέσουμε ότι σχεδιάζουμε ένα σύστημα που απεικονίζει εταιρικές φόρμες σε μια οθόνη για να τις συμπληρώσουνε οι υπάλληλοι. Ο υπάλληλος χρησιμοποιεί ένα μενού για να διαλέξει τη φόρμα που θέλει να συμπληρώσει. Στο σχεδιασμό μας, έχουμε μια κλάση Σύστημα και μια κλάση Φόρμα. Ανάμεσα στις πολλαπλές της λειτουργίες, η κλάση Σύστημα έχει και την παρουσίασηΦόρμας(φ:Φόρμα). Η φόρμα που εμφανίζει το σύστημα προφανώς εξαρτάται από το ποια φόρμα διαλέγει ο χρήστης. Η σημειολογία της UML γι'αυτό είναι μια διακεκομμένη γραμμή με ένα βέλος που δείχνει προς την κλάση που είναι είναι εξαρτημένη.



Σχήμα 5.4.1.6-1: Αναπαράσταση μιας εξάρτησης

5.4.1.7 Περίληψη

Τα διαγράμματα κλάσης δείχνουν τη στατική δομή του μοντέλου και πιο συγκεκριμένα τα πράγματα που υπάρχουν (όπως κλάσεις και τύποι), την εσωτερική δομή τους και τις σχέσεις τους με άλλα πράγματα. Τα διαγράμματα κλάσης δε δείχνουν χρονική πληροφορία αν και ίσως περιλαμβάνουν συγκεκριμενοποιημένες εμφανίσεις πραγμάτων που έχουν ή πράγματα που περιγράφουν χρονική συμπεριφορά. Ένα διάγραμμα αντικειμένων δείχνει στιγμιότυπα συμβατά με ένα συγκεκριμένο διάγραμμα κλάσεων.

Αυτό το κεφάλαιο περιλαμβάνει τις κλάσεις και τις μεταβολές τους, όπως και πρότυπα και κλάσεις στιγμιότυπων και τις σχέσεις ανάμεσα στις κλάσεις: διασύνδεση και γενικοποίηση. Περιλαμβάνει τα περιεχόμενα των κλάσεων: γνωρίσματα και λειτουργίες.

Διάγραμμα κλάσης

Ένα διάγραμμα κλάσεων είναι ένα γράφημα στοιχείων ταξινομητών (classifier) συνδεδεμένα από διάφορες στατικές σχέσεις. (Σημειώστε ότι ένα διάγραμμα κλάσης μπορεί να συμπεριλαμβάνει επίσης διασυνδέσεις, πακέτα, σχέσεις και ακόμη στιγμιότυπα όπως αντικείμενα και συνδέσμους. Ίσως μια καλύτερη ονομασία θα ήταν «στατικό δομικό διάγραμμα» αλλά ο όρος «διάγραμμα κλάσης» είναι μικρότερος και καθιερωμένος).

Σημειολογία

Ένα διάγραμμα κλάσης είναι μια γραφική απεικόνιση ενός στατικού δομικού μοντέλου. Τα ανεξάρτητα διαγράμματα κλάσης δεν αντιπροσωπεύουν υποδιαιρέσεις του υποκείμενου μοντέλου.

Συμβολισμός

Ένα διάγραμμα κλάσης είναι μια συλλογή από (στατικά) δηλωτικά στοιχείων μοντέλου, όπως οι κλάσεις, από διασυνδέσεις και τις συσχετίσεις τους, συνδεδεμένα μεταξύ τους και με τα περιεχόμενά τους ως ένα γράφημα. Τα διαγράμματα κλάσης

μπορούν να οργανωθούν σε πακέτα είτε με τα υποκείμενά τους μοντέλα ή ως ξεχωριστά πακέτα που χτίζουν επί των υποκείμενων πακέτων των μοντέλων.

5.4.2 ΔΙΑΓΡΑΜΜΑΤΑ ΣΕΝΑΡΙΩΝ

5.4.2.1 Ορισμός διαγραμμάτων σεναρίου

Τα διαγράμματα σεναρίων (Use case diagrams) είναι το αρχικό σημείο όταν σχεδιάζεται ένα νέο σύστημα χρησιμοποιώντας την UML . Αυτά τα διαγράμματα προέρχονται από την δουλειά του Invar Jacobson και εφαρμόζονταν στην αρίθμηση των επιχειρηματικών απαιτήσεων ενός συστήματος με τρόπο που θα γίνονταν κατανοητός από όλους όσους θα χρησιμοποιούσαν το σύστημα. Η αναγνώριση των απαιτήσεων του συστήματος γίνεται το πρώτο στάδιο της ενοποιημένης διεργασίας . Από τα διαγράμματα σεναρίου που θα δημιουργηθούν σε αυτή τη φάση θα προκύψουν πολλά αλλά διαγράμματα στην επόμενη φάση .

Για να αντιληφθούμε καλυτέρα αυτές αλλά και άλλες έννοιες που θα ακολουθήσουν ίσως πρώτα θα πρέπει να ασχοληθούμε με τι είναι σύστημα σύμφωνα με την UML . Θα μπορούσε να είναι ένα πρόγραμμα σε έναν υπολογιστή που θα εκτύπωνε σε εβδομαδιαία βάση επιταγές η ακόμα και μια κυβέρνηση με τις βασικές της λειτουργίες , περισσότερο θα ασχοληθούμε με τα συστήματα σε σχέση με τον υπολογιστή αν και η UML μπορεί να χρησιμοποιηθεί για να μοντελοποιήσουμε οτιδήποτε εκτελεί μια εργασία όπως ένα αυτοκίνητο , ένα κινητό , μια εταιρεία ή ένα άτομο .

Ένα διάγραμμα σεναρίων είναι η υψηλότερη μορφή (δεν είναι αναλυτικό) λεπτομέρειας για ένα σύστημα . Είναι ένας εξαίρετος τρόπος για να επικοινωνήσουν με τους διαχειριστές , οι πελάτες και αλλά άτομα που δεν ασχολούνται με τον σχεδιασμό του συστήματος και να τους δείξουν τι θα κάνει το σύστημα όταν ολοκληρωθεί . Το διάγραμμα δεν μπαίνει σε λεπτομέρειες για το πώς ένα σύστημα θα κάνει κάτι αλλά δείχνει ποιος θα χρησιμοποιήσει το σύστημα και τι θα είναι ικανός να κάνει αυτό .

5.4.2.1.1 Γιατί χρησιμοποιούμε τα διαγράμματα σεναρίων

Σχεδιάζουμε τα διαγράμματα σεναρίων για να δώσουμε ένα έναυσμα στο σχεδιασμό του συστήματος η για να περιγράψουμε ένα ήδη υπάρχον σύστημα . Θα πρέπει να απεικονίζουν στην ομάδα ανάπτυξης ακριβώς τι προσδοκούμε από μια εφαρμογή που αναπτύσσουν . Δεν είναι ασυνήθιστο να δημιουργούνται τα διαγράμματα από μια ομάδα που δεν ασχολείτο με την ανάπτυξη των εφαρμογών . Αλλά τις περισσότερες φορές βρίσκουμε τους εαυτούς μας να δουλεύουν σε ένα πρόγραμμα που απασχολεί λίγους ανθρώπους και η ομάδα ανάπτυξης είναι αναγκασμένη να παίξει το ρολό του διαχειριστή του προϊόντος και να πάρει συνέντευξη από μελλοντικούς χρηστές για το τι θα κάνει το σύστημα .

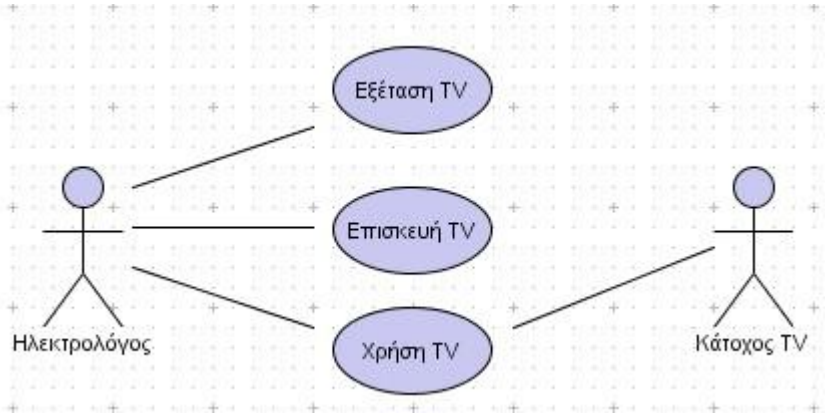
Κατά την διάρκεια της εναρκτήριας φάσης προσπαθούμε να δούμε ποιος θα το χρησιμοποιεί και τι θα κάνει με αυτό . Αφού μαζέψουμε τις πληροφορίες αυτές μπορούμε να τις μετατρέψουμε σε διάγραμμα σεναρίων και να συνεχίσουμε να σχεδιάζουμε το σύστημα έτσι ώστε να ανταποκρίνεται στις προηγούμενες

απαιτήσεις. Από τα διαγράμματα σεναρίων μπορούμε να βρούμε τα σημεία που θα επικοινωνεί το σύστημα με τους χρηστές .

Τα διαγράμματα αυτά επίσης ,εξαιτίας της ικανότητας τους να δείχνουν τη σχέση μεταξύ χρηστή και συστήματος , βοηθούν πολύ και τους δοκιμαστές του συστήματος . Γιατί ενώ τα διαγράμματα σεναρίων δημιουργούνται για να δείξουν την ροή των εργασιών μπορούν να δείξουν και εξαιρούμενες διεργασίες ή διεργασίες χειρισμού λαθών . Ο χειρισμός λαθών είναι μια διεργασία που συνήθως παραβλέπει στην αρχή αλλά με την χρήση των διαγραμμάτων σεναρίου κινεί την απαιτούμενη προσοχή .

5.4.2.2 Σημειολογικά στοιχεία διαγραμμάτων σεναρίου

Τώρα που γνωρίζουμε τι είναι ένα σενάριο και γιατί πρέπει να το χρησιμοποιούμε, ας ρίξουμε μια ματιά σε ένα.



Σχήμα 5.4.2.2-1: Διάγραμμα σεναρίου

Δεν υπάρχουν πολλά να πούμε για ένα σενάριο. Είναι πάνω κάτω απλό με πολύ λίγα σημειολογικά συστατικά στοιχεία. Μετά από μια γρήγορη ματιά στο προηγούμενο διάγραμμα, θα πρέπει να έχετε μια σχετικά καλή ιδέα του τι αντιπροσωπεύει.
Αναλύοντας σε μέρη το διάγραμμα σεναρίου, βρίσκουμε ότι αποτελείται από τέσσερα βασικά συστατικά στοιχεία:

- Σύστημα
- Δράστες
- Σενάρια
- Συσχέτιση

Ενώ τα σενάρια είναι σχετικά απλά, θα εξερευνήσουμε το καθένα από τα προηγούμενα συστατικά στοιχεία για να βεβαιωθούμε ότι έχετε καταλάβει τι αντιπροσωπεύουν.

5.4.2.2.1 Σύστημα

Ένα σύστημα (system), όπως ξεκινήσαμε να αναφέρουμε νωρίτερα σε αυτήν την ενότητα, είναι κάτι που εκτελεί μια λειτουργία. Έτσι ένα σύστημα μπορεί να είναι και

ένα κομμάτι ή πολλά κομμάτια ενός λογισμικού που εκτελεί κάποιου είδους λειτουργίες για τους χρήστες του.

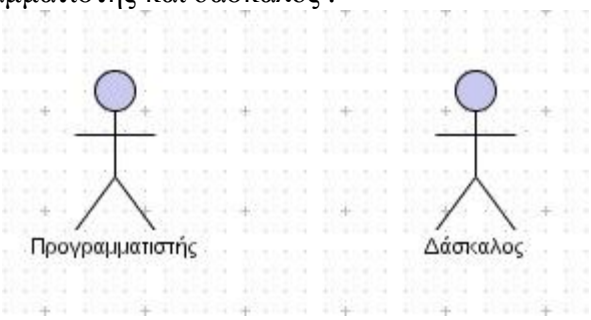
Σε κάθε διάγραμμα σεναρίου, οι δράστες και τα σενάρια είναι μέρη ενός απλού συστήματος που δεν είναι καν αναγνωρισμένο. Παίρνουμε ως δεδομένο ότι το σύστημα για το οποίο συζητάμε είναι το σύστημα που σχεδιάζουμε και ότι το σενάριο είναι μέρος μιας μεγαλύτερης λύσης μοντελοποίησης ενός απλού συστήματος.

Είναι πιθανόν ένα σύστημα να έχει υποκατηγορίες, οι οποίες είναι οργανωμένα συστατικά στοιχεία μέσα στο σύνολο του συστήματος. Για τις υποκατηγορίες θα μιλήσουμε με λεπτομέρειες στο κεφάλαιο «Υποκατηγορίες», πιο κάτω σε αυτήν την ενότητα.

5.4.2.2.2 Δράστες

Το πιο κοινό συστατικό στοιχείο των σεναρίων είναι ο δράστης (actor). Ο δράστης αντιπροσωπεύει συνήθως κάτι που χρησιμοποιεί το σύστημά μας. Ένας δράστης μπορεί να είναι ένα πρόσωπο ή ένα άλλο σύστημα. Ένας δράστης απεικονίζεται με μια φιγούρα (ένα ανθρωπάκι) με το όνομα του ρόλου του δράστη από κάτω.

Όταν ονομάζετε τους δράστες στα σενάρια σας, είναι σημαντικό να τους δίνετε ένα κατάλληλο όνομα που να περιγράφει καλύτερα τη λειτουργία του ως χρήστης ενός συστήματος. Αποφύγετε να δώσετε στους δράστες ονόματα που αντιπροσωπεύουν πραγματικά ονόματα ανθρώπων. Αντί αυτού, πρέπει να ονομάσετε τους δράστες σύμφωνα με τον τίτλο εργασίας τους όταν χρησιμοποιούν το σύστημα. Για παράδειγμα, προγραμματιστής και δάσκαλος .



Σχήμα 5.4.2.2.2-1: Δράστες

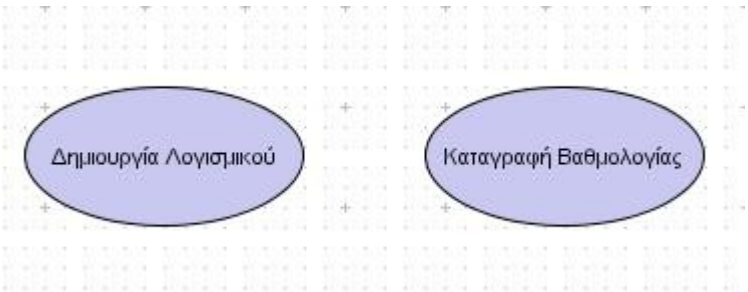
Όταν κατασκευάζετε τα σενάρια, θέλετε να απεικονίσετε κάθε πιθανό δράστη (χρήστη) του συστήματός σας. Αν δίνατε σε όλους κανονικά ονόματα, θα είχατε μπροστά σας πολλή δουλειά προσπαθώντας να κάνετε μια λίστα με όλους τους developers που ξέρατε ότι θα χρησιμοποιούσαν το σύστημα (Νίκος, Γιάννης και ούτω καθεξής). Αν απλώς κάνετε τη λίστα “developer”, θα σας καλύπτει για όλους.
Οι δράστες δεν είναι απαραιτήτως άνθρωποι. Μπορεί να είναι άλλα συστήματα εξωτερικά από το σύστημα που μοντελοποιείτε.

5.4.2.2.3 Υποδείγματα εργασίας

Τα σενάρια είναι οι πράξεις που ένας χρήστης αναλαμβάνει να εκτελέσει σε ένα σύστημα. Για παράδειγμα, ένας developer θα δημιουργούσε λογισμικό με ένα σύστημα ανάπτυξης, ένας δάσκαλος θα κατέγραφε τις βαθμολογίες με ένα σύστημα βαθμολογίας και ένα τέρας θα έκανε ένα σωρό διαφορετικά πράγματα με ένα σύστημα που προκαλεί το φόβο.

Οι περιπτώσεις χρήσης μπορεί να είναι τόσο ειδικές που δεν μπορούν να περιλαμβάνουν κάποια άλλη λειτουργικότητα, όπως Βάλτε το Γράμμα Χ, ούτε μπορούν να είναι πολύ ανώτερου επιπέδου, όπως Δακτυλογραφήστε Γράμματα. Μερικές φορές, τα σενάρια αντιστοιχούν σε συγκεκριμένη λειτουργικότητα που θα βλέπατε στο τελικό προϊόν, όπως οι λειτουργίες που παρατίθενται στα στοιχεία του μενού (όπως Print Preview, Page Setup και Print Document) αλλά πιο συχνά είναι αφηρημένα κομμάτια μιας λειτουργικότητας που είναι πολύ πιο γενική, όπως Allow Printing. Πώς το προϊόν επιτρέπει την εκτύπωση είναι μια τεχνική σχεδίασης, όχι μια τεχνική ανάλυσης της μοντελοποίησης των σεναρίων.

Το παρακάτω σχήμα απεικονίζει μερικά διαφορετικά σενάρια που ανήκουν στα τρία συστήματα που περιγράψαμε. Όπως θα προσέξουμε, οι περιπτώσεις χρήσης είναι ένα οβάλ με ένα όνομα μέσα.



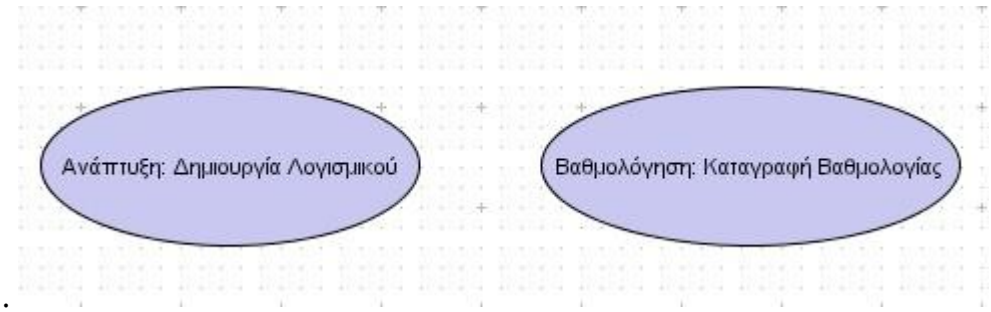
Σχήμα 5.4.2.2.3-1: *Περιπτώσεις χρήσης*

Μπορούμε να μετατρέψουμε αυτή τη σημειολογία λίγο τοποθετώντας το όνομα των σενάριων έξω από το οβάλ, κάτω από αυτό. Αν αποφασίσουμε να χρησιμοποιήσουμε αυτή την τεχνική, όμως, σας παροτρύνουμε να μην τη μπερδέψετε με την προηγούμενη τεχνική τοποθέτησης ονόματος των σενάριων μέσα στο οβάλ. Τοποθετώντας και τις δύο σημειολογίες στο ίδιο διάγραμμα, ρισκάρουμε να μπερδέψουμε πιθανούς αναγνώστες του μοντέλου.

Η ονομασία των σενάριων είναι το ίδιο σημαντική με την ονομασία ενός δράστη. Τα σενάρια πρέπει να είναι αλγοριθμικά κείμενα με αριθμούς, γράμματα και οποιαδήποτε σημεία στίξης εκτός της άνω κάτω τελείας, που είναι περιορισμένη.
Όταν κατασκευάζουμε σενάρια, προσπαθήσουμε να χρησιμοποιισουμε ρήματα στην ενεργητική φωνή σε συνδυασμό με τη λειτουργικότητα που εκτελεί το σύστημα, για παράδειγμα Τρακάρω Αυτοκίνητο, Χάνω Λεφτά και Κάνω Ασφάλεια.

Έχοντας τα τρία προηγούμενα σενάρια αιωρούμενα γύρω από το ίδιο διάγραμμα σεναρίου θα μπορούσε να γίνει πολύ περίπλοκο γιατί, όπως είπαμε νωρίτερα, το σύστημα δεν αναγνωρίζεται και υποτίθεται ότι το διάγραμμα απεικονίζει σενάρια του συνόλου του συστήματος. Είναι, όμως, προφανές διαβάζοντας αυτήν ενότητα ότι τα

σενάρια Δημιουργώ Λογισμικό, Καταγράφω Βαθμολογίες και Φοβίζω Κάποιον δεν είναι μέρος του ίδιου συστήματος ή τουλάχιστον όχι μέρος του ίδιου πακέτου.
Ένα πακέτο είναι μια υποκατηγορία ενός συστήματος που έχει συγγενή λειτουργικότητα. Μπορούμε να αναγνωρίσουμε από ποιο πακέτο προέρχονται τα σενάρια βάζοντας ένα πρόθεμα στο όνομα του πακέτου και δύο άνω κάτω τελείες, όπως Ανάπτυξη::, Βαθμολόγηση::

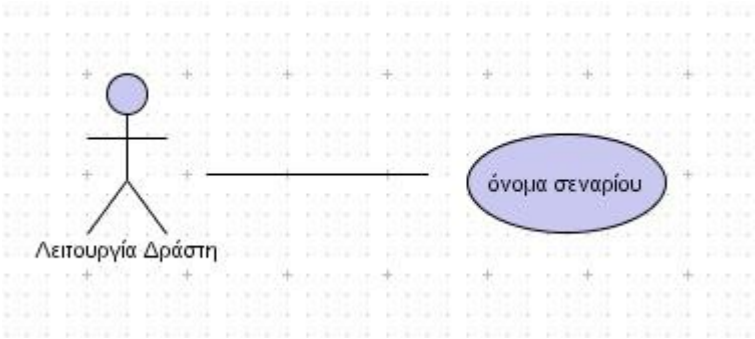


Σχήμα 5.4.2.2.3-2: Χρήση προθέματος

Η χρήση της διπλής άνω κάτω τελείας στο όνομα του πακέτου είναι ο λόγος που οι άνω κάτω τελείες δεν είναι έγκυροι χαρακτήρες στο όνομα των σεναρίων.

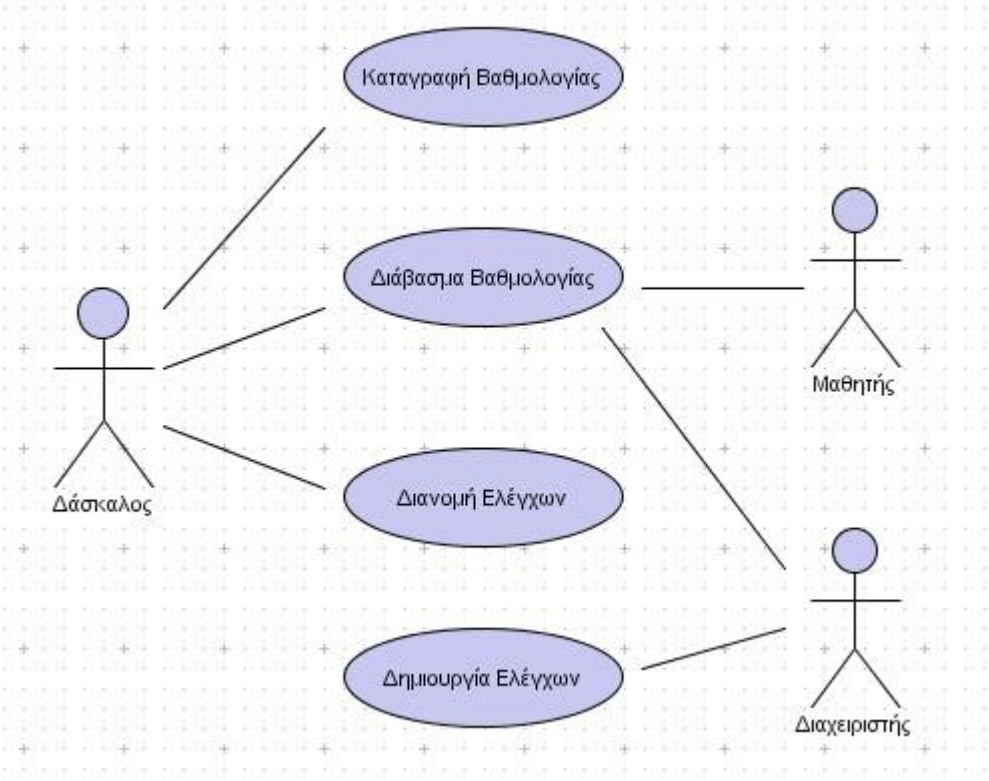
5.4.2.2.4 Συσχετίσεις

Ας έρθουμε τώρα στο «ψητό» των σεναρίων, τις συσχετίσεις. Οι συσχετίσεις απεικονίζονται απλώς με μια γραμμή που συνδέει τους δράσεις με τα σενάρια



Σχήμα 5.4.2.2.4-1: Συσχέτιση ανάμεσα σε δράστη και σενάριο

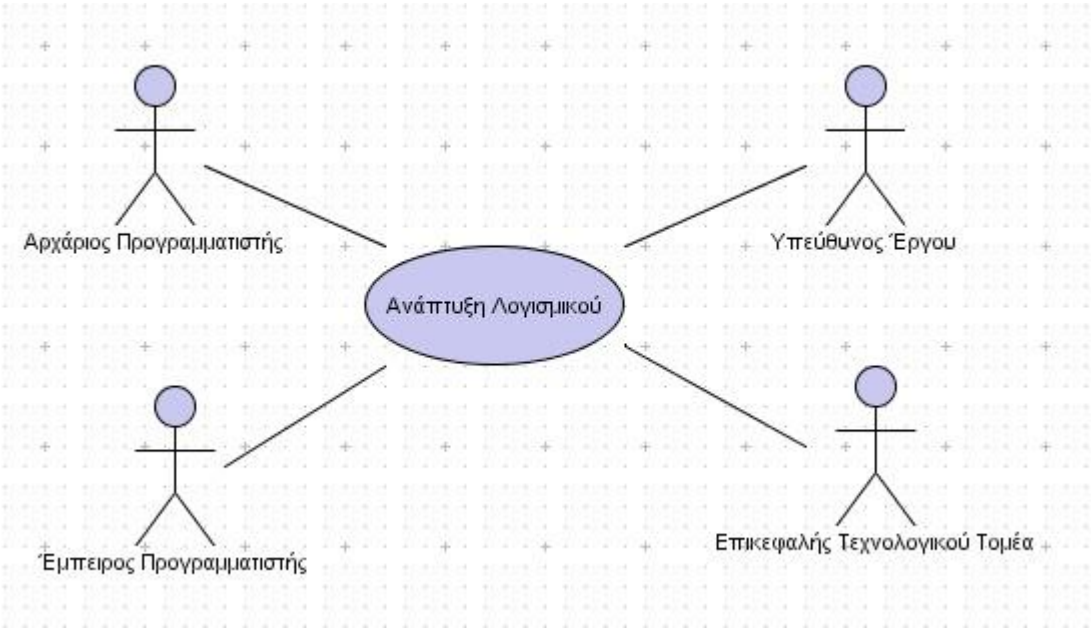
Στο παραπάνω παράδειγμα, βλέπουμε ότι ένας developer δημιουργεί λογισμικό, ένας δάσκαλος καταγράφει βαθμολογίες και ένα τέρας τρομάζει κάποιον. Οι δράστες μπορούν να συσχετιστούν με πολλά σενάρια και τα σενάρια με πολλούς δράστες, όπως βλέπουμε όταν αρχίζουμε να διευρύνουμε το σύστημα βαθμολογίας.



Σχήμα 5.4.2.2.4-2: Σσυσχέτιση ενός δράστη με πολλά σενάρια

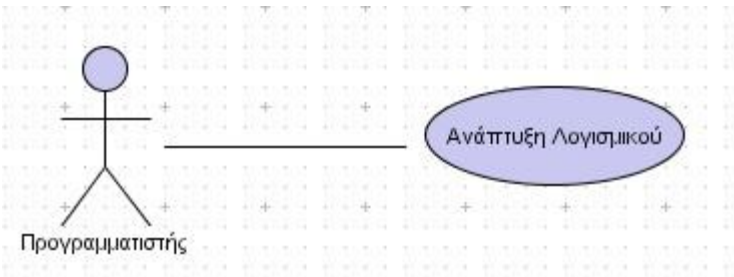
Όπως μπορείτε να δείτε, έχουμε τρεις δράστες: ένα δάσκαλο, ένα μαθητή και έναν διαχειριστή εκτύπωσης. Ο δάσκαλος μπορεί να καταγράψει βαθμολογίες, να δει βαθμολογίες και να μοιράσει ελέγχους. Ένας μαθητής μπορεί να δει βαθμολογίες (ίσως μέσα από ένα πρόγραμμα online). Ένας διαχειριστής εκτύπωσης μπορεί να δημιουργήσει ελέγχους και να δει τις βαθμολογίες για να δει αν όλοι οι έλεγχοι είναι σωστοί.

Καθώς ξεκινάμε να δημιουργούμε συσχετίσεις ανάμεσα στους δράστες μας και τα σενάρια, ξεκινάμε να συνειδητοποιούμε πόσο σημαντικό είναι να διαλέγουμε καλά ονόματα για τους δράστες μας. Μερικές φορές ο τίτλος ενός ατόμου δεν είναι το κατάλληλο όνομα για ένα δράστη. Μια πιο ακριβής λύση θα ήταν να ονομάσουμε ένα δράστη σύμφωνα με το ρόλο που παίζει σε ένα σύστημα – έστω κι αν αυτό αντιτίθεται με τον τίτλο του χρήστη στην πραγματική ζωή. Η παρακάτω απεικόνιση παρουσιάζει κακώς επιλεγμένα ονόματα δραστών



Σχήμα 5.4.2.2.4-3: Συσχέτιση πολλών δραστών με ένα σενάριο

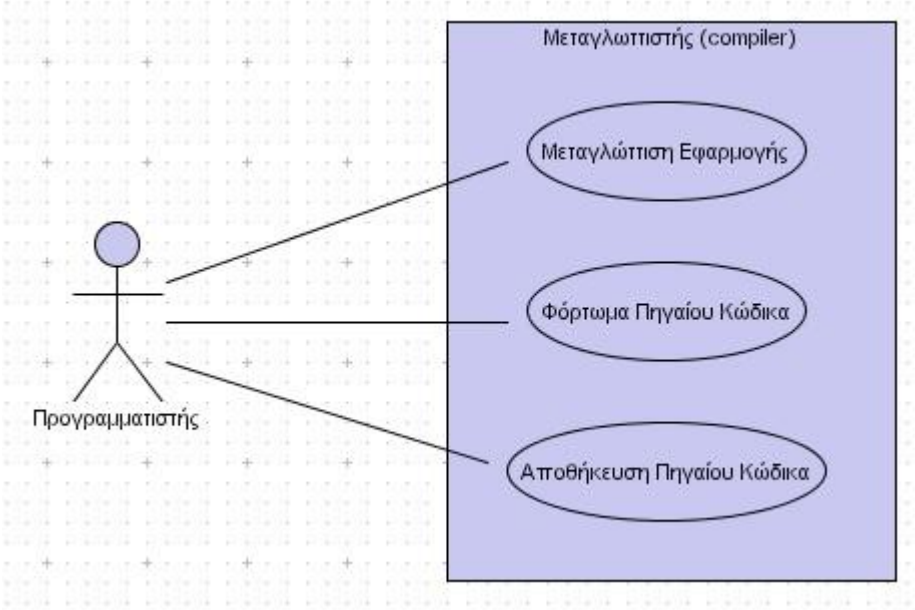
Αντί αυτού, θα θέλατε να γενικεύσετε τα ονόματα των δραστών έτσι ώστε τα ονόματα να περιγράφουν καλύτερα το ρόλο που οι δράστες παίζουν στο σύστημα. Αντί των Αρχάριος προγραμματιστής, Έμπειρος προγραμματιστής, Υπεύθυνος έργου και Επικεφαλής τεχνολογικού τομέα ως ξεχωριστών δραστών, πρέπει απλώς να δημιουργήσετε έναν που ονομάζεται Προγραμματιστής γιατί χρησιμοποιούν το ίδιο σύστημα για την ανάπτυξη του λογισμικού. Αυτό το όνομα είναι κατάλληλο ακόμη για τον Επικεφαλής τεχνολογικού τομέα – αν κι αυτό το άτομο ίσως να μην έχει τον τίτλο του προγραμματιστή, αλλά κάνει την ίδια δουλειά όταν χρησιμοποιεί το σύστημα.



Σχήμα 5.4.2.2.4-4: Γενίκευση σχήματος 5.4.2.2.4-3

5.4.2.2.5 Υποκατηγορίες

Μερικές φορές μπορεί να μοντελοποιείτε ένα πολύ μεγάλο σύστημα που είναι ευκολότερο να μελετηθεί αν διασπαστεί σε περισσότερα διατηρούμενα κομμάτια. Μπορείτε να δημιουργήσετε τέτοια κομμάτια, που ονομάζονται υποκατηγορίες.



Σχήμα 5.4.2.2.5-1: Χρήση συστήματος

Στο προηγούμενο παράδειγμα, βλέπουμε ότι ο προγραμματιστής χρησιμοποιεί το υποσύστημα Μεταγλωτιστής (compiler). Είναι ενα υποσυστήματα του συνολικού συστήματός μας, Περιβάλλον Ανάπτυξης. Άλλα υποσυστήματα που μπορεί να εμπεριέχονται στο σύστημα Περιβάλλον Ανάπτυξης θα μπορούσαν να είναι ο Συντάκτης (editor), ο Διορθωτής (debugger) και η Πηγή Ελέγχου (source control).

5.4.2.3 Κατανοώντας τη γενίκευση

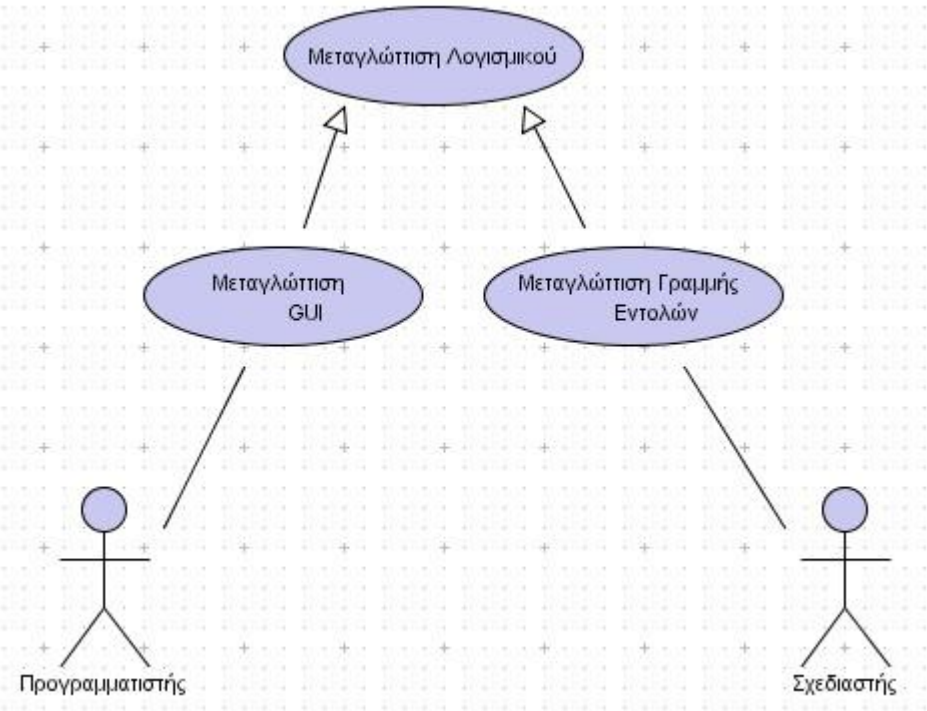
Η γενίκευση (generalization) είναι μια τεχνική που χρησιμοποιείται για να υποδείξει την κληρονομικότητα ενός στοιχείου στη UML. Η γενίκευση μπορεί να εφαρμοστεί και στους δράστες και στα σενάρια για να υποδείξουν ότι θυγατρικά στοιχεία κληρονομούν λειτουργικότητες από τα γονικά στοιχεία. Επιπλέον, αυτό υποδεικνύει ότι κάθε θυγατρικό στοιχείο του γονικού έχει ελαφρώς διαφορετική λειτουργικότητα ή σκοπό από το επόμενο έτσι ώστε να εξασφαλίσει τη μοναδικότητά του.

5.4.2.3.1 Συσχετισμός σεναρίου με σενάριο

Είναι ευκολότερο να κατανοήσει κανείς τη γενίκευση ενώ την εφαρμόζει στα σενάρια συγκρινόμενη με τους δράστες. Πάρετε, για παράδειγμα, την περίπτωση ενός δράστη που είναι ένα σύστημα.

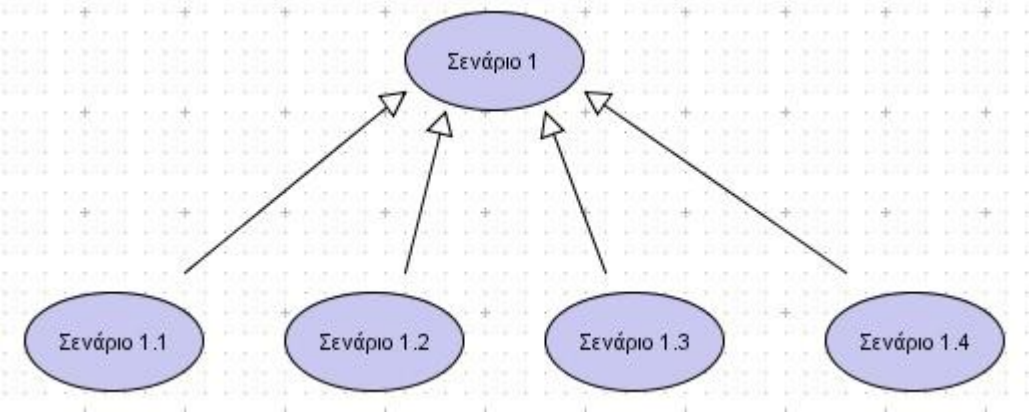
Σχήμα 5.4.2.3.1-1: Συσχετισμός σεναρίων

Αυτό το διάγραμμα ίσως έρθει σε αντίθεση με την προηγούμενη συμβουλή ότι όταν δύο ή περισσότεροι δράστες χρησιμοποιούν τα ίδια σενάρια, πρέπει να είναι περισσότερο γενικευμένοι για να αφαιρέσουν τους ατομικούς δράστες. Παρόλα' αυτά, με τη γενίκευση, αυτό δεν εφαρμόζεται σε αυτό το διάγραμμα, γιατί αν και τόσο ο προγραμματιστής όσο και ο σχεδιαστής μεταγλωττίζουν την εφαρμογή, ο καθένας το κάνει διαφορετικά.



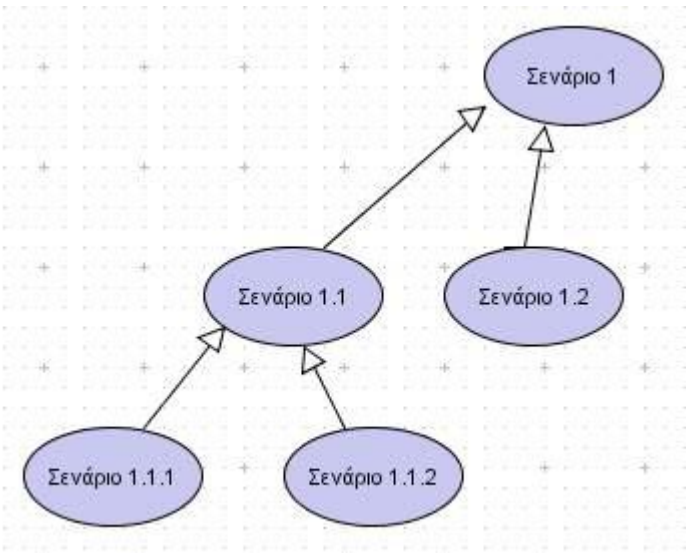
Σχήμα 5.4.2.3.1-2: Σχεδιάγραμμα με γενίκευση

Όπως μπορείτε να δείτε, η σημειολογία για τη γενίκευση είναι εύκολη. Το καθένα από τα δύο νέα σενάρια είναι θυγατρικά των αρχικών γονικών σεναρίων.
Η σημειολογία που τα συσχετίζει μεταξύ τους ονομάζεται γενίκευση και φαίνεται με ένα κενό βελάκι και μια γραμμή. Προχωρώντας κι άλλο αυτό το παράδειγμα, βλέπουμε ότι αυτή η γενίκευση μπορεί να διασπαστεί σε περισσότερα από ένα σενάρια.



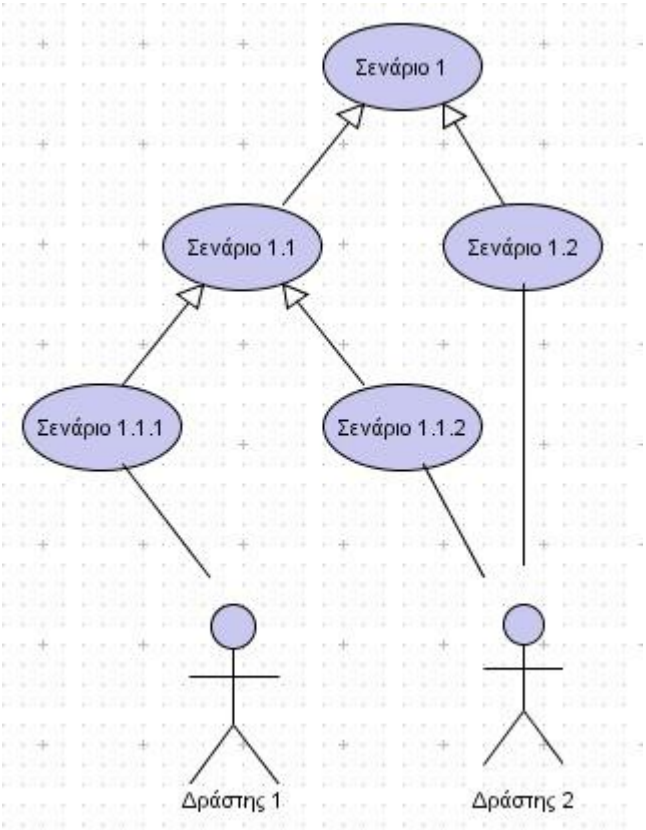
Σχήμα 5.4.2.3.1-3: Σχεδιάγραμμα με γενίκευση

Η γενίκευση μπορεί να είναι ιεραρχική, όπου τα θυγατρικά σενάρια ενός γονικού σεναρίου μπορούν να έχουν τους δικούς τους απογόνους.



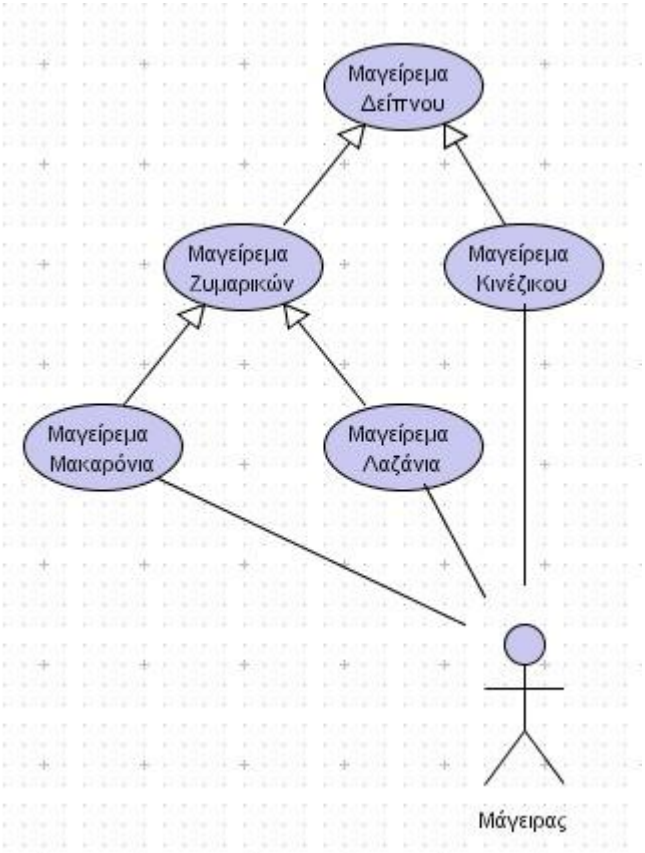
Σχήμα 5.4.2.3.1-4: Σχεδιάγραμμα με ιεραρχική γενίκευση

Αν έχετε γενικεύσει σενάρια και ένας δράστης χρησιμοποιεί ένα γενικευμένο σενάριο, δεν πρέπει να αφήσετε αυτό το δράστη να χρησιμοποιήσει ένα θυγατρικό αυτού του γενικευμένου σεναρίου. Στο προηγούμενο παράδειγμα, αυτό θα σήμαινε ότι αν ο δράστης σας χρησιμοποιεί το σενάριο 1.1.1., τότε αυτός ο δράστης δε θα πρέπει να χρησιμοποιεί το 1.1 ή 1.



Σχήμα 5.4.2.3.1-5: Σχεδιάγραμμα με γενίκευση

Στο προηγούμενο παράδειγμα, εάν ο δράστης 2 χρησιμοποιεί τα σενάρια 1.1.2 και 1.2, τι θα σήμαινε αν ο δράστης χρησιμοποιούσε και τα σενάρια 1;
Ίσως αυτό είναι καλύτερο να το εξηγήσουμε με ένα παράδειγμα. Στο επόμενο διάγραμμα, χρησιμοποιούμε την ίδια δομή σεναρίων όπως στα προηγούμενα, με ιεραρχική γενίκευση, και τα γεμίζουμε με διαφορετικές περιπτώσεις για να μαγειρέψουμε δείπνο.

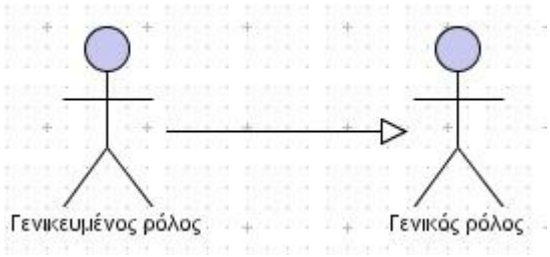


Σχήμα 5.4.2.3.1-6: Σχεδιάγραμμα γενίκευσης

Στο παράδειγμα, χάσαμε έναν από τους δράστες αλλά θα τον ανακτήσουμε στο επόμενο κομμάτι αυτής της ενότητας. Έχουμε έναν δράστη, Μάγειρα, που θα Μαγειρέψει Δείπνο, Μαγειρεύοντας Μακαρόνια, Μαγειρεύοντας Λαζάνια ή Μαγειρεύοντας Κινέζικο. Τι θα σήμαινε αν ο Μάγειρας χρησιμοποιούσε Μαγειρεύω Μακαρόνια και Μαγειρεύω Λαζάνια; Γενικεύοντας με το Μαγειρεύω Ζυμαρικά, το διάγραμμα απεικονίζει ότι τα θυγατρικά σενάρια, Μαγειρεύω Μακαρόνια και Μαγειρεύω Λαζάνια, κληρονομούν κοινή λειτουργικότητα με την ονομασία Μαγειρεύω Ζυμαρικά. Υπονοεί ότι η χρήση του σεναρίου Μαγειρεύω Ζυμαρικά από μόνη της δεν είναι αρκετή, αλλά μάλλον ένα μέρος της διαδικασίας που επιτρέπει στο δείπνο να μαγειρευτεί.

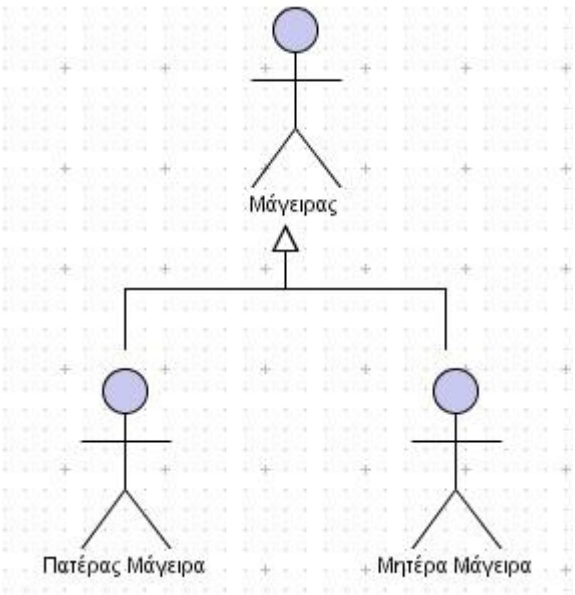
5.4.2.3.2 Συσχετισμός δράστη με δράστη

Τώρα, ας σκεφτούμε τη γενίκευση λίγο παραπέρα και ας την εφαρμόσουμε στους δράστες. Αυτό που θα ακολουθήσει είναι η σημειολογία για τη γενίκευση των δραστών. Είναι σαν τη γενίκευση των σεναρίων, εκτός του ότι αντικαθιστούμε τα δύο σενάρια με δύο δράστες.



Σχήμα 5.4.2.3.2-1: Γενίκευση δραστών

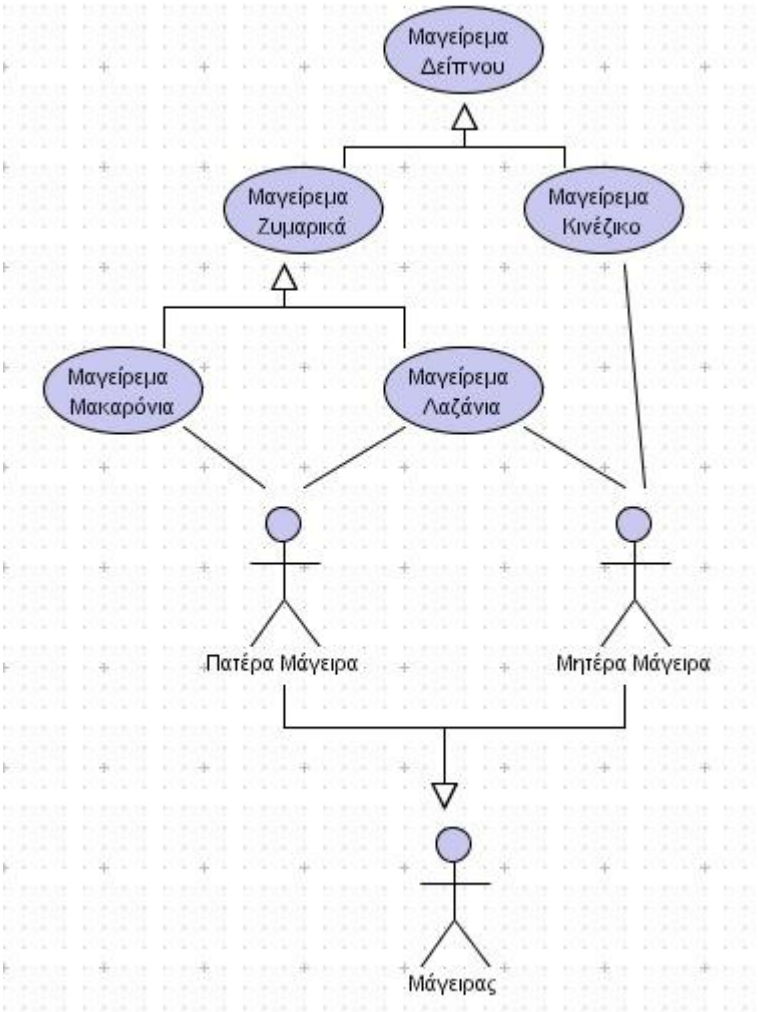
Η σημειολογία δείχνει ότι υπάρχει ένας γενικός δράστης και ένας γενικευμένος που παίζει έναν πιο συγκεκριμένο ρόλο στο σύστημα. Αν επεκτείνουμε το προηγούμενο μας παράδειγμα, μπορούμε να γενικεύσουμε το Μάγειρα σε Μητέρα Μάγειρα και Πατέρα Μάγειρα, τυπικοί στα περισσότερα νοικοκυριά.



Σχήμα 5.4.2.3.2-2: Γενίκευση δραστών

Σημειώστε ότι έχουμε αρχίσει να βλέπουμε μια διαφορετική σημειολογία για τη γενίκευση που είναι επίσης αποδεκτή. Σε αυτήν τη σημειολογία, βλέπουμε ότι όλοι οι γενικευμένοι δράστες ενώνουν το κενό βέλος της γενικευμένης σημειολογίας τους με ένα που δείχνει προς τον γενικό δράστη. Αυτό μπορεί να εφαρμοστεί και στη γενίκευση σεναρίων.

Τώρα, παίρνοντας τα προηγούμενα παραδείγματα, συνδυάζουμε τους γενικευμένους δράστες ούτως ώστε να ενωθούν με τα γενικευμένα σενάρια για ένα παράδειγμα ενός πολύ γενικευμένου διαγράμματος σεναρίου όπου ο Πατέρας Μάγειρας ξέρει πώς να Μαγειρέψει Σπαγγέτι ενώ η Μητέρα Μάγειρας ξέρει πώς να Μαγειρέψει Λαζάνια και να Μαγειρέψει Κινέζικο.



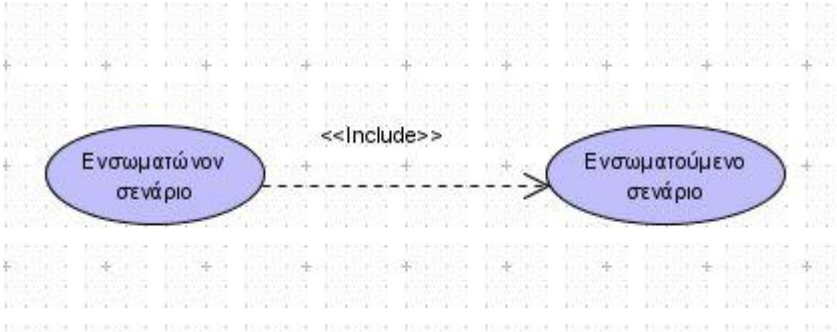
Σχήμα 5.4.2.3.2-3: Συνδυασμός γενικεύσεων

5.4.2.4 Σχέσεις ενσωμάτωσης και επέκτασης

Η ενσωμάτωση (inclusion) και η επέκταση (extension) είναι δύο τρόποι να συσχετίζετε τα σενάρια μεταξύ τους όταν είναι πολύ συγγενείς. Η σχέση ενσωμάτωσης χρησιμοποιείται για να υποδείξει ότι τα σενάρια θα συμπεριλάβουν λειτουργικότητα από ένα επιπλέον σενάριο για να εκτελέσουν τη λειτουργία της. Παρομοίως, η σχέση επέκτασης υποδεικνύει ότι ένα σενάριο μπορεί να επεκταθεί από ένα άλλο σενάριο.

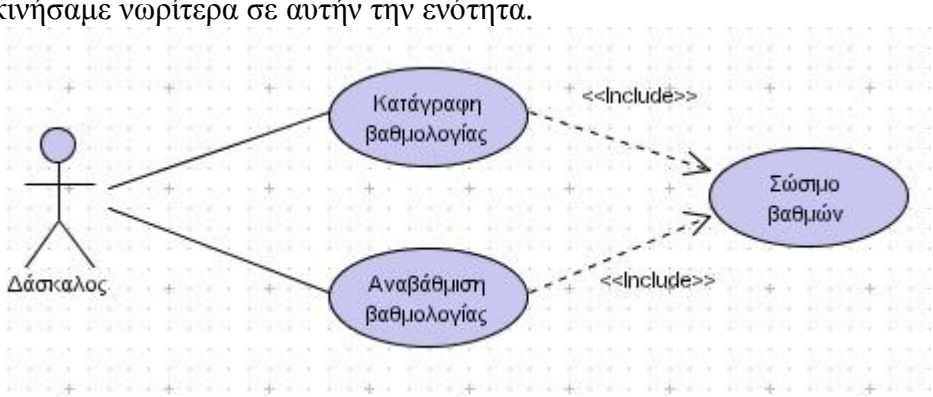
5.4.2.4.1 Σχέση ενσωμάτωσης

Το παρακάτω σχήμα είναι η απεικόνιση για τη σχέση ενσωμάτωσης:



Σχήμα 5.4.2.4.1-1: Σχέση ενσωμάτωσης

Υπάρχουν δύο τύποι σεναρίων, όπως φαίνονται σε αυτή τη σημειολογία. Ο πρώτος τύπος ονομάζεται ενσωματώνον σενάριο. Αυτό είναι ένα σενάριο που απαιτεί λειτουργικότητα (ενσωμάτωση λειτουργικότητας) από ένα άλλο σενάριο. Ο δεύτερος τύπος σεναρίου είναι το ενσωματωμένο σενάριο. Αυτό είναι το σενάριο που ενσωματώνεται στο πρώτο,. Τα δύο συνδέονται με μια διακεκομμένη γραμμή και ένα ανοιχτό βέλος, που ξεκινά από το ενσωματώνον σενάριο και καταλήγει στο ενσωματούμενο σενάριο. Τέλος, η λέξη «ενσωματώνω» («include») γράφεται κατά μήκους του βέλος και ενθυλακώνεται με εισαγωγικά («…»).
Για να κατανοήσετε πώς η σχέση ενσωμάτωσης λειτουργεί, κοιτάξτε το παρακάτω διάγραμμα σεναρίου που μοντελοποιεί ένα κομμάτι του συστήματος βαθμολογίας που ξεκινήσαμε νωρίτερα σε αυτήν την ενότητα.



Σχήμα 5.4.2.4.1-2: Διάγραμμα σεναρίου με τη χρήση ενσωμάτωσης

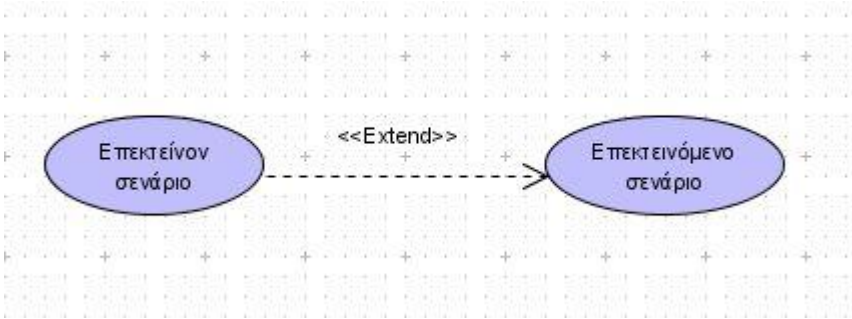
Αυτό το παράδειγμα δείχνει ότι ένας δάσκαλος θα καταγράψει βαθμολογίες και θα ενημερώσει βαθμολογίες. Δείχνει επίσης ότι και τα δύο σενάρια ενσωματώνουν ένα κοινό κομμάτι λειτουργικότητας από ένα σενάριο που ονομάζεται Αποθηκεύω Βαθμούς. Όποτε ένας δάσκαλος ενημερώνει (αναβαθμίζει) βαθμολογίες, οι βαθμολογίες πάντα θα αποθηκεύονται. Όποτε ένας δάσκαλος καταγράφει βαθμολογίες, οι βαθμολογίες πάντα θα αποθηκεύονται.

Είναι μια καλή ιδέα να αναγνωρίσετε τα ενσωματωμένα σενάρια στο μοντελοποιημένο σύστημά σας γιατί σας βοηθάει να αναγνωρίσετε πού μπορείτε να ξαναχρησιμοποιήσετε λειτουργικότητα όταν αναπτύσσετε το σύστημα σε μια εφαρμογή. Η επαναχρησιμοποίηση του κώδικα είναι ένα από τα μεγαλύτερα

πλεονεκτήματα του σχεδίου βασισμένου σε συστατικά στοιχεία του σχεδιασμού και της ανάπτυξης.

5.4.2.4.2 Σχέση επέκτασης

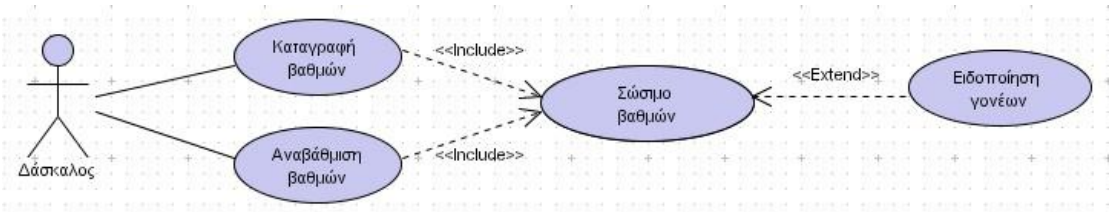
Πολύ παρόμοια με τη σχέση ενσωμάτωσης είναι η σχέση επέκτασης (extension). Η σημειολογία της σχέσης επέκτασης είναι όμοια με τη σημειολογία της σχέση ς ενσωμάτωσης, εκτός του ότι η λέξη επέκταση χρησιμοποιείται μέσα στα εισαγωγικά αντί της λέξης ενσωμάτωσης, για να υποδείξετε ότι ένα σενάριο μπορεί να επεκταθεί από ένα άλλο.



Σχήμα 5.4.2.4.2-1: Σχέση επέκτασης

Το σενάριο από όπου το διακεκομμένο βέλος προέρχεται αναφέρεται ως επεκτείνον σενάριο και το σενάριο στο τέλος της διακεκομμένης γραμμής, όπου το ανοιχτό βέλος δείχνει αναφέρεται ως επεκτεινόμενο σενάριο. Αυτή η σημειολογία χρησιμοποιείται για να αναγνωρίζετε πότε ένα σενάριο μπορεί κατ' επιλογή να επεκταθεί από τη λειτουργικότητα ενός άλλου σενάρια. Συγκρινόμενη με τη συσχέτιση ενσωμάτωσης, η συσχέτιση επέκτασης έχει την επιλογή χρήσης του επεκτεινόμενου σεναρίου.

Το παρακάτω παράδειγμα θα σας βοηθήσει να κατανοήσετε πώς η συσχέτιση επέκτασης χρησιμοποιείται στο πλαίσιο του συστήματος βαθμολόγησης. Το σενάριο Αποθηκεύω Βαθμολογίες επεκτείνεται από το σεναρίου Ειδοποιώ Κηδεμόνες. Διαβάζοντας αυτό το διάγραμμα, βλέπουμε ότι όταν ένας δάσκαλος καταγράφει ή ενημερώνει βαθμολογίες, οι βαθμοί αποθηκεύονται και, σε μερικές περιπτώσεις, οι κηδεμόνες ειδοποιούνται.

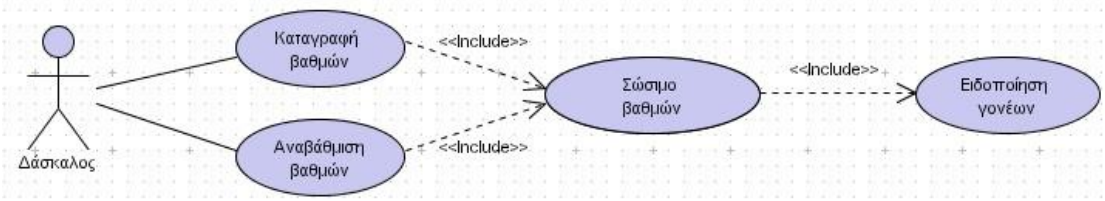


Σχήμα 5.4.2.4.2-2: Διάγραμμα σεναρίου με τη χρήση της επέκτασης

Ίσως αναρωτηθείτε γιατί το βέλος δείχνει από το σενάριο Ειδοποιώ Κηδεμόνες στο σενάριο Αποθηκεύω Βαθμολογίας και όχι αντίστροφα – άλλωστε, οι κηδεμόνες ειδοποιούνται αφού οι βαθμολογίες έχουν αποθηκευτεί (μερικές φορές). Ο λόγος είναι ότι το σενάριο Ειδοποιώ Κηδεμόνες επεκτείνεται στο σενάριο Αποθηκεύω Βαθμούς. Το σενάριο Ειδοποιώ Κηδεμόνες είναι μια λειτουργικότητα που

προστίθεται στη λειτουργία Αποθηκεύω Βαθμούς. Το σενάριο Αποθηκεύω Βαθμούς δεν προστίθεται στο σενάριο Ειδοποιώ Κηδεμόνες.

Αν οι κηδεμόνες ειδοποιούνται κάθε φορά που οι βαθμολογίες αποθηκεύονταν, η απεικόνιση θα χρησιμοποιούσε και μία συσχέτιση ενσωμάτωσης, όπως φαίνεται εδώ.

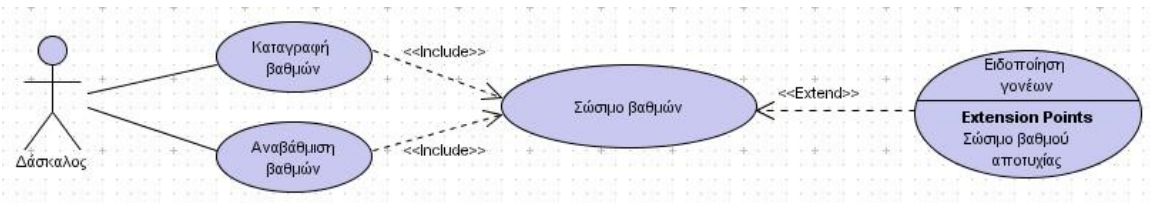


Σχήμα 5.4.2.4.2-3: Διάγραμμα σεναρίου με τη χρήση της επέκτασης και ενσωμάτωσης

Τώρα που καταλαβαίνετε τι είναι ένα επεκτείνον σενάριο και γιατί χρησιμοποιείται, ίσως διερωτηθείτε πώς μπορείτε να γνωρίζετε πότε οι κηδεμόνες ειδοποιούνται. Άλλωστε, ειδοποιούνται μόνο μερικές φορές και δεν πρέπει να είναι τυχαίο. Ας υποθέσουμε ότι θέλουμε να ειδοποιούμε τους κηδεμόνες ενός μαθητή όταν αποθηκεύεται ένας βαθμός αποτυχίας. Μπορούμε να το κάνουμε αυτό χρησιμοποιώντας τα σημεία επέκτασης, που επιτρέπουν να υποδείξουμε μια προϋπόθεση που θα επέτρεπε στην περίπτωση χρήσης να εισέλθει σε ένα επεκταθείς σενάριο. Η παρακάτω απεικόνιση χρησιμοποιείται για τα σημεία επέκτασης.

Σχήμα 5.4.2.4.2-4: Σημείο επέκτασης

Όπως μπορείτε να δείτε, μία οριζόντια γραμμή χωρίζει το επεκτεινόμενο σενάριο και το όνομα του σεναρίου μετακινείται στο πάνω μισό του οβάλ. Μέσα στο κάτω μισό του οβάλ, οι λέξεις «Σημεία Επέκτασης:» τονίζονται με bold με τη λίστα των πιθανών σημείων επέκτασης που θα οδηγούσαν σε οποιαδήποτε επεκτείνοντα σενάρια. Το παρακάτω είναι το επεκτεινόμενο σενάριο χρησιμοποιώντας απεικόνιση των σημείων επέκτασης για να υποδείξουν ότι το επεκτείνον σενάριο πρέπει να ακολουθείται αν αποθηκευτεί ένας βαθμός αποτυχίας.



Σχήμα 5.4.2.4.2-5: Διάγραμμα σεναρίου με τη χρήση της επέκτασης και ενσωμάτωσης και σημείου επέκτασης

5.4.2.5 Μοντελοποίηση διαγραμμάτων σεναρίων

Υπάρχουν πέντε στάδια για να δημιουργήσουμε σενάρια:

1. Προσδιορίζουμε τους δράστες και τα σενάρια στο σύστημα.
2. Δίνουμε προτεραιότητα στα σενάρια.
3. Καταγράφουμε λεπτομέρειες για κάθε σενάριο.
4. Σχεδιάζουμε το μοντέλο της περίπτωσης.
5. Προτοτυποποιούμε τις επαφές του χρήστη.

Θυμηθείτε ότι ένα άτομο μπορεί να παίξει το ρόλο ενός ή περισσοτέρων στο διάγραμμα και σε ορισμένες περιπτώσεις περισσότερα από ένα άτομο είναι απαραίτητα για να συμπληρώσουν ένα ρόλο.

5.4.2.6 Περίληψη

Ένα διάγραμμα σεναρίου δείχνει τη σχέση ανάμεσα σε δράστες και σενάρια μέσα σε ένα σύστημα.

Σημειολογία

Τα διαγράμματα σεναρίων δείχνουν στοιχεία από ένα μοντέλο σεναρίου. Το μοντέλο σεναρίου αντιπροσωπεύει τη λειτουργικότητα ενός συστήματος ή μίας κλάσης όπως εκδηλώθηκε από δράστες σε εξωτερικές αλληλεπιδράσεις με το σύστημα.

Συμβολισμός

Ένα διάγραμμα σεναρίου είναι ένα γράφημα δραστών, ένα σύνολο σεναρίων που περιέχονται σε ένα οριοθετημένο μέρος του συστήματος, διασυνδέσεις επικοινωνίας (συμμετοχή) ανάμεσα σε δράστες και σενάρια και γενικεύσεις ανάμεσα στα σενάρια.

.

5.4.3 ΔΙΑΓΡΑΜΜΑΤΑ ΑΚΟΛΟΥΘΙΑΣ

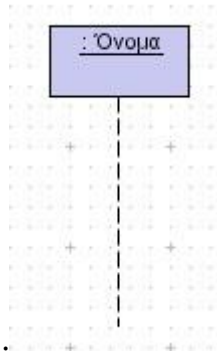
5.4.3.1 Τί είναι ένα διάγραμμα ακολουθίας

Το διάγραμμα ακολουθίας (sequence diagram) αποτελείται από αντικείμενα που απεικονίζονται με το συνηθισμένο τρόπο (ως ορθογώνια με το όνομα υπογραμμισμένο), μηνύματα που σημειώνονται ως πλήρη ίσια βέλη και χρόνο που σημειώνεται με κατακόρυφη γραμμή.

5.4.3.1.1 Ενεργά αντικείμενα

Τα αντικείμενα σχεδιάζονται στο πάνω μέρος του διαγράμματος, από τα αριστερά προς τα δεξιά. Τακτοποιούνται με οποιοδήποτε τρόπο που απλουστεύει το διάγραμμα.

Επεκτεινόμενη προς τα κάτω από κάθε αντικείμενο είναι μια διακεκομμένη γραμμή που ονομάζεται γραμμή ζωής (lifeline) του αντικειμένου. Κατά μήκος της γραμμής ζωής βρίσκεται ένα στενό ορθογώνιο που ονομάζεται ενεργοποίηση. Το μήκος του ορθογωνίου παριστάνει την εκτέλεση μιας λειτουργίας του αντικειμένου. Η διάρκεια και ο χρόνος γενικά, παριστάνονται με τακτικό τρόπο. Αυτό σημαίνει ότι κάθε "παύλα" σε μια γραμμή ζωής συνήθως δε παριστάνει μια συγκεκριμένη μονάδα χρόνου αλλά δίνει μια γενική αίσθηση της διάρκειας. Το παρακάτω σχήμα δείχνει ένα αντικείμενο, γραμμή ζωής και ενεργοποίηση.



Σχήμα 5.4.3.1.1-1: Ένα αντικείμενο σε ένα διάγραμμα ακολουθίας

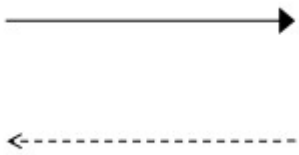
5.4.3.1.2 Μηνύματα

Ένα μήνυμα (message) που πηγαίνει από ένα αντικείμενο σε ένα άλλο, πάει από τη γραμμή ζωής του ενός αντικειμένου στη γραμμή ζωής του άλλου αντικειμένου. Ένα αντικείμενο μπορεί ακόμη να στείλει ένα μήνυμα στον εαυτό του, δηλαδή από τη δική του γραμμή ζωής πίσω στη δική του γραμμή ζωής.

Η UML παριστάνει ένα μήνυμα ως ένα βέλος που ξεκινάει από μία γραμμή ζωής και καταλήγει σε μια άλλη. Το σχήμα της κεφαλής του βέλους δείχνει τι είδους μήνυμα είναι. Στη UML 1.1 τρία είδη κεφαλών βέλους ήταν διαθέσιμα. Η UML 2.0 κατήργησε ένα από αυτά και ουσιαστικά διευκόλυνε την κατάσταση.

Ένας τύπος μηνύματος είναι η κλήση**.** Αυτό είναι ένα αίτημα του αντικειμένου που στέλνει το μήνυμα στο αντικείμενο που λαμβάνει το μήνυμα. Το αίτημα είναι για τον παραλήπτη να πραγματοποιήσει μια από τις λειτουργίες του (του παραλήπτη). Συνήθως, αυτό συνεπάγεται ότι ο αποστολέας περιμένει από τον παραλήπτη να πραγματοποιήσει αυτή τη λειτουργία. Γι' αυτό το λόγο, επειδή ο αποστολέας περιμένει τον παραλήπτη (ο αποστολέας "συγχρονίζεται" με τον παραλήπτη"), αυτό το μήνυμα λέγεται και συγχρονισμένο.

Η UML δηλώνει αυτό τον τύπο μηνύματος με συνεχόμενη γραμμή και μια "γεμάτη" κεφαλή βέλους. Είναι τυπικά η περίπτωση που μια κλήση περιλαμβάνει ένα μήνυμα επιστροφής από τον παραλήπτη. Το σύμβολο του μηνύματος επιστροφής είναι ένα βέλος με διακεκομμένη γραμμή και ανοιχτή κεφαλή. Παρακάτω φαίνονται αυτά τα σύμβολα:



Σχήμα 5.4.3.1.2-1: Η σημειολογία για μια κλήση κι ένα μήνυμα επιστροφής

Ένας άλλος τύπος μηνύματος είναι το ασύγχρονο**.** Με αυτό, ο αποστολέας μεταφέρει τον έλεγχο στον παραλήπτη και δεν περιμένει να ολοκληρωθεί η λειτουργία**.** Το σύμβολο γι' αυτό το μήνυμα είναι ένα βέλος με ανοιχτή κεφαλή, όπως δείχνεται παρακάτω:

Σχήμα 5.4.3.1.2-2: Ασύγχρονο μήνυμα

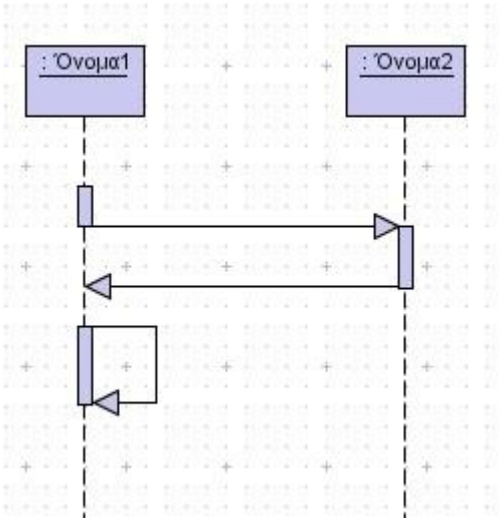
5.4.3.1.3 Χρόνος

Το διάγραμμα παριστάνει το χρόνο με κάθετη κατεύθυνση: ο χρόνος (time) ξεκινάει στο επάνω μέρος και προοδεύει προς τα κάτω. Ένα μήνυμα που είναι πιο κοντά στο επάνω μέρος πραγματοποιείται πιο πριν από ένα άλλο που είναι πιο κοντά στο κάτω μέρος.

Έτσι, το διάγραμμα ακολουθίας είναι δισδιάστατο. Στην διάσταση από αριστερά προς τα δεξιά παρατίθενται τα αντικείμενα και η διάσταση από πάνω προς τα κάτω δείχνει το πέρασμα του χρόνου.

Το σχέδιο παρακάτω δείχνει το κύριο συμβολισμό ενός διαγράμματος ακολουθίας, με τα σύμβολα να αλληλεπιδρούν. Τα αντικείμενα έχουν παραταχθεί από τα αριστερά προς τα δεξιά, στο επάνω μέρος. Η γραμμή ζωής του κάθε αντικειμένου είναι μια διακεκομμένη γραμμή που εκτείνεται προς τα κάτω από το αντικείμενο. Η

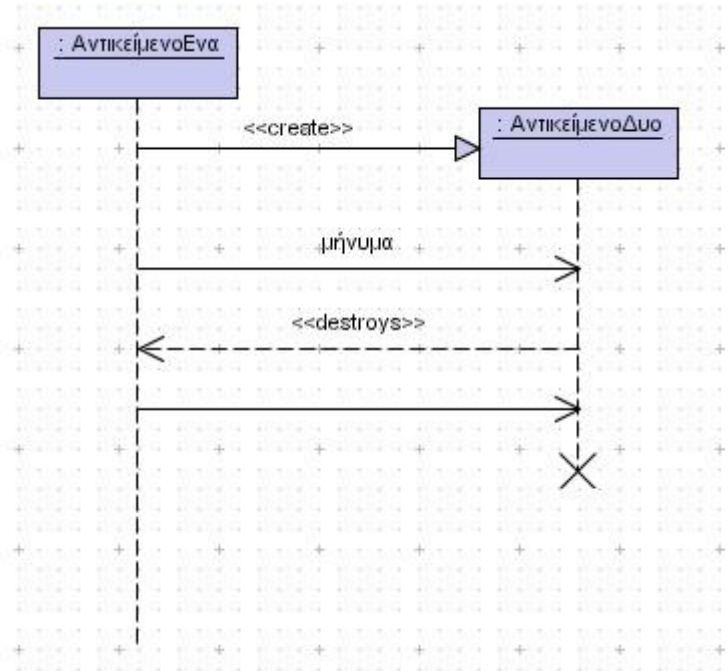
συνεχόμενη γραμμή με τη γεμάτη κεφαλή συνδέει μια γραμμή ζωής με μια άλλη και παριστάνει ένα μήνυμα από ένα αντικείμενο σε ένα άλλο.



Σχήμα 5.4.3.1.3-1: Τα σύμβολα σε ένα διάγραμμα ακολουθίας

5.4.3.2 Δημιουργία και διαγραφή αντικειμένων

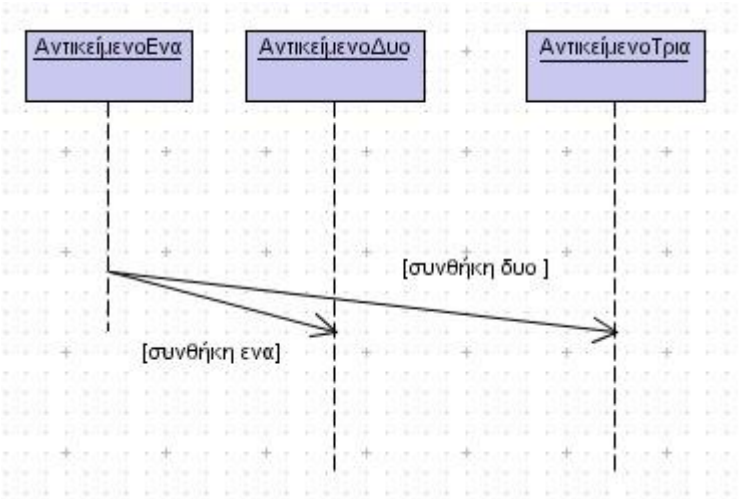
Υπάρχει ένα απαραίτητο βήμα που πρέπει να γίνει. Αυτό είναι να στείλει το ένα αντικείμενο το <<create>> μήνυμα στο νέο αντικείμενο. Όταν το αντικείμενο δημιουργηθεί, του δίνεται μια γραμμή ζωής, όπως κάθε άλλο αντικείμενο σε ένα διάγραμμα ακολουθίας. Απ' τη στιγμή που έχει δημιουργηθεί ένα νέο αντικείμενο, μπορούμε να στείλουμε το μήνυμα <<destroy>> για να το διαγράψουμε. Για να το υποδείξουμε αυτό, βάζουμε ένα Χ στη γραμμή ζωής στο σημείο που αυτό καταστρέφεται. Απ' το σημείο αυτό και μετά, δεν υπάρχει λόγος να συνεχίσουμε να επεκτείνουμε τη γραμμή ζωής του προς τα κάτω.



Σχήμα 5.4.3.2-1: Δημιουργία και καταστροφή ενός αντικειμένου

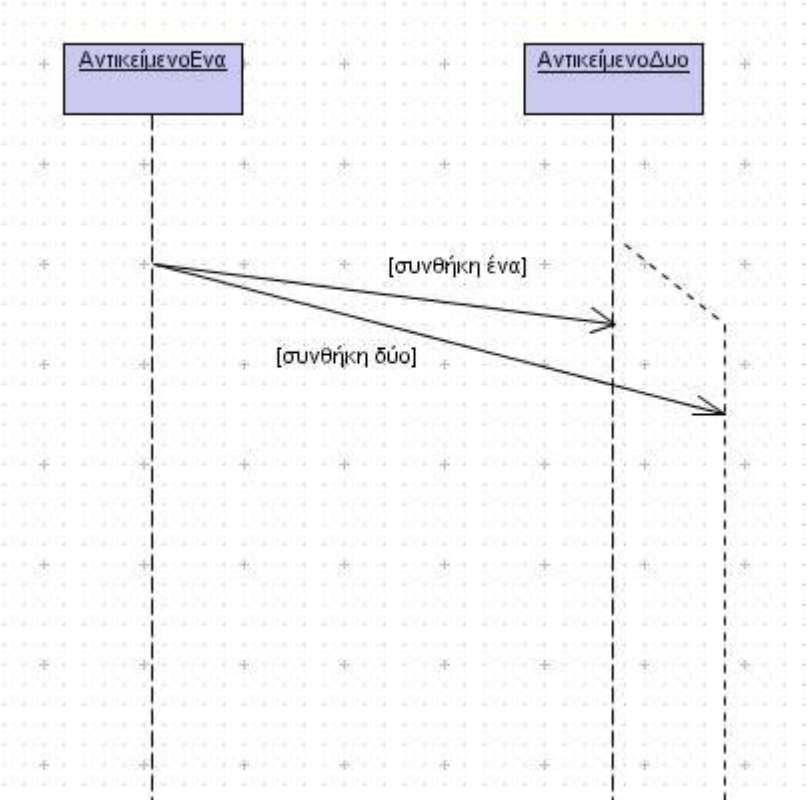
5.4.3.3 Διακλάδωση και εναλλακτικές ροές

Υπάρχουν δύο τρόποι ώστε τα διαγράμματα ακολουθίας να αλλάξουν τη ροή ελέγχου τους: με τη διακλάδωση (branching) και με τις εναλλακτικές ροές (alternative flows). Τα δυό αυτά είναι παρόμοια αλλά το καθένα έχει ελαφρώς διαφορετική απεικόνιση. Αλλαγές στη ροή ελέγχου συμβαίνουν όταν διαφορετικές καταστάσεις προκαλούν τη ροή να κινηθεί προς διαφορετικές κατευθύνσεις.
Η διακλάδωση επιτρέπει στη ροή να πάει σε διαφορετικά αντικείμενα.



Σχήμα 5.4.3.3-1: Διακλάδωση

Η εναλλακτική ροή επιτρέπει τη ροή ελέγχου να αλλάξει σύμφωνα με τις εκάστοτε συνθήκες, αλλά επιτρέπει στη ροή να αλλάξει σε μια διαφορετική γραμμή ζωής του ίδιου αντικειμένου.



Σχήμα 5.4.3.2-2: Εναλλακτική ροή

5.4.3.4 Μοντελοποίηση διαγραμμάτων ακολουθίας

Υπάρχουν τέσσερεις εργασίες για τη δημιουργία διαγραμμάτων ακολουθίας:

1. Αποφασίζουμε ποιες ροές εργασίας θα μοντελοποιηθούν.
2. Κατατάσσουμε τα αντικείμενα από αριστερά προς τα δεξιά
3. Συμπεριλάμβάνουμε μηνύματα και συνθήκες για την κάθε ροή εργασίας.
4. Σχεδιάζουμεένα γενικό διάγραμμα για να συνδυάσουμε τα ξεχωριστά διαγράμματα.

5.4.3.5 Περίληψη

Ένα πρότυπο αλληλεπιδράσεων ανάμεσα στα αντικείμενα φαίνεται στα διαγράμματα αλληλεπιδράσεων. Τα διαγράμματα αλληλεπίδρασης εμφανίζονται με δύο φόρμες βασισμένες στην ίδια υποκείμενη πληροφορία αλλά η καθεμία δίνει έμφαση σε μια συγκεκριμένη διάσταση αυτής: διαγράμματα ακολουθίας και διαγράμματα συνεργασίας.

Ένα διάγραμμα ακολουθίας δείχνει μια αλληλεπίδραση τοποθετημένη σε χρονική ακολουθία. Πιο συγκεκριμένα, δείχνει τα αντικείμενα που συμμετέχουν στην αλληλεπίδραση με τις «γραμμές επικοινωνίας» και τα μηνύματα που ανταλλάσουν διευθετημένα σε χρονικές ακολουθίες. Δε δείχνει τις συσχετίσεις ανάμεσα στα αντικείμενα.

Τα διαγράμματα ακολουθίας εμφανίζονται σε διαφορετικές μορφές, η καθεμιά από αυτές εξυπηρετεί διαφορετικούς σκοπούς.
Ένα διάγραμμα ακολουθίας μπορεί να υπάρχει σε γενική μορφή (περιγράφει όλες τις πιθανές ακολουθίες) και σε στιγμιαία μορφή (περιγράφει μία πραγματική ακολουθία σύμφωνη με τη γενική μορφή). Σε περιπτώσεις χωρίς βρόχους ή διακλαδώσεις, οι δύο μορφές είναι ισόμορφες.

Τα διαγράμματα ακολουθιών και τα διαγράμματα συνεργασίας εκφράζουν παρόμοιες πληροφορίες αλλά τις δείχνουν με διαφορετικούς τρόπους. Τα διαγράμματα ακολουθίας δείχνουν τη διεξοδική ακολουθία των μηνυμάτων και είναι καλύτερα για διευκρινίσεις πραγματικού χρόνου και για περίπλοκα σενάρια. Τα διάγραμμα συνεργασίας δείχνουν τις συσχετίσεις ανάμεσα στα αντικείμενα και είναι καλύτερα για την κατανόηση όλων των επιδράσεων σε ένα δεδομένο αντικείμενο και για το διαδικαστικό σχεδιασμό.

Σημειολογία

Ένα διάγραμμα ακολουθίας αντιπροσωπεύει μία αλληλεπίδραση που είναι ένα σύνολο μηνυμάτων, τα οποία ανταλλάσσονται ανάμεσα στα αντικείμενα σε συνεργασία για να πραγματοποιήσουν μια επιθυμητή λειτουργία ή αποτέλεσμα.

Συμβολισμός

Ένα διάγραμμα ακολουθίας έχει δύο διαστάσεις: την κάθετη διάσταση που αντιπροσωπεύει το χρόνο, την οριζόντια που αντιπροσωπεύει διαφορετικά αντικείμενα. Συνήθως ο χρόνος προχωρά όσο κανείς κατεβαίνει στη σελίδα (οι διαστάσεις μπορούν να αντιστραφούν αν το επιθυμείτε). Συνήθως μόνο οι χρονικές ακολουθίες είναι σημαντικές αλλά σε εφαρμογές πραγματικού χρόνου ο άξονας του χρόνου θα μπορούσε να είναι μία πραγματική μετρική. Δεν έχει κανένα νόημα η οριζόντια διάταξη των αντικειμένων. Τα αντικείμενα μπορούν να συγκροτηθούν σε λωρίδες (κουλουάρ) σε ένα διάγραμμα.

5.4.4 ΔΙΑΓΡΑΜΜΑΤΑ ΕΠΙΚΟΙΝΩΝΙΑΣ

Όπως το διάγραμμα ακολουθίας, το διάγραμμα επικοινωνίας (collaboration diagram) δείχνει πώς τα αντικείμενα αλληλεπιδρούν. Δείχνει τα αντικείμενα μαζί με τα μηνύματά τους που πάνε από το ένα αντικείμενο στο άλλο.
Οι δύο τύποι διαγραμμάτων – τα διαγράμματα ακολουθίας και επικοινωνίας – είναι παρόμοιοι. Στην πράξη παρουσιάζουν τις ίδιες πληροφορίες, και μπορούμε να μετατρέψουμε ένα διάγραμμα ακολουθίας σε ένα αντίστοιχο διάγραμμα επικοινωνίας, και το αντίστροφο.

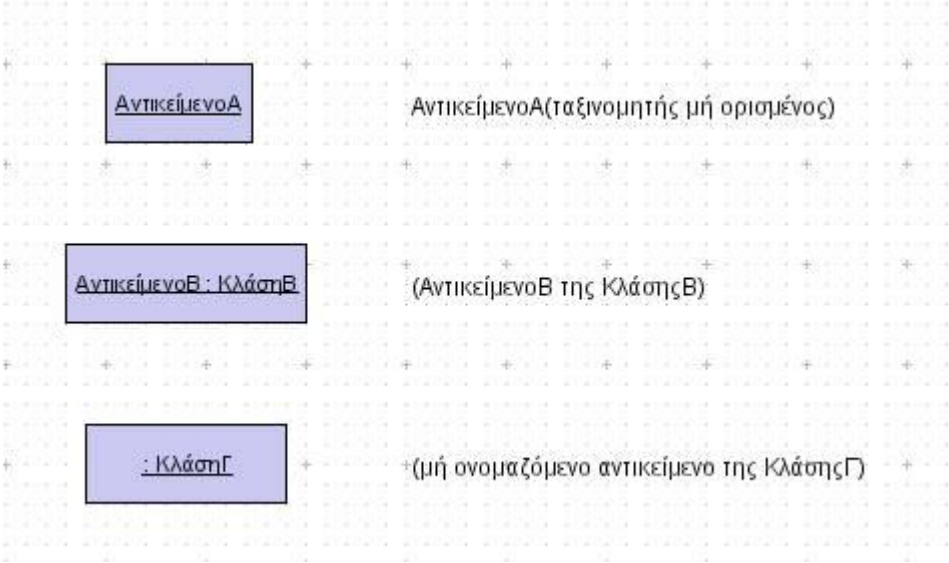
Είναι καλό να έχουμε και τους δύο τύπους. Το διάγραμμα ακολουθίας δίνει έμφαση στη σειρά των αλληλεπιδράσεων. Το διάγραμμα επικοινωνίας δίνει έμφαση στη γενικότερη οργάνωση των αντικειμένων που αλληλεπιδρούν. Και τα δύο απεικονίζουν τις αλληλεπιδράσεις ανάμεσα στα αντικείμενα και γι' αυτό το λόγο κάθε ένα από αυτά είναι ένας τύπος διαγράμματος αλληλεπίδρασης (interaction diagram).

5.4.4.1 Τι είναι ένα διάγραμμα επικοινωνίας

Ένα διάγραμμα επικοινωνίας είναι μια προέκταση ενός διαγράμματος αντικειμένων και δείχνει τα μηνύματα που στέλνουν μεταξύ τους τα αντικείμενα.
Για να οπτικοποιήσουμε ένα μήνυμα, σχεδιάζουμε ένα βέλος δίπλα στη σύνδεση ανάμεσα στα δύο αντίκειμενα. Το βέλος δείχνει προς το δεχόμενο αντικείμενο. Μια ετικέτα δίπλα στο βέλος δείχνει ποιο είναι το μήνυμα. Το μήνυμα τυπικά λέει στο δεχόμενο αντικείμενο να εκτελέσει μια από τις λειτουργίες του (του αποδέκτη).
Όπως αναφέραμε νωρίτερα, μπορούμε να μετατρέψουμε ένα διάγραμμα ακολουθίας σε διάγραμμα επικοινωνίας. Για το λόγο αυτό, προσθέτουμε ένα νούμερο στην ετικέτα ενός μηνύματος, με το νούμερο να αντιπροσωπεύει τη σειρά του μηνύματος στην ακολουθία. Μια άνω-κάτω τελεία χωρίζει το νούμερο απ'το μήνυμα.

5.4.4.1.1 Αντικείμενα και ρόλοι

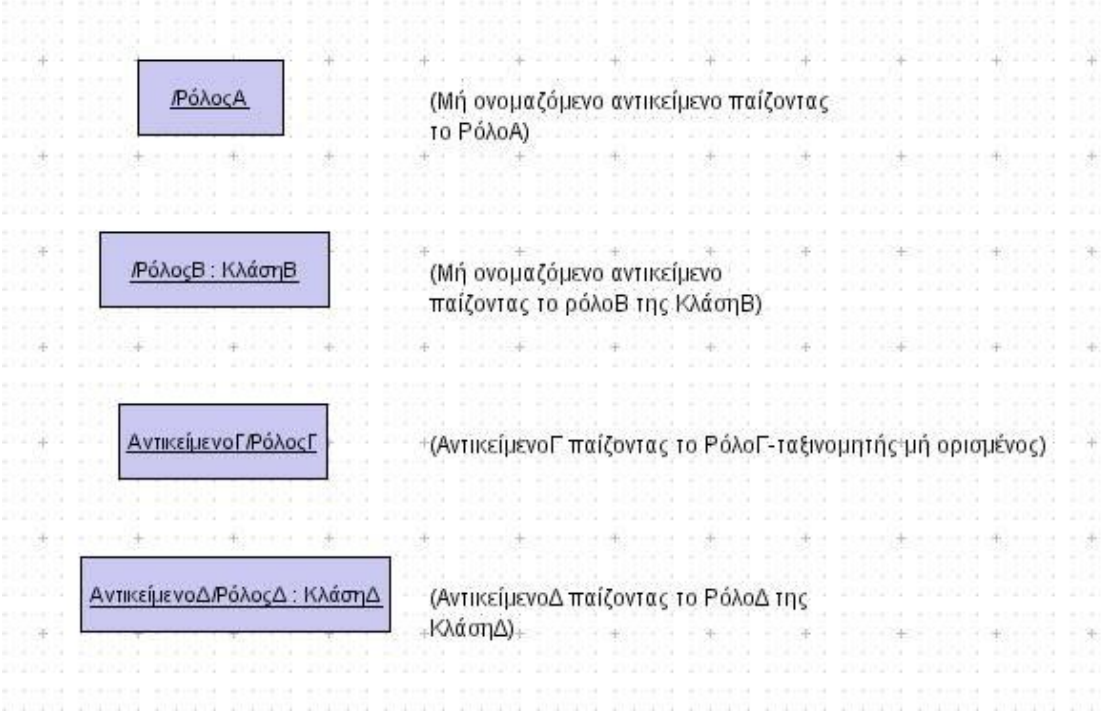
Επειδή ένα διάγραμμα επικοινωνίας μοντελοποιεί αλληλεπιδράσεις του συστήματος, πρέπει να κάνει το ίδιο με τα στιγμιότυπα κλάσεων που έχουμε σχεδιάσει έως τώρα. Υπάρχουν τρείς τύποι στιγμιοτύπων αντικειμένων (object instance) που μπορούμε να χρησιμοποιήσουμε στα διαγράμματά μας.



Σχήμα 5.4.4.1.1-1: Οι τρείς τύποι στιγμιοτύπων αντικειμένων

Ο πρώτος τύπος στιγμιότυπου αντικειμένου είναι τα αταξινόμητα στιγμιότυπα (unclassified instances). Το όνομά τους είναι απλώς υπογραμμισμένο ως ΑντικείμενοΑ. Αυτή η σημειολογία υποδεικνύει ότι η κλάση από την οποία αυτό το αντικείμενο είναι στιγμιότυπο είναι ασήμαντη ή άγνωστη στο συγκεκριμένο μοντέλο. Ο δεύτερος τύπος στιγμιότυπου αντικειμένου είναι το πλήρως κατάλληλο αντικείμενο (fully qualified object) που περιλαμβάνει ένα όνομα του αντικειμένου και τον ταξινομητή του (classifier). Αυτή η σημειολογία αναφέρεται σε ένα συγκεκριμένο , μοναδικό και ονομαζόμενο στιγμιότυπο. Σημειώνεται ως ΑντικείμενοΒ:ΚλάσηΒ.
Ο τρίτος τύπος στιγμιοτύπου αντικειμένου είναι το μη ονομαζόμενο, ταξινομημένο αντικείμενο (unnamed, classified instance). Αυτό το αντικείμενο είναι ονομαζόμενο με το όνομα της κλάσης του, υπογραμμισμένο και ύστερα από μια άνω-κάτω τελεία :ΚλάσηΓ.

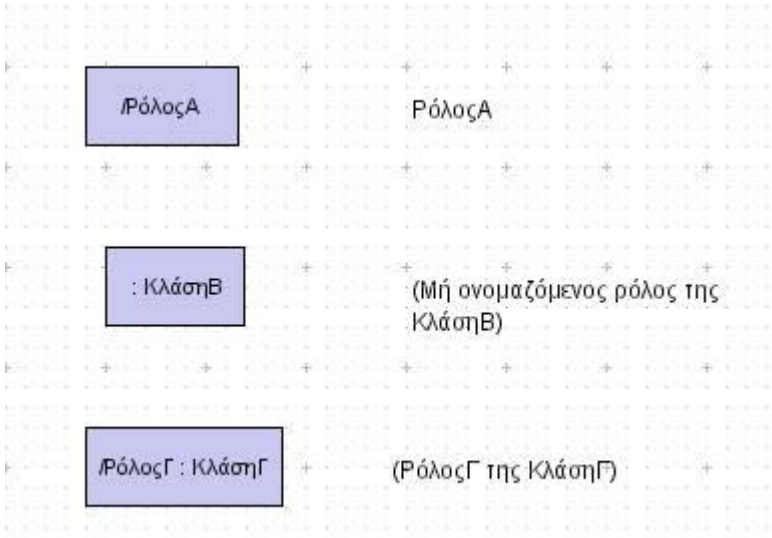
Επιπλέον, μπορούμε να δείξουμε τους ρόλους των στιγμιοτύπων των αντικειμένων στα διαγράμματα επικοινωνίας. Υπάρχουν τέσσερις τρόποι με τους οποίους μπορούμε να απεικονίσουμε τους ρόλους των στιγμιοτύπων των αντικειμένων.



Σχήμα 5.4.4.1.1-2: Οι τέσσερις τρόποι απεικόνισης των ρόλων των στιγμιοτύπων αντικειμένων

Η πρώτη σημειολογία που απεικονίζεται είναι ένα μη-ονομαζόμενο αντικείμενο που παίζει κάποιο ρόλο. Η δεύτερη σημειολογία είναι ένα μη-ονομαζόμενο αντικείμενο που παίζει ένα ρόλο μιας συγκεκριμένης κλάσης.

Οι τελευταίες δύο σημειολογίες για τους ρόλους των στιγμιοτύπων των αντικειμένων σχετίζονται άμεσα με τους πρώτους δύο, αλλά το αντικείμενο είναι ονομαζόμενο. Οι ρόλοι των κλάσεων μπορούν να χρησιμοποιηθούν κι αυτοί στα διαγράμματα. Σε αυτή την περίπτωση τους απεικονίζουμε δίχως την υπογράμμιση. Υπάρχουν τρείς επιλογές για τη σημειολογία των ρόλων των κλάσεων.



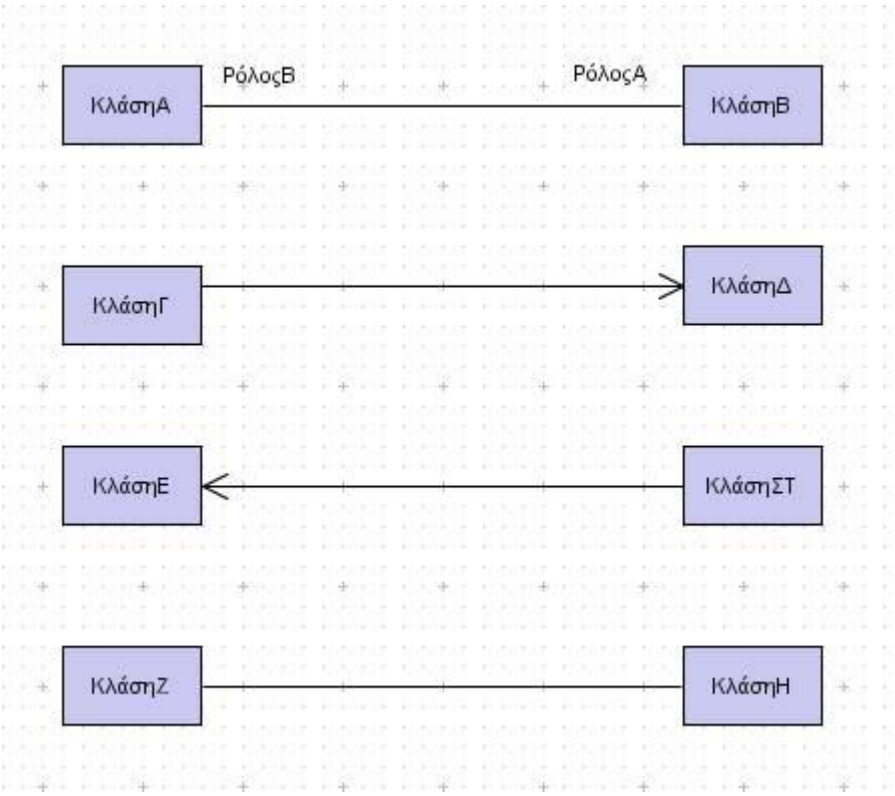
Σχήμα 5.4.4.1.1-3: Οι τρείς επιλογές για τη σημειολογία ρόλων των κλάσεων

Η πρώτη επιλογή είναι απλώς η ονομασία ενός ρόλου που δεν υποδεικνύει την κλάση που αντιπροσωπεύει. Ξέρουμε παρ' όλα αυτά ότι ο ρόλος είναι μιας κλάσης, επειδή

δεν είναι υπογραμμισμένος, όπως δείχνεται στην πρώτη κλάση του σχήματος. Μπορούμε να χρησιμοποιήσουμε το ακριβώς αντίθετο στο διάγραμμα επικοινωνίας, έναν μη ονομαζόμενο ρόλο με έναν ταξινομητή. Ακόμη μπορούμε να χρησιμοποιήσουμε ένα πλήρως κατάλληλο ρόλο κλάσης με το να διευκρινίζουμε το όνομα του ρόλου και το όνομα της κλάσης.

5.4.4.1.2 Ρόλοι συσχέτισης

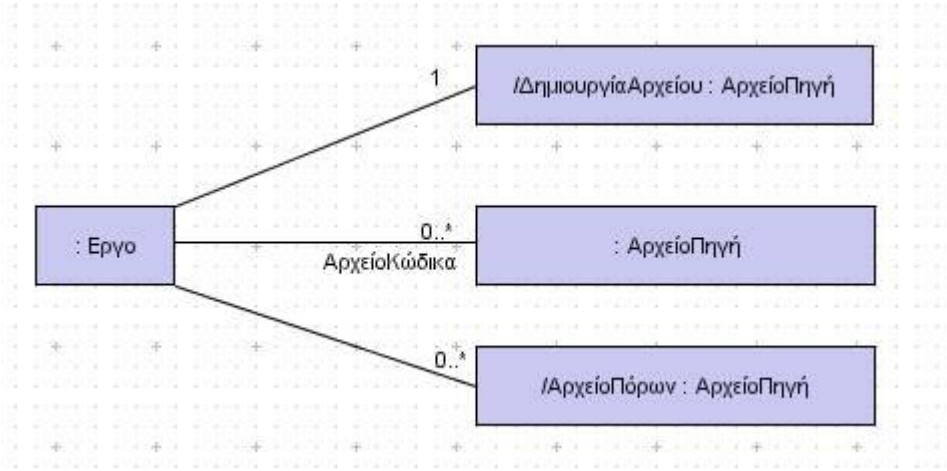
Οι συσχετίσεις μπορούν να μεταφερθούν από ένα διάγραμμα κλάσης σε ένα διάγραμμα επικοινωνίας με τη μορφή ρόλων συσχέτισης (association roles). Στο ακόλουθο σημειολογικό παράδειγμα, η ΚλάσηΑ παίζει το ρόλο ΡόλοςΒ στην ΚλάσηΒ όπως και η ΚλάσηΒ παίζει το ρόλο ΡόλοςΑ στην ΚλάσηΑ.



Σχήμα 5.4.4.1.2-1: Ρόλοι συσχέτισης

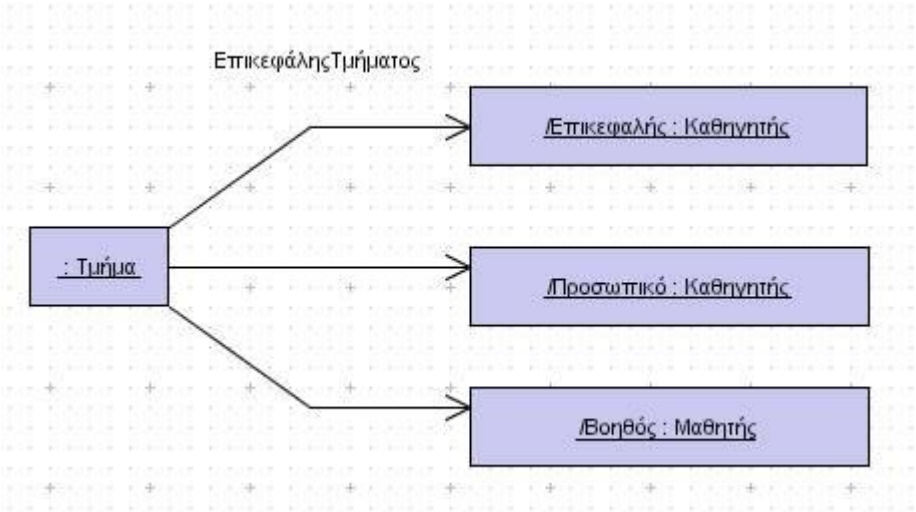
Επιπλέον, όπως δείχνεται εδώ, οι ρόλοι συσχέτισης μπορούν να υποδηλώνουν πλοήγηση. Μπορούν, μέσω ενός ανοιχτού βέλους, να υποδηλώνουν προς ποια κατεύθυνση ρέουν τα μηνύματα από ένα ρόλο κλάσης σε έναν άλλο.

Πολλαπλότητα μπορεί να προστεθεί στο ρόλο συσχέτισης για να υποδείξει πόσα αντικείμενα μιας κλάσης σχετίζονται με ένα αντικείμενο μιας άλλης κλάσης. Το επόμενο παράδειγμα υποδεικνύει ότι η κλάση Έργο έχει μία κλάση αρχείου-πηγής που παίζει το ρόλο του φτιάξε αρχείο, μηδέν έως πολλά αρχεία-πηγές που παίζουν το ρόλο κωδικοποίησης αρχείων, και μηδέν έως πολλά αρχεία-πηγές που παίζουν το ρόλο πόρων αρχείων.



Σχήμα 5.4.4.1.2-2: Πολλαπλότητα στους ρόλους συσχέτισης

Για να οπτικοποιήσουμε την πλοήγηση, το επόμενο παράδειγμα δείχνει ότι η κλάση Τμήμα μπορεί να επικοινωνήσει με την κλάση Καθηγητής παίζοντας το ρόλο ενός Επικεφαλής ή Προσωπικό ενώ μπορεί να επικοινωνήσει με ένα Μαθητή που παίζει το ρόλο Βοηθός.



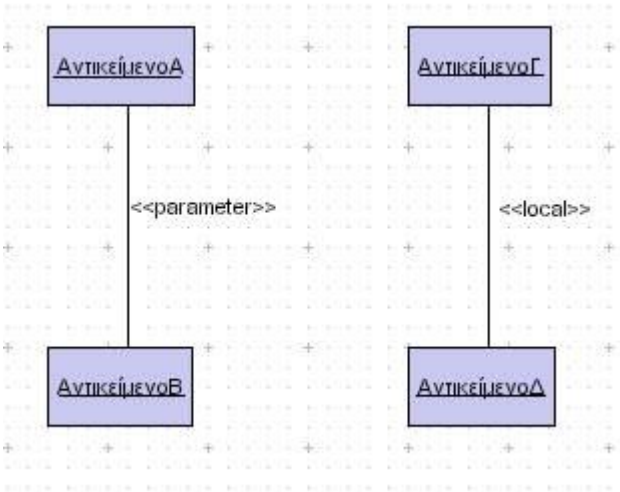
Σχήμα 5.4.4.1.2-3: Διαφορετικοί τρόποι επικοινωνίας

Το διάγραμμα δεν υποδεικνύει ένα τρόπο με τον οποίο ο ρόλος Καθηγητής ή ο ρόλος μαθητής μπορούν να επικοινωνήσουν με την κλάση Τμήμα. Η επικοινωνία πρέπει να προέρχεται από την κλάση Τμήμα.

5.4.4.1.3 Σύνδεσμοι

Οι σύνδεσμοι (links) χρησιμοποιούνται για να συσχετίσουν αντικείμενα σε ένα διάγραμμα επικοινωνίας. Μπορούν να αντιπροσωπεύουν ένα στιγμιότυπο μιας συσχέτισης από ένα διάγραμμα κλάσης. Ένας σύνδεσμος μπορεί να γίνει στερεότυπο (stereotype) με είτε «παράμετρο» («parameter») ή «τοπικό» («local»).Το στερεότυπο «παράμετρος» υποδεικνύει ότι ένα αντικείμενο είναι παράμετρος ενός άλλου, καθώς και το στερεότυπο «τοπικό» υποδεικνύει ότι ένα αντικείμενο έχει τοπικό ρόλο, ως μεταβλητή, μέσα στο άλλο αντικείμενο. Αυτό μπορεί να υποδεικνύει ότι η σχέση και

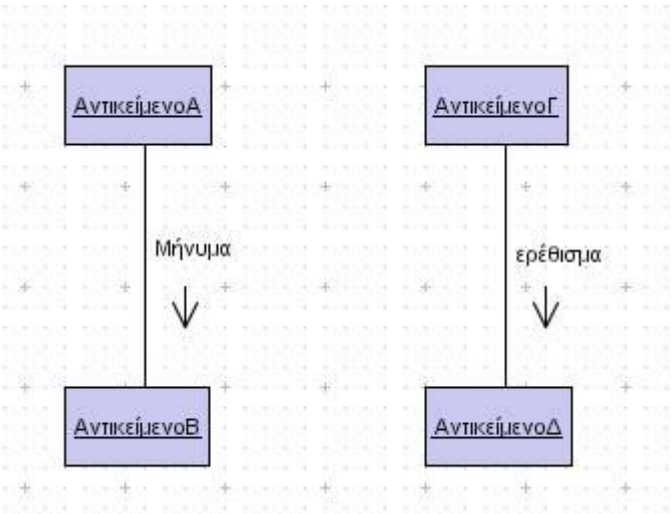
το αντικείμενο της μεταβλητής είναι μονάχα προσωρινό και ότι θα καταστραφεί μαζί με το κυρίως αντικείμενο.



Σχήμα 5.4.4.1.3-1: Σύνδεσμοι με στερεότυπα

5.4.4.2 Μηνύματα

Τα μηνύματα (messages) είναι ένα μέσο επικοινωνίας των αντικειμένων μεταξύ τους ή των ρόλων των κλάσεων μεταξύ τους μέσα στα διαγράμματα επικοινωνίας. Ένα μήνυμα απεικονίζεται, σε ένα διάγραμμα επικοινωνίας, ως κείμενο κατά μήκος ενός συνδέσμου ή ενός ρόλου συσχέτισης, με ένα βέλος που δείχνει την κατεύθυνση προς την οποία το μήνυμα 'κινείται' κατά μήκος της σχέσης.



Σχήμα 5.4.4.2-1: Μήνυμα

Ένα μήνυμα μπορεί να πάρει τη μορφή της λειτουργίας μιας κλάσης, στην οποία μπορεί να μεταφέρει παραμέτρους ως μεταβλητές ή τιμές μεταβλητών. Το παρακάτω παράδειγμα δείχνει πώς το μη-ονομαζόμενο αντικείμενο Καθηγητής δίνει το μήνυμα, ΑντιστοίχισηΒαθμού, στο μη-ονομαζόμενο αντικείμενο Μαθητής μαζί με τις παραμέτρους Κλάση, Αντιστοίχιση και Βαθμός.

5.4.4.2.1 Διαφορετικά είδη μηνυμάτων

Τα ακόλουθα είναι τρία διαφορετικά είδη μηνυμάτων σε ένα διάγραμμα επικοινωνίας, που είναι παρόμοια με τα μηνύματα του διαγράμματος ακολουθίας.

Συγχρονισμένο (synchronous) είναι το μήνυμα που χρησιμοποιείται όταν η επικοινωνία γίνεται κατά βήματα, με το να ολοκληρώνεται ένα βήμα πριν προχωρήσει στο επόμενο. Η UML δηλώνει αυτό τον τύπο μηνύματος με συνεχόμενη γραμμή και μια “γεμάτη” κεφαλή βέλους.

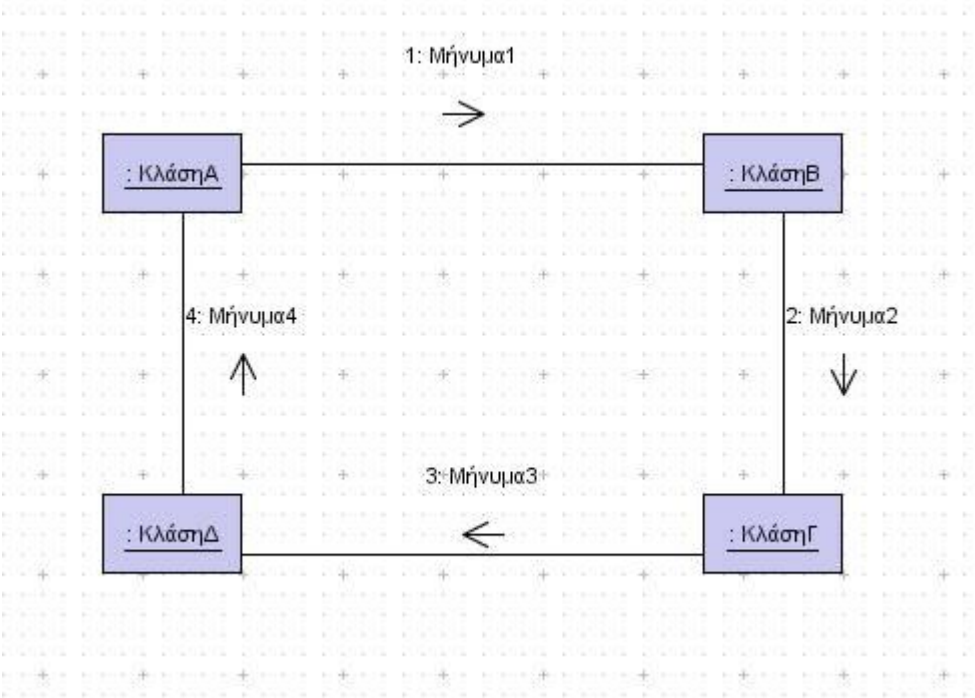
Ασύγχρονο (asynchronous) είναι το μήνυμα που χρησιμοποιείται όταν θέλουμε να επικοινωνήσουμε σε μια παράλληλη διαδικασία. Το σύμβολο γι’ αυτό το μήνυμα είναι ένα βέλος με ανοιχτή κεφαλή πάνω σε συνεχόμενη γραμμή.

5.4.4.2.2 Περισσότερα για τα μηνύματα

Υπάρχουν αρκετά άλλα πράγματα που μπορούμε να κάνουμε με τα μηνύματα. Μπορούμε να αριθμήσουμε μηνύματα για να τους δώσουμε μια σειρά με την οποία να εκτελεστούν (sequencing). Μπορούμε να συμπεριλάβουμε συνθήκες φύλαξης (guard conditions) για να απαγορέψουμε σε μηνύματα να σταλούν. Μπορούμε να δημιουργήσουμε στιγμιότυπα (instances) αντικειμένων με μηνύματα και μπορούμε να βάλουμε μηνύματα για να επαναλάβουμε μια διαδικασία (iteration).

5.4.4.2.3 Ακολουθία

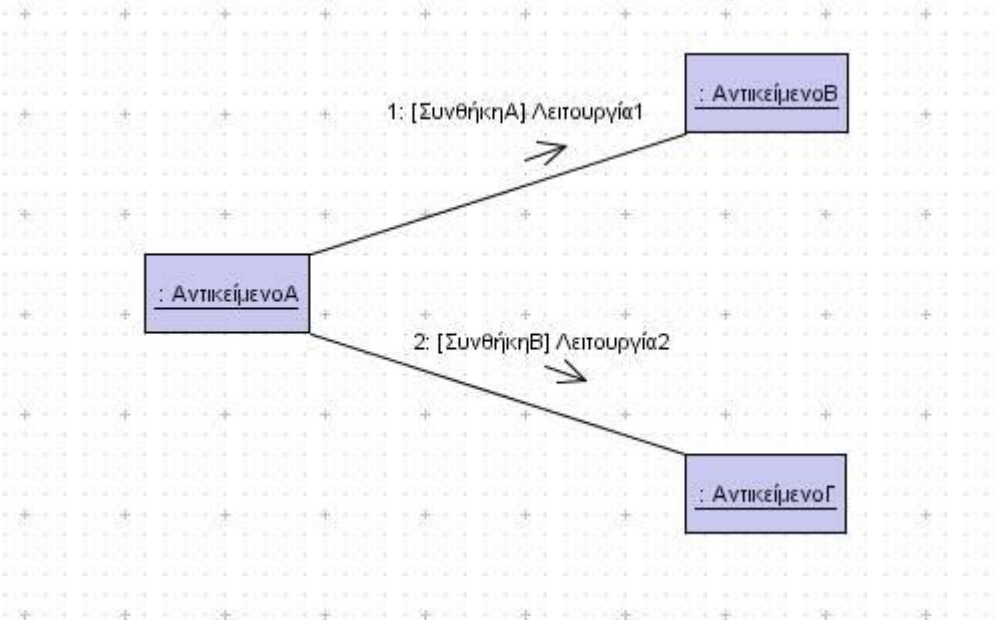
Για να βάλουμε μηνύματα σε μια ακολουθία, πρέπει απλά να τους δώσουμε ένα αναγνωριστικό ακολουθίας (sequencing ID). Ο ευκολότερος τρόπος για να το κάνουμε αυτό είναι να τα αριθμήσουμε με τη σειρά με την οποία θέλουμε να εκτελεστούν.



Σχήμα 54.4.2.3-1: Μια ακολουθία μηνυμάτων

5.4.4.2.4 Συνθήκες φύλαξης

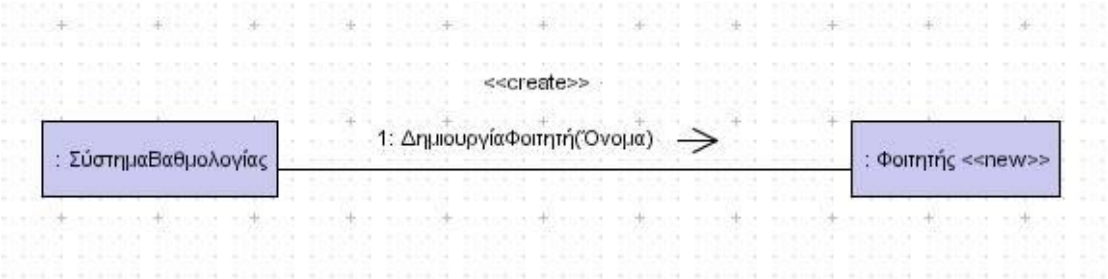
Χρησιμοποιούνται για να περιορίσουμε ένα μήνυμα να σταλεί μόνο βάσει της συνθήκης που έχει οριστεί.
Η παρακάτω σημειολογία υποδεικνύει πως από το ΑντικείμενοΑ είτε το μήνυμα 1^{α}, 1β ή κανένα απολύτως μήνυμα δε θα σταλεί σύμφωνα με τις συνθήκες. Αν η ΣυνθήκηΑ είναι αληθής (true), τότε το ΑντικείμενοΑ θα στείλει το μήνυμα Λειτουργία1 στο ΑντικείμενοΒ. Εάν η ΣυνθήκηΒ είναι αληθής, τότε το ΑντικείμενοΑ θα στείλει το μήνυμα Λειτουργία2 στο ΑντικείμενοΓ. Αν καμιά από τις συνθήκες ΣυνθήκηΑ ή ΣυνθήκηΒ δεν είναι αληθής, κανένα μήνυμα δε θα σταλεί.



Σχήμα 5.4.4.2.4-1: Συνθήκες φύλαξης μεταξύ αντικειμένων

5.4.4.2.5 Δημιουργώντας στιγμιότυπα

Τα μηνύματα μπορούν να χρησιμοποιηθούν για να δημιουργήσουμε στιγμιότυπα (instances) αντικειμένων μέσα σε ένα διάγραμμα επικοινωνίας. Για να το κάνουμε αυτό, ένα μήνυμα στέλνεται στο πρόσφατα δημιουργημένο στιγμιότυπο αντικειμένου. Το αντικείμενο φέρει το στερεότυπο (stereotype) «new» (νέο) και το μήνυμα φέρει το στερεότυπο «create» (δημιουργία) για να δείξει στον αναγνώστη ότι το αντικείμενο δημιουργείται στην πορεία.



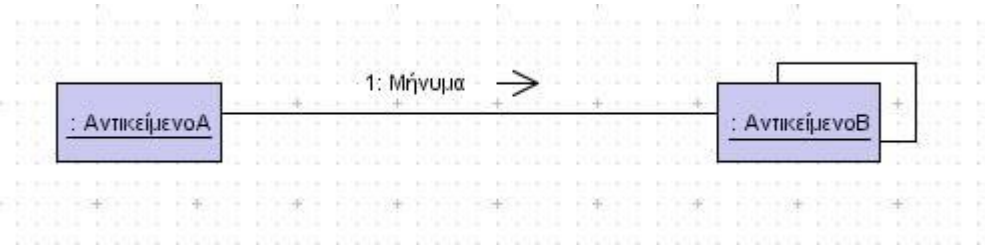
Σχήμα 5.4.4.2.5-1: Δημιουργία στιγμιοτύπου αντικειμένου

Εάν από το μήνυμα που στέλνεται είναι φανερό πως το αντικείμενο που το δέχεται δημιουργείται στην πορεία, τα στερεότυπα δεν είναι αναγκαία.

5.4.4.2.6 Επανάληψη

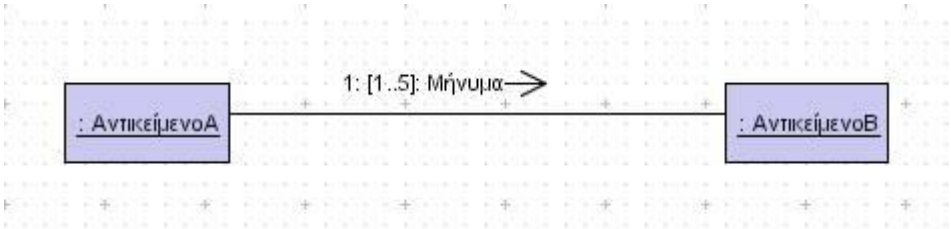
Η επανάληψη (iteration) είναι σημαντικός τύπος της ροής ελέγχου (control flow) σε οποιοδήποτε σύστημα ή συστατικό. Η επανάληψη μπορεί εύκολα να μοντελοποιηθεί στα διαγράμματα επικοινωνίας και πρέπει να χρησιμοποιείται για να υποδείξει επαναλαμβανόμενη διαδικασία.
Υπάρχουν δύο ειδών απεικονίσεις για την επανάληψη στη UML. Η πρώτη σημειολογία χρησιμοποιείται όταν ένα αντικείμενο στέλνει ένα μήνυμα σε μια ομάδα

άλλων αντικειμένων. Στο παρακάτω παράδειγμα η πολλαπλότητα μπορεί να είναι οποιουδήποτε τύπου, και υποδεικνύεται από τον αστερίσκο δίπλα στο σύνδεσμο μεταξύ του ΑντικείμενοΑ και ΑντικείμενοΒ.



Σχήμα 5.4.4.2.6-1: Η πρώτη σημειολογία απεικόνισης της επανάληψης

Το μήνυμα δηλώνει πως για κάθε ΑντικείμενοΒ, το Μήνυμα1 θα σταλεί από το ΑντικείμενοΑ στο ΑντικείμενοΒ.
Η δεύτερη, παρόμοια σημειολογία για επανάληψη στη UML δηλώνει πως το μήνυμα στέλνεται πολλές φορές από το ένα αντικείμενο στο άλλο, όπως δείχνεται παρακάτω.



Σχήμα 5.4.4.2.6-2: Η δεύτερη σημειολογία απεικόνισης της επανάληψης

Βλέπουμε ότι το Μήνυμα στέλνεται από το ΑντικείμενοΑ στο ΑντικείμενοΒ πέντε φορές.

5.4.4.3 Μοντελοποίηση διαγραμμάτων επικοινωνίας

Υπάρχουν βασικά τρία βήματα για τη μοντελοποίηση διαγραμμάτων επικοινωνίας.

1. Αναγνώριση των στοιχείων που ανήκουν στο διάγραμμά μας.
2. Μοντελοποίηση των δομικών σχέσεων μεταξύ αυτών των στοιχείων.
3. Μοντελοποίηση του διαγράμματος σε επίπεδο στιγμιοτύπων.

5.4.4.4 Περίληψη

Ένα διάγραμμα επικοινωνίας δείχνει μια αλληλεπίδραση οργανωμένη γύρω από τα αντικείμενα στην αλληλεπίδραση και τους συνδέσμους τους μεταξύ τους. Διαφορετικά από ένα διάγραμμα ακολουθίας, ένα διάγραμμα επικοινωνίας δείχνει τις σχέσεις ανάμεσα στους ρόλους των αντικειμένων. Από την άλλη, ένα διάγραμμα επικοινωνίας δεν παρουσιάζει το χρόνο ως μια ξεχωριστή διάσταση, έτσι η ακολουθία των μηνυμάτων και των ταυτόχρονων αλληλουχιών πρέπει να προσδιορίζεται χρησιμοποιώντας αριθμούς ακολουθίας.

Σημειολογία

Ένα διάγραμμα επικοινωνίας αντιπροσωπεύει μια συνεργασία, η οποία είναι ένα σύνολο αντικειμένων που σχετίζονται σε ένα συγκεκριμένο πλαίσιο και μία αλληλεπίδραση, η οποία είναι ένα σύνολο μηνυμάτων που ανταλλάσσονται ανάμεσα στα αντικείμενα μέσα σε μια συνεργασία για να καταλήξουν σε μια επιθυμητή λειτουργία ή αποτέλεσμα.

Συμβολισμός

Ένα διάγραμμα επικοινωνίας είναι ένα γράφημα από αναφορές σε αντικείμενα και συνδέσμους με ροή μηνυμάτων επισυναπτόμενων στους συνδέσμους του. Το διάγραμμα δείχνει τα αντικείμενα που σχετίζονται με την επίδοση μιας λειτουργίας, όπως και τα αντικείμενα που επηρεάζονται έμμεσα ή που είναι προσπελάσιμα κατά τη διάρκεια της λειτουργίας. Η επικοινωνίας που χρησιμοποιείται για να περιγραφεί μια λειτουργία περιλαμβάνει τις παραμέτρους της και τοπικές μεταβλητές που δημιουργήθηκαν κατά την εκτέλεσή της όπως και συνήθεις συσχετίσεις. Τα αντικείμενα που δημιουργήθηκαν στη διάρκεια της εκτέλεσης μπορεί να προσδιορίζονται ως {new}, τα αντικείμενα που καταστράφηκαν ως {destroyed}, τα αντικείμενα που δημιουργήθηκαν στη διάρκεια της εκτέλεσης και μετέπειτα καταστράφηκαν ως {transient}. Αυτές οι αλλαγές στην κατάσταση ζωής πηγάζουν από λεπτομερή μηνύματα σταλμένα ανάμεσα στα αντικείμενα. Προβάλλονται ως συμβολικές χρησιμότητες.

Το διάγραμμα δείχνει επίσης τους συνδέσμους ανάμεσα στα αντικείμενα όπως και τους μεταβατικούς συνδέσμους που αντιπροσωπεύουν διαδικασίες των παραμέτρων, τοπικές μεταβλητές και δικούς του συνδέσμους. Επειδή τα διαγράμματα ακολουθίας χρησιμοποιούνται συχνά για να βοηθήσουν στο σχεδιασμό, δείχνου τυπικά ικανότητα για «πλοήγηση» (navigation) χρησιμοποιώντας «τόξα» πάνω στους συνδέσμους (Ένα τόξο πάνω σε μια γραμμή ανάμεσα σε κουτάκια αντικειμένου υποδεικνύει ένα σύνδεσμο με δυνατότητα πλοήγησης σε μια μεριά. Ένα βέλος δίπλα σε μια γραμμή υποδεικνύει ένα μήνυμα που ρέει σε μια δεδομένη κατεύθυνση πάνω στο σύνδεσμο. Προφανώς ένα βέλος ενός μηνύματος δεν μπορεί να ρέει προς τα πίσω σε ένα σύνδεσμο μιας κατευθύνσεως).

Ξεχωριστές αξίες γνωρισμάτων δεν παρουσιάζονται συνήθως διεξοδικά. Αν τα μηνύματα πρέπει να σταλούν στις αξίες γνωρισμάτων, τα γνωρίσματα θα έπρεπε να μοντελοποιηθούν χρησιμοποιώντας, όμως, συσχετίσεις.

Τα εσωτερικά μηνύματα που θέτουν σε ισχύ μία μέθοδο είναι αριθμημένα ξεκινώντας με το νούμερο 1. Για μια διαδικαστική ροή ελέγχου τα νούμερα του επακόλουθου μηνύματος εμφωλεύονται σύμφωνα με ένθετες κλήσεις. Για μια μη-διαδικαστική ακολουθία μηνυμάτων που ανταλλάσσονται ανάμεσα σε ταυτόχρονα αντικείμενα όλα τα νούμερα ακολουθίας βρίσκονται στο ίδιο επίπεδο (άρα, δεν είναι ένθετα).
Ένα διάγραμμα επικοινωνίας χωρίς μηνύματα δείχνει το πλαίσιο στο οποίο μπορούν να λάβουν χώρα οι αλληλεπιδράσεις χωρίς να δείχνουν συγκεκριμένες αλληλεπιδράσεις. Ίσως χρησιμοποιηθεί για να δείξει το πλαίσιο μιας μοναδικής λειτουργίας ή ακόμη και όλων των λειτουργιών μιας κλάσης ή ενός γκρουπ κλάσεων.

5.4.5 ΔΙΑΓΡΑΜΜΑΤΑ ΚΑΤΑΣΤΑΣΗΣ

Τα διαγράμματα κατάστασης (statechart diagrams) έχουν πολλές ομοιότητες με τα διαγράμματα δραστηριοτήτων στην απεικόνισή τους και γι'αυτο πολλές φορές συγχέονται. Όμως, ενώ ένα διάγραμμα δραστηριοτήτων χρησιμοποιείται για να δείξει πώς διαφορετικές περιοχές ενός έργου συνεργάζονται μεταξύ τους , ένα διάγραμμα κατάστασης δείχνει ένα αντικείμενο και πως η συμπεριφορά του αλλάζει καταστάσεις.

5.4.5.1 Ορισμός των διαγραμμάτων κατάστασης

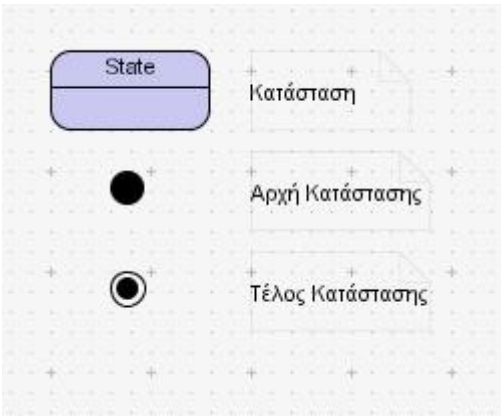
Τα διαγράμματα κατάστασης χρησιμοποιούνται για να δείξουν πώς ένα αντικείμενο αλλάζει καταστάσεις. Ως κατάσταση ορίζουμε την εικόνα της συμπεριφοράς του αντικειμένου σε μια συγκεκριμένη χρονική στιγμή. Για παράδειγμα οι καταστάσεις ενός υπολογιστή μπορούν να είναι : άνοιγμα, φόρτωμα ,επεξεργασία ,ανενεργός ,κλείσιμο ,κλειστός. Το πως ο υπολογιστής πάει από την μια κατάσταση στην άλλη είναι δουλειά ενός διαγράμματος κατάστασης να μας το δείξει.

5.4.5.1.1 Σημειολογία διαγραμμάτων κατάστασης

Ένα διάγραμμα κατάστασης αποτελείται από καταστάσεις , μεταβάσεις και γεγονότα. Με τον συνδυασμό των καταστάσεων και των μεταβάσεων δημιουργούμε τα βασικά μοντέλα ,με την προσθήκη και άλλων σημείων όπως είναι τα σημεία αποφάσεων και οι μπάρες συγχρονισμού προσθέτουμε ένα υψηλότερο επίπεδο λεπτομέρειας στο μοντέλο.

5.4.5.1.2 Καταστάσεις

Υπάρχουν τρία διαφορετικά σημεία για τις καταστάσεις (states)



Σχήμα 5.4.5.1.2-1: Τα σημεία των καταστάσεων

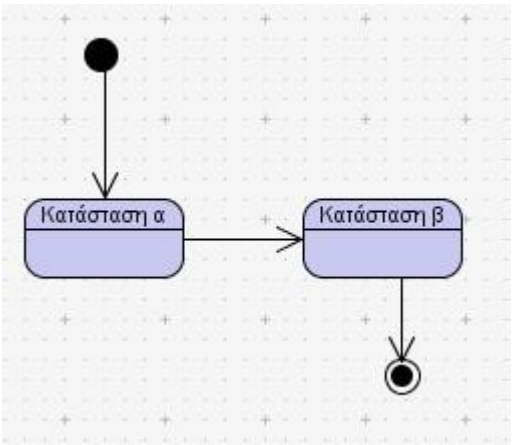
Το βασικό σημείο είναι ένα ορθογώνιο με στρογγυλευμένες γωνιές. Το όνομα της κατάστασης τοποθετείται μέσα στο ορθογώνιο και δείχνει ένα σημείο στο μοντέλο που ικανοποιεί μια συνθήκη. Τα ονόματα μερικών βασικών καταστάσεων δείχνουν τι συνθήκη συναντάμε, όπως στο παράδειγμα:

Σχήμα 5.4.5.1.2-2: Παραδείγματα καταστάσεων

Τα υπόλοιπα δυο σημεία για την κατάσταση είναι ειδικά σημεία που δείχνουν την αρχική και την τελική κατάσταση του μοντέλου. Η αρχική κατάσταση η σημείο εκκίνησης είναι μια απλή μαύρη κουκίδα. Η τελική κατάσταση σε ένα μοντέλο είναι μια μαύρη κουκίδα με ένα κύκλο. Ένα μοντέλο δεν είναι υποχρεωμένο να έχει αρχική η τελική κατάσταση μπορεί να τρέχει για πάντα , μπορεί επίσης να έχει παραπάνω από μια αρχικές καταστάσεις αλλά και παραπάνω από μια τελικές καταστάσεις.

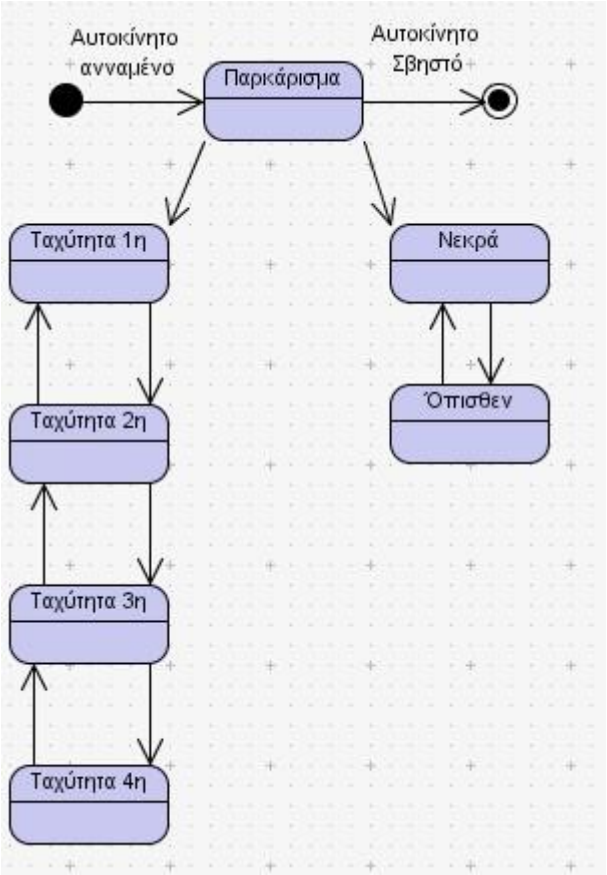
5.4.5.1.3 Μεταβάσεις

Οι μεταβάσεις (transitions) χρησιμοποιούνται για να δείξουν τη ροή από μια κατάσταση σε μια άλλη. Μια μετάβαση σχεδιάζεται με ένα ανοιχτό βέλος που οδηγεί από μια κατάσταση σε μια άλλη. Στο ακόλουθο σχέδιο, υπάρχει μια διαδρομή από την αρχική κατάσταση στην τελική κατάσταση, που συμπεριλαμβάνει μια στάση στην Κατάσταση Α και μια στάση στην Κατάσταση Β.



Σχήμα 5.4.5.1.3-1: Μεταβάσεις

Το επόμενο παράδειγμα μεταβάσεων απεικονίζει μια τυπική μετάδοση ταχυτήτων σ'ένα αυτοκίνητο και τις έγκυρες καταστάσεις της.

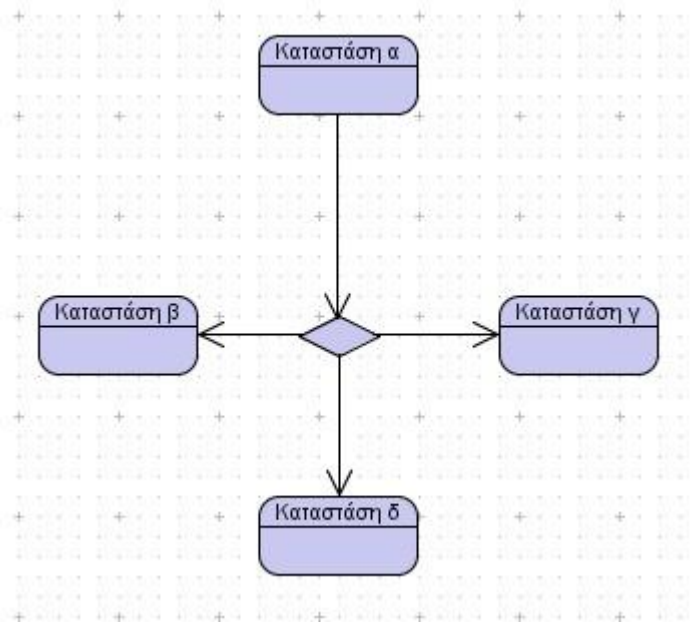


Σχήμα 5.4.5.1.3-2: Παράδειγμα μεταβάσεων μοντέλου ταχυτήτων

Προσέξτε οτι αυτό το παράδειγμα δεν έχει μια αρχική και μια τελική κατάσταση που να σχετίζεται με το εάν το αυτοκίνητο θα βρίσκεται σε λειτουργία ή όχι. Είναι πιθανό να ξεκινήσει το αυτοκίνητο (βάζοντάς το στην κατάσταση "παρκάρισμα") και να σβήσει το αυτοκίνητο χωρίς να έχει μπεί σε καμία από τις άλλες πιθανές καταστάσεις. Δεν είναι όμως πιθανό να εισέλθει το αυτοκίνητο στην κατάσταση "Ταχύτητα 4" χωρίς να έχει εισέλθει πρώτα στην κατάσταση "Ταχύτητα 3".

5.4.5.1.4 Σημεία αποφάσεων

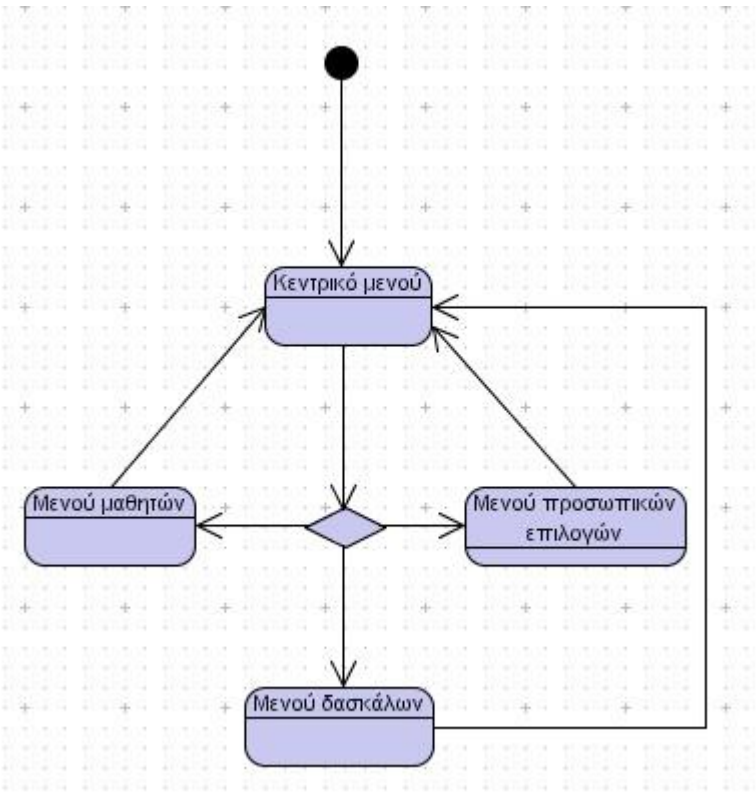
Τα σημεία αποφάσεων (decision points) διευκολύνουν όταν μοντελοποιούμε διαγράμματα καταστάσεων, επειδή κάνουν το διάγραμμα πιο ευχάριστο οπτικά με το να ομαδοποιούν τις μεταβάσεις σε ένα κεντρικό σημείο. Όπως φαίνεται στο επόμενο σχέδιο, η Κατάσταση Α μπορεί να μεταβεί σε οποιαδήποτε από τις υπόλοιπες καταστάσεις, Κατάσταση Β, Γ, Δ. Η σημειολογία ενός σημείου αποφάσεων είναι ένα κενό "διαμάντι" με ένα ή περισσότερα σημεία εισόδου και δύο ή περισσότερα σημεία εξόδου.



Σχήμα 5.4.5.1.4-1: Σημειολογία σημείου αποφάσεων

Το προηγούμενο σχέδιο μπορεί να μοντελοποιηθεί με το να βάλουμε τις τρείς μεταβάσεις σε καθεμία από τις καταστάσεις προορισμού, να ξεκινήσουν απ' την Κατάσταση Α. Το σημείο αποφάσεων χρησιμοποιείται για να αποφευχθεί σύγχιση στο διάγραμμα και συνήθως χρησιμοποιείται σε περιπτώσεις όπου πολλές μεταβάσεις ξεκινούν από μια κατάσταση.

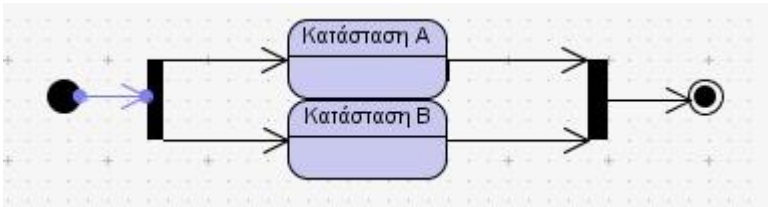
Το επόμενο παράδειγμα μας δείχνει πως μπορούμε να χρησιμοποιήσουμε την κατάσταση αποφάσεων για να πλοηγηθούμε μέσα στο μενού ενός συστήματος που χρησιμοποιεί το σημείο αποφάσεων. Στο σχέδιο, το Κεντρικό Μενού μπορεί να χρησιμοποιηθεί για την πλοήγηση σε οποιοδήποτε από τα άλλα 3 μενού και το καθένα από αυτά τα μενού μπορούν να χρησιμοποιηθούν για να επιστρέψουμε στο Κεντρικό Μενού.



Σχήμα 5.4.5.1.4-2: Παράδειγμα χρήσης του σημείου αποφάσεων

5.4.5.1.5 Συγχρονισμός

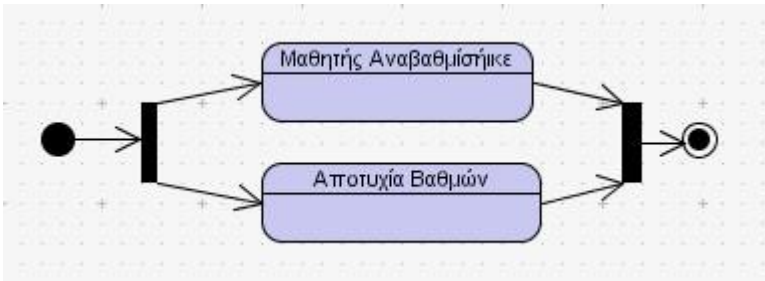
Οι μπάρες συγχρονισμού (synchronization) χρησιμοποιούνται στα διαγράμματα καταστάσεων για να δείξουν πού οι καταστάσεις χρειάζεται να ενωθούν ή να περιμένουν για άλλες καταστάσεις. Οι μπάρες συγχρονισμού χρησιμοποιούνται για να δείξουν συντρέχουσες καταστάσεις. Το επόμενο σχέδιο δείχνει τη σημειολογία για τις μπάρες συγχρονισμού.



Σχήμα 5.4.5.1.5-1: Σημειολογία για τις μπάρες συγχρονισμού

Μία παχιά μπάρα με μία μετάβαση να εισέρχεται σ' αυτή δείχνει την αρχή μιας περιόδου συγχρονισμού. Δύο ή περισσότερες μεταβάσεις μπορούν να εξέρχονται από την άλλη πλευρά της πρώτης μπάρας συγχρονισμού. Όλες οι μπάρες συγχρονισμού πρέπει να πηγαίνουν σε ζευγάρια, έτσι ώστε να υπάρχει ένας τρόπος να συνδυάζονται πάλι σε μία μετάβαση. Αυτό φαίνεται στην αντίθετη μορφή όπου δύο ή περισσότερες μεταβάσεις εισέρχονται σε μια μαύρη μπάρα και μόνο μία μετάβαση εξέρχεται.
Στο επόμενο παράδειγμα, το διάγραμμα καταστάσεων δείχνει μια βαθμολογία έχει αναβαθμιστεί, αλλά επίσης δείχνει ότι μια βαθμολογία έχει αποτύχει. Αυτή η

κατάσταση μπορεί να υπάρξει επειδή προειδοποιήσεις ή άλλες ενέργειες μπορούν να συμβούν κάτω απ' τις συγκεκριμένες συνθήκες.



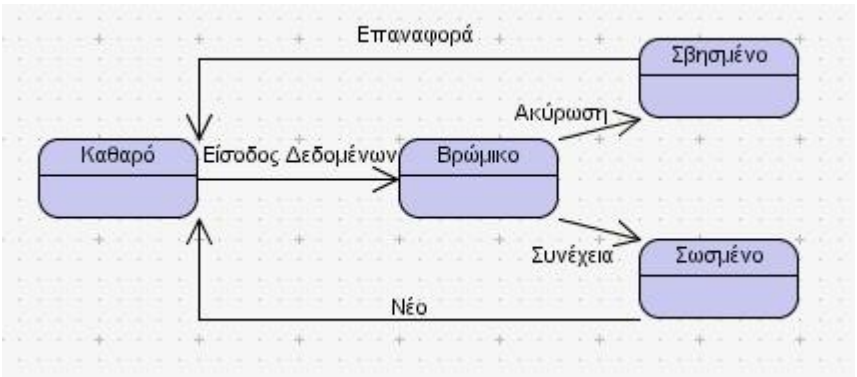
Σχήμα 5.4.5.1.5-2: Παράδειγμα χρήσης συγχρονισμού

5.4.5.2 Πως ορίζονται ενέργειες και γεγονότα για καταστάσεις

Οι ενέργειες και τα γεγονότα χρησιμοποιούνται για να περιγράψουν το πώς φτάνουμε σε μια κατάσταση, τι συμβαίνει όταν φτάνουμε εκεί και τι συμβαίνει όταν βγαίνουμε απ'την κατάσταση. Τα γεγονότα χρησιμοποιούνται για να δείξουν τι ενεργοποιεί μια μετάβαση και οι ενέργειες χρησιμοποιούνται για να δείξουν τι συμβαίνει όταν ένα γεγονός έχει ήδη συμβεί.

5.4.5.2.1 Γεγονότα

Τα γεγονότα (events) συνήθως τοποθετούνται κατευθείαν στη διαδρομή μιας μετάβασης από μια κατάσταση σε μια άλλη κατάσταση και χρησιμοποιούνται για να δείξουν τι προκάλεσε την αλλαγή της κατάστασης σε ένα μοντέλο. Το παρακάτω παράδειγμα απεικονίζει αυτή τη σημειολογία.



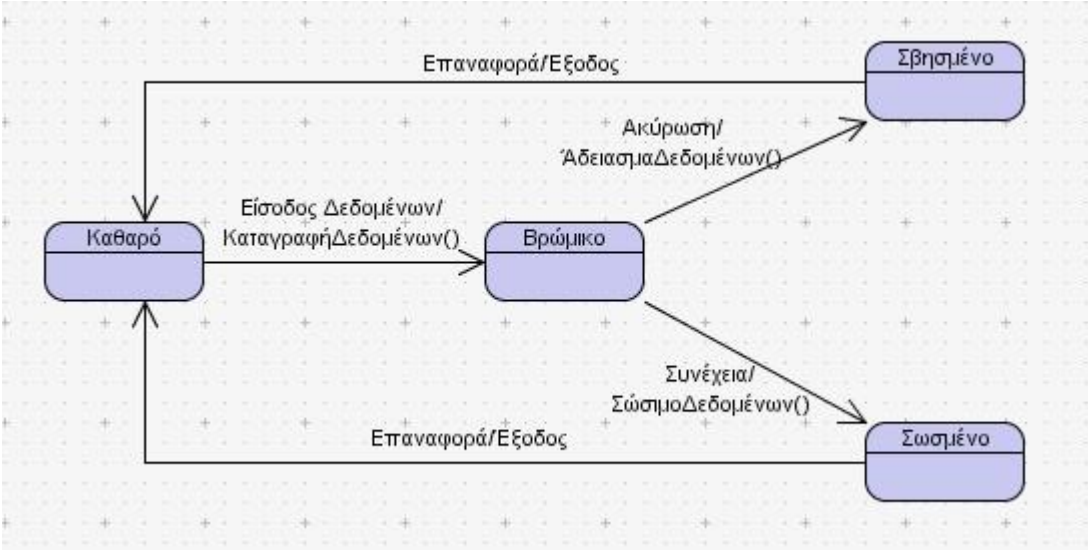
Σχήμα 5.4.5.2.1-1: Παράδειγμα χρήσης γεγονότων

Σε αυτό το παράδειγμα, το αντικείμενό μας έχει 4 καταστάσεις: Καθαρό, Βρώμικο, Σβησμένο και Σωσμένο (αποθηκευμένο). Εάν το αντικείμενο είναι Καθαρό, μόνο η Είσοδος δεδομένων θα μπορεί να αλλάξει την κατάσταση και να πάει κατευθείαν στην κατάσταση Βρώμικο. Από εδώ, είτε το γεγονός Ακύρωση ή το γεγονός Συνέχεια μπορεί να αλλάξει πάλι την κατάσταση είτε σε Σβησμένο ή σε Σωσμένο, αντίστοιχα. Τα γεγονότα μπορεί τελικά να γραφτούν σαν λειτουργίες μιας κλάσης, αλλά δεν είναι αναγκαστικό να έχουνε μια ένα-προς-ένα σχέση με τις λειτουργίες.

5.4.5.2.2 Ενέργειες

Μια ενέργεια (actions) δείχνει τι προκύπτει όταν γίνεται ένα γεγονός. Υπάρχουν 5 βασικοί τύποι ενεργειών που μπορεί να ξεκινήσει μια κατάσταση.

- Είσοδος (entry): Χρησιμοποιείται για να δείξει την ενέργεια που γίνεται όταν εισερχόμαστε σε μια κατάσταση.
- Έξοδος (exit): Χρησιμοποιείται για να δείξει την ενέργεια που γίνεται όταν εγκαταλείπουμε μια κατάσταση για μια άλλη.
- Κάνε (do): Χρησιμοποιείται για να δείξει την ενέργεια που γίνεται όταν η κατάσταση είναι κατειλημμένη.
- Συμπεριέλαβε (include): Επικαλείται ένα υποσύστημα, που αντιπροσωπεύεται από ένα άλλο διάγραμμα κατάστασης.
- Γεγονός (event): Χρησιμοποιείται για να δείξει την ενέργεια που γίνεται όταν ένα συγκεκριμένο γεγονός ενεργοποιείται.



Σχήμα 5.4.5.2.2-1: Παράδειγμα ενεργειών

Βλέπουμε ότι όταν μετακινηθούμε από την Καθαρή στη Βρώμικη κατάσταση, το γεγονός Είσοδος δεδομένων ενεργοποιεί την ενέργεια ΚαταγραφήΔεδομένων, εάν ενεργοποιηθεί το γεγονός Ακύρωση ενεργοποιείται επίσης και η ενέργεια ΚαθαρισμόΔεδομένων. Από την άλλη πλευρά, εάν ενεργοποιηθεί το γεγονός Συνέχεια η κατάσταση ΣώσιμοΔεδομένων ενεργοποιείται επίσης. Κατά την έξοδο από τις καταστάσεις Σβήσιμο και Σώσιμο, η ενέργεια Επαναφορά ενεργοποιείται και επανερχόμαστε στην κατάσταση Καθαρό.

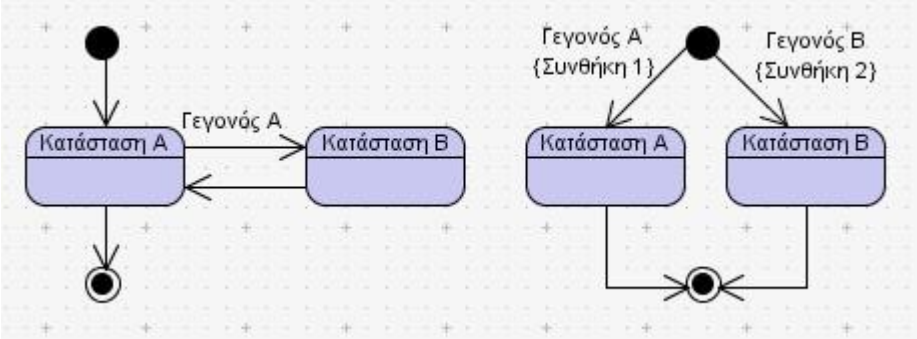
5.4.5.3 Κατανοώντας τη χρήση σύνθετων καταστάσεων

Ένα αντικείμενο μπορεί να έχει παραπάνω από μία έγκυρες καταστάσεις ανά στιγμή, μέσω συγχρονισμού. Μερικές φορές, τα αντικείμενα μπορεί να έχουν ιεραρχικές καταστάσεις. Οι ιεραρχικές καταστάσεις παρουσιάζονται σε ένα διάγραμμα όταν μία

κατάσταση μπορεί να έχει επιπλέον υποκαταστάσεις που συναντώνται μονάχα σε αυτή την κατάσταση.

5.4.5.3.1 Υποκαταστάσεις

Υποκαταστάσεις (substates) ή καταστάσεις που συναντώνται σε μία συγκεκριμένη κατάσταση ενός διαγράμματος, μπορούν να μοντελοποιηθούν χρησιμοποιώντας πολλαπλά διαγράμματα καταστάσεων, όπως φαίνεται και στο παρακάτω σχήμα.

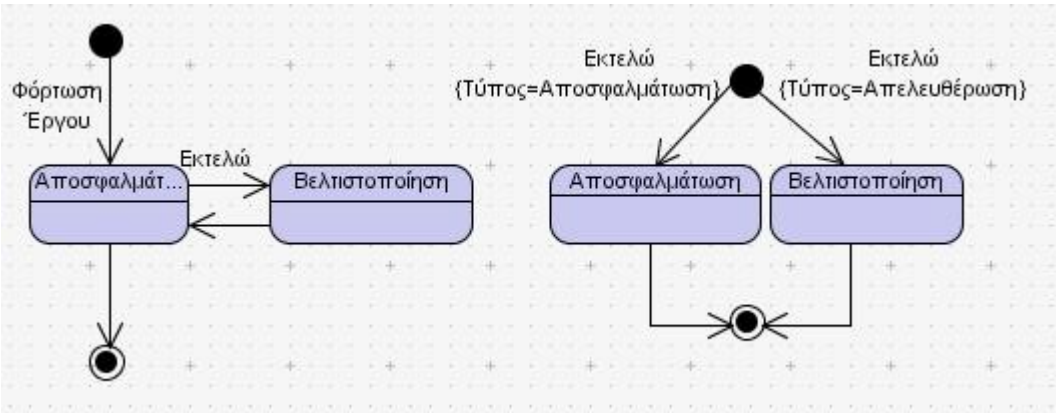


Σχήμα 5.4.5.3.1-1: Δυο τρόποι απεικόνισης υποκαταστάσεων

Στην απεικόνιση αυτή το αντικείμενο μπορεί να είναι σε οποιαδήποτε από τα ακόλουθα σετ καταστάσεων:

- Κατάσταση Α
- Κατάσταση Β και Κατάσταση 1
- Κατάσταση Β και Κατάσταση 2

Στο αριστερό διάγραμμα του προηγούμενου σχεδίου, το Γεγονός Α επιφέρει την αλλαγή της κατάστασης του αντικειμένου από Α σε Β. Στο δεξί διάγραμμα το Γεγονός Α επιφέρει την μετακίνηση του αντικειμένου από την αρχική κατάσταση είτε στην Κατάσταση Α είτε στην Κατάσταση Β εξαρτώμενο από την Συνθήκη 1 ή τη Συνθήκη 2. Το παρακάτω παράδειγμα χρησιμοποιεί την παρακάτω σημειολογία για να το δείξει:

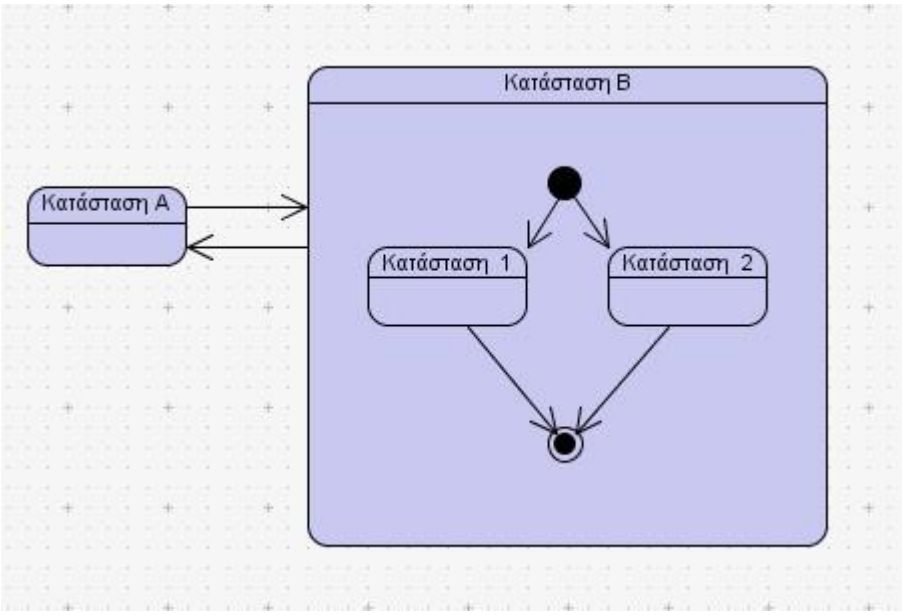


Σχήμα 5.4.5.3.1-2: Παράδειγμα υποκαταστάσεων

Όταν ένα έργο (λογισμικό) φορτώνεται, το αντικείμενο εισέρχεται στην κατάσταση Επεξεργασία. Από εκεί το έργο μπορεί να ξεφορτωθεί (unloaded) έτσι ώστε να εξέλθει από το διάγραμμα κατάστασης, αλλά μπορεί επίσης να λειτουργήσει (το έργο) έτσι ώστε να μπεί στην κατάσταση Λειτουργεί. Η κατάσταση Λειτουργεί περιγράφεται περαιτέρω στο δεύτερο διάγραμμα κατάστασης και το γεγονός Λειτουργία φυλάσσεται από εντολές γραμμένες με τη χρήση της OCL (Object Constraint Language), που καθορίζουν ποια μετάβαση θα επιλεχθεί. Εάν ο τύπος του αντικειμένου είναι Αποσφαλμάτωση, τότε επιλέγουμε την κατάσταση Αποσφαλμάτωση, έτσι κάνουμε το αντικείμενο να βρίσκεται στις καταστάσεις Λειτουργία και Αποσφαλμάτωση ταυτόχρονα. Παρομοίως εάν ο τύπος ήταν ίσος με Κυκλοφορεί, τότε η κατάσταση Βελτιστοποίηση θα ήταν ενεργή μαζί με την κατάσταση Λειτουργία.

5.4.5.3.2 Σύνθετες καταστάσεις

Ένας άλλος τρόπος για να δείξουμε τις υποκαταστάσεις είναι να χρησιμοποιήσουμε τη σημειολογία για τις σύνθετες καταστάσεις (composite states). Οι σύνθετες καταστάσεις μοντελοποιούνται με τη δημιουργία μιας μεγάλης κατάστασης που εμπεριέχει την υποκατάσταση του διαγράμματος κατάστασης. Αυτό φαίνεται στο παρακάτω σχέδιο:

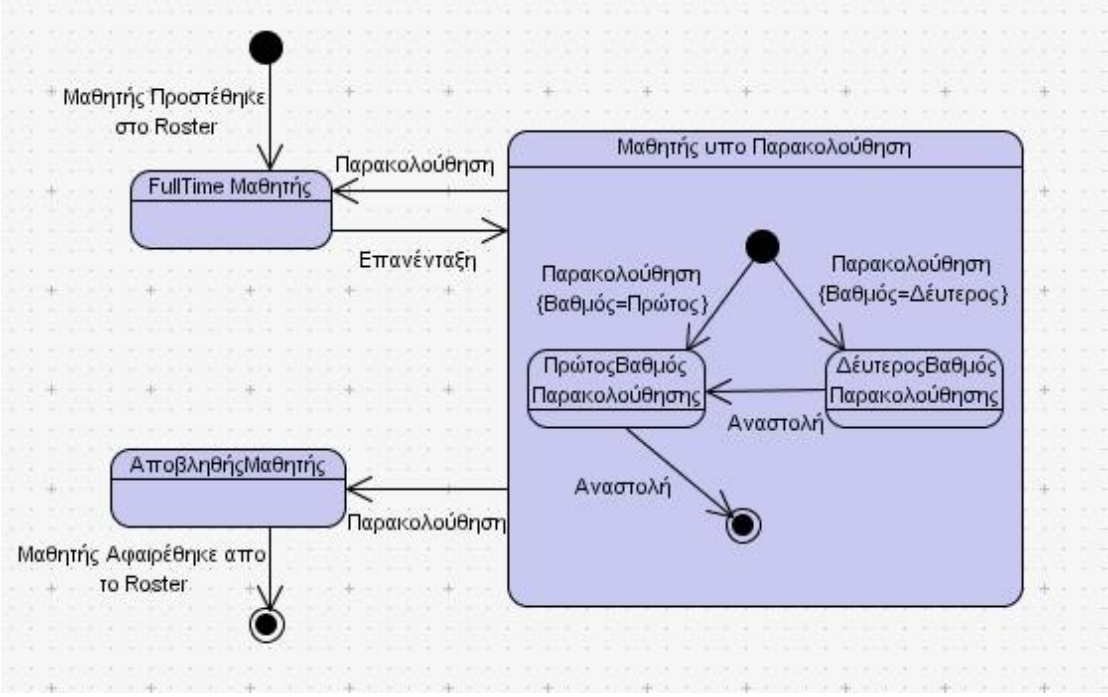


Σχήμα 5.4.5.3.2-1: Σημειολογία υποκαταστάσεων

Το σχέδιο δείχνει ότι για την κλάση που μοντελοποιούμε οι παρακάτω καταστάσεις είναι έγκυρες.

- Κατάσταση A
- Κατάσταση B και Κατάσταση 1
- Κατάσταση B και Κατάσταση 2

Το επόμενο παράδειγμα δείχνει πώς σύνθετες καταστάσεις μπορούν να χρησιμοποιηθούν για τη μοντελοποίηση υποκαταστάσεων και ιεραρχικών καταστάσεων.



Σχήμα 5.4.5.3.2-2: Παράδειγμα υποκαταστάσεων

Οι ακόλουθες είναι οι έγκυρες καταστάσεις (εξαιρουμένων της αρχικής και τελικής κατάστασης) για τις ακόλουθες τοποθεσίες:

- Full Time Μαθητής
- Μαθητής υπό Παρακολούθηση και Πρώτος Βαθμός Παρακολούθησης
- Μαθητής υπό Παρακολούθηση και Δεύτερος Βαθμός Παρακολούθησης
- Αποβληθής μαθητής

Το διάγραμμα δείχνει ότι όταν ένας μαθητής προστεθεί στην ομάδα, η κατάσταση Full Time Μαθητής είναι έγκυρη. Από εδώ ο μόνος δρόμος είναι προς τα κάτω. Το γεγονός Παρακολούθηση θα φέρει το μαθητή στην κατάσταση Μαθητής υπό Παρακολούθηση, ανάλογα με τη σοβαρότητα της πράξης του. Από εκεί μπορεί να συμβεί το γεγονός Αναστολή και να πάει πίσω στην κατάσταση Full Time Μαθητής, αλλά ένα ακόμα γεγονός Παρακολούθηση θα τον πάει στην κατάσταση Αποβληθείς μαθητής. Πρέπει να σημειωθεί ότι ένας Full Time Μαθητής δεν μπορεί να αποβληθεί χωρίς πρώτα να γίνει Μαθητής υπό Παρακολούθηση και ότι ένας Μαθητής υπό Παρακολούθηση θα αποβληθεί μετά από Παρακολούθηση, ανεξάρτητα από την αυστηρότητα της υποκατάστασης (Πρώτος ή Δεύτερος Βαθμός Παρακολούθησης).

5.4.5.4 Μοντελοποίηση διαγραμμάτων κατάστασης

Τα ακόλουθα βήματα ακολουθούνται για τη μοντελοποίηση διαγραμμάτων καταστάσεων.

1. Αναγνώριση των οντοτήτων που χρειάζονται περαιτέρω επεξήγηση
2. Αναγνώριση των αρχικών και τελικών καταστάσεων για κάθε οντότητα
3. Καθορισμός των γεγονότων που σχετίζονται με κάθε οντότητα
4. Δημιουργία του διαγράμματος κατάστασης, ξεκινώντας με ένα αρχικό γεγονός
5. Δημιουργία σύνθετων καταστάσεων όπου χρειάζονται

Αυτές οι οδηγίες αναφέρονται σε πολλαπλές οντότητες, αν και τα διαγράμματα καταστάσεων αντιπροσωπεύουν μία μοναδική οντότητα ανά διάγραμμα.

5.4.5.5 Περίληψη

Ένα διάγραμμα καταστάσεων δείχνει τις ακολουθίες των καταστάσεων που ένα αντικείμενο ή μια αλληλεπίδραση περνάει στη διάρκεια της ζωής του σε απάντηση των ληφθέντων ερεθισμάτων, μαζί με τις αποκρίσεις του και τις ενέργειες.
Η σημειολογία και ο συμβολισμός που περιγράφονται σε αυτό το κεφάλαιο είναι στην ουσία αυτοί των διαγραμμάτων καταστάσεων του David Harel με μετατροπές ώστε να γίνουν αντικειμενοστραφή. Ο συμβολισμός της κατάστασης θέτει σε εφαρμογή και τις δύο πτυχές των μηχανών του Moore και του Mealy, παραδοσιακά μοντέλα μηχανών καταστάσεων.

Σημειολογία

Μια μηχανή κατάστασης είναι ένα γράφημα καταστάσεως και μεταβάσεων που περιγράφει την απόκριση ενός αντικειμένου μιας δεδομένης κλάσης στη λήψη των εξωτερικών ερεθισμάτων. Μια μηχανή κατάστασης επισυνάπτεται σε μια κλάση ή μια μέθοδο.

Συμβολισμός

Ένα διάγραμμα καταστάσεων αντιπροσωπεύει μια μηχανή κατάστασης. Οι καταστάσεις αντιπροσωπεύονται από σύμβολα καταστάσεων και οι μεταβάσεις αντιπροσωπεύονται από βέλη που συνδέουν τα σύμβολα καταστάσεων. Οι καταστάσεις μπορούν ακόμη να περιέχουν υπο-διαγράμματα από φυσικό περιορισμό και παραθέσεις.

5.4.6 ΔΙΑΓΡΑΜΜΑΤΑ ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ

Αυτή η ενότητα σας εισάγει στα διαγράμματα δραστηριοτήτων (activity diagrams), που χρησιμοποιούνται για τη μοντελοποίηση της ροής ανάμεσα σε διαφορετικά στοιχεία ενός αντικειμενοστρεφούς συστήματος. Με τα διαγράμματα δραστηριοτήτων, θα είστε ικανοί να απεικονίσετε πού υπάρχει λειτουργικότητα στο σύστημά σας και πώς η λειτουργικότητα, συντονισμένη με τη λειτουργικότητα των άλλων μερών του συστήματός σας, θα χρησιμοποιηθεί για να ικανοποιήσει επιχειρηματικά αιτήματα που μοντελοποιήσατε νωρίτερα χρησιμοποιώντας διαγράμματα σενάριων.

5.4.6.1 Εισαγωγή στα διαγράμματα δραστηριοτήτων

Τα διαγράμματα δραστηριοτήτων παρέχουν περισσότερο χρήσιμες περιγραφές ενός συστήματος παρέχοντας σας το επόμενο βήμα στην ανάλυση του συστήματος, ακολουθώντας τα διαγράμματα σενάριων. Ένα διάγραμμα δραστηριοτήτων επιτρέπει στον αναγνώστη να δει την εκτέλεση του συστήματος και πώς αλλάζει κατεύθυνση βασισμένο σε διαφορετικές συνθήκες και ερεθίσματα. Κατ' αυτόν τον τρόπο, τα διαγράμματα δραστηριοτήτων χρησιμοποιούνται για τη μοντελοποίηση ροής εργασιών των σεναρίων. Αν και τα διαγράμματα δραστηριότητων μπορούν να χρησιμοποιηθούν και για τη μοντελοποίηση σύνθετης συμπεριφοράς αντικειμένου όταν φτάσετε στο σχεδιαστικό τμήμα της μοντελοποίησής σας, αυτή η ενότητα αφορά κυρίως τη φάση ανάλυσης χρησιμοποιώντας τα διαγράμματα δραστηριοτήτων ως μέσα για να προχωρήσετε τα σενάρια στο επόμενο επίπεδο.

5.4.6.1.1 Γιατί μοντελοποιούμε διαγράμματα δραστηριοτήτων

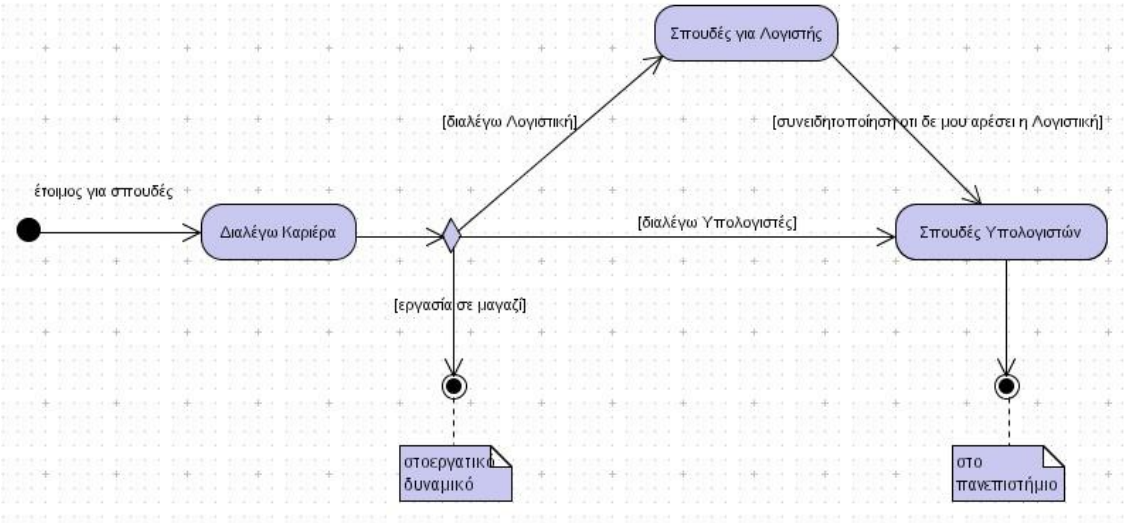
Τα διαγράμματα δραστηριοτήτων είναι χρήσιμα, ειδικότερα στα διαγραμματα σεναριων, επειδή δίνουν στον αναγνώστη μία κατάσταση αρχής και τέλους. Όταν χρησιμοποιούνται για τη μοντελοποίηση των ροών εργασίας των σενάριων, τα διαγράμματα δραστηριοτήτων μπορούν να φανερώσουν τις οδούς μέσα στα διαγραμματα σενάριων όπως και ανάμεσα στα σενάρια.

Τα διαγράμματα δραστηριοτήτων μπορούν να εξηγήσουν στον αναγνώστη ποιες συνθήκες πρέπει να πληρούνται για να είναι έγκυρο ένα σενάριο, όπως και ποιες συνθήκες, ή καταστάσεις, παραμένουν σε ένα σύστημα αμέσως μόλις το σενάριο έχει ολοκληρωθεί.

Επειδή γνωρίζουμε ότι η Ενιαία Διεργασία είναι μια διαδοχικά επαναλαμβανόμενη διαδικασία, όταν μοντελοποιούμε διαγράμματα δραστηριοτήτων, συχνά ανακαλύπτουμε επιπρόσθετα σενάρια που δεν είχαμε σκεφτεί νωρίτερα. Σε ορισμένες περιπτώσεις, ανακαλύπτουμε μια κοινή λειτουργικότητα που μπορεί να αποτελέσει ένα ξεχωριστό σενάριο. Έτσι εξοικονομούμε χρόνο, όταν αναπτύσσουμε την εφαρμογή.

5.4.6.2 Τα σημειολογικά στοιχεία των διαγραμμάτων δραστηριοτήτων

Ένα διάγραμμα δραστηριοτήτων θυμίζει σε πολλούς το παραδοσιακό διάγραμμα ροής εκτός του ότι η σημειολογία είναι ελαφρώς διαφορετική.



Σχήμα 5.4.6.2-1: Παράδειγμα διαγράμματος δραστηριότητας

Υπάρχουν τρία πρωταρχικά σημειολογικά στοιχεία στα διαγράμματα δραστηριοτήτων: δραστηριότητες, καταστάσεις και μεταβάσεις.

Μία δραστηριότητα ονομάζεται επίσης κατάσταση δράσης. Είναι μια ένδειξη μέσα στο διάγραμμα ότι κάτι πρέπει να γίνει. Στο προηγούμενο παράδειγμα, Διαλέγω Καριέρα είναι η πρώτη δραστηριότητα. Οι καταστάσεις είναι μια ένδειξη των εσωτερικών τιμών. Μία κατάσταση μπορεί να υποδείξει αν ένα πεδίο είναι ακάθαρτο ή μπορεί να επιδείξει επιτυχία ή αποτυχία. Όταν μια κατάσταση μοντελοποιείται σε ένα διάγραμμα δραστηριοτήτων, χρησιμοποιείται επειδή είναι σημαντική. Ίσως να χρησιμοποιείται για να υποδείξει ότι το σύστημα συμπεριφέρεται διαφορετικά ή ότι πληρεί όλα τα κριτήρια. Για να συνδυάσετε δραστηριότητες και καταστάσεις, οι μεταβάσεις δείχνουν τις οδούς μετανάστευσης γύρω από το διάγραμμα δραστηριοτήτων.

5.4.6.2.1 Δραστηριότητες: καταστάσεις δράσης

Η απεικόνιση για μια δραστηριότητα είναι ένα ορθογώνιο με πολύ στρογγυλοποιημένες γωνίες («πολύ στρογγυλοποιημένες» σε σχέση με τη σημειολογία των καταστάσεων που τοποθετούνται δίπλα, οι οποίες είναι πολύ παρόμοιες αλλά με λιγότερο στρογγυλοποιημένες γωνίες). Μπορείτε, μάλιστα, να σκεφτείτε τη δραστηριότητα σαν ένα κύκλο κομμένη στη μέση και επιμηκυμένο στο κέντρο, σαν μια τεντωμένη λιμουζίνα.

Σχήμα 5.4.6.2-2: Μια δραστηριότητα

Οι δραστηριότητες υποδεικνύουν δράση και για αυτό πρέπει να ονομάζονται αναλόγως. Όταν αποφασίζετε τα ονόματα των δραστηριοτήτων σας, διαλέξτε μερικές λέξεις που με ακρίβεια περιγράφουν τη δράστη που λαμβάνει χώρα. Για παράδειγμα, Αποθηκεύω Φάκελο ή Δημιουργώ Νέο Αρχείο θα ήταν περιγραφικές ονομασίες δραστηριοτήτων, σε σύγκριση με Τρέχω ή Ενημερώνω, που μοιάζουν στον αναγνώστη ημιτελείς.

5.4.6.2.2 Καταστάσεις

Μία κατάσταση (state) σχεδιάζεται παρόμοια με μια δραστηριότητα με τη διαφορά οτι, όπως αναφέρθηκε και προηγουμένως, οι γωνίες μιας κατάστασης είναι λιγότερο στρογγυλοποιημένες.

Σχήμα 5.4.6.2.2-1: Μια κατάσταση

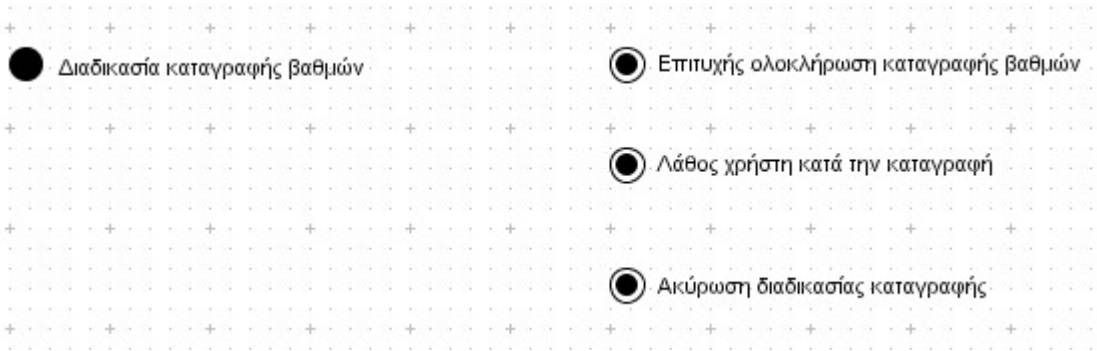
Οι καταστάσεις αναγνωρίζονται συνήθως με μία λέξη ή φράση που υποδεικνύει την τωρινή ύπαρξη ενός συστήματος. Για παράδειγμα, Σταματημένος θα ήταν μια κατάσταση, ενώ Σταματώ θα ήταν μια δραστηριότητα.

Οι καταστάσεις μπορούν να διαβιβάζουν ορόσημα σε ένα διάγραμμα δραστηριοτήτων για τον αναγνώστη ή μπορούν να χρησιμοποιηθούν για να σημειώσουν άλλες συνθήκες αργότερα στη ροή εργασιών. Για παράδειγμα, οι καταστάσεις των περιπτώσεων χρήσεις, Το Αυτοκίνητο μπορεί να Σταθμευθεί, Ρελαντί, Πρώτη Ταχύτητα, Δεύτερη Ταχύτητα, Τρίτη Ταχύτητα, Τέταρτη Ταχύτητα και Όπισθεν.

Η UML περιγράφει δύο ειδικές καταστάσεις, τις καταστάσεις αρχής και τέλους. Μία κατάσταση αρχής υποδεικνύεται με μια μαύρη κουκίδα ενώ η κατάσταση τέλους με μια μαύρη κουκίδα και ένα κύκλο γύρω της.

Σχήμα 5.4.6.2.2-2: Αρχική και Τελική κατάσταση

Κάθε διάγραμμα δραστηριοτήτων μπορεί να έχει μόνο μια κατάσταση αρχής αλλά μπορεί να έχει αναρίθμητες καταστάσεις τέλους. Η μοντελοποίηση υποτίθεται ότι κάνει τη ζωή μας ως αναλυτή και σχεδιαστή ευκολότερη – όχι δυσκολότερη.

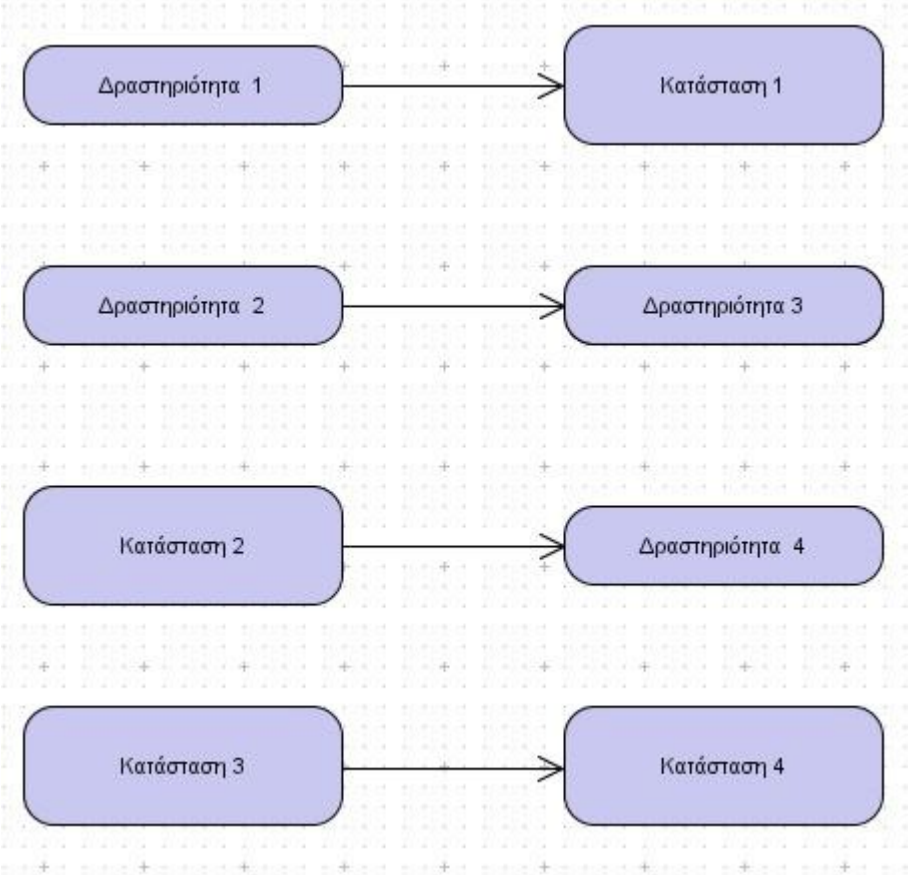


Σχήμα 5.4.6.2.2-3: Διάφορα παραδείγματα ονομασίας καταστάσεων

Όπως μπορούμε να δούμε από το προηγούμενο παράδειγμα, μπορούμε να βάλουμε ετικέτα σε κάθε κατάσταση αρχής και τέλους σα να ήταν απλές καταστάσεις.

5.4.6.2.3 Μεταβάσεις

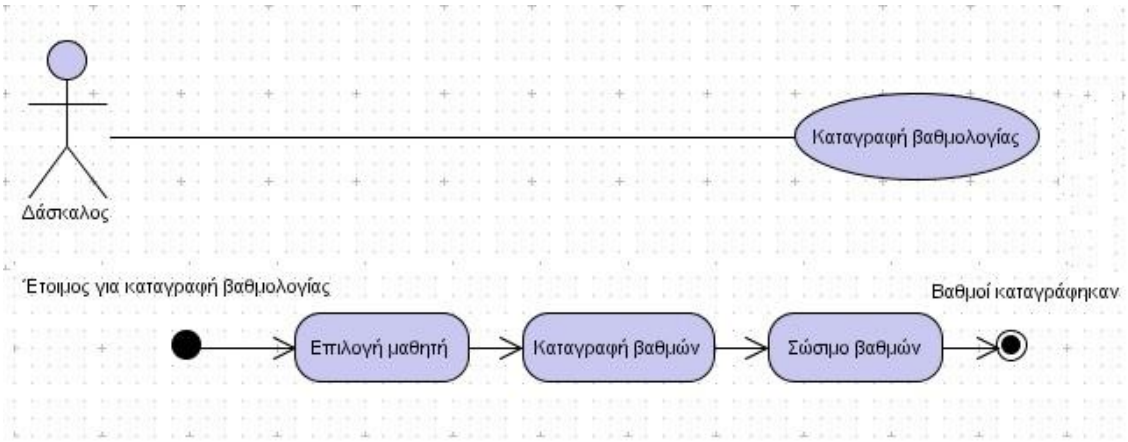
Οι μεταβάσεις (transitions) χρησιμοποιούνται για να δείξουν τη ροή ελέγχου από τη μια κατάσταση στην άλλη. Μπορούν να δείξουν τη ροή από μια κατάσταση σε μια δραστηριότητα, ανάμεσα σε δραστηριότητες ή ανάμεσα σε καταστάσεις. Η σημειολογία για μια μετάβαση είναι ένα ανοιχτό βέλος που δείχνει προς την κατεύθυνση της ροής ελέγχου.



Σχήμα 5.4.6.2.3-1: Μορφές μεταβάσεων

5.4.6.2.4 Συνθέτοντας τα σημειολογικά στοιχεία

Τώρα που έχουμε δει και τα τρία σημειολογικά στοιχεία ενός διαγράμματος δραστηριοτήτων, μπορούμε να τα συνθέσουμε για να κάνουμε το πρώτο μας διάγραμμα δραστηριοτήτων.



Σχήμα 5.4.6.2.4-1: Παράδειγμα διαγράμματος δραστηριότητας

Σε αυτό το παράδειγμα, αποδεικνύουμε ότι υπάρχουν περισσότερες πληροφορίες για τα σενάρια από ό,τι μοντελοποιείται στο διάγραμμα σενάριων. Ξεκινώντας με ένα σενάριο που δείχνει ότι ένα Δάσκαλος καταγράφει βαθμολογίες, έχουμε δημιουργήσει ένα διάγραμμα δραστηριοτήτων που δείχνει τα τρία απαραίτητα

βήματα για να καταγράψει ο δάσκαλος βαθμολογίες (επιλέγω μαθητή, καταγράφω βαθμολογίες και αποθηκεύω βαθμολογίες).

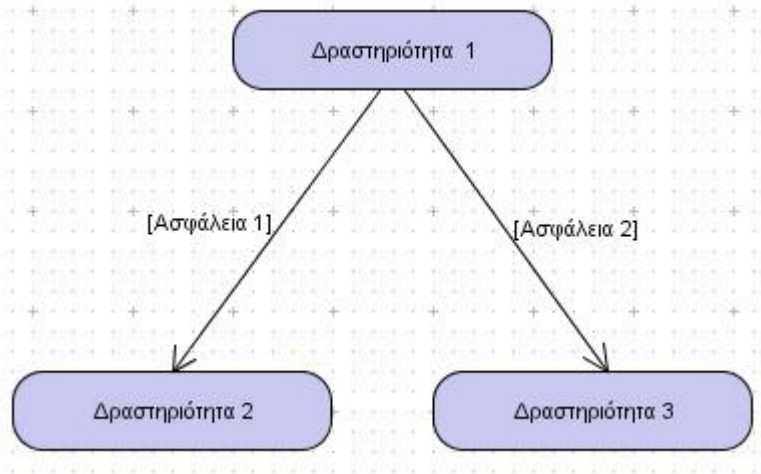
5.4.6.3 Χρησιμοποιώντας συνθήκες

Οι συνθήκες είναι απαραίτητες προσθήκες για να αλλάξετε την κατεύθυνση της ροής κάθε εργασίας. Χωρίς συνθήκες, τα πάντα θα ξεκινούσαν σε ένα σημείο Α και θα συνέχιζαν σε ένα σημείο Β χωρίς καμία περίπτωση παρέκκλισης. Επειδή, όμως, εδώ μοντελοποιούμε ροές εργασιών, η έλλειψη παρέκκλισης σπάνια μας βοηθάει, γι'αυτό θα μάθουμε πώς να χρησιμοποιούμε τις συνθήκες για να ενισχύσουμε τα διαγράμματά μας.

Με τη χρήση σημείων φύλαξης και απόφασης, μπορούμε να μοντελοποιούμε συνθήκες που αλλοιώνουν τη ροή του διαγράμματός μας. Τα σημεία φύλαξης χρησιμοποιούνται για να επιτρέπουν στον έλεγχο να ρέει μόνο προς την κατεύθυνση που πληρεί τις προϋποθέσεις, ενώ τα σημεία απόφασης απαιτούν μία απόφαση για να προσδιοριστεί σε ποια κατεύθυνση θα συνεχίσει η ροή.

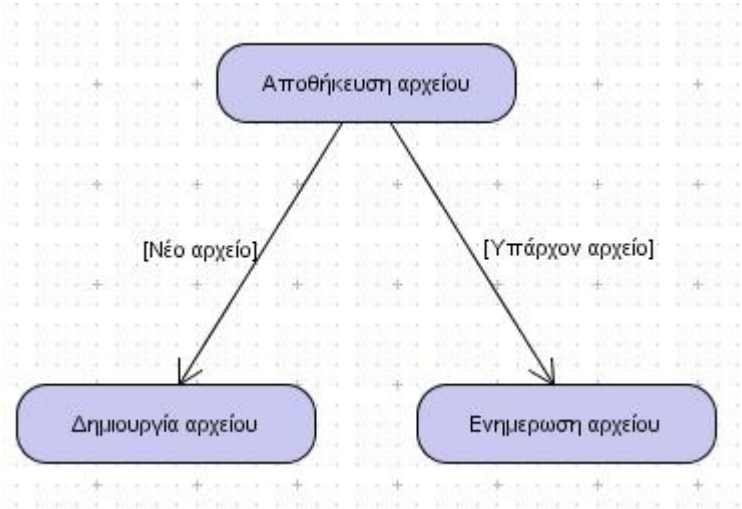
5.4.6.3.1 Σημεία φύλαξης

Τα σημεία φύλαξης (guards) σημειώνονται ανάμεσα σε δύο δραστηριότητες ή καταστάσεις και τοποθετούνται ανάμεσα σε αγκύλες (για παράδειγμα [όνομα σημείου φύλαξης]).



Σχήμα 5.4.6.3.1-1: Σημειολογία σημείων φύλαξης

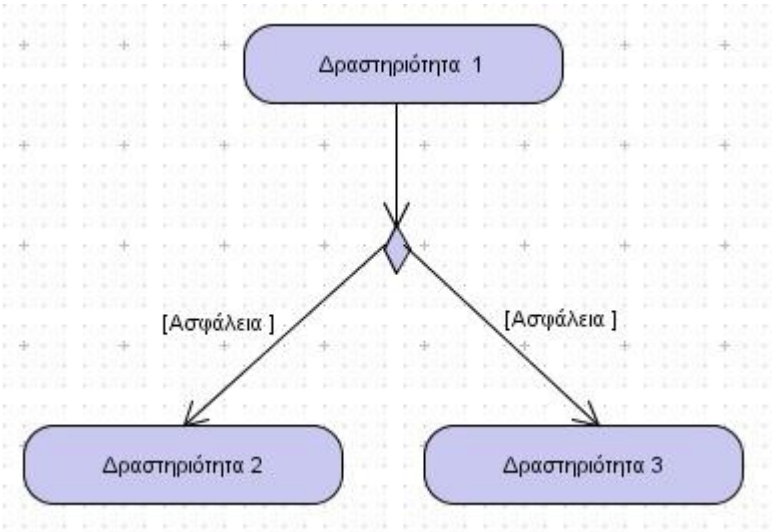
Στο προηγούμενο παράδειγμα, η ροή ελέγχου θα πάει από την Κατάσταση Ενέργειας Ένα στην Κατάσταση Ενέργειας Δύο αν το [σημείοφύλαξης1] είναι Αληθές αλλά η ροή θα πάει από την Κατάσταση Ενέργειας Ένα στην Κατάσταση Ενέργειας 3 αν το [σημείοφύλαξης2] είναι Αληθές. Στο επόμενο παράδειγμα, όταν ένας χρήστης αποθηκεύει ένα αρχείο, τα σημεία φύλαξης χρησιμοποιούνται για να προσδιορίσουν πώς να αποθηκευτεί το αρχείο. Αν το αρχείο ήδη υπάρχει, τότε ενημερώνεται· αν το αρχείο δεν υπάρχει, ένα νέο δημιουργείται.



Σχήμα 5.4.6.3.1-2: Παράδειγμα σημείων φύλαξης

5,4.6.3.2 Σημεία απόφασης

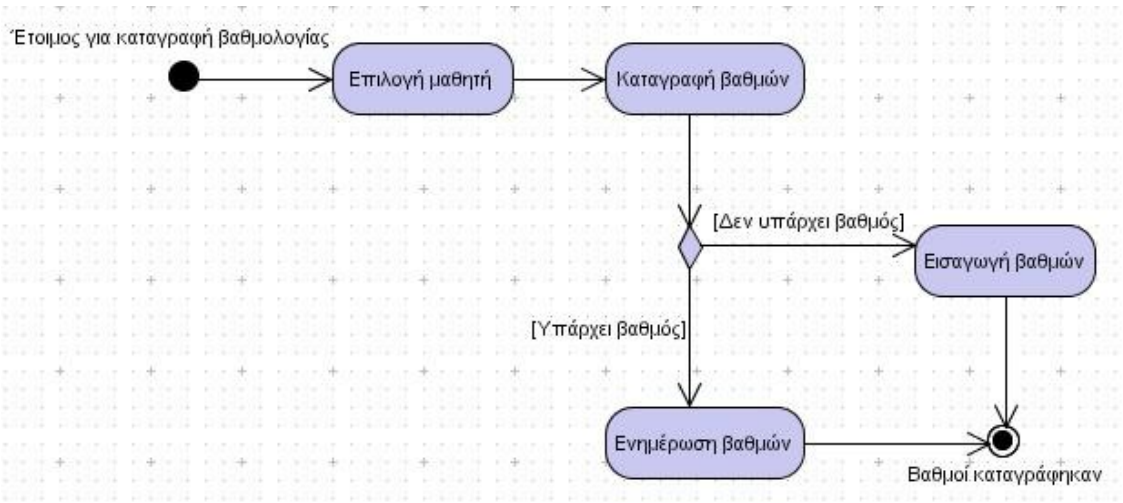
Τα σημεία απόφασης (decision points), σε κάθε περίπτωση, κάνουν το ίδιο πράγμα όπως τα κανονικά σημεία φύλαξης αλλά πιο τακτικά, ειδικά αν έχετε ένα μεγάλο διάγραμμα με πολλές διαφορετικές συνθήκες. Χρησιμοποιώντας τη σημειολογία ενός διαμαντιού, από το οποίο διακλαδώνονται όλα τα σημεία φύλαξης υπό συνθήκες, οι μεταβάσεις διατάσσονται από την προερχόμενη δραστηριότητα και εστιάζονται σε ένα σημείο όπου μεταβαίνει η δραστηριότητα. Αυτό δίνει στον αναγνώστη την εντύπωση ότι η ενέργεια ολοκληρώνεται με τη λήψη μιας απόφασης.



Σχήμα 5.4.6.3.2-1: Σημειολογία σημείων απόφασης

Θα προσέξετε ότι τα σημεία φύλαξης χρησιμοποιούνται ακόμη με τα σημεία απόφασης αλλά και διακλαδώνονται από τις μεταβάσεις μέχρι το σημείο απόφασης και όχι τόσο από τη αρχική δραστηριότητα. Όταν εκθέτετε το διάγραμμα δραστηριοτήτων για ανάγνωση, το σημείο απόφασης διευκολύνει γιατί τάσσει κατά

διαστήματα τις υπό συνθήκη μεταβιβάσεις μεταξύ τους, δίνοντας κάποιο διάστημα και στο ίδιο.



Σχήμα 5.4.6.3.2-2: Παράδειγμα σημείων απόφασης

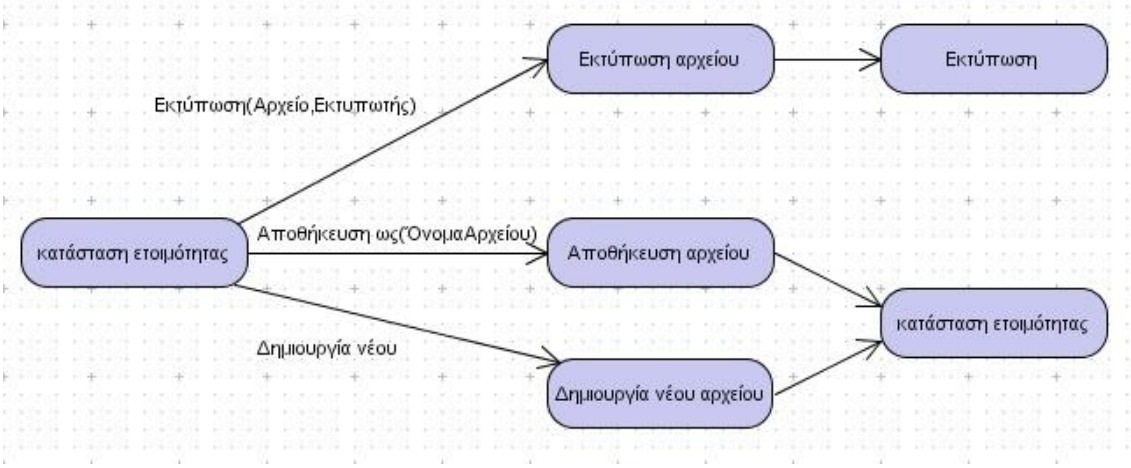
5.4.6.4 Επιπλέον σημειολογίες

Επιπροσθέτως των βασικών που έχετε μάθει μέχρι τώρα για τα διαγράμματα δραστηριοτήτων, υπάρχουν αρκετές επιπλέον σημειολογίες που μπορείτε να χρησιμοποιήσετε για να ενισχύσετε τις τεχνικές των διαγραμμάτων σας. Η χρήση συμβάντων και διεγερτών είναι παρόμοια με τη χρήση των στοιχείων φύλαξης, εκτός που αντί να ελέγχουν τη ροή, τα συμβάντα και οι διεγέρτες «πυροδοτούνται» για να μετακινήσουν τη ροή ελέγχου στη σχετική κατεύθυνση. Οι στήλες χρησιμοποιούνται για να απομονώσουν δραστηριότητες σύμφωνα με το πεδίο ορισμού ή το αντικείμενο. Και, τελικά, οι διασπάσεις και οι ενώσεις χρησιμοποιούνται για την παράλληλη επεξεργασία μεταβιβάσεων (όταν δύο μεταβιβάσεις μπορούν να συμβούν τον ίδιο χρόνο).

5.4.6.4.1 Συμβάντα και διεγέρτες

Τα συμβάντα (events) είναι πολύ παρόμοια με τις λειτουργίες ή μεθόδους (καθώς μπορεί να είστε εξοικειωμένοι από την εμπειρία σας στον προγραμματισμό). Τα συμβάντα είναι στοιχείο υψηλότερου επιπέδου σημειολογικής ανάλυσης (βρισκόμαστε ακόμη στη φάση της ανάλυσης) από τις λειτουργίες. Τα συμβάντα είναι ενδείξεις ότι μια ενέργεια λαμβάνει χώρα. Μπορούν να συμπεριλάβουν ένα ή περισσότερα ορίσματα, τοποθετημένα μέσα σε παραθέσεις και ακολουθώντας το όνομα του συμβάντος.

Τα συμβάντα μπορούν να συμπεριληφθούν στις μεταβάσεις για να υποδείξουν ότι η διαδικασία υποχρεώνει τον έλεγχο να κινηθεί προς μια συγκεκριμένη κατεύθυνση.

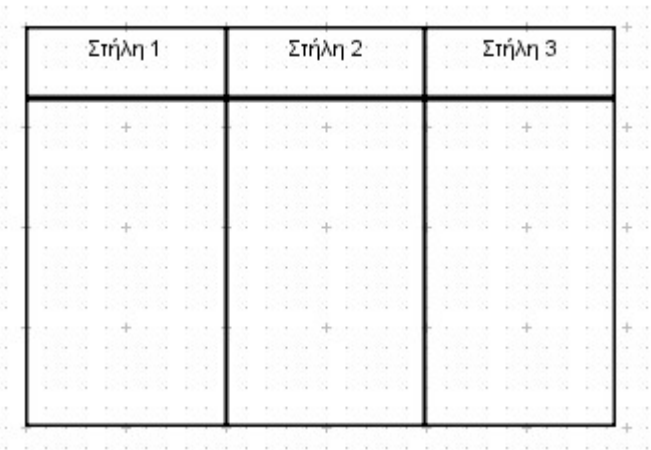


Σχήμα 5.4.6.4.1-1: Συμβάντα

Στο προηγούμενο παράδειγμα, η ροή ελέγχου μπορεί να πάει προς τις τρεις κατευθύνσεις βασισμένη στο συμβάν ότι διεγείρει τη ροή για να αφήσει την κατάσταση Έτοιμος και να μετακινηθεί στην αντίστοιχη δραστηριότητα. Το πρώτο συμβάν, Εκτυπώνω, χρειάζεται δύο ορίσματα, το αρχείο και τον εκτυπωτή, για τη δραστηριότητα Εκτυπώνω Αρχείο. Το δεύτερο συμβάν, Αποθηκεύω Ως, καταγράφει απλώς το όνομα του αρχείο στη δραστηριότητα Αποθηκεύω Αρχείο. Το τελικό συμβάν, ΔημιουργώΝέο, δε χρειάζεται κανένα όρισμα γιατί φέρνει στη ροή ελέγχου τη δραστηριότητα Δημιουργώ Νέο Αρχείο.

5.4.6.4.2 Στηλοθέτηση

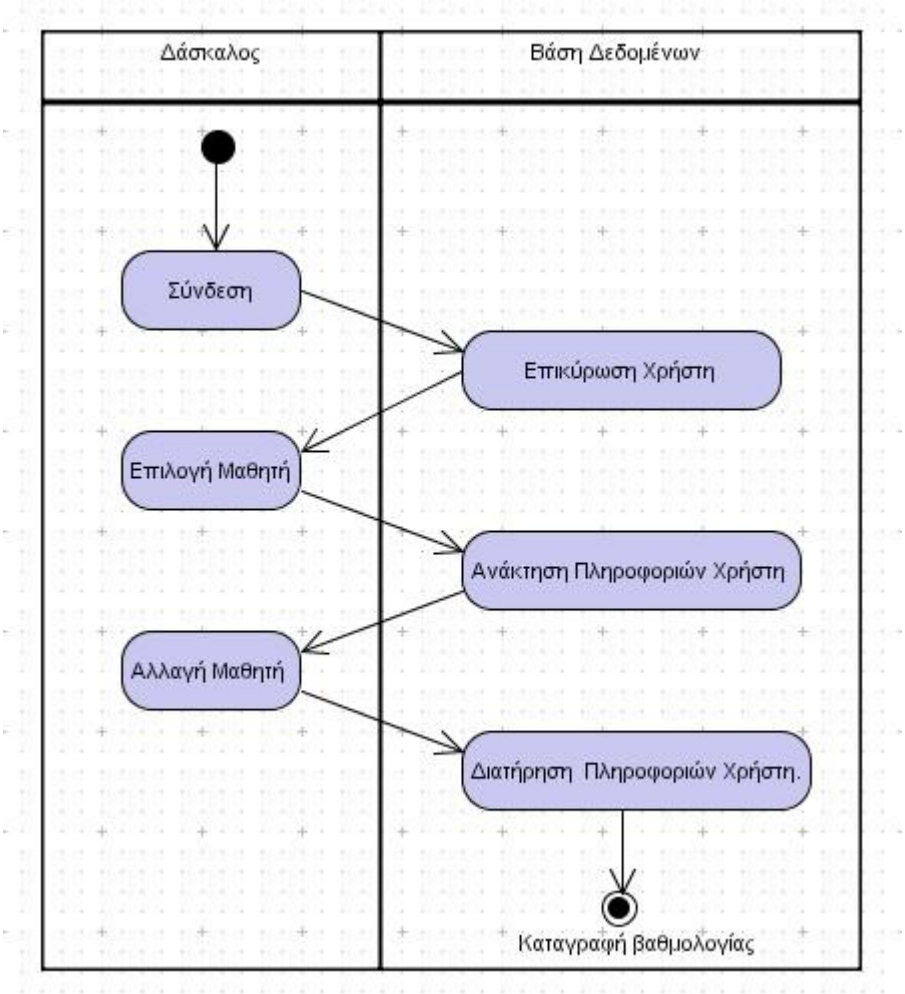
Οι στήλες νοικοκυρεύουν και ενισχύουν την αναγνωσιμότητα ενός διαγράμματος δραστηριοτήτων ένα μεγάλο βαθμό. Με την απεικόνισή τους ως μεγάλα ορθογώνια κουτιών με το όνομα του αντικειμένου ή του πεδίου στην κορυφή τους, οι στήλες χρησιμοποιούνται για να συμπεριλαμβάνουν δραστηριότητες στο αντίστοιχο «σπίτι» τους.



Σχήμα 5.4.6.4.2-1: Σημειολογία στηλοθέτησης

Το παράδειγμα που ακολουθεί δείχνει ένα διάγραμμα δραστηριοτήτων ενώ εξελίσσεται προς τα εμπρός και πίσω ανάμεσα στη διεπαφή Δασκάλου και Ιστού,

διασταυρώνοντας τη ροή ελέγχου. Χωρίς τις στήλες, αυτό το διάγραμμα δραστηριοτήτων δε θα μπορούσε να σας πει ότι ο Δάσκαλος χρησιμοποιεί τις δραστηριότητες Συνδέομαι, Επιλέγω Μαθητή και Αλλάζω Μαθητή και η διεπαφή του Ιστού χρησιμοποιεί τις δραστηριότητες Επικυρώνω Χρήστη, Ανακτώ τις Πληροφορίες του Χρήστη και Διατηρώ τις Πληροφορίες του Χρήστη.



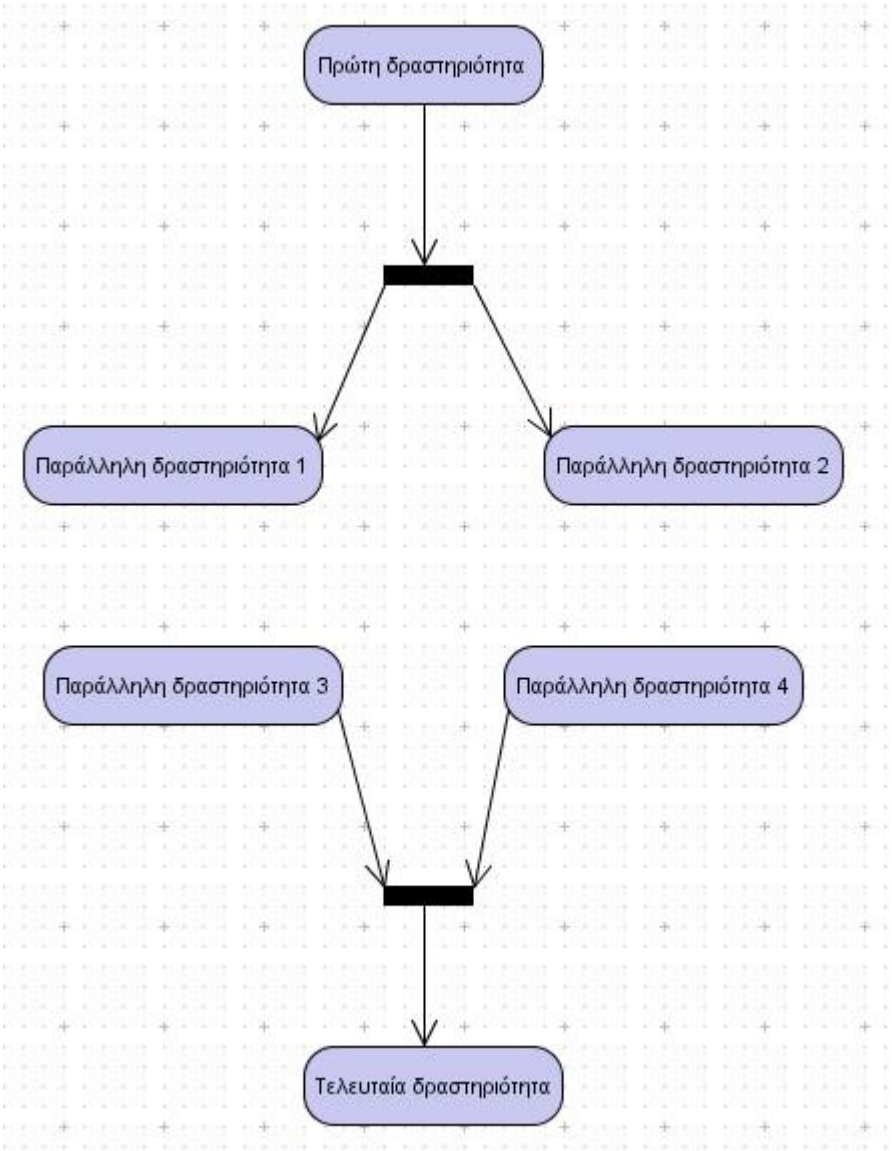
Σχήμα 5.4.6.4.2-2: Παράδειγμα στηλοθέτησης

Αν και ίσως θα μπορούσατε να συμπεράνετε σε ποιο αντικείμενο ή σε ποιο πεδίο ανήκει μια δραστηριότητα ενώ διαβάζετε αυτά τα παραδείγματα, καθώς αποκτάτε μεγαλύτερη εξοικείωση με την UML, θα ανακαλύψετε ότι τα διαγράμματα γίνονται όλο και δυσκολότερα. Η ικανότητά σας, βέβαια, θα μεγαλώσει επίσης αλλά η ικανότητα να διαβάσετε το μυαλό του μοντελιστή όχι. Οι στήλες είναι ένας πολύ καλός τρόπος να κάνετε σαφή τα σημεία σε οποιοδήποτε διάγραμμα δραστηριοτήτων και σας προτείνω να τις χρησιμοποιείτε κάθε φορά που το μοντέλο σας αφορά περισσότερα από ένα αντικείμενα.

5.4.6.4.3 Διασπάσεις και Ενώσεις

Οι διασπάσεις (forks) επιτρέπουν σε παράλληλες διαδικασίες να ξεκινήσουν και οι ενώσεις επιτρέπουν σε παράλληλες διαδικασίες να προφτάσουν και να ξαναρχίσουν τη ροή μιας μονής διαδικασίας. Όταν μια διάσπαση συναντάται με ένα διάγραμμα δραστηριοτήτων, κάθε διακλάδωση της ένωσης είναι, στην ουσία, το δικό του ξεχωριστό διάγραμμα δραστηριοτήτων, χωρίς σεβασμό στα άλλα. Καθεμιά από τις ροές ελέγχου δε χρειάζεται να περιμένει για την άλλη – μέχρι δηλαδή να έρθει κάποια για να συνδεθεί.

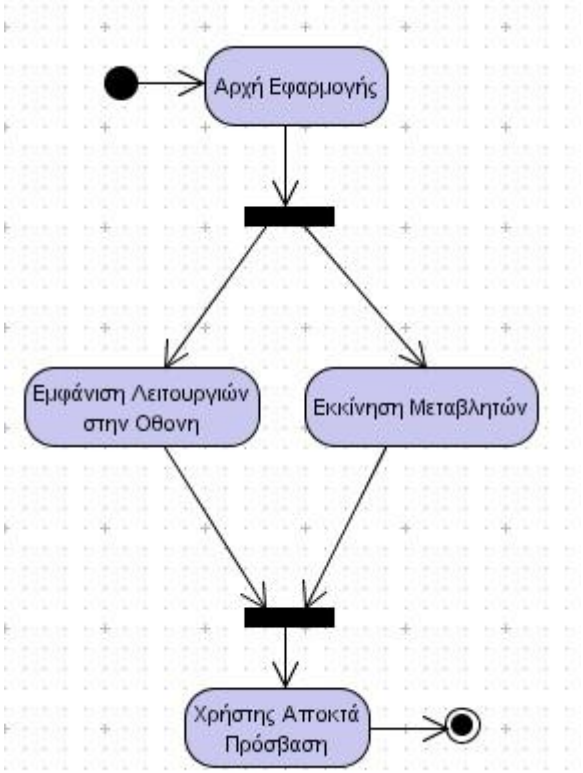
Τόσο οι διασπάσεις όσο και οι ενώσεις (joins) αναπαρίστανται παρόμοια στη UML. Έχουν και οι δύο μια παχιά μαύρη γραμμή. Οι διασπάσεις έχουν μία μετάβαση που εισέρχεται και δύο ή περισσότερες που εξέρχονται. Αυτό απεικονίζει τη μονή διαδικασία ροής ελέγχου που διαχωρίζεται σε πολλαπλές ροές ελέγχου. Οι ενώσεις, από την άλλη, είναι το ακριβώς αντίθετο. Έχουν δύο ή περισσότερες μεταβάσεις που εισέρχονται και μόνο μία που εξέρχεται. Αυτό απεικονίζει τις ξεχωριστές διαδικασίες που ενώνονται για να ξανασχηματίσουν μία μονή διαδικασία.



Σχήμα 5.4.6.4.3-1: Σημειολογία διασπάσεων και ενώσεων

Αν μια διαδικασία φτάσει στην ένωση πριν από την άλλη, περιμένει και ελέγχει να μην ξεπεράσει ποτέ την ένωση αν δεν είναι έτοιμες και οι δύο διαδικασίες.

Το παράδειγμα που ακολουθεί απεικονίζει πώς μία διαδικασία διάσπασης ίσως χρειαστεί περισσότερη ώρα από κάποια άλλη. Αυτό, βεβαίως, είναι τελείως σχετικό με τον αριθμό των δραστηριοτήτων σε κάθε διαδικασία. Επειδή δεν ξέρουμε πόσο διαρκεί κάθε δραστηριότητα, δεν μπορούμε να εγγυηθούμε ποια θα ολοκληρωθεί πρώτα. Εξαιτίας αυτού, έχουμε μια ένωση ακριβώς πριν δώσουμε πρόσβαση στην εφαρμογή στο χρήστη, για να σιγουρέψουμε ότι οι δύο ξεχωριστές διαδικασίες θα προλάβουν η μία την άλλη.



Σχήμα 5.4.6.4.3-2: Παράδειγμα διασπάσεων και ενώσεων

5.4.6.5 Μοντελοποίηση διαγραμμάτων δραστηριότητας

Υπάρχουν πέντε εργασίες για τη δημιουργία διαγραμμάτων δραστηριοτήτων:

1. Αναγνωρίστε τα σενάρια που απαιτούν διάγραμμα δραστηριοτήτων.
2. Μοντελοποιήστε τις πρωταρχικές διαδρομές για κάθε σενάριο.
3. Μοντελοποιήστε τις εναλλακτικές διαδρομές για κάθε σενάριο.
4. Προσθέστε στήλες για να αναγνωρίσετε τις επιχειρηματικές περιοχές για τις δραστηριότητες.
5. Εκλεπτύνετε τις υψηλού επιπέδου δραστηριότητες σε περισσότερα διαγράμματα δραστηριοτήτων.

5.4.6.6 Περίληψη

Σημειολογία

Ένα μοντέλο δραστηριοτήτων είναι μια παραλλαγή μιας μηχανής καταστάσεων στην οποία οι καταστάσεις είναι Δραστηριότητες που εκπροσωπούν την αποδοτικότητα των λειτουργιών και στην οποία οι μεταβάσεις λαμβάνουν το έναυσμα από τη συμπλήρωση των λειτουργιών. Αντιπροσωπεύει μια μηχανή κατάστασης μιας διαδικασίας. Η διαδικασία αποτελεί την υλοποίηση μιας λειτουργίας της κλάσης.

Συμβολισμός

Ένα διάγραμμα δραστηριοτήτων είναι μια ειδική περίπτωση ενός διαγράμματος καταστάσεων στο οποίο όλες (ή τουλάχιστον οι περισσότερες) από τις καταστάσεις είναι καταστάσεις ενέργειας και στο οποίο όλες (ή τουλάχιστον οι περισσότερες) από τις μεταβάσεις παίρνουν το έναυσμα από τη συμπλήρωση των ενεργειών στις πηγαίες καταστάσεις. Ολόκληρο το διάγραμμα δραστηριοτήτων επισυνάπτεται (μέσω του μοντέλου) σε μία κλάση ή στην υλοποίηση μιας λειτουργίας ή ενός σεναρίου. Ο σκοπός αυτού του διαγράμματος είναι να δώσει έμφαση σε καταστάσεις όπου όλα ή τα περισσότερα γεγονότα αντιπροσωπεύουν τη συμπλήρωση ενεργειών που προέρχονται από μέσα (διαδικαστική ροή ελέγχου, δηλαδή). Χρησιμοποιείστε κανονικά διαγράμματα καταστάσεων σε περιπτώσεις που λαμβάνουν χώρα ασύγχρονα γεγονότα.

5.4.7 ΔΙΑΓΡΑΜΜΑΤΑ ΣΥΣΤΑΤΙΚΩΝ

Σε προηγούμενα κεφάλαια, μάθαμε για τα διαγράμματα που ασχολούνται με ιδεατές οντότητες. Για παράδειγμα, ένα διάγραμμα κλάσεων αντιπροσωπεύει μια ιδέα – μία αφαίρεση ειδών που ταιριάζουν σε μια κατηγορία. Ένα διάγραμμα καταστάσεων επίσης αντιπροσωπεύει μια ιδέα-αλλαγές της κατάστασης του αντικειμένου.
Σε αυτό το κεφάλαιο, θα μάθετε για ένα διάγραμμα της UML που αντιπροσωπεύει ένα διαφορετικό είδος οντότητας: ένα συστατικό λογισμικού (component).

5.4.7.1 Τι είναι συστατικό

Ένα συστατικό λογισμικού είναι ένα μέρος συνάρτησης ενός συστήματος. Επειδή είναι η λογισμική υλοποίηση ενός ή περισσοτέρων κλάσεων, ένα συστατικό εδρεύει στον υπολογιστή, όχι στο μυαλό ενός αναλυτή. Ένα συστατικό παρέχει διασύνδεση με τα άλλα συστατικά.

Στην UML 1.x., αρχεία δεδομένων, πίνακες, εκτελέσεις, αρχεία και δυναμικά link βιβλιοθηκών προσδιορίστηκαν ως συστατικά. Στην πραγματικότητα, οι μοντελιστές κατέτασσαν αυτού του είδους τα αντικείμενα ως συστατικά ανάπτυξης, συστατικά προϊόντος εργασίας και συστατικά εκτέλεσης. Η UML 2.0 αναφέρεται σε αυτά ως τεχνούργημα (artifact), κομμάτια πληροφορίας που ένα σύστημα χρησιμοποιεί ή παράγει.
Ένα συστατικό, εν αντιθέσει, χαρακτηρίζει τη λειτουργικότητα ενός συστήματος. Όπως ένα εξάρτημα είναι η υλοποίηση ενός ή περισσοτέρων κλάσεων, ένα τεχνούργημα (αν είναι εκτελέσιμο) είναι η υλοποίηση ενός συστατικού.
Μοντελοποιείτε συστατικά και τις σχέσεις τους κατά τέτοιο τρόπο που:

- Οι πελάτες να μπορούν να δουν ξεκάθαρα τη δομή και τη λειτουργικότητα του τελικού συστήματος.
- Οι προγραμματιστές να έχουν μια δομή με την οποία μπορούν να δουλέψουν περαιτέρω.
- Οι συγγραφείς που πρέπει να παρέχουν κείμενα και αρχεία για βοήθεια να μπορούν να καταλαβαίνουν τι γράφουν.
- Να είστε έτοιμοι για επαναχρησιμοποίηση.

Ας εξετάσουμε το τελευταίο. Μία από τις πιο σημαντικές πτυχές των συστατικών είναι η δυνατότητα που παρέχουν για επαναχρησιμοποίηση. Στη σημερινή γρήγορα αναπτυσσόμενη επαγγελματική αρένα, όσο γρηγορότερα καρποφορήσει το σύστημά σας, τόσο μεγαλύτερη είναι η διάκρισή σας. Αν μπορείτε να δομήσετε ένα συστατικό για ένα σύστημα και να το επαναχρησιμοποιήσετε για ένα άλλο, συμβάλλετε σε αυτή τη διάκριση. Με το να ασχοληθείτε και με το να προσπαθήσετε να μοντελοποιήσετε ένα συστατικό η επαναχρησιμοποίηση είναι πιθανότερη.

5.4.7.2 Συστατικά και διασυνδέσεις

Όταν έχετε να κάνετε με συστατικά, πρέπει να ασχολείστε και με τις διασυνδέσεις του. Νωρίς στη συζήτησή μας για κλάσεις και αντικείμενα, μιλήσαμε για διασυνδέσεις. Ένα αντικείμενο κρύβει ό,τι κάνει από άλλα αντικείμενα και από τον έξω κόσμο. Το αντικείμενο πρέπει να παρουσιάσει ένα «πρόσωπο» στον έξω κόσμο έτσι ώστε τα άλλα αντικείμενα (συμπεριλαμβανομένων και των ανθρώπων) να μπορούν να ζητήσουν από το αντικείμενο να εκτελέσει τις λειτουργίες του. Αυτό το «πρόσωπο» αποτελεί τη διασύνδεση (interface) του αντικειμένου.

5.4.7.2.1 Επαναπροσδιορίζοντας τις διασυνδέσεις

Μία διασύνδεση είναι ένα σύνολο λειτουργιών που σας επιτρέπει να έχετε πρόσβαση στη συμπεριφορά μίας κλάσης – όπως το κουμπί ελέγχου που σας επιτρέπει να κάνετε το πλυντήριο να εκτελέσει λειτουργίες πλυντηρίου. Σκεφτείτε μια διασύνδεση όπως μία κλάση που έχει μόνο λειτουργίες και όχι ιδιότητες. Εν συντομία: Η διασύνδεση είναι ένα σύνολο λειτουργιών που μια κλάση παρουσιάζει σε άλλες κλάσεις.

Η σχέση μεταξύ μιας κλάσης και της διασύνδεσής της ονομάζεται πραγματοποίηση. Στην πραγματικότητα, μια διασύνδεση μπορεί να είναι ιδεατή ή φυσική. Η διασύνδεση που χρησιμοποιεί μία κλάση είναι η ίδια με τη διασύνδεση που χρησιμοποιεί η εφαρμογή του λογισμικού του. Για σας, ως μοντελιστές αυτό σημαίνει ότι ο τρόπος που παρουσιάζετε μία διασύνδεση για μία κλάση είναι ο ίδιος με τον τρόπο που παρουσιάζετε μια διασύνδεση για ένα εξάρτημα. Αν και ο συμβολισμός της UML διαφοροποιείται ανάμεσα σε μια κλάση και ένα εξάρτημα, δεν κάνει καμία διάκριση ανάμεσα σε μια ιδεατή και μία φυσική διασύνδεση.

Εδώ έχουμε ένα σημαντικό σημείο για να θυμάστε όσον αφορά τα συστατικά και τις διασυνδέσεις: μπορείτε να φτάσετε τις λειτουργίες ενός συστατικού μόνο μέσα από τις διασυνδέσεις του. Όπως σε μία κλάση και στη διασύνδεσή της, η σχέση μεταξύ ενός συστατικού και της διασύνδεσής του ονομάζεται πραγματοποίηση.

Κι άλλο ένα σημαντικό σημείο: ένα συστατικό μπορεί να κάνει τη διασύνδεσή του διαθέσιμη έτσι ώστε άλλα συστατικά να μπορούν να χρησιμοποιήσουν τις λειτουργίες της διασύνδεσης. Με άλλα λόγια, ένα συστατικό μπορεί να έχει πρόσβαση στις υπηρεσίες ενός άλλου συστατικού. Το συστατικό που προσφέρει τις υπηρεσίες του παρουσιάζει μια παρέχουσα διασύνδεση. Το συστατικό που αποκτά πρόσβαση χρησιμοποιεί μία προαπαιτούμενη διασύνδεση.

5.4.7.2.2 Αντικατάσταση και Επαναχρησιμοποίηση

Οι διασυνδέσεις εμφανίζονται έντονα στις σημαντικές ιδέες της αντικατάστασης και επαναχρησιμοποίησης των συστατικών. Μπορούμε να αντικαταστήσουμε ένα συστατικό με ένα άλλο εάν το νέο συστατικό εναρμονίζεται με τις ίδιες διασυνδέσεις όπως και το παλιό.

Για να δείξουμε την αντικατάσταση και τις διασυνδέσεις, ορίστε ένα παράδειγμα από το μηχανοκίνητο κόσμο. Πριν μερικά χρόνια, ένας είχε ένα συγκεκριμένο κλασικό σπορ αυτοκίνητο από τη δεκαετία του 60. Γρήγορα ανακάλυψε ότι ένα επιπλέον κομμάτι του εξοπλισμού ήταν άκρως απαραίτητο – άλλο ένα αυτοκίνητο ώστε να μπορεί να επισκέπτεται το σπορ αυτοκίνητο στο μαγαζί! Γιατί; Η μηχανή ήταν, για να το θέσου ήπια, «υψηλού πνεύματος» και χρειαζόταν συνέχεια σέρβις. Η λύση του φίλου ήταν να πάρει μία συνήθη μηχανή από μια άλλη μάρκα αυτοκινήτου – λιγότερο ισχυρή αλλά περισσότερο αξιόπιστη – και να αντικαταστήσει την αρχική μηχανή. Μπόρεσε να το κάνει αυτό επειδή η νέα μηχανή, αν και σχεδιασμένη και κατασκευασμένη για ένα εντελώς διαφορετικό αυτοκίνητο, έτυχε απλώς να διασυνδέεται ορθώς με άλλα εξαρτήματα του σπορ αυτοκινήτου.

Αυτή είναι μια επίσης πολύ καλή επίδειξη επαναχρησιμοποίησης. Μπορείτε να επαναχρησιμοποιήσετε ένα συστατικό σε ένα άλλο σύστημα (όπως η αντικατάσταση της μηχανή για το σπορ αυτοκίνητο) αν το νέο σύστημα μπορεί να έχει πρόσβαση στο επαναχρησιμοποιημένο συστατικό μέσω των διασυνδέσεων του συστατικού. Αν μπορείτε να τελειοποιήσετε τη διασύνδεση ενός συστατικού ώστε μια μεγάλη τάξη να μπορεί να έχει πρόσβαση, μπορείτε να επεξεργαστείτε κατά τέτοιο τρόπο το συστατικό ώστε να είναι δυνατή η επαναχρησιμοποίηση σε εργασίες ανάπτυξης σε όλη την επιχείρησή σας.

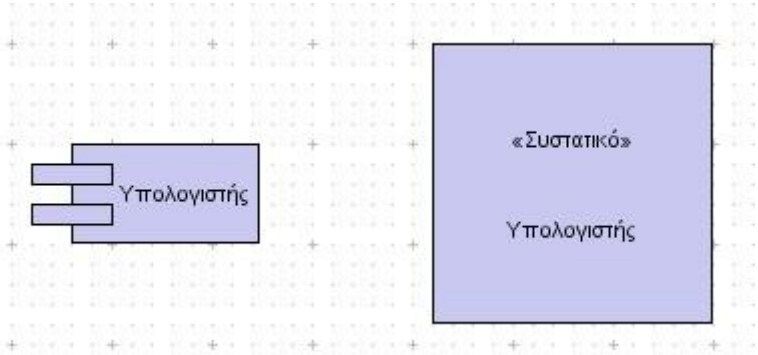
Σε αυτό το σημείο το σημείο χρειαζόμαστε τη μοντελοποίηση διασυνδέσεων. Η ζωή γίνεται ευκολότερο για ένα προγραμματιστή που προσπαθεί να αντικαταστήσει ή να επαναχρησιμοποιήσει ένα συστατικό αν οι πληροφορίες για τη διασύνδεση του συστατικού είναι γραπτώς διαθέσιμες με τη μορφή ενός μοντέλου. Αν όχι, ο προγραμματιστής πρέπει να περάσει από μια χρονοβόρο διαδικασία μέσω του κώδικα.

5.4.7.3 Τι είναι ένα διάγραμμα συστατικών

Ένα διάγραμμα συστατικών μοντελοποιεί ενότητες λογισμικού και τις μεταξύ τους σχέσεις. Το συστατικό είναι ένα κομμάτι λογισμικού. Μπορεί να είναι ένα αρχείο, ένα προϊόν, μια εντολή εκτέλεσης κτλ.

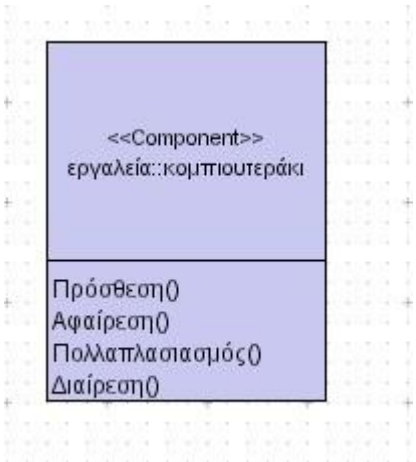
5.4.7.3.1 Διαγράμματα συστατικών στη UML 1.x και UML 2.0

Στη UML 1.x, το βασικό εικονίδιο ενός διαγράμματος συστατικών είναι ένα ορθογώνιο που έχει δύο άλλα ορθογώνια να επικαλύπτουν την αριστερή μεριά του. Πολλοί μοντελιστές βρήκαν το σύμβολο 1.x υπερβολικά άβολο, ιδιαίτερα όταν έπρεπε να δείξουν τη σύνδεση με την αριστερή πλευρά. Για αυτό το λόγο, η UML 2.0 παρέχει ένα νέο εικονίδιο συστατικού. Στη UML 2.0, το εικονίδιο είναι ένα ορθογώνιο με τη λέξη κλειδί «συστατικό» κοντά στην κορυφή. Το σχέδιο παρακάτω δείχνει αυτά τα εικονίδια.



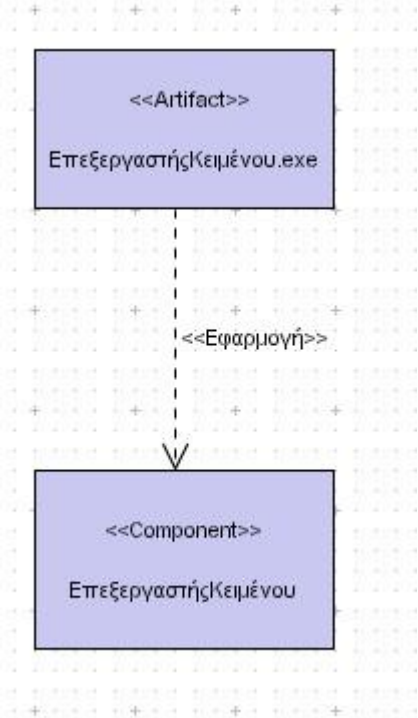
Σχήμα 5.4.7.3.1-1: Το εικονίδιο συστατικού στη UML 1.x και στη UML 2.0

Το σχέδιο δείχνει ότι εάν το συστατικό είναι μέρος του πακέτου, μπορείτε να προτάσσετε το όνομα του συστατικού με το όνομα του πακέτου. Μπορούμε επίσης, να δείξουμε τις λειτουργίες του συστατικού σε ξεχωριστό πλαίσιο.



Σχήμα 5.4.7.3.1-2: Προσθέτοντας πληροφορίες στο εικονίδιο συστατικού

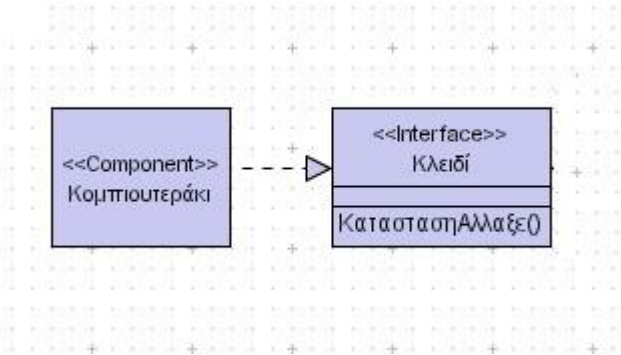
Μιλώντας για τεχνουργήματα, το σχήμα δείχνει μερικούς τρόπους για να τα προβάλλετε και δείχνει επίσης πώς να μοντελοποιείτε τη σχέση ανάμεσα σε ένα συγκεκριμένο είδος ενός τεχνουργήματος (artifact, ενός εκτελέσιμου) και στο συστατικό εφαρμογής. Όπως μπορείτε να δείτε, μπορούμε να βάλουμε ένα συμβολισμό στο εικονίδιο του τεχνουργήματος, ανάλογο με το συστατικό του συμβόλου στο εικονίδιο του συστατικού στη UML 1.x



Σχήμα 5.4.7.3.1-3: Μοντελοποιώντας τη σχέση μεταξύ ενός τεχνουργήματος και ενός συστατικού

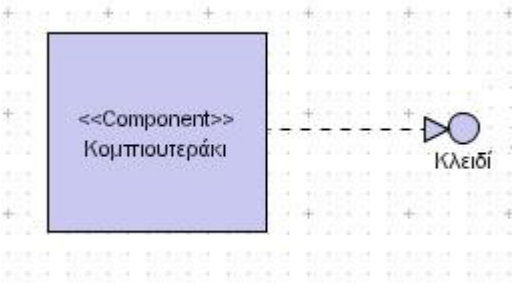
5.4.7.3.2 Αναπαριστώντας διασυνδέσεις

Ένα συστατικό και οι διασυνδέσεις που υλοποιεί αναπαρίστανται με δύο τρόπους. Ο πρώτος δείχνει τη διασύνδεση ως ορθογώνιο που περιέχει πληροφορίες σχετικές με τη διασύνδεση. Συνδέεται με το συστατικό με μια διακεκομμένη γραμμή κι ένα μεγάλο ανοιχτό τρίγωνο που προσδίδει την υλοποίηση.



Σχήμα 5.4.7.3.2-1: Αναπαράσταση διασύνδεσης

Το σχήμα παρακάτω δείχνει το δεύτερο τρόπο. Είναι εικονικός: Αναπαριστάτε τη διασύνδεση ως ένα μικρό κύκλο συνδεδεμένο με το συστατικό με μια απλή γραμμή.

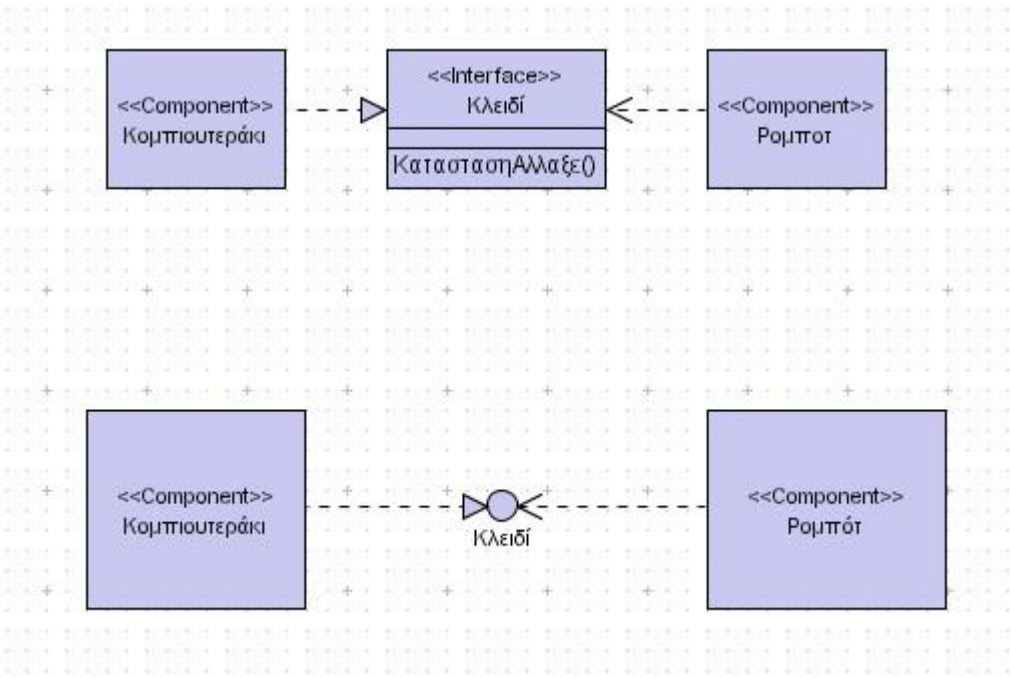


Σχήμα 5.4.7.3.2-2: Αναπαράσταση διασύνδεσης με άλλο τρόπο

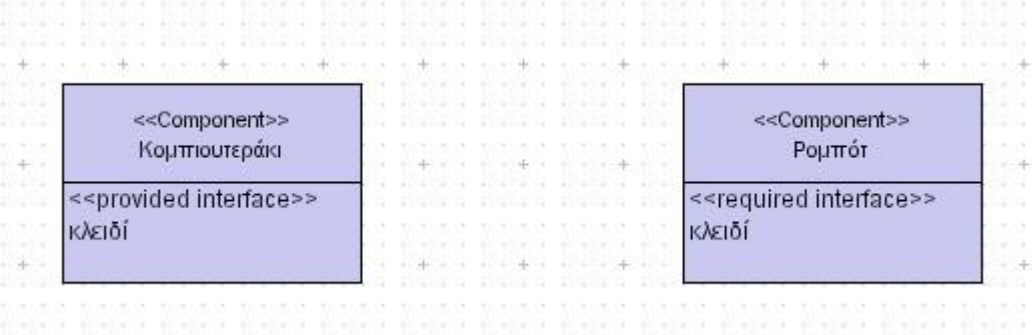
Επιπλέον, μπορείτε να αναπαραστήσετε την εξάρτηση – τη σχέση ανάμεσα σε ένα συστατικό και τη διασύνδεση μέσω της οποίας έχει πρόσβαση σε άλλο συστατικό. Η εξάρτηση απεικονίζεται ως μία διακεκομμένη γραμμή με ένα βέλος. Μπορεί να δείξετε την υλοποίηση και την εξάρτηση στο ίδιο διάγραμμα, όπως στο παρακάτω διάγραμμα. Το παρακάτω διάγραμμα δείχνει την ισοδύναμη «σημειογραφία» της «σφαιρικής άρθρωσης». Στην ορολογία που αναφέρθηκε νωρίτερα, «η σφαίρα» αναπαριστά μία παρέχουσα διασύνδεση και «η άρθρωση» μια προαπαιτούμενη διασύνδεση.

5.4.7.3.3 Πλαίσια – Μαύρα και Άσπρα

Όταν μοντελοποιούμε τις διασυνδέσεις ενός συστατικού, όπως στο παρακάτω σχήμα δείχνουμε αυτό που η UML ονομάζει μία εξωτερική, ή «μαύρου πλαισίου», μορφή. Έχετε επίσης την επιλογή να δείξετε μια εσωτερική, ή «άσπρου πλαισίου», μορφή. Αυτή η μορφή δείχνει τη λίστα των διασυνδέσεων μέσα στο εικονίδιο του συστατικού και μάλιστα οργανωμένη με λέξεις – κλειδιά. Το τελευταίο σχήμα δείχνει μια εσωτερική μορφή σχεδιασμού των διαγραμμάτων συστατικών.



Σχήμα 5.4.7.3.3-1: Δυο τρόποι να δείξουμε το ίδιο



Σχήμα 5.4.7.3.3-2: Μια οπτική «άσπρου-πλαισίου» των συστατικών του παραπάνω σχήματος

5.4.7.4 Μοντελοποίηση διαγραμμάτων συστατικών

Υπάρχουν τέσσερα βήματα για τη μοντελοποίηση διαγραμμάτων συστατικών.

1. Αναγνωρίζουμε και προσθέτουμε τα συστατικά.
2. Αναγνωρίζουμε και προσθέτουμε τις συσχετίσεις.
3. Προσθέτουμε τα στερεότυπα.

5.4.7.5 Περίληψη

Σημειολογία

Ένα διάγραμμα συστατικών δείχνει τις εξαρτήσεις ανάμεσα στα συστατικά του λογισμικού συμπεριλαμβανομένων και του πηγαίου κώδικα συστατικών, των διττών κωδικών συστατικών και των εκτελέσιμων συστατικών. Μερικά συστατικά υπάρχουν σε χρόνο μεταγλώττισης, άλλα σε χρόνο δεσμού κι άλλα σε χρόνο εκτέλεσης. Μερικά υπάρχουν παραπάνω από μία φορά. Ένα συστατικό – μεταγλωττιστής είναι αυτό που έχει σημασία μόνο σε χρόνο μεταγλώττισης. Σε αυτήν την περίπτωση ένα συστατικό εκτέλεσης θα ήταν ένα εκτελέσιμο πρόγραμμα.
Ένα διάγραμμα συστατικών έχει μόνο ένα τύπο μορφής, όχι μια στιγμιαία μορφή. Για να δείξετε τις στιγμές των συστατικών, χρησιμοποιείστε ένα διάγραμμα ανάπτυξης (πιθανώς ένα εκφυλισμένο χωρίς κόμβους).

Συμβολισμός

Ένα διάγραμμα συστατικών είναι ένα γράφημα συστατικών συνδεδεμένο από εξαρτημένες σχέσεις. Τα συστατικά μπορούν ακόμη να συνδέονται με τα συστατικά από φυσικό περιορισμό που αντιπροσωπεύει τη συσχέτιση των συνθέσεων.
Ένα διάγραμμα που περιλαμβάνει τύπους συστατικών και τύπους κόμβων μπορεί να χρησιμοποιηθεί για να παρουσιάσει εξαρτήσεις του μεταγλωττιστή, οι οποίες εμφανίζονται ως διακεκομμένα βέλη (εξαρτήσεις) από ένα συστατικό εξυπηρετούμενου σε ένα συστατικό προμηθευτή από το οποίο και εξαρτάται κατά κάποιο τρόπο. Τα είδη των εξαρτήσεων είναι γλωσσικά συγκεκριμενοποιημένα και ίσως παρουσιαστούν ως στερεότυπα των εξαρτήσεων.

Το διάγραμμα μπορεί ακόμη να χρησιμοποιηθεί για να δείξει διασυνδέσεις και τις λεγόμενες εξαρτήσεις ανάμεσα στα συστατικά χρησιμοποιώντας διακεκομμένα βέλη από τα συστατικά στις διασυνδέσεις άλλων συστατικών.

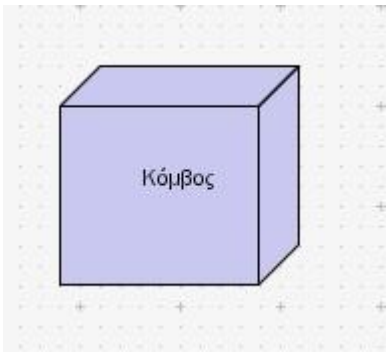
5.4.8 ΔΙΑΓΡΑΜΜΑΤΑ ΑΝΑΠΤΥΞΗΣ

Τα διαγράμματα ανάπτυξης (deployment diagrams) χρησιμοποιούνται για να μοντελοποιήσουμε τα μηχανήματα (hardware) όπως αυτά σχετίζονται με την ανάπτυξη του συστήματος που αναπτύσσουμε στην UML .Ένα διάγραμμα ανάπτυξης έχει δυο βασικά τμήματα στη σημειολογία του ,τον κόμβο και τη συσχέτιση επικοινωνίας.

5.4.8.1 Σημειολογικά στοιχεία διαγραμμάτων ανάπτυξης

5.4.8.1.1 Κόμβος

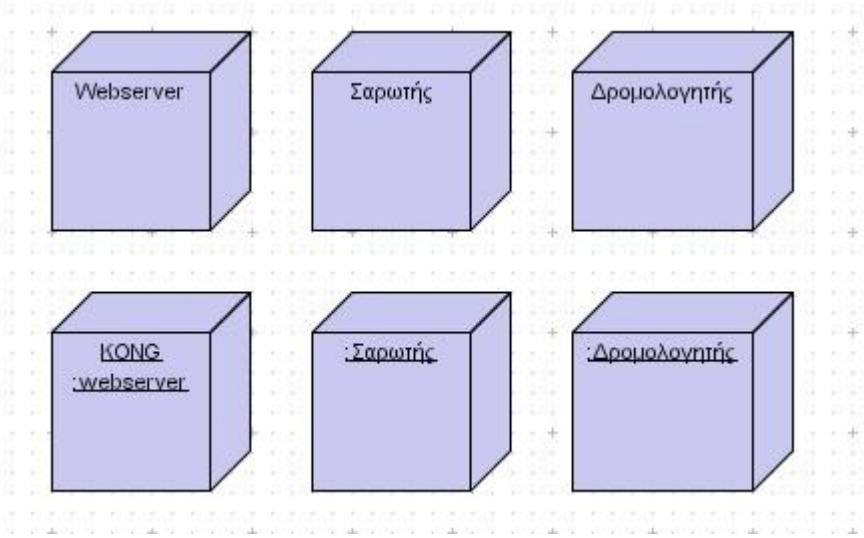
Ο κόμβος (node) χρησιμοποιείται για να δείξει ένα μηχάνημα, όπως είναι ένας εκτυπωτής ή ένας σαρωτής.



Σχήμα 5.4.8.1.1-1: Ένας κόμβος

Οι κόμβοι μπορούν να σχεδιαστούν σαν γενικές μορφές μηχανημάτων , όπως ένας webserver , ένας δρομολογητής (router) ,ένας σαρωτής (scanner). Μπορούν να δείχνουν όμως και πιο συγκεκριμένα στιγμιότυπα μηχανημάτων με το να αλλάξουμε τον τρόπο γραφής του ονόματος του κόμβου.

Στο παρακάτω παράδειγμα οι τρεις πάνω κόμβοι είναι γενικοί ενώ οι κάτω τρεις είναι συγκεκριμένα στιγμιότυπα των πάνω. Βλέπουμε ότι στους κάτω κόμβους τα ονόματα είναι υπογραμμισμένα στην αρχή έχουν την άνω-κάτω τελεία .

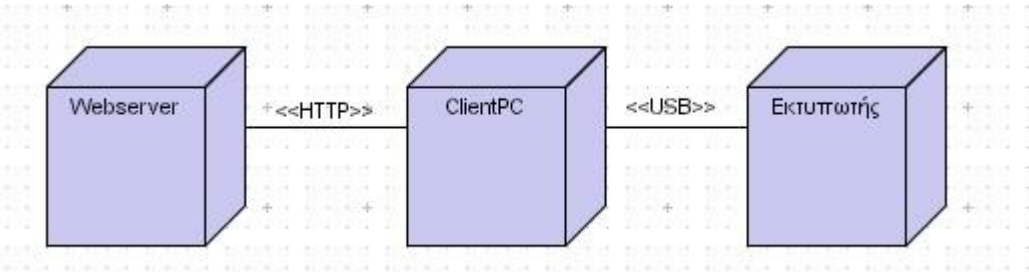


Σχήμα 5.4.8.1.1-2: Κόμβοι και στιγμιότυπα κόμβων

Στο προηγούμενο παράδειγμα , KONG είναι το όνομα του web server , υπάρχει μονό ένας KONG αλλά πολλοί webservers. Ο σαρωτής και ο δρομολόγησης δεν έχουν ιδιαίτερο όνομα αλλά ο αναγνώστης μπορεί να καταλάβει ότι είναι μοναδικοί γιατί είναι υπογραμμισμένοι και έχουν και άνω κάτω τελεία στην αρχή.

5.4.8.1.2 Συσχετίσεις επικοινωνίας

Οι κόμβοι συνδέονται με συσχετίσεις επικοινωνίας (communication associations), συμπαγείς γραμμές που σχεδιάζονται από τον ένα κόμβο στον άλλο. Αυτή η σχέση χρησιμοποιείται για να δείξει ότι δυο ή περισσότερα τμήματα εξοπλισμού επικοινωνούν μεταξύ τους με κάποιο τρόπο που συνήθως αναγράφεται πάνω στη συσχέτιση.

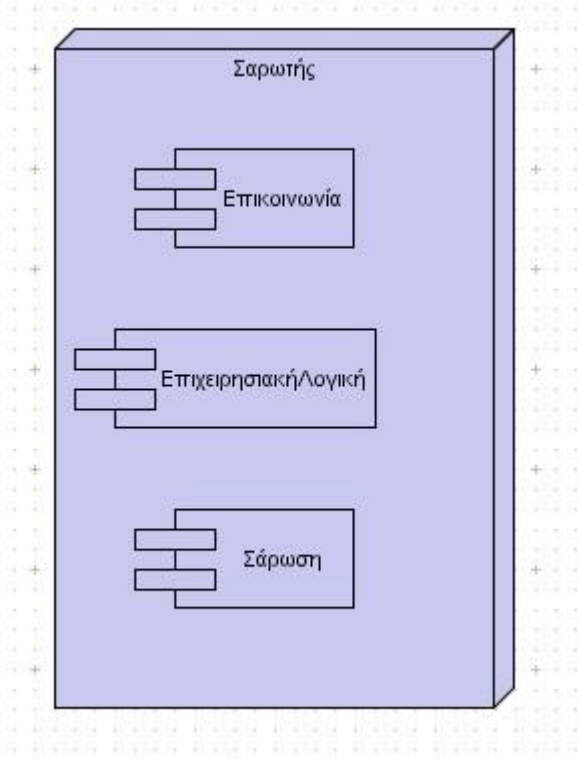


Σχήμα 5.4.8.1.2-1: Αλληλεπίδραση κόμβων

5.4.8.2 Συνδυάζοντας τα διαγράμματα συστατικών και ανάπτυξης

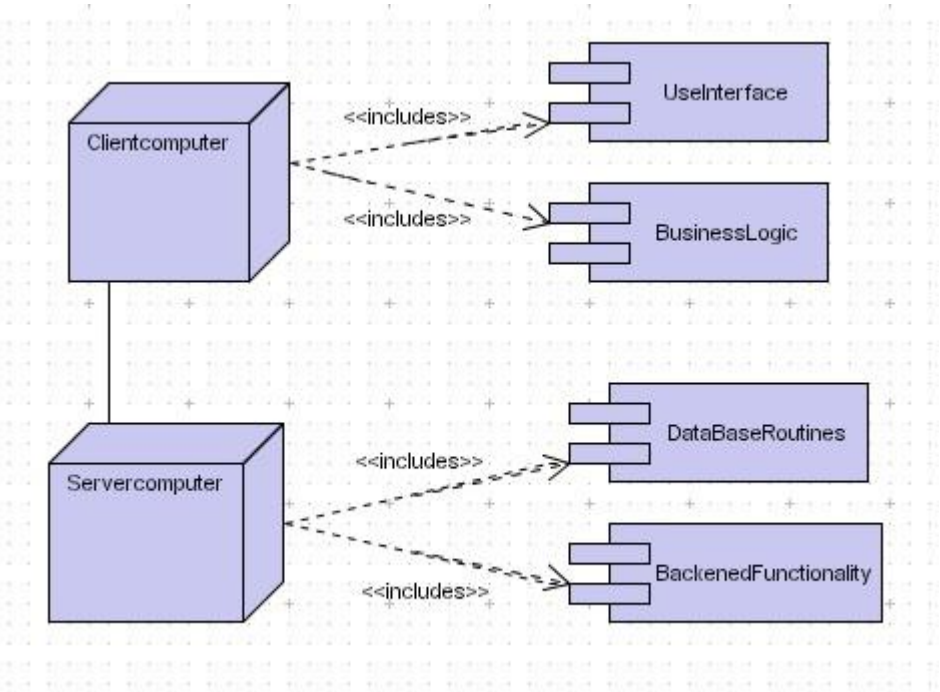
Τα διαγράμματα αυτά μπορούν να συνδυαστούν για να δείξουν πως το λογισμικό διανέμεται πάνω σε ένα συγκεκριμένο σύστημα. Όταν το λογισμικό συνδυάζεται με τα μηχανήματα, τεχνικά έχουμε ένα διάγραμμα ανάπτυξης.

Τα συστατικά μοντελοποιούνται ως μέρος ενός κόμβου. Το ακόλουθο διάγραμμα δείχνει πως η Επικοινωνία , η ΕπιχειρησιακηΛογικη και η Σάρωση ως συστατικά ενσωματώνονται στο μηχάνημα Σαρωτής.



Σχήμα 5.4.8.2-1: Ενσωμάτωση συστατικών σε ένα μηχάνημα

Ο άλλος τρόπος για να σχεδιάσουμε ένα μοντέλο είναι το λογισμικό που χρησιμοποιεί ένα μηχάνημα να είναι μέσα στον κόμβο που αντιπροσωπεύει το μηχάνημα αλλά να συνδέσουμε αυτό με μια συσχέτιση (η συσχέτιση αυτή αποτελείται από μια διακεκομμένη γραμμή και ένα ανοιχτό βέλος). Στο επόμενο διάγραμμα απεικονίζεται αυτός ο τρόπος σχεδίασης με την χρήση ενός ΠελατηΥπολογιστη(client) και ενός server.



Σχήμα 5.4.8.2-2: Άλλος τρόπος απεικόνισης αλληλεπίδρασης συστατικών και κόμβων

5.4.8.3 Μοντελοποίηση διαγραμμάτων ανάπτυξης

Υπάρχουν τρία βήματα στην κατασκευή διαγραμμάτων ανάπτυξης.

1. Αναγνωρίζουμε και προσθέτουμε γενικούς κόμβους.
2. Αναγνωρίζουμε και προσθέτουμε στιγμιότυπα κόμβων.
3. Αναγνωρίζουμε και προσθέτουμε συσχετίσεις επικοινωνίας

5.4.8.4 Περίληψη

Σημειολογία

Τα διαγράμματα ανάπτυξης δείχνουν σύνθεση επεξεργασίας στοιχείων σε χρόνο εκτέλεσης και τα συστατικά λογισμικού, διαδικασίες κι αντικείμενα που ζουν σε αυτά. Τα στιγμιότυπα των συστατικών του λογισμικού αντιπροσωπεύουν εκδηλώσεις σε χρόνο εκτέλεσης των μονάδων του κώδικα. Τα συστατικά που δεν υπάρχουν ως οντότητες χρονικής ακολουθίας (επειδή έχουν μεταγλωττιστεί) δεν εμφανίζονται σε αυτά τα διαγράμματα. Θα πρέπει να εμφανίζονται σε διαγράμματα συστατικών.

Συμβολισμός

Ένα διάγραμμα ανάπτυξης είναι ένα γράφημα κόμβων συνδεδεμένο από διασυνδέσεις επικοινωνίας. Οι κόμβοι μπορεί να περιέχουν στιγμιότυπα συστατικών. Αυτό αποδεικνύει ότι το αντικείμενο είναι μέρος του συστατικού. Τα συστατικά είναι συνδεδεμένα με άλλα συστατικά με διακεκομμένα βέλη εξάρτησης (πιθανόν μέσω διασυνδέσεων). Αυτό δείχνει και ότι ένα συστατικό χρησιμοποιεί τις υπηρεσίες ενός

άλλου συστατικού. Ένα στερεότυπο μπορεί να χρησιμοποιηθεί για να δείξει την ακριβή εξάρτηση αν χρειαστεί.
Ο τύπος του διαγράμματος ανάπτυξης μπορεί ακόμη να χρησιμοποιηθεί για να δείξει ποια συστατικά μπορούν να τρέξουν πάνω σε ποιους κόμβους, χρησιμοποιώντας διακεκομμένα βέλη με στερεότυπη «υποστήριξη».
Η μετάβαση των συστατικών από κόμβο σε κόμβο ή αντικειμένων από συστατικό σε συστατικό μπορεί να δειχθεί χρησιμοποιώντας το στερεότυπο «κατεστημένο» μιας συσχέτισης εξάρτησης. Σε αυτήν την περίπτωση το συστατικό ή αντικείμενο εδρεύει στον κόμβο του ή το συστατικό μόνο ως μέρος όλου του χρόνου.

5.5 ΔΙΑΘΕΣΙΜΑ ΕΡΓΑΛΕΙΑ ΣΧΕΔΙΑΣΜΟΥ ΓΙΑ ΤΗ UML

Υπάρχει στην αγορά ένας αριθμός από διαθέσιμα εργαλεία , οι τιμές τους κυμαίνονται από τα δωρεάν προγράμματα μέχρι προγράμματα αξίας χιλιάδων ευρώ για ατομικές άδειες χρήσης . Το εργαλείο που θα διαλέξει κάποιος να χρησιμοποιήσει για να δουλέψει με τη UML θα εξαρτηθεί από τις ανάγκες του , τις προτιμήσεις του καθώς και τα χρήματα που διαθέτει.

Τα πιο σημαντικά εργαλεία που κυκλοφορούν στην αγορά είναι :

- Visual Paradigm (http://www.visual-paradigm.com/)
- Rational Rose (http://www-306.ibm.com/software/rational/)
- Poseidon (http://www.gentleware.com/index.php)
- ARTiSAN real-time studio (http://www.artisansw.com/)
- Microsoft Visio (http://office.microsoft.com/en-_us/FX010857981033.aspx)

Από τα παραπάνω εργαλεία τα τέσσερα πρώτα παράγουν κώδικα, ενώ το Visio κάνει μονάχα αναπαράσταση των διαγραμμάτων.

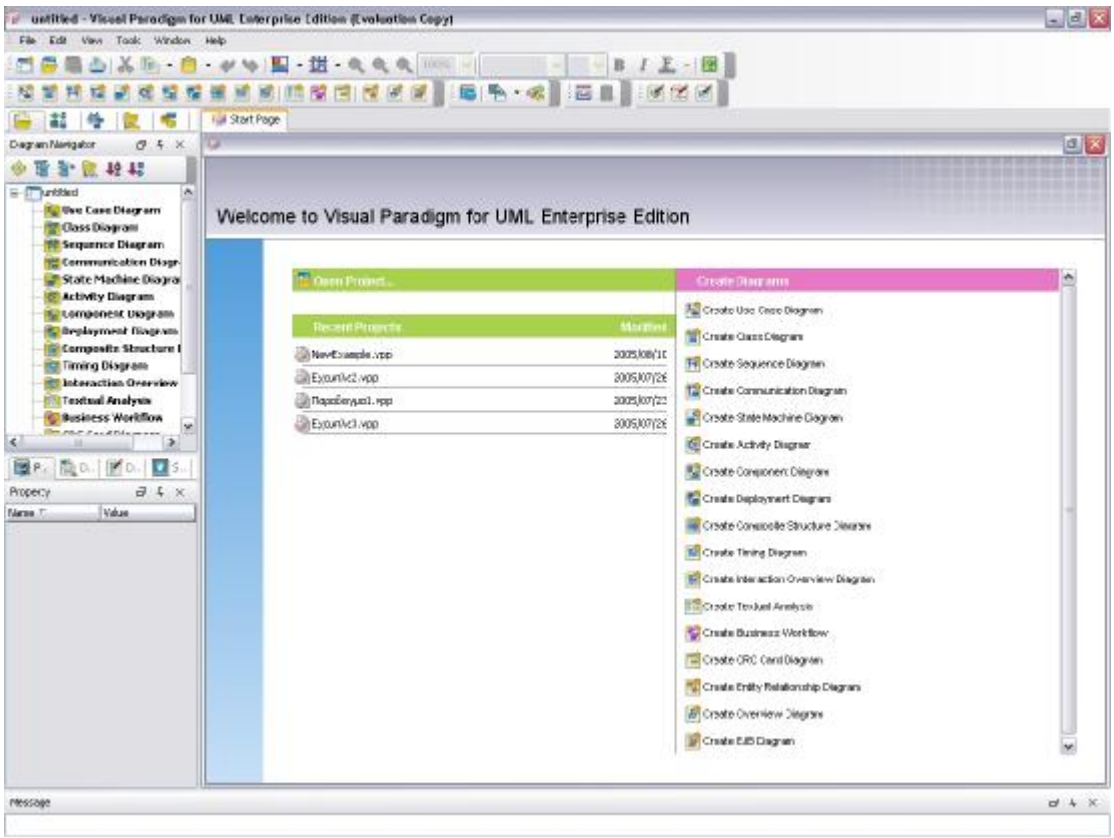
Μπορεί μια εταιρεία να σπαταλήσει χιλιάδες ευρώ στην αγορά προγραμμάτων αλλά και πάλι οι σχεδιαστές θα χρειαστούν τα πιο βασικά, ένα μολυβί και χαρτί. Δεν πρέπει ποτέ να παραβλέπει κανείς την ευκολία της χρήσης αυτών των εργαλείων καθώς και ότι μπορεί να τα έχει διαθέσιμα κάθε στιγμή, οπού και αν βρίσκεται .

5.5.1 Visual Paradigm

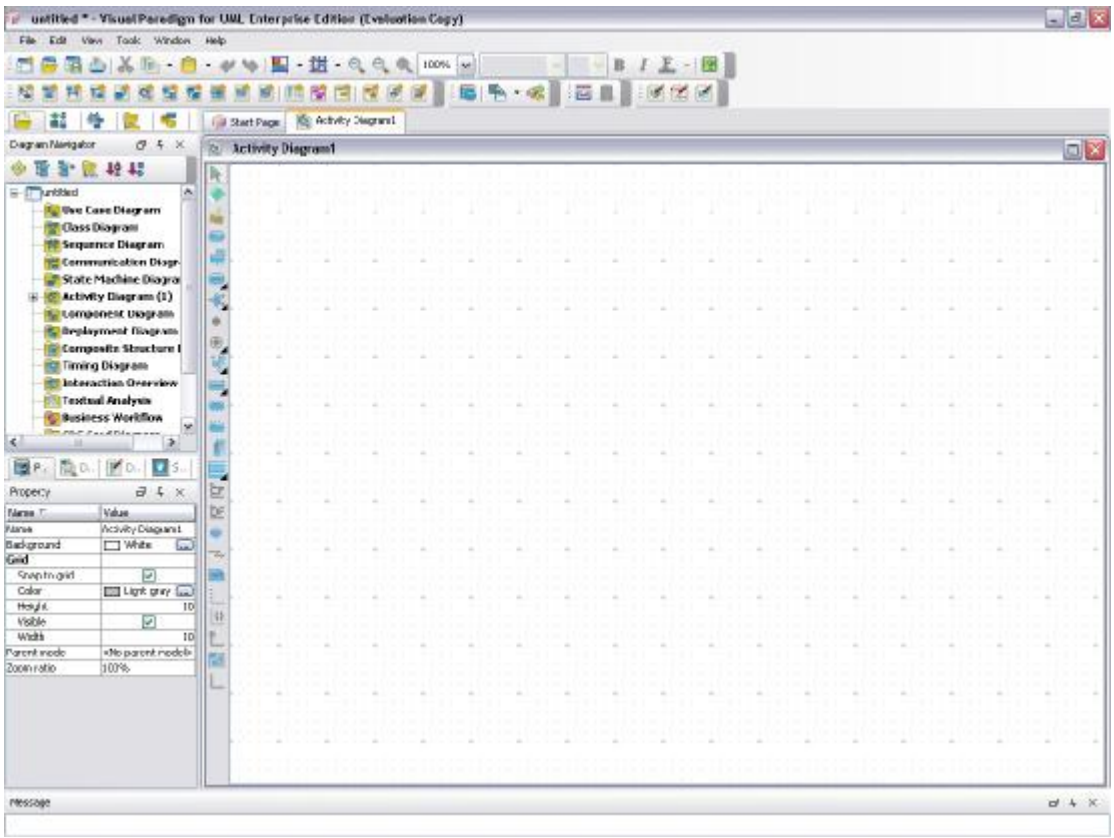
Το Visual Paradigm for Unified Modeling Language (VP-UML) είναι ένα εργαλείο της UML. Το εργαλείο είναι σχεδιασμένο για ένα ευρύ φάσμα χρηστών, συμπεριλαμβανομένων των μηχανικών λογισμικού, αναλυτών συστημάτων, επιχειρησιακών αναλυτών και αρχιτεκτόνων συστημάτων, οι οποίοι ενδιαφέρονται για να χτίσουν τα συστήματα λογισμικού μεγάλης κλίμακας αποδοτικά, μέσω της χρήσης της αντικειμενοστρεφούς προσέγγισης. Το VP-UML υποστηρίζει τα πιό πρόσφατα πρότυπα της Java και της σημειολογίας της UML και παρέχει την πλήρη παραγωγή κώδικα (code generation) της βιομηχανίας και την υποστήριξη αντίστροφης εφαρμοσμένης μηχανικής κώδικα (code reverse engineering) για την Java.

Επιπλέον, το VP-UML είναι ενσωματωμένο με τα Eclipse/IBM WebSphere®, Borland® JBuilder®, NetBeans IDE/Sun™ ONE, IntelliJ IDEA™, Oracle JDeveloper και BEA WebLogic Workshop™ για να υποστηρίξει τη φάση εφαρμογής της ανάπτυξης λογισμικού. Οι μεταβάσεις από την ανάλυση στο σχεδιασμό και έπειτα στην υλοποίηση είναι ενσωματωμένες μέσα στο εργαλείο, κατά συνέπεια μειώνοντας σημαντικά τις προσπάθειες σε όλα τα στάδια του κύκλου ζωής ανάπτυξης λογισμικού.

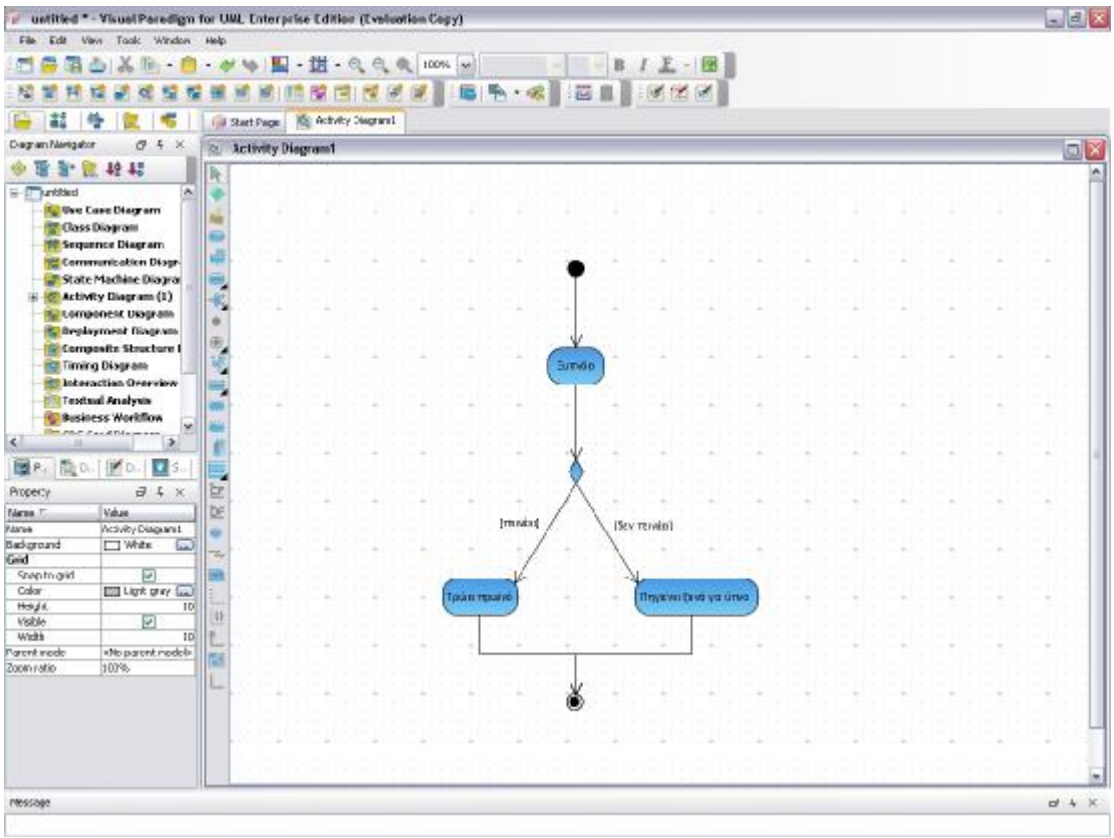
Παρακάτω απεικονίζονται μερικές εικόνες (screenshots) από το περιβάλλον του Visual Paradigm, από την έναρξη της εφαρμογής έως την τελική μορφή ενός διαγράμματος δραστηριότητας.



Σχήμα 5.5.1-1 : Εισαγωγική σελίδα του Visual Paradigm



Σχήμα 5.5.1-2 : Σελίδα δημιουργίας ενός διαγράμματος δραστηριότητας



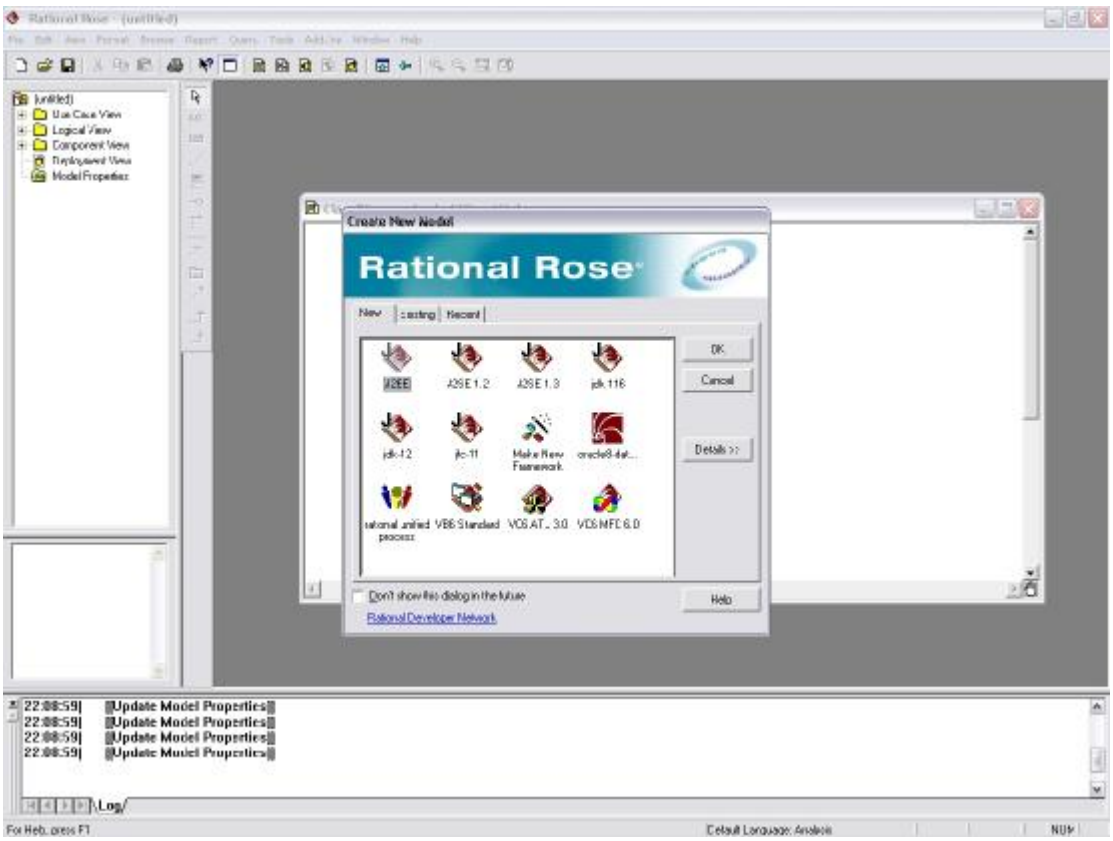
Σχήμα 5.5.1-3 : Ένα σχεδιάγραμμα δραστηριότητας στο Visual Paradigm

5.5.2 Rational Rose

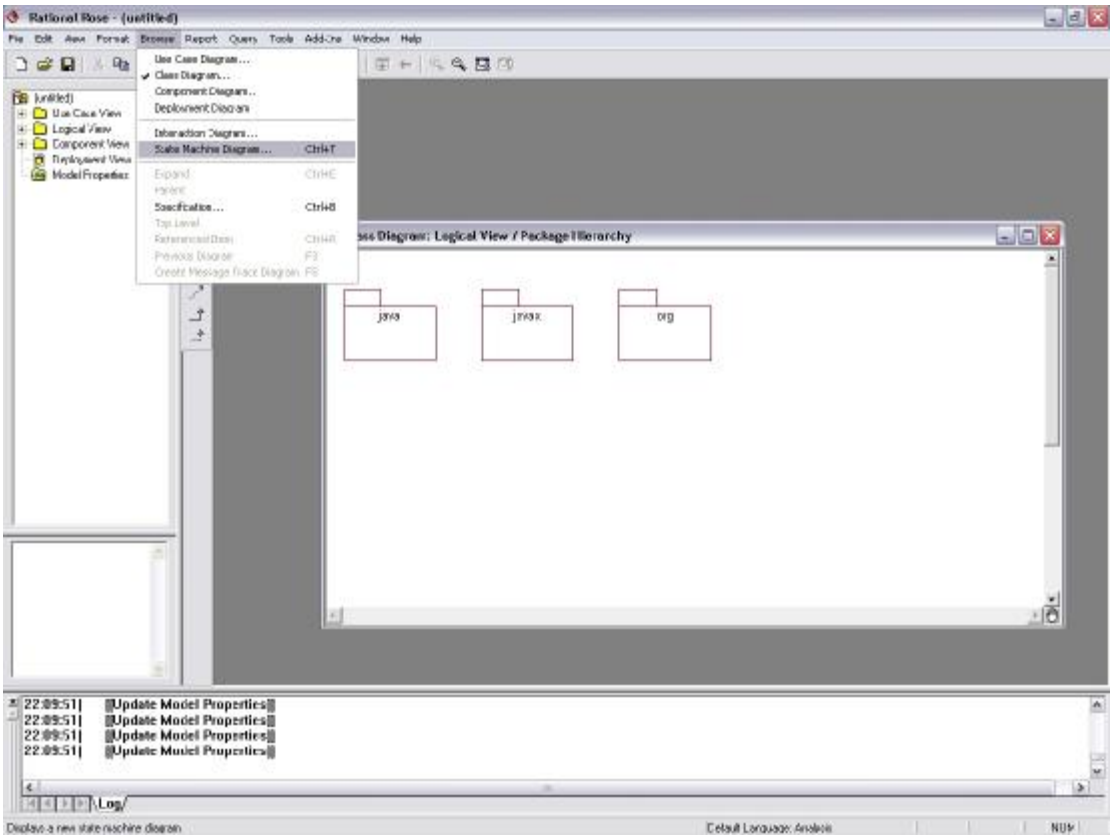
Τον Οκτώβριο του 2004, τα IBM Rational Software Architect and IBM Rational Software Modeler εισήχθησαν ως νέα προϊόντα διαμόρφωσης υπέρ των πιό πρόσφατων προτύπων συμπεριλαμβανομένης της UML 2.0.
Το Rational Rose περιλαμβάνει:

- Αξιόπιστη λύση ανάπτυξης μοντέλων
- Πλήρως αυτοματοποιημένη παραγωγή κώδικα για τη Java, C και C++.
- Εκτέλεση μοντέλου runtime, πλήρως εκτελέσιμη παραγωγή κώδικα και οπτική διόρθωση (visual debugging)
- Αυτόματα δημιουργεί τους οδηγούς (drivers) και τα πραγματικά χειρόγραφα δοκιμής (test scripts)
- Βελτιστοποιημένο για τις event-driven , ταυτόχρονες (concurrent) και διανεμημένες (distributed) εφαρμογές
- Προηγμένα κατασκευάσματα διαμόρφωσης για την κάλυψη των αυστηρών απαιτήσεων για τη λανθάνουσα κατάσταση (latency), τη ρυθμοαπόδοση (throughput), και την αξιοπιστία.
- Σχεδιασμένο για τις πιό τεχνολογικά προκλητικές εφαρμογές
- Πραγματικού μεγέθους μοντελοποιημένη ανάπτυξη με την Java, τη C και τη C++

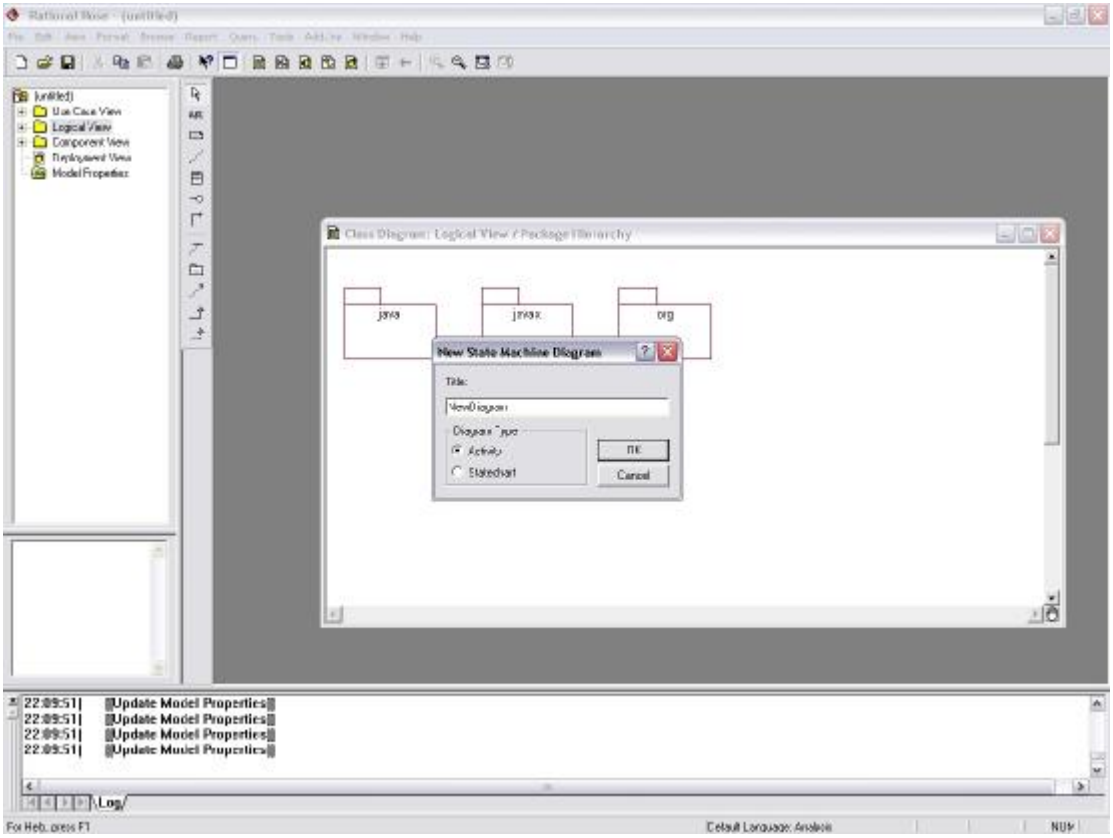
Παρακάτω απεικονίζονται μερικές εικόνες (screenshots) από το περιβάλλον του Rational Rose, από την έναρξη της εφαρμογής έως την τελική μορφή ενός διαγράμματος δραστηριότητας.



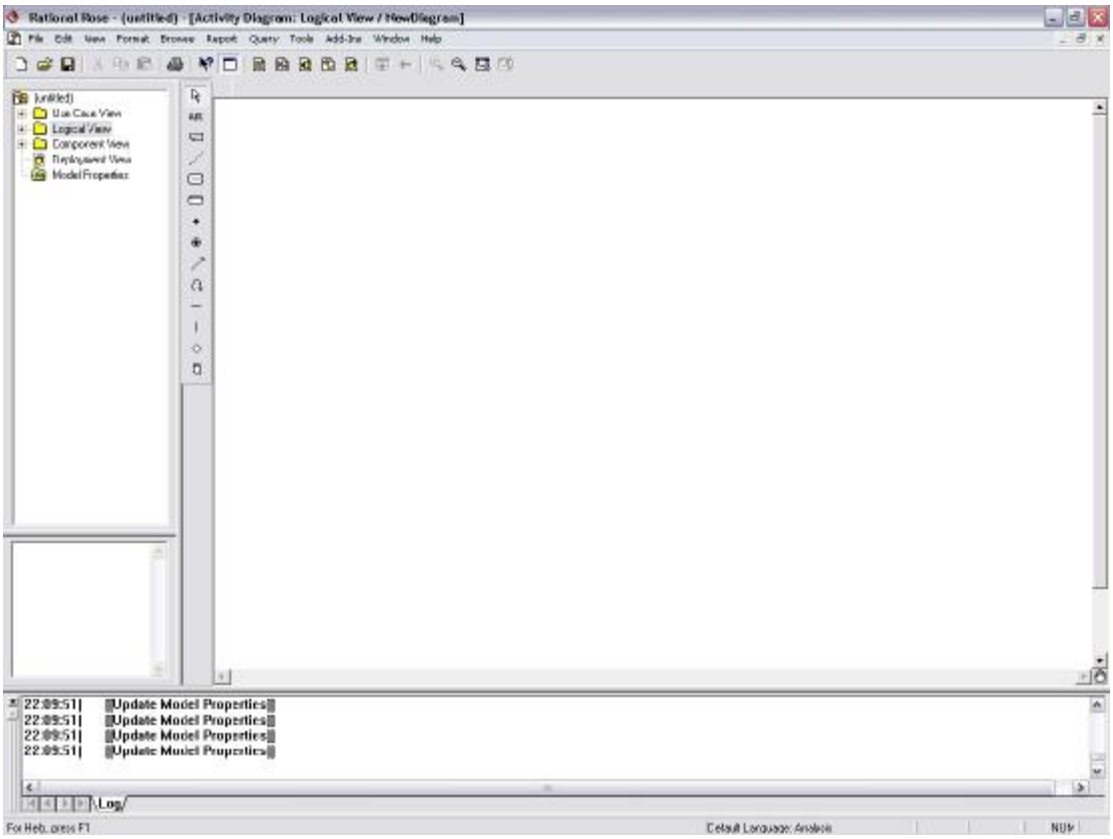
Σχήμα 5.5.2-1: Εισαγωγική σελίδα του Rational Rose



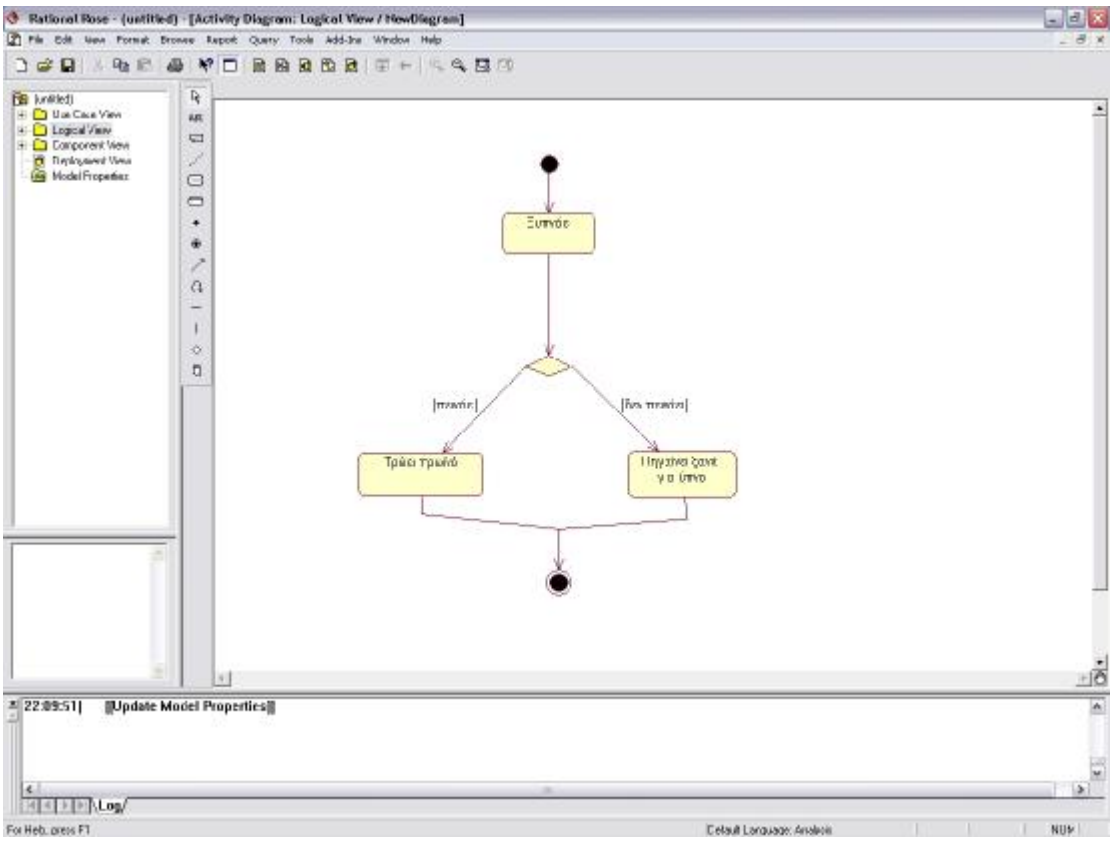
Σχήμα 5.5.2-2: Επιλέγοντας το μενού για την επιλογή των διαγραμμάτων δραστηριότητας



Σχήμα 5.5.2-3: Διαλέγοντας τη σχεδίαση διαγράμματος δραστηριότητας



Σχήμα 5.5.2-4: Το περιβάλλον σχεδίασης του διαγράμματος δραστηριότητας



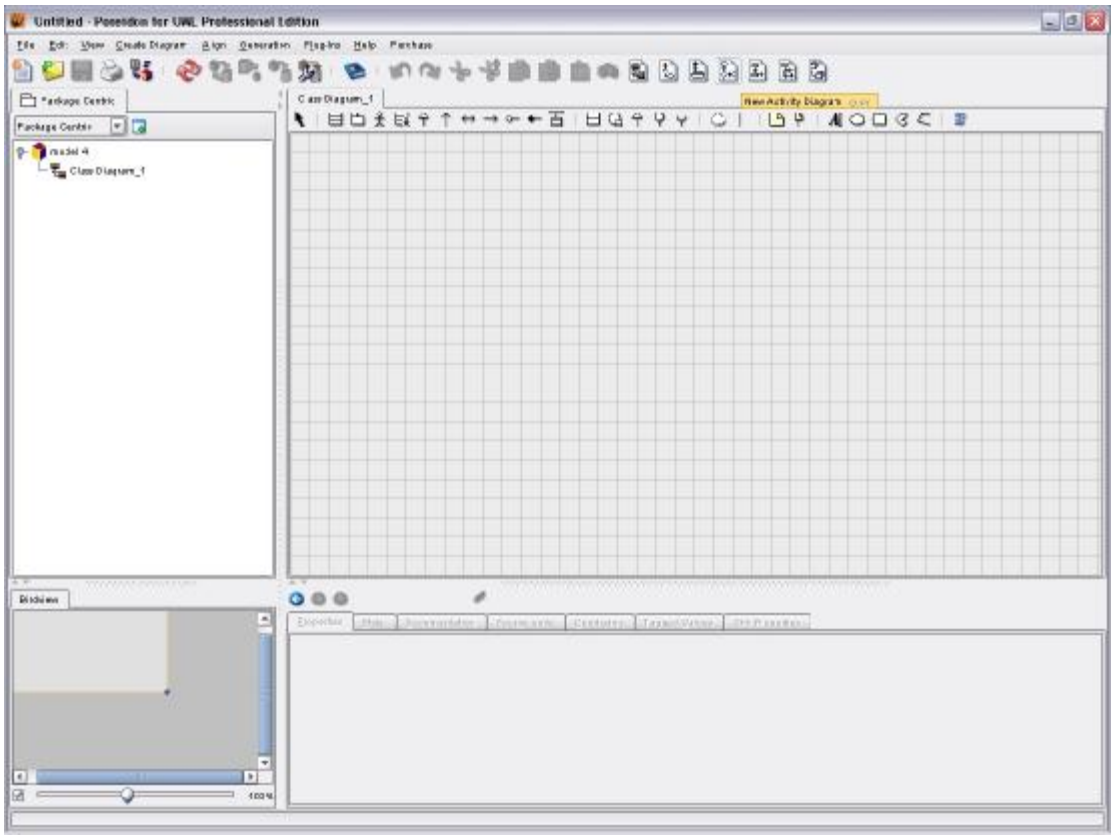
Σχήμα 5.5.2-5: Μετά τη σχεδίαση του διαγράμματος δραστηριότητας

5.5.3 Poseidon

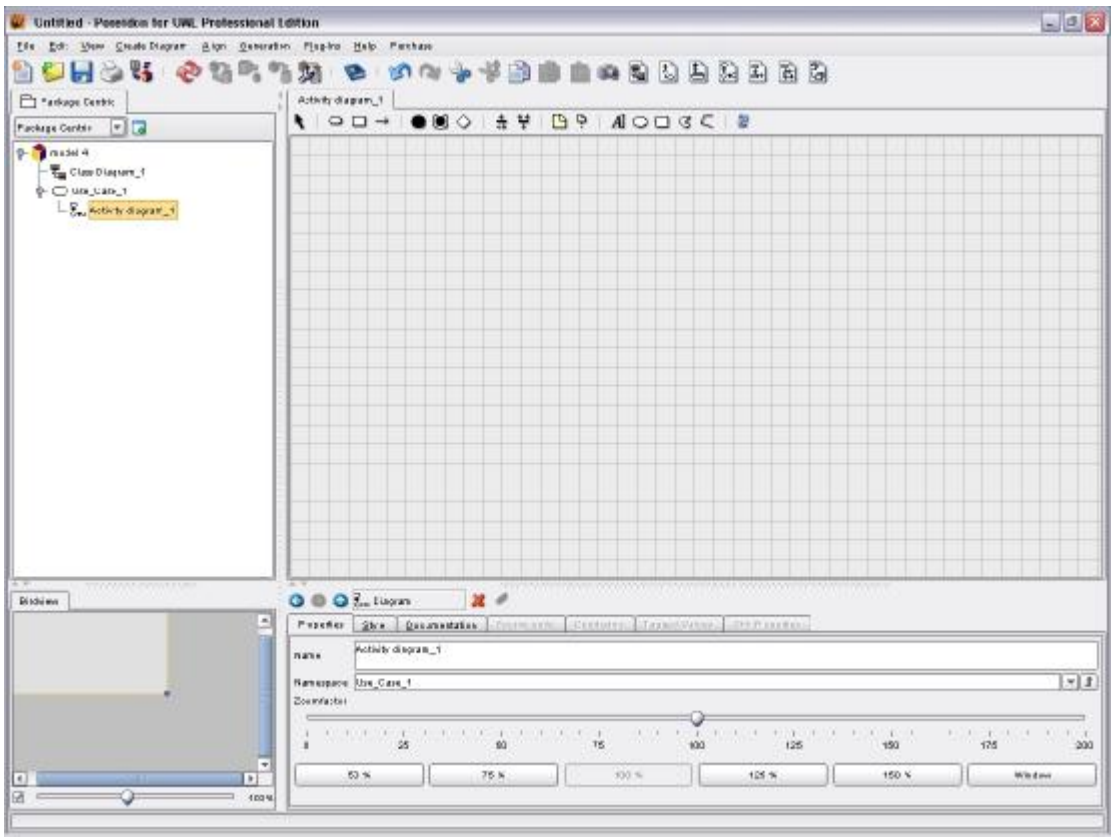
Τα κυριότερα γνωρίσματα του εργαλείου Poseidon for UML είναι τα ακόλουθα:

- Πλήρως κατασκευασμένο σε Java
- Ανεξάρτητο της πλατφόρμας στην οποία θα εγκατασταθεί
- Μηχανική για τη Java
- Παραγωγή κώδικα για: C++, C, COBRA IDL, Delphi, Perl, PHP4, SQL DDL και VB.NET
- Οπτική ενοποίηση με το Eclipse IDE
- Εισαγωγή αρχείων τύπου JAR και MDL
- Δυνατότητα προσωρινής άδειας
- Υποστήριξη από την μονάδα υποστήριξης της Gentleware μέσω email
- Προσαρμοσμένη παραγωγή κώδικα HTML

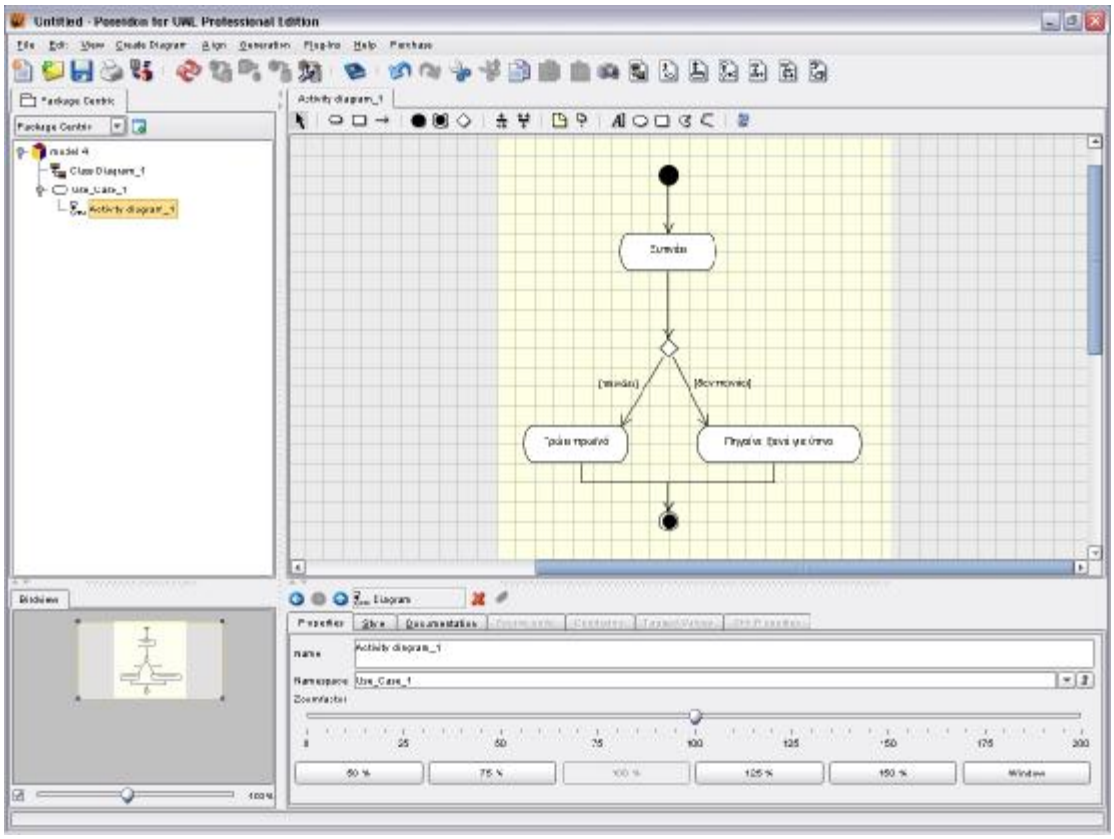
Παρακάτω απεικονίζονται μερικές εικόνες (screenshots) από το περιβάλλον του Poseidon for UML, από την έναρξη της εφαρμογής έως την τελική μορφή ενός διαγράμματος δραστηριότητας.



Σχήμα 5.5.3-1: Επιλέγοντας Νέο διάγραμμα Δραστηριότητας



Σχήμα 5.5.3-2: Σελίδα σχεδίασης του διαγράμματος δραστηριότητας



Σχήμα 5.5.3-3: Μετά τη σχεδίαση του διαγράμματος δραστηριότητας

5.5.4 ARTiSAN real-time studio

Το ARTiSAN real-time studio είναι μια ενσωματωμένη “σουίτα” από εργαλεία μοντελοποίησης για τη UML®, που στοχεύουν να ικανοποιήσουν τις αναπτυξιακές ανάγκες των τεχνικών συστημάτων. Με την προσαρμογή στα πιό πρόσφατα πρότυπα UML 2.0 και SysML, το ARTiSAN real-time studio είναι το εργαλείο για τα μεγάλης κλίμακας συστήματα και την τεχνολογία λογισμικού.

Τα χαρακτηριστικά του εργαλείου είναι τα ακόλουθα:

- Ρυθμιζόμενη αποθήκη πολλών χρηστών με την αποδεδειγμένη ευρωστία και σταθερότητα
- Υποστήριξη εφαρμοσμένης μηχανικής για τις μικτές γλώσσες σε ένα ενιαίο μοντέλο
- Ενσωματωμένη διαχείριση και υποστήριξη έκδοσης για τα βιομηχανικώς τυποποιημένα εργαλεία CM
- Πλήρης διαχείριση απαιτήσεων κύκλου ζωής και ώριμος συγχρονιστής DOORS®
- Προσομοίωση και οπτικοποίηση καταστάσεων με την πλήρη παραγωγή κώδικα
- Υψηλής ποιότητας αναφορές με έναν εύκαμπτο γενικευτή εγγράφων
- Επέκταση μέσω προφίλ και επιλεγμένων στυλ
- Εύχρηστο για τη γρήγορη πλοήγηση μοντέλων
- Υποστήριξη για την ανάπτυξη και τη αρχιτεκτονική των συστατικών της UML 2.0
- Προηγμένα γνωρίσματα της SysML για την εφαρμοσμένη μηχανική συστημάτων
- Εξατομικεύσιμα πρότυπα για τη C, C++, Java, Ada και Spark Ada
- XMI και Rational Rose εισαγωγή/εξαγωγή για ανταλλαγή μοντέλων

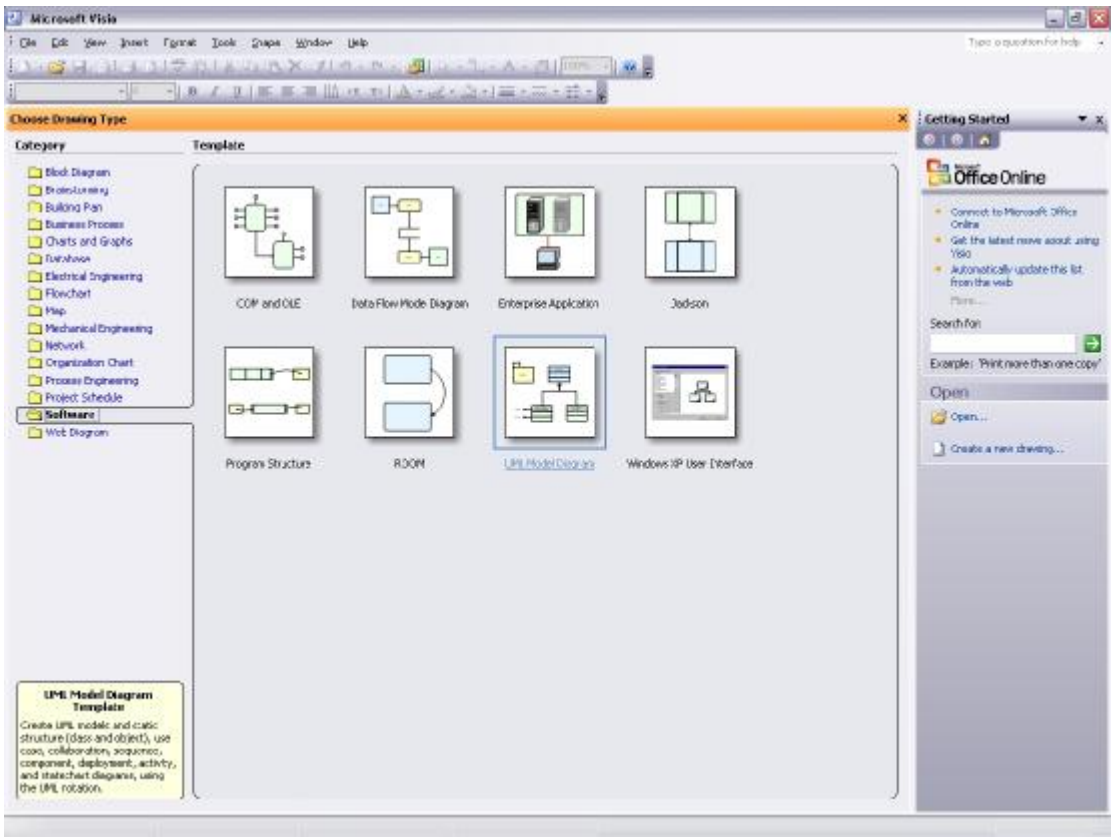
5.5.5 Microsoft Visio

Το Microsoft Visio είναι ένα πρόγραμμα σχεδίασης διαγραμμάτων που μπορεί να σας βοηθήσει να δημιουργήσετε επιχειρησιακά και τεχνικά διαγράμματα που τεκμηριώνουν και οργανώνουν σύνθετες ιδέες, διαδικασίες και συστήματα. Τα διαγράμματα που δημιουργούνται σε Visio μας επιτρέπουν να απεικονίσουν και να μεταβιβάσουν τις πληροφορίες σαφώς, συνοπτικά, και αποτελεσματικά στους τρόπους που το κείμενο και οι αριθμοί δεν μπορούν. Το Visio επίσης αυτοματοποιεί την απεικόνιση στοιχείων με το να συγχρονίζεται άμεσα με τις πηγές στοιχείων για να παρέχει ενημερωμένα διαγράμματα, και μπορεί να προσαρμοστεί για να ικανοποιήσει τις ανάγκες της οργάνωσής σας.

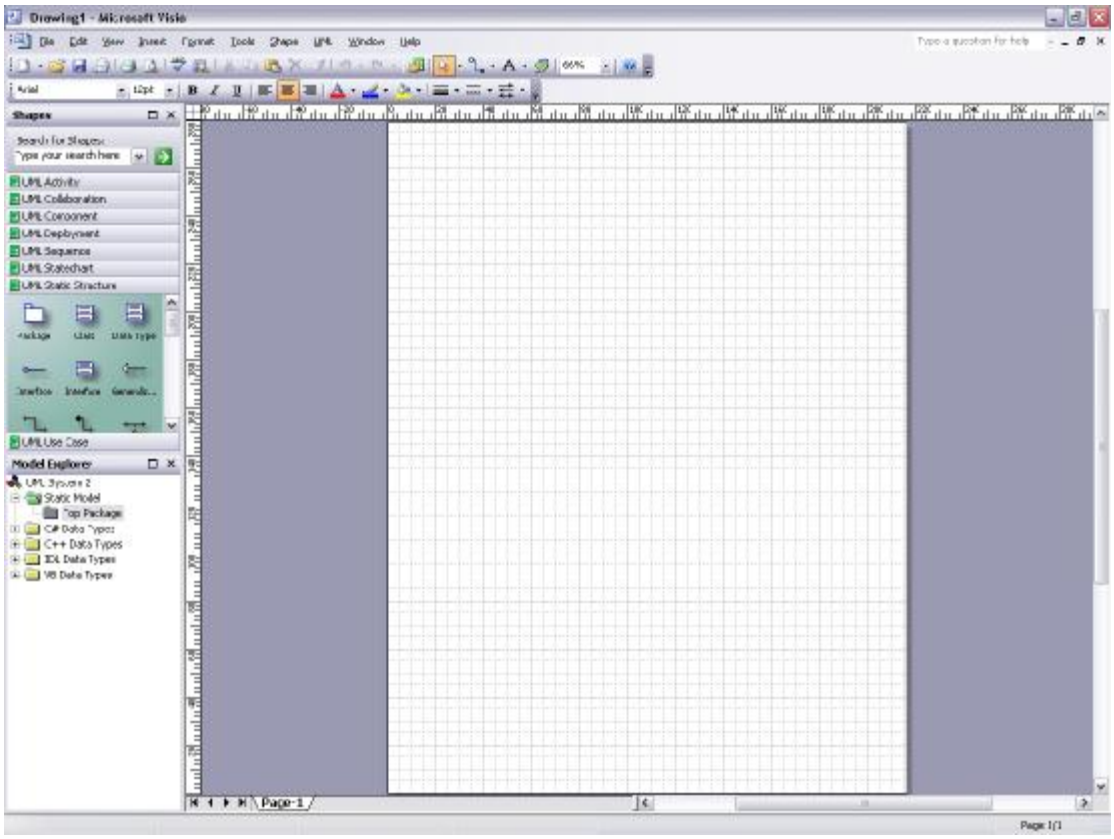
Το Visio έχει τα ακόλουθα κύρια χαρακτηριστικά:

- Εύκολη συγκέντρωση διαγραμμάτων με τη χρήση των προκαθορισμένων συμβόλων Microsoft SmartShapes®
- Εργαλεία χρήσης που σχεδιασμένα για τις συγκεκριμένες επαγγελματικές πειθαρχίες για την επιχείρηση και τις τεχνικές διαγραμματικές απαιτήσεις για όλο τον οργανισμό σας
- Δημιουργήστε οπτικά πλούσια διαγράμματα για το μέγιστο αντίκτυπο στο ακροατήριό σας
- Εισάγετε και εξάγετε διαγράμματα τύπου (SVG), ένα νέο πρότυπο XML για γραφικά του διαδικτύου
- Ενσωματώστε τις επιχειρησιακές διαδικασίες και τα συστήματα με την εξαγωγή των στοιχείων από τα διαγράμματα Visio και την εισαγωγή τους στο Microsoft Word, Microsoft Excel, Microsoft SQL Server™, και XML

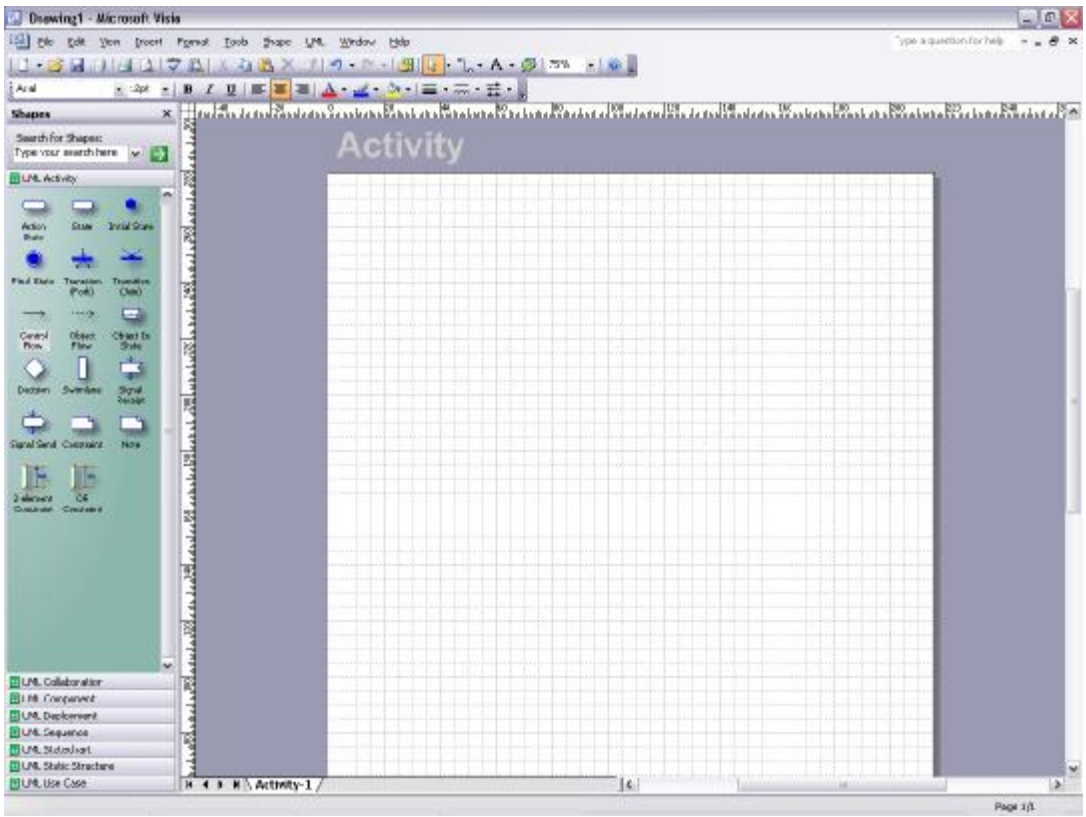
Παρακάτω απεικονίζονται μερικές εικόνες (screenshots) από το περιβάλλον του Microsoft Visio, από την έναρξη της εφαρμογής έως την τελική μορφή ενός διαγράμματος δραστηριότητας.



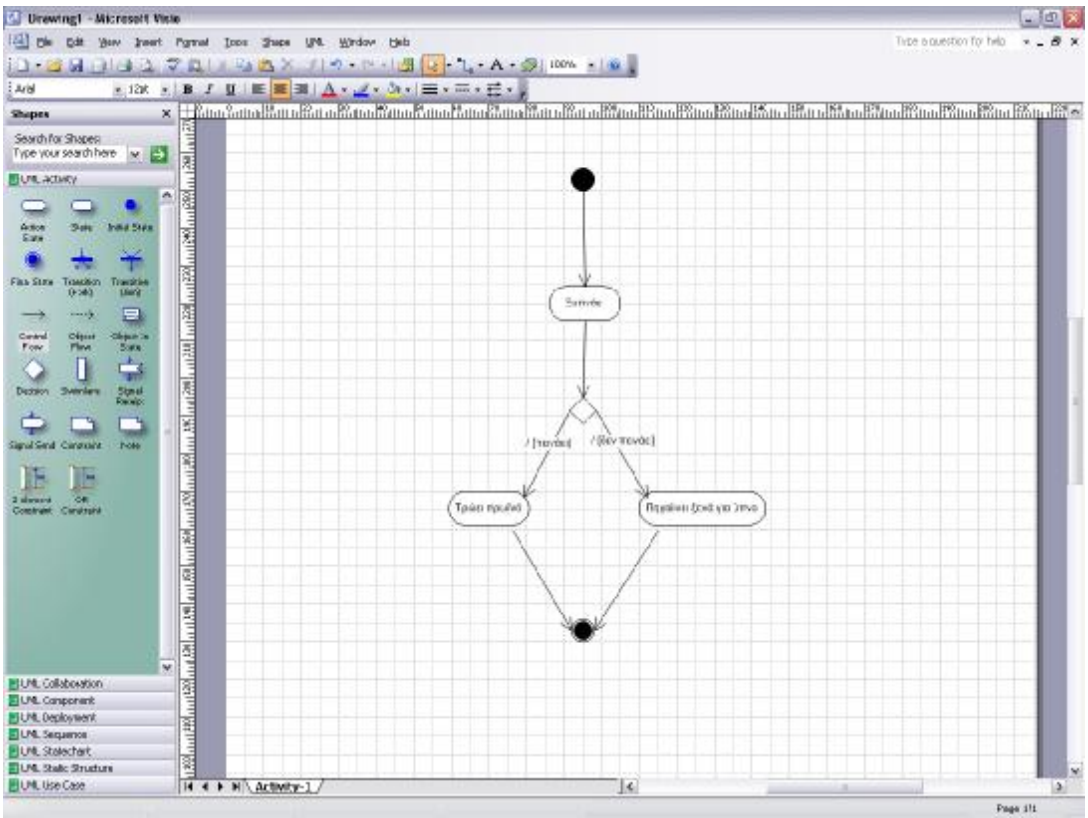
Σχήμα 5.5.5-1: Επιλογή διαγραμμάτων UML από την εισαγωγική σελίδα του Microsoft Visio



Σχήμα 5.5.5-2: Επιλογή διαγραμμάτων δραστηριότητας



Σχήμα 5.5.5-3: Σελίδα σχεδίασης διαγραμμάτων δραστηριότητας



Σχήμα 5.5.5-4: Η τελική μορφή του διαγράμματος δραστηριότητας

5.6 ΜΕΘΟΔΟΛΟΓΙΕΣ ΑΝΑΠΤΥΞΗΣ ΜΕ ΒΑΣΗ ΤΗ UML

Η UML είναι ένα θαυμάσιο εργαλείο αλλά δεν χρησιμοποιείται μεμονωμένα. Χρησιμοποιείται για να διευκολύνει την ανάπτυξη του λογισμικού. Σε αυτό το κεφάλαιο θα μάθουμε για τις διαδικασίες ανάπτυξης και τις μεθοδολογίες σαν ένα μέσο για την κατανόηση της UML σε ανάλογο πλαίσιο.

Φανταστείτε αυτή την κατάσταση: Η οργάνωσή σας χρειάζεται ένα καινούργιο σύστημα υπολογιστών. Καινούργιο υλικό και λογισμικό θα είναι ένα ανταγωνιστικό πλεονέκτημα και θέλετε αυτό το πλεονέκτημα. Πρέπει να αρχίσει η ανάπτυξη και μάλιστα σύντομα.

Είστε αυτός που αποφάσισε να χτίσει το νέο σύστημα. Κάνατε μια ομάδα ανάπτυξης, αποτελούμενη από διευθυντή έργου, προγραμματιστές, αναλυτές, προγραμματιστές και μηχανικούς συστήματος.

Είστε, με άλλα λόγια, ένας πελάτης. Τι προϊόντα εργασίας περιμένετε να δείτε από την ομάδα; Πώς θέλετε να σας δίνει αναφορά ο διευθυντής έργου; Στο τέλος, βέβαια, θα θέλετε να δείτε το σύστημα να δουλεύει. Πριν από αυτό, θέλετε ενδείξεις ότι η ομάδα καταλαβαίνει το πρόβλημα που προσπαθείτε να λύσετε και αντιλαμβάνεται ξεκάθαρα τη δική σας οπτική γωνία όσον αφορά την επίλυση. Θα θέλετε να ρίξετε μια ματιά στην πρόοδο της επίλυσης και κατά πόσο η ομάδα απέχει από αυτήν. Αυτές είναι κοινές ανησυχίες για οποιοδήποτε πελάτη και για οποιοδήποτε σύστημα ανάπτυξης που απαιτεί μια σεβαστή ποσότητα χρόνου, χρημάτων και ανθρώπινου δυναμικού.

5.6.1 Παλιές και νέες μεθοδολογίες

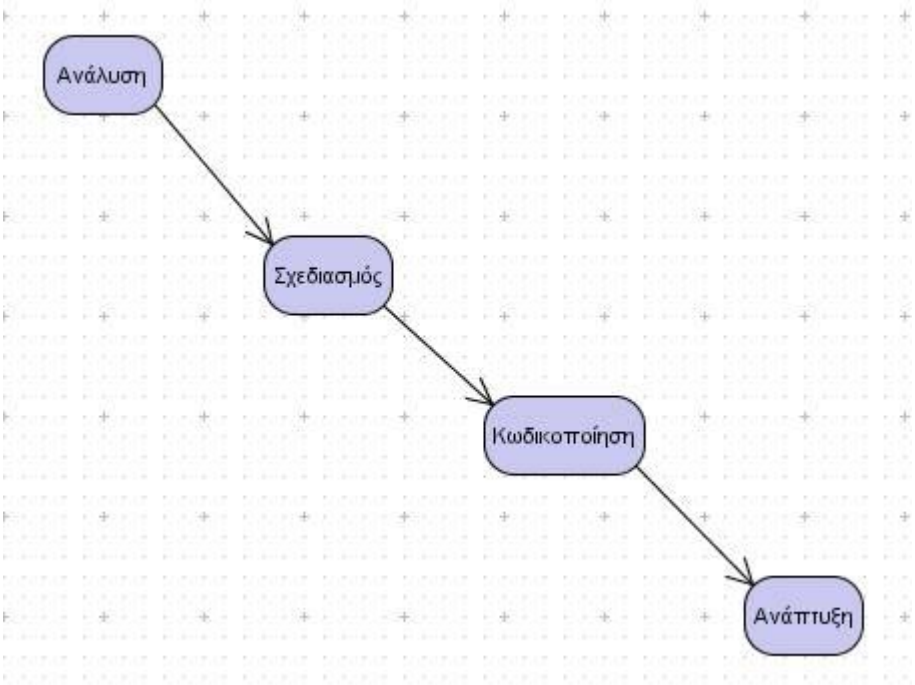
Δε θα θέλετε η ομάδα ανάπτυξης να βιαστεί και να ξεκινήσει να κωδικοποιεί. Στο κάτω κάτω, τι θα κωδικοποιήσει; Η ομάδα θα πρέπει να προχωρήσει με ένα δομημένο και μεθοδικό τρόπο. Η δομή και η φύση των βημάτων σε μια προσπάθεια ανάπτυξης είναι αυτό που ονομάζουμε μεθοδολογία.

Πριν ξεκινήσουν τον προγραμματισμό, πρέπει να έχουν αντιληφθεί πλήρως το πρόβλημα. Αυτό προϋποθέτει ότι κάποιος θα αναλύει τις ανάγκες και τις απαιτήσεις σας. Αφού γίνει αυτή η ανάλυση, μπορεί να αρχίσει η κωδικοποίηση; Όχι. Κάποιος πρέπει να μετατρέψει την ανάλυση σε σχέδιο. Οι κωδικοποιητές δουλεύουν από το σχέδιο για να παράγουν κώδικα, που ύστερα από τεστ και ανάπτυξη, γίνεται σύστημα.

5.6.1.1 Ο παλιός τρόπος

Αυτή η πάρα πολύ απλουστευμένη ματιά σε μια προσπάθεια ακολουθίας των τμημάτων, ίσως σας δώσει μια ιδέα ότι τα τμήματα πρέπει να συμβαίνουν σε αυστηρά καθορισμένα χρονικά σημεία, το ένα μετά το άλλο. Στην πραγματικότητα, οι πρώτες μεθοδολογίες ήταν δομημένες κατά αυτόν τον τρόπο. Το σχήμα δείχνει ένα τρόπο

σκέψης που επηρέασε πολύ για πολλά χρόνια. Με την ονομασία μεθοδολογία *καταρράκτης*, δείχνει ότι η ανάλυση, ο σχεδιασμός, η κωδικοποίηση και ανάπτυξη ακολουθούν το ένα το άλλο σαν δραστηριότητες σε ένα διάγραμμα δραστηριοτήτων. Μόνο όταν το ένα ολοκληρώνεται ξεκινάει το άλλο.



Σχήμα 5.6.1.1-1: Ο παλιός τρόπος

Αυτός ο τρόπος ακολουθείται από δυσοίωνους συσχετισμούς. Κατ' αρχήν, ενθαρρύνει τη διαίρεση μιας προσπάθειας. Εάν ένας αναλυτής παραδώσει μια ανάλυση σε ένα σχεδιαστή, ο οποίος θα την παραδώσει σε έναν προγραμματιστή, το πιθανότερο είναι ότι σπανίως θα δουλεύουν μαζί τα μέλη της ομάδας και σπανίως θα μοιραστούν τις αντιλήψεις τους.

Ένα άλλο πρόβλημα με αυτή τη μέθοδο είναι ότι ελαχιστοποιεί τον αντίκτυπο της κατανόησης που έχει κερδιθεί κατά τη διάρκεια του έργου. Αν η διεργασία δεν μπορεί να επιστρέψει και να ξαναεπισκεφθεί παλαιότερα στάδια, είναι πιθανόν ότι κάποιες εξελισσόμενες ιδέες δε θα αξιοποιηθούν. Η προσπάθεια να στριμώξετε νέες αντιλήψεις σε ένα έργο κατά τη διάρκεια της ανάπτυξης είναι στην καλύτερη περίπτωση δύσκολο. Θα έχετε μεγαλύτερη πιθανότητα επιτυχίας αν ξαναδείτε μια ανάλυση και ένα σχέδιο και μετά ενσωματώσετε μια εξελισσόμενη ιδέα.

5.6.1.2 Ένας νέος τρόπος

Σε αντίθεση με τη μέθοδο του καταρράκτη, η σύγχρονη μηχανική του λογισμικού προσδίδει έμφαση στη διαρκή αλληλεπίδραση ανάμεσα στα στάδια ανάπτυξης. Οι αναλυτές και οι σχεδιαστές, για παράδειγμα, κάνουν συνεχείς επεξεργασίες για να εξελίξουν στέρεα θεμέλια για τους προγραμματιστές. Οι προγραμματιστές, με τη σειρά τους, συνεργάζονται με τους αναλυτές και τους σχεδιαστές για να μοιραστούν τις αντιλήψεις τους, να διαφοροποιήσουν τα σχέδια τους και να ενδυναμώσουν τον κώδικά τους.

Το πλεονέκτημα είναι ότι ενώ διευρύνεται η κατανόηση, η ομάδα ενσωματώνει νέες ιδέες και χτίζει ένα πιο δυνατό σύστημα. Η αρνητική πλευρά (αν υπάρχει κάποια) είναι ότι σε μερικούς ανθρώπους αρέσει η διακοπή των εργασιών και θέλουν να βλέπουν τα ενδιάμεσα στάδια να φτάνουν σε ένα διακριτό τέλος. Μερικές φορές, οι διευθυντές έργου θέλουν να μπορούν να πουν στους πελάτες κάτι όπως, «Η ανάλυση ολοκληρώθηκε και προχωράμε στο σχεδιασμό. Δύο ή τρεις μέρες για το σχεδιασμό και θα ξεκινήσουμε την κωδικοποίηση».

Αυτή η νοοτροπία είναι γεμάτη κινδύνους. Το «στήσιμο» τεχνητών φραγμών ανάμεσα στα στάδια θα επιδράσει τελικώς σε ένα σύστημα που δε θα κάνει επακριβώς αυτό που ο πελάτης θέλει.

Ο παλιός τρόπος ευνοεί κι άλλο ένα πρόβλημα: είναι συνήθως η περίπτωση όπου οι υποστηρικτές της μεθόδου του καταρράκτη καταμερίζουν τη μερίδα του λέοντος όσον αφορά το χρόνο του έργου στην κωδικοποίηση. Η καθαρή επίδραση αυτού είναι η αφαίρεση πολύτιμου χρόνου από την ανάλυση και το σχεδιασμό.

5.6.2 Τι πρέπει να κάνει μια μεθοδολογία ανάπτυξης

Τα πρώτα χρόνια προγραμματισμού των υπολογιστών, ένα άτομο μπορούσε να αναλύσει ένα πρόβλημα, να βρει μια λύση και να γράψει ένα πρόγραμμα. Τα πρώτα χρόνια της κατασκευής των σπιτιών, ένα άτομο μπορούσε να χτίσει ένα αρκετά εξυπηρετικό σπίτι, επίσης.

Τι γίνεται σήμερα, είναι μια άλλη ιστορία. Με σκοπό να αναπτύξει κανείς τα είδη των περίπλοκων συστημάτων που ο επιχειρηματικός κόσμος απαιτεί, μια ομαδική προσέγγιση είναι απαραίτητη. Διότι η γνώση έχει γίνει τόσο εξειδικευμένη που ένα άτομο δεν μπορεί να ξέρει όλες τις πτυχές της επιχείρησης, να κατανοήσει το πρόβλημα, να σχεδιάσει μια λύση, να μεταφράσει τη λύση σε ένα πρόγραμμα, να αναπτύξει την εκτελούμενη εκδοχή σε υλικό και να βεβαιωθεί ότι όλα τα στοιχεία του υλικού δουλεύουν σωστά μαζί.

Η ομάδα πρέπει να συμπεριλαμβάνει αναλυτές που θα επικοινωνούν με τον πελάτη και θα καταλαβαίνουν το πρόβλημά του, αναλυτές που θα οικοδομούν μια λύση, προγραμματιστές που θα κωδικοποιούν τη λύση και μηχανικούς του συστήματος που θα εξελίσσουν τη λύση. Μια διεργασία ανάπτυξης πρέπει να λάβει υπόψη της όλους αυτούς τους ρόλους, να τους αξιοποιήσει σωστά και να κατανέμει τον κατάλληλο χρόνο για κάθε στάδιο της προσπάθειας. Η διεργασία πρέπει, επίσης, να επιδράσει σε έναν αριθμό προϊόντων της δουλειάς που υποδεικνύουν πρόοδο και σχηματίζουν ευθύνες.

Τέλος, η διεργασία πρέπει να διασφαλίζει ότι τα στάδια της προσπάθειας δεν είναι διακριτά. Αντιθέτως, η αναθεώρηση πρέπει να λαμβάνει χώρα ανάμεσα στα στάδια για να τροφοδοτεί δημιουργικότητα και να αυξάνει την ευκολία χτισίματος νέων ιδεών μέσα στην προσπάθεια. Με δυο λόγια: Είναι ευκολότερο να γίνουν αλλαγές σε ένα προσχέδιο και μετά να κάνουμε τις αλλαγές στο σπίτι, παρά να αλλάξουμε το σπίτι ενώ χτίζουμε τη φυσική δομή.

Φτάνοντας σε μια διεργασία, ο πειρασμός είναι να χτίσετε ένα σύνολο από στάδια που θα καταλήγουν σε απίστευτη γραφειοκρατία. Μερικές διαθέσιμες μεθοδολογίες το κάνουν αυτό, αφήνοντας έτσι τους διευθυντές έργου να συμπληρώνουν ατελείωτες φόρμες. Η γραφειοκρατία από μόνη της οδηγεί στο τέλμα.

Ένας λόγος για αυτή τη λανθασμένη πορεία είναι ότι υπάρχει μια μεθοδολογία για όλα. Κάθε οργάνωση είναι μοναδική. Κάθε οργάνωση έχει τη δική της κουλτούρα, κριτήρια, ιστορία και ανθρώπους. Μια μεθοδολογία που ταιριάζει σε μια πολυεθνική κοινοπραξία επιχειρήσεων πιθανότατα θα είναι μια αποτυχία για μια μικρή επιχείρηση και αντιστρόφως. Προσπαθώντας να προσαρμόσετε μια μεθοδολογία για να ταιριάξει σε μια οργάνωση, η παρανόηση είναι ότι ο μεγάλος αριθμός εγγράφων θα βοηθήσει κατά κάποιο τρόπο.

Κι εδώ είναι η πρόκληση. Μια διεργασία ανάπτυξης πρέπει:

- Να διασφαλίζει ότι η ομάδα ανάπτυξης έχει κατανοήσει ξεκάθαρα το πρόβλημα που προσπαθεί να επιλύσει.
- Να επιτρέπει τη δημιουργία μιας ομάδας με προκαθορισμένους ρόλους.
- Να ενθαρρύνει την επικοινωνία ανάμεσα στα μέλη της ομάδας που καταλαμβάνουν αυτούς τους ρόλους.
- Να επιτρέπει την αναθεώρηση των σταδίων κατά τη διάρκεια της ανάπτυξης.
- Να αναπτύσσει τα προϊόντα εργασίας που παρέχουν πρόοδο στον πελάτη αλλά να εξαλείφουν την περιττή γραφειοκρατία.
-

Επιθυμητό βεβαίως είναι η διεργασία να παράγει ένα ολοκληρωμένο προϊόν μέσα σε μια μικρή χρονική περίοδο.

5.6.3 Η μεθοδολογία RUP

5.6.3.1 Εισαγωγή

Η ενοποιημένη μέθοδος της Rational (Rational Unified Process) είναι απόγονος της μεθόδου αντικειμένων της Rational (Rational Objective Process). Η οποία ήταν αποτέλεσμα συγχώνευσης των Rational software corporation και Objectory process το 1995.

Η ενοποιημένη μέθοδος της Rational ,που για λογούς συντομίας θα αναφέρεται ως RUP , είναι μια μέθοδος δημιουργίας λογισμικού. Παρέχει μια πειθαρχημένη προσέγγιση στο να ορίζονται στόχοι και ευθύνες μέσα σε ένα οργανισμό που αναπτύσσει μοντέλα. Ο βασικός στόχος είναι η δημιουργία υψηλής ποιότητας λογισμικού σύμφωνο με τις ανάγκες του χρηστή.

Η RUP είναι μια μέθοδος που συνεχώς εξελίσσεται, από την Rational software , και συμπεριλαμβάνεται στη σουίτα με τα εργαλεία ανάπτυξης λογισμικού. έχει ένα περιθώριο διεργασιών που μπορεί να προσαρμοστεί και επεκταθεί για να καλύψει τις ανάγκες κάθε οργανισμού.

Η RUP εμπεριέχει πολλές από τις «καλύτερες πρακτικές» της σύγχρονης ανάπτυξης λογισμικού σε μια μορφή που είναι κατάλληλη για μια ευρεία γκάμα έργων και οργανισμών. Πιο συγκεκριμένα οι πρακτικές αυτές είναι :

- Ανάπτυξη λογισμικού με την επαναληπτική μέθοδο.
- Διαχείριση απαιτήσεων .
- Χρήση αρχιτεκτονικών βασισμένων σε συστατικά.
- Οπτική μοντελοποίησης λογισμικού.
- Επαλήθευση ποιότητας λογισμικού.
- Έλεγχος των αλλαγών στο λογισμικό.
-

Παρακάτω θα αναλυθούν περισσότερο τα πιο βασικά στάδια από τη μεθοδολογία αυτή.

5.6.3.2 Ανάπτυξη λογισμικού με την επαναληπτική μέθοδο

Η προσέγγιση που ακολουθείται από την RUP είναι γενικώς ανώτερη από τις άλλες γνωστές μεθόδους, την γραμμική και του καταρράκτη, για διαφόρους λογούς όπως :

- Επιτρέπει να λάβουμε υπόψη μας τις αλλαγές των απαιτήσεων. Γιατί είναι γεγονός ότι οι απαιτήσεις συνήθως αλλάζουν.
- Στην RUP η ενσωμάτωση (integration) δεν είναι μια μεγάλη έκρηξη στο τέλος του έργου. Άντυτου τα στοιχειά ενσωματώνονται προοδευτικά στο σύνολο. Η επαναληπτική προσέγγιση είναι μια διαδικασία συνεχούς ενσωμάτωσης.
- Η επαναληπτική προσέγγιση μετριάζει εκ των προτέρων τα ρίσκα επειδή η ενσωμάτωση είναι γενικά η μονή περίπτωση στη οποία τα ρίσκα ανακαλύπτονται.
- Παρέχει στο μάνατζμεντ ένα μέσο για την εφαρμογή τακτικών αλλαγών στο προϊόν.
- Διευκολύνει την επαναχρησιμοποίηση επειδή είναι ευκολότερο να αναγνωριστούν κοινά μέρη όπως αυτά σχεδιάζονται η υλοποιούνται αντί να αναγνωριστούν όλα τα κοινά μέρη στην αρχή.
- Έχει ως αποτέλεσμα μια πιο γερή αρχιτεκτονική επειδή τα λάθη διορθώνονται υστέρα από αρκετές επαναλήψεις. Τα σφάλματα ανιχνεύονται ακόμα και στις πρώτες επαναλήψεις κατά την διάρκεια που το προϊόν φεύγει από το στάδιο της σύλληψης και εισέρχεται στο στάδιο της επεξεργασίας και όχι σε ένα μαζικό τεστ στο τέλος.
- Οι υπεύθυνοι για την σχεδίαση του έργου μπορούν να μάθουν κατά την πρόοδο του , και οι διαφορές ικανότητες και ειδικεύσεις τους χρησιμοποιούνται πλήρως καθ'όλη την διάρκεια του.
- Η διαδικασία ανάπτυξης μπορεί να βελτιωθεί στην πορεία. Η αξιολόγηση στο τέλος μιας επανάληψης αναλύει τη θα έπρεπε να αλλαχθεί στην οργάνωση και στην ιδία την διαδικασία για να αποδώσει καλυτέρα στην επόμενη επανάληψη

Οι διαχειριστές του έργου συχνά αντιστέκονται στην επαναληπτική προσέγγιση, θεωρώντας την ανεξέλεγκτη. Στην RUP η επαναληπτική προσέγγιση είναι πλήρως ελεγχόμενη, οι επαναλήψεις καταμερίζονται σε ποσότητα , διάρκεια και στόχους.

5.6.3.3 Διαχείριση απαιτήσεων

Η διαχείριση των απαιτήσεων είναι μια συστηματική προσέγγιση στην οργάνωση, την επικοινωνία, και τις αλλαγές σε ένα σύστημα λογισμικού η μια εφαρμογή.
Τα βασικά κέρδη από την αποτελεσματική διαχείριση των απαιτήσεων είναι κυρίως :

- Ο καλύτερος έλεγχος των πολύπλοκων έργων
- Βελτιωμένη ποιότητα λογισμικού και ικανοποίηση του πελάτη
- Μειωμένο κόστος και καθυστέρηση έργου
- Καλύτερη επικοινωνία στον εργασιακό χώρο

5.6.3.4 Αρχιτεκτονική λογισμικού και χρήση συστατικών

Τα σενάρια χρησιμοποιούνται από την RUP σε όλη την διάρκεια ζώνης του έργου. Αλλά οι σχεδιαστικές ενέργειες επικεντρώνονται στην αρχιτεκτονική, είτε συστήματος είτε λογισμικού. Ο κύριος σκοπός των πρωίμων επαναλήψεων των διεργασιών είναι να παράγουν και να επικυρώσουν μια αρχιτεκτονική λογισμικού η οποία θα έχει τη μορφή εκτελέσιμου αρχιτεκτονικού προτύπου, που θα εξελιχθεί στην τελική μορφή του έργου.

Η RUP παρέχει ένα μεθοδικό συστηματικό τρόπο για την σχεδίαση και ανάπτυξη μιας αρχιτεκτονικής. Προσφέρει φόρμες για την περιγραφή μιας αρχιτεκτονικής από την πλευρά των πολλών οπτικών ,αρχιτεκτονικών , όψεων.

Ένα συστατικό λογισμικού μπορεί να οριστεί ως ένα κομμάτι λογισμικού , ένα υποσύστημα , ένα πακέτο που εκπληρώνει μια λειτουργία και έχει καθαρά όρια. Η ανάπτυξη λογισμικού βάση συστατικών μπορεί να έχει διαφορές μορφές.
Η RUP υποστηρίζει την ανάπτυξη λογισμικού βάση συστατικών με διαφόρους τρόπους. :

- Η κατά τμήματα προσέγγιση επιτρέπει στους σχεδιαστές να αναγνωρίζουν προοδευτικά τα συστατικά και να αποφασίζουν ποια θέλουν μεγαλύτερη ανάπτυξη.
- Η στόχευση στην αρχιτεκτονική λογισμικού επιτρέπει την άρθρωση της δομής
- Ιδέες όπως τα πακέτα, τα υποσυστήματα, και τα επίπεδα χρησιμοποιούνται στην ανάλυση και τον σχεδιασμό για την οργάνωση των συστατικών.
- Οι δοκιμές οργανώνονται γύρω από τα συστατικά ένα στην αρχή και σταδιακά επεκτείνεται για να συμπεριλάβει μεγαλύτερα σετ συστατικών.

5.6.3.5 Ποιότητα διεργασιών και προϊόντων

Η ποιότητα στην RUP είναι υπευθυνότητα κάθε μέλους της αναπτυξιακής ομάδας και όχι μονό ενός υπευθύνου. Τα βασικότερα πεδία που αφορούν την ποιότητα στην RUP είναι η ποιότητα του προϊόντος και η ποιότητα των διεργασιών.
Στόχος της RUP είναι να επαληθεύει και αντικειμενικά να προσδιορίζει εάν ένα αντικείμενο συναντά το απαιτούμενο επίπεδο ποιότητας.

5.6.3.6 Μοντελοποίηση

Ένα μεγάλο μέρος της RUP είναι η ανάπτυξη και διατήρηση μοντέλων του συστήματος που αναπτύσσεται. Τα μοντέλα ενός έργου μας βοηθάνε να καταλάβουμε καλυτέρα αλλά και να αντιμετωπίσουμε κάποιο πρόβλημα που εμφανίζεται. Το μοντέλο είναι μια απλοποίηση της πραγματικότητας που επιτρέπει τον έλεγχο ενός κατά τα αλλά μεγάλου και πολύπλοκου συστήματος.

Η UML είναι μια κατά βάση γραφική γλωσσά που μπορεί να εκφράσει διαφορά μοντέλα αλλά δεν λέει πως να αναπτύσσεις λογισμικό, παρέχει το λεξιλόγιο αλλά εσύ πρέπει να δεις πως θα το χρησιμοποιήσεις. Γι'αυτο αναπτύχθηκε η RUP, που στην ουσία είναι ένας οδηγός για την δημιουργία μοντέλων.

5.6.4 Η μεθοδολογία GRAPPLE

Το GRAPPLE περιλαμβάνει πέντε τμήματα. Χρησιμοποιούμε τη λέξη τμήματα και όχι στάδια για να ξεφύγουμε από την ιδέα ότι πρέπει ένα στάδιο να ολοκληρωθεί πριν ξεκινήσει κάποιο άλλο. Κάθε τμήμα με τη σειρά του περιλαμβάνει πολλές δράσεις. Κάθε δράση παράγει ένα προϊόν εργασίας και κάθε δράση είναι υπεύθυνη για ένα συγκεκριμένο παίκτη.

Σε πολλές περιπτώσεις, ο διευθυντής έργου μπορεί να συνδυάσει τα προϊόντα του έργου σε μια αναφορά που παρουσιάζει στον πελάτη. Τα προϊόντα εργασίας εξυπηρετούν τον ίδιο σκοπό όπως μια στοίβα χαρτιού χωρίς να τελματώνεται το έργο σε γραφειοκρατικές διαδικασίες.

Για να προσαρμόσετε το GRAPPLE, ένας διευθυντής έργου μπορεί να προσθέσει δράση σε κάθε τμήμα. Άλλη μια πιθανότητα είναι να διαπεράσει άλλο ένα επίπεδο και να υποδιαιρέσει κάθε δράση σε υποδράση. Οι ανάγκες μιας οργάνωσης θα υποδείξουν την πορεία που θα ακολουθηθεί.

Το GRAPPLE προορίζεται για αντικειμενοστρεφή συστήματα. Έτσι, οι δράσεις μέσα σε ένα τμήμα συνδέονται με παράγωγα προϊόντα εργασίας αντικειμενοστρεφούς φύσης.

Τα τμήματα είναι:

1. Συγκέντρωση απαιτήσεων (Requirements gathering)
2. Ανάλυση (Analysis)
3. Σχεδιασμός (Design)
4. Ανάπτυξη (Development)
5. Εξέλιξη (Deployment)

Το ακρωνύμιο που προκύπτει είναι RADDD ή RAD^3. Μετά το τρίτο τμήμα, ο διευθυντής έργου συνδυάζει τα προϊόντα εργασίας σε ένα σχέδιο που δίνει στον πελάτη και στους προγραμματιστές. Όταν ολοκληρώνονται όλα τα τμήματα του RAD^3, όλα τα προϊόντα εργασίας συνδυάζονται για να σχηματίσουν ένα κείμενο που προσδιορίζει το σύστημα.

Πριν ξεκινήσουν όλα αυτά τα τμήματα, υποθέτετε ότι ο πελάτης έχει σκεφτεί ένα σενάριο για το νέο σύστημα. Υποθέτετε, επίσης, ότι τα μέλη της ομάδας και ειδικότερα οι αναλυτές, έχουν μελετήσει όσο το δυνατόν περισσότερα δεδομένα. Ας εξετάσουμε κάθε τμήμα από πιο κοντά, με μια ματιά προς τα μέρη της UML που ταιριάζουν το ένα μέσα στο άλλο.

5.6.5 Η δομή της GRAPPLE (RAD3)

Για να καλύψουμε την πολυπρόσωπη πρόκληση δημιουργίας μιας διεργασίας ανάπτυξης, παρουσιάζουμε το Guidelines for Rapid APPLication Engineering (GRAPPLE). Οι ιδέες μέσα στο GRAPPLE δεν είναι πρωτότυπες. Είναι η ουσία ανάμεσα σε άλλες ιδέες. Οι εμπνευστές της UML δημιούργησαν ένα Λογικό Ενοποιημένο Σύστημα και πριν από αυτή είχε ο κάθε ένας τη δική του διεργασία. Οι ιδέες σε αυτές τις διαδικασίες είναι παρόμοιες με το GRAPPLE.

Η πρώτη λέξη στο όνομα GRAPPLE, *Guidelines* (κατευθυντήριες γραμμές) είναι σημαντική: Δεν υπάρχει σταθερή μεθοδολογία. Αντιθέτως, πρόκειται για ένα σύνολο ευπροσάρμοστων και προσαρμοστικών ιδεών. Θυμίζει σαν ένα απλοποιημένο σκελετό μιας διεργασίας ανάπτυξης. Την παρουσιάζουμε σαν ένα μέσο για την επίδειξη της UML μέσα σε περιεχόμενο. Με κάποιες προσαρμογές η UML μπορεί να λειτουργήσει σε πολλές οργανώσεις (αλλά ίσως όχι σε όλες). Αφήνει το χώρο σε ένα δημιουργικό διευθυντή έργου να προσθέσει τις δικές του ιδέες όσον αφόρα τη λειτουργία της συγκεκριμένης οργάνωσης και να αφαιρέσει ό,τι δεν του ταιριάζει.

5.6.5.1 Συγκέντρωση απαιτήσεων

Αν είχατε σκοπό προσδώσετε σημαντικότητα σε κάθε τμήμα, αυτό εδώ είναι ένας καλός υποψήφιος για την πρώτη επιλογή. Αν δεν καταλαβαίνετε τι θέλει ο πελάτης, δε θα χτίσετε ποτέ το σωστό σύστημα. Όλοι οι αναλυτές σεναρίου στον κόσμο δε θα βοηθήσουν αν δεν καταλάβετε τις απαιτήσεις του πελάτη και τα προβλήματα που θέλει να λύσετε.

5.6.5.1.1 Ανακαλύψτε τις διεργασίες μιας επιχείρησης

Είναι μια καλή ιδέα να ξεκινήσετε την προσπάθεια ανάπτυξης καταλαβαίνοντας τις διεργασίες της επιχείρησης του πελάτη, ειδικότερα αυτές που προσπαθείτε να ενσωματώσετε στο προτεινόμενο σύστημα. Για να κερδίσει αυτή την γνώση, ένας αναλυτής τυπικά παίρνει συνέντευξη από τον πελάτη ή κάποιο άτομο που γνωρίζει και έχει υποδειχθεί από τον πελάτη και ρωτάει το συνεντευξιαζόμενο τις σχετικές διεργασίες βήμα προς βήμα.

Ένα σημαντικό αποτέλεσμα είναι ότι ο αναλυτής αποκτά ένα λεξιλόγιο με το οποίο μπορεί να δουλέψει σε ένα υποσύνολο της ορολογίας του πελάτη. Ο αναλυτής χρησιμοποιεί το λεξιλόγιο όταν παίρνει συνέντευξη από τον πελάτη στην επόμενη δράση.

Το προϊόν εργασίας για αυτή τη δράση είναι ένα διάγραμμα δραστηριοτήτων ή ένα σύνολο διαγραμμάτων δραστηριοτήτων που αιχμαλωτίζει τα βήματα και τα σημεία αποφάσεων στη διεργασία.

5.6.5.1.2 Εκτέλεση Ανάλυσης Πεδίων Ορισμού

Ο σκοπός είναι να αποκτηθεί όσο πιο ξεκάθαρη αντίληψη γίνεται του πεδίου ορισμού του πελάτη. Σημειώστε ότι αυτή η δράση και η προηγούμενη αφορούν έννοιες, δεν αφορούν ένα σύστημα που πρόκειται να χτίσετε. Ο αναλυτής πρέπει να νοιώθει άνετος στον κόσμο του πελάτη αφού θα είναι ο απεσταλμένος του πελάτη στην ομάδα ανάπτυξης.

Ο αναλυτής παίρνει συνέντευξη από τον πελάτη με σκοπό την κατανόηση κύριων οντοτήτων του πεδίου ορισμού του πελάτη. Κατά τη διάρκεια της συζήτησης ανάμεσα στον πελάτη και τον αναλυτή, άλλο ένα μέλος της ομάδας κρατάει σημειώσεις (σε ένα laptop καλύτερα) και ένας μοντελιστής αντικειμένων χτίζει ένα υψηλού επιπέδου διάγραμμα κλάσης. Αν μπορείτε να έχετε παραπάνω από ένα άτομο της ομάδας για τις σημειώσεις, θα ήταν καλό.

Ο μοντελιστής του αντικειμένου ακούει ουσιαστικά και πιάνει δουλειά μετατρέποντας το κάθε ουσιαστικό που ακούει σε κλάση. Εν τέλει, μερικά ουσιαστικά θα γίνουν ιδιότητες. Ο μοντελιστής ψάχνει, επίσης, ρήματα που θα γίνουν λειτουργίες των κλάσεων. Σε αυτό το σημείο, ένα εργαλείο μοντελισμού του υπολογιστή είναι εξαιρετικά χρήσιμο.

Το προϊόν εργασίας είναι ένα υψηλού επιπέδου διάγραμμα κλάσης και ένα σύνολο σημειώσεων.

5.6.5.1.3 Συνεργαζόμενα Συστήματα

Τα σύγχρονα επιχειρησιακά συστήματα δεν εμφανίζονται μεμονωμένα. Πρέπει να συνεργάζονται με άλλα. Αρχικά στη διαδικασία, η ομάδα ανάπτυξης ανακαλύπτει επακριβώς από ποια συστήματα θα εξαρτάται το νέο σύστημα και ποια συστήματα θα εξαρτώνται από αυτό. Ένας μηχανικός συστήματος φροντίζει για αυτή τη δράση και παράγει ένα διάγραμμα εξέλιξης ως το προϊόν εργασίας. Το διάγραμμα δείχνει το σύστημα ως κόμβο, με γραμμές διακομβικής επικοινωνίας, μονίμων συστατικών στοιχείων και διαμονίμων εξαρτήσεων.

5.6.5.1.4 Ανακαλύψτε τις Απαιτήσεις του Συστήματος

Σε αυτή τη δράση, η ομάδα περνάει την πρώτη της χρονική περίοδο JAD (Joint Application Development). Μια χρονική περίοδος JAD φέρνει κοντά όσους παίρνουν αποφάσεις από την οργάνωση του πελάτη, τους υποψήφιους πελάτες, και τα μέλη της ομάδας. Υπάρχει κάποιος που διευκολύνει και συντονίζει. Η δουλειά του είναι να εξάγει από αυτούς που παίρνουν αποφάσεις και τους χρήστες τι θέλει να κάνει το σύστημα. Τουλάχιστον δύο μέλη της ομάδας πρέπει να κρατάνε σημειώσεις και ο

μοντελιστής του αντικειμένου πρέπει να κάνει πιο ξεκάθαρο το διάγραμμα που παρήχθη προηγουμένως.

Το προϊόν εργασίας είναι ένα διάγραμμα πακέτου. Το κάθε πακέτο αντιπροσωπεύει ένα υψηλού επιπέδου πεδίο λειτουργικότητας του συστήματος (για παράδειγμα, «Βοήθεια με το σέρβις πελατών»). Κάθε πακέτο συγκεντρώνει ένα σύνολο σεναρίων (για παράδειγμα, «Βρες το ιστορικό του πελάτη» και «Συνομίλησε με τον πελάτη»). Η πολυπλοκότητα του συστήματος καθορίζει το μάκρος της χρονικής περιόδου. Δε διαρκεί ποτέ λιγότερο από μισή ημέρα και μπορεί να κρατήσει όσο μια ολόκληρη εργασιακή εβδομάδα. Η οργάνωση του πελάτη πρέπει να δεσμευτεί για να επενδύσει τον απαραίτητο χρόνο.

Γιατί να χρησιμοποιούμε μια χρονική περίοδο JAD για την ανάπτυξη των απαιτήσεων του συστήματος; Γιατί να μην παίρναμε συνέντευξη από κάθε άτομο ξεχωριστά; Όπως θυμάστε το τελευταίο μέρος της πρόκλησης για μια διαδικασία ανάπτυξης είναι να εξοπλίζετε ένα σύστημα σε σύντομο χρονικό διάστημα. Οι ατομικές συνεντεύξεις μπορούν να διαρκέσουν εβδομάδες ή ακόμη περισσότερο αν δε συμπίπτουν τα προγράμματα των ατόμων. Περιμένοντας τα αποτελέσματα των ατομικών συνεντεύξεων καταλήγει σε χάσιμο χρόνου και μαζί του σε χάσιμο πιθανού ανταγωνιστικού πλεονεκτήματος για γρήγορη ολοκλήρωση του συστήματος. Οι ατομικές συνεντεύξεις θα περιλαμβάνουν αντικρουόμενες απόψεις και χάνετε έτσι ακόμη περισσότερο χρόνο ενώ η ομάδα προσπαθεί να λύσει τις διαφορές. Μαζεύοντας τους πάντες μαζί δημιουργείται ένα σύνολο που ξεπερνάει το σύνολο των μερών και η αλληλεπίδραση ανάμεσα στους συμμετέχοντες JAD καταλήγει σε μια συμβίωση συμφέρουσα για όλους.

5.6.5.1.5 Παρουσίαση των αποτελεσμάτων στον πελάτη

Όταν η ομάδα τελειώσει με τις Απαιτήσεις, ο διευθυντής έργου παρουσιάζει τα αποτελέσματα στον πελάτη. Μερικές οργανώσεις ίσως απαιτούν την έγκριση του πελάτη σε αυτό το σημείο ώστε να προχωρήσει η ανάπτυξη. Άλλες οργανώσεις ίσως να απαιτήσουν μία εκτίμηση του κόστους βασισμένη στα αποτελέσματα. Το προϊόν εργασίας, τότε, θα ποικίλλει ανάλογα με την οργάνωση.

5.6.5.2 Ανάλυση

Σε αυτό το τμήμα, η ομάδα διαπερνά τα αποτελέσματα του τμήματος των Απαιτήσεων και διευρύνει την κατανόησή της επάνω στο πρόβλημα. Στην πραγματικότητα, μέρη αυτού του τμήματος ξεκινάνε στη διάρκεια του τμήματος των Απαιτήσεων, αφού ο μοντελιστής ξεκινά να ξεκαθαρίζει το σχεδιάγραμμα κλάσης στη χρονική περίοδο JAD των Απαιτήσεων.

5.6.5.2.1 Καταλαβαίνοντας τη χρήση του συστήματος

Αυτή η δράση είναι μια ανάλυση σεναρίου υψηλού επιπέδου. Σε μια χρονική περίοδο JAD με υποψήφιους χρήστες, η ομάδα ανάπτυξης εργάζεται με τους χρήστες για να ανακαλύψει τους δράστες που εισάγουν κάθε σενάριο από τη χρονική περίοδο

Απαιτήσεων JAD και τους δράστες μπορούν να επωφεληθούν από αυτά σενάρια. (Ενας δράστης, θυμηθείτε, μπορεί να είναι τόσο ένα σύστημα όσο κι ένα άτομο). Αυτός που διευκολύνει συντονίζει την περίοδο και δύο μέλη της ομάδας κρατούν σημειώσεις. Μετά από μερικά project, αυτός που διευκολύνει αυτήν την περίοδο θα εξελιχθεί σε αναλυτής σεναρίου.

Η ομάδα προσπαθεί επίσης να αναπτύσσει νέα σενάρια. Το προϊόν εργασίας είναι ένα σύνολο διαγραμμάτων σεναρίων που δείχνει δράστες και κάθε στερεότυπη εξάρτηση ανάμεσα στα σενάρια.

5.6.5.2.2 Ανάλυση διαγραμμάτων σεναρίων

Σε αυτή τη δράση, η ομάδα ανάπτυξης συνεχίζει τη δουλειά της με τους χρήστες. Ο σκοπός είναι να αναλύσει τη συχνότητα των βημάτων σε κάθε σενάριο. Αυτή η περίοδος JAD μπορεί να είναι συνέχεια της προηγουμένης. Προσοχή: είναι συνήθως η πιο δύσκολο περίοδος για τους χρήστες. Δεν είναι μάλλον συνηθισμένοι να διασπάνε μια λειτουργία σε συστατικά βήματα και διεξοδικά να απαριθμούν όλα αυτά τα βήματα. Το προϊόν έργου είναι ένα κείμενο που περιγράφει τα βήματα σε κάθε σενάριο.

5.6.5.2.3 Αποσαφηνίζοντας ένα διάγραμμα κλάσης

Στη διάρκεια των περιόδων JAD, ο μοντελιστής αντικειμένων ακούει όλες τις συνομιλίες και προχωρά στην αποσαφήνιση των διαγραμμάτων κλάσης. Σε αυτό ο σημείο, ένας μοντελιστής αντικειμένων πρέπει να ονομάσει τις συσχετίσεις, τις αφηρημένες κλάσεις, τις πολυπλοκότητες, τις γενικοποιήσεις και τις συναθροίσεις. Το προϊόν εργασίας είναι ένα αποσαφηνισμένο διάγραμμα κλάσεων.

5.6.5.2.4 Αναλύστε τις Αλλαγές Καταστάσεων στα Αντικείμενα

Ο μοντελιστής των αντικειμένων αποσαφηνίζει περαιτέρω το μοντέλο δείχνοντας αλλαγές κατάστασης όπου είναι απαραίτητο. Το προϊόν εργασίας είναι ένα διάγραμμα κατάστασης.

5.6.5.2.5 Καθορίστε την αλληλεπίδραση ανάμεσα στα αντικείμενα

Τώρα που η ομάδα έχει ένα σύνολο διαγραμμάτων σεναρίου και ένα αποσαφηνισμένο διάγραμμα κλάσεων, είναι ώρα να καθορίσετε πώς αλληλεπιδρούν τα αντικείμενα. Ο μοντελιστής αντικειμένων αναπτύσσει ένα σύνολο διαγραμμάτων ακολουθιών και διαγραμμάτων συνεργασίας για να απεικονίσει την αλληλεπίδραση. Πρέπει να συμπεριλαμβάνονται και διαγράμματα κατάστασης. Αυτά τα διαγράμματα σχηματίζουν το προϊόν εργασίας για αυτή τη δράση.

5.6.5.2.6 Αναλύστε την Ενσωμάτωση των Συστημάτων Συνεργασίας

Προχωρώντας παράλληλα με όλα τα προηγούμενα βήματα, ο μηχανικός συστημάτων αποκαλύπτει συγκεκριμένες λεπτομέρειες με τα συστήματα συνεργασίας. Τι είδους επικοινωνία εμπλέκεται; Τι είναι η αρχιτεκτονική ενός δικτύου; Αν το σύστημα πρέπει να έχει πρόσβαση σε βάση δεδομένων, ένας αναλυτής βάσης δεδομένων καθορίζει την αρχιτεκτονική (φυσική και λογική) αυτών των βάσεων δεδομένων. Τα προϊόντα εργασίας είναι λεπτομερή διαγράμματα εξέλιξης και (αν είναι απαραίτητο) μοντέλα δεδομένων.

5.6.5.3 Σχεδιασμός

Σε αυτό το τμήμα, η ομάδα δουλεύει με τα αποτελέσματα του τμήματος της Ανάλυσης για να σχεδιάσει τη λύση. Σχεδιασμός και ανάλυση πρέπει να επεξεργάζονται συνεχώς μέχρι την ολοκλήρωση του σχεδίου. Μερικές μεθοδολογίες, στην πραγματικότητα, συνδυάζουν Ανάλυση και Σχεδιασμό σε ένα τμήμα.

5.6.5.3.1 Ανάπτυξη και Αποσαφήνιση των Διαγραμμάτων Αντικειμένων

Οι προγραμματιστές παίρνουν το διάγραμμα κλάσης και παράγουν κάθε απαραίτητο διάγραμμα αντικειμένων. Εμπλουτίζει τα διαγράμματα αντικειμένων εξετάζοντας την κάθε λειτουργία και αναπτύσσοντας ένα ανάλογο διάγραμμα δραστηριοτήτων. Τα διαγράμματα δραστηριοτήτων θα λειτουργήσουν σα βάση για το μεγαλύτερο μέρος του τμήματος της Ανάπτυξης. Τα προϊόντα εργασίας είναι τα διαγράμματα αντικειμένων και τα διαγράμματα δραστηριοτήτων.

5.6.5.3.2 Αναπτύξτε Διαγράμματα Συστατικών

Οι προγραμματιστές παίζουν ένα σημαντικό ρόλο σε αυτή τη δράση. Η δουλειά εδώ είναι η οπτικοποιήση των στοιχείων, της οποίας το αποτέλεσμα θα φανεί από τα επόμενο τμήμα και θα δείξει τις εξαρτήσεις ανάμεσά τους. Τα διαγράμματα συστατικών είναι το προϊόν εργασίας.

5.6.5.3.3 Προσχέδιο για ανάπτυξη

Όταν το διάγραμμα συστατικών είναι ολοκληρωμένο, ο μηχανικός συστήματος ξεκινάει το προσχέδιο για εξέλιξη και ενσωμάτωση στα συστήματα συνεργασίας. Δημιουργεί ένα διάγραμμα ανάπτυξης που δείχνει πού θα εδρεύσουν τα στοιχεία. Το προϊόν εργασίας είναι ένα διάγραμμα που είναι μέρος του διαγράμματος ανάπτυξης που αναπτύχθηκε νωρίτερα.

5.6.5.3.4 Σχεδιασμός και Πρωτότυπη Διασύνδεση με το Χρήστη

Αυτό αφορά άλλη μια περίοδο JAD με τους χρήστες. Αν και αυτό είναι μέρος του Σχεδιασμού, αυτή η περίοδος μπορεί να είναι συνέχεια των προηγουμένων περιόδων

JAD με χρήστες – μία ένδειξη της αλληλεπίδρασης ανάμεσα στην Ανάλυση και το Σχεδιασμό.

Η διασύνδεση με το χρήστη πρέπει να επιτρέπει την ολοκλήρωση όλων των σεναρίων. Για να εκτελέσει αυτή τη δράση, ένας αναλυτής GUI (Graphical User Interphase = Γραφικό Περιβάλλον Διασύνδεσης) δουλεύει με τους χρήστες για να αναπτύξει πρότυπα ανακοινώσεων της οθόνης που αντιστοιχούν σε σύνολα σεναρίων. Οι χρήστες τοποθετούν αυτοκόλλητα χαρτάκια με σημειώσεις που αντιπροσωπεύουν στοιχεία οθόνης (κουμπάκια, check boxes, λίστα drop down, μενού και ούτω καθεξής). Όταν οι χρήστες είναι ικανοποιημένοι με την τοποθέτηση των στοιχείων, οι προγραμματιστές χτίζουν πρότυπα οθόνης για την έγκριση των χρηστών. Τα προϊόντα εργασίας είναι στιγμιότυπα οθόνης των πρότυπων της οθόνης.

5.6.5.3.5 Τεστ σχεδιασμού

Τα σενάρια διευκολύνουν το σχεδιασμό των τεστ για το λογισμικό. Ο σκοπός είναι να υπολογίσετε αν το αναπτυγμένο λογισμικό επιτελεί ή όχι αυτό που πρέπει – δηλαδή κάνει αυτό που προσδιορίζουν τα σενάρια. Κατά προτίμηση, ένας προγραμματιστής ή ένας ειδικός δοκιμών εκτός της ομάδας χρησιμοποιούν τα διαγράμματα σεναρίων για να αναπτύσσουν δοκιμαστικά τεστ για αυτοματοποιημένα εργαλεία δοκιμής. Τα δοκιμαστικά σενάρια αποτελούν το προϊόν εργασίας.

5.6.5.3.6 Αρχή Τεκμηρίωσης

Δεν είναι ποτέ νωρίς να ξεκινήσετε να τεκμηριώνετε το σύστημα για τους τελικούς χρήστες και τους administrators του συστήματος. Οι ειδικοί τεκμηρίωσης δουλεύουν με σχεδιαστές για να ξεκινήσουν να μετατρέψουν σε πίνακες την τεκμηρίωση και να φτάνουν σε μια δομή υψηλού επιπέδου για κάθε έγγραφο. Η δομή του εγγράφου είναι το προϊόν εργασίας.

5.6.5.4 Ανάπτυξη

Εδώ αναλαμβάνουν οι προγραμματιστές. Με αρκετή ανάλυση και σχεδιασμό, αυτό το τμήμα πρέπει να ολοκληρωθεί με τρόπο γρήγορο και ομαλό.

5.6.5.4.1 Χτίζοντας τον κώδικα

Με τα διαγράμματα κλάσεων, τα διαγράμματα αντικειμένων, τα διαγράμματα δραστηριοτήτων και τα διαγράμματα εξαρτημάτων χτίζουν τον κώδικα για το σύστημα. Ο κώδικας είναι το προϊόν εργασίας από αυτή τη δράση.

5.6.5.4.2 Δοκιμάζοντας τον κώδικα

Οι ειδικοί δοκιμών (όχι οι προγραμματιστές) τρέχουν τα σενάρια για να υπολογίσουν αν ο κώδικας κάνει ό,τι πρέπει ή όχι. Τα αποτελέσματα των δοκιμών είναι τα προϊόντα εργασίας. Αυτή η δράση τροφοδοτείται από την προηγούμενη δράση και αντιστρόφως μέχρι που ο κώδικας περνάει όλα τα επίπεδα δοκιμών.

5.6.5.4.3 Χτίστε Διασύνδεση με το Χρήστη , Συνδεθείτε στον Κώδικα και Δοκιμάστε

Αυτή η δράση αναφέρεται στην εγκεκριμένη διασύνδεση με το χρήστη. Ο ειδικός GUI τη χτίζει και τη συνδέει στον κώδικα. Περαιτέρω δοκιμές επιβεβαιώνει ότι η διασύνδεση με το χρήστη δουλεύει καλά. Το λειτουργικό σύστημα μαζί με τη διασύνδεση με το χρήστη είναι το προϊόν εργασίας.

5.6.5.4.4 Ολοκληρωμένη Τεκμηρίωση

Κατά τη διάρκεια του τμήματος της Ανάπτυξης, οι ειδικοί τεκμηρίωσης δουλεύουν παράλληλα με τους προγραμματιστές για να επιβεβαιώσουν την έγκαιρη μεταβίβαση Η τεκμηρίωση είναι το προϊόν εργασίας για αυτή τη δράση.

5.6.5.5 Επέκταση

Όταν η ανάπτυξη ολοκληρωθεί, το σύστημα επεκτείνεται με το κατάλληλο υλικό και αλληλεπιδρά με τα συνεργαζόμενα συστήματα. Η πρώτη δράση σε αυτό το τμήμα, όμως, μπορεί να αρχίσει πολύ πριν την εκκίνηση του τμήματος Ανάπτυξης.

5.6.5.5.1 Εφεδρεία και Επανόρθωση

Ο μηχανικός του συστήματος δημιουργεί ένα προσχέδιο με βήματα που πρέπει να ακολουθηθούν σε περίπτωση που το σύστημα καταρρεύσει. Το προσχέδιο, το προϊόν εργασίας για αυτή τη δράση, συγκεκριμενοποιεί τι πρέπει να γίνει για το backup του συστήματος και να επανέλθει από την κατάρρευση.

5.6.5.5.2 Εγκατάσταση του Τελικού Συστήματος

Ο μηχανικός του συστήματος, με την απαραίτητη βοήθεια από τους προγραμματιστές, εξελίσσει το τελειωμένο σύστημα στους ανάλογους υπολογιστές. Το προϊόν εργασίας είναι το πλήρως εξελισσόμενο σύστημα.

5.6.5.5.3 Δοκιμή στο Εγκαταστημένο Σύστημα

Τέλος, η ομάδα ανάπτυξης δοκιμάζει το εγκαταστημένο σύστημα. Λειτουργεί όπως θα έπρεπε; Λειτουργεί το πλάνο backup και recovery; Τα αποτελέσματα αυτών των

δοκιμών προσδιορίζουν αν περαιτέρω εκλέπτυνση είναι απαραίτητη και αυτά τα αποτελέσματα των τεστ αποτελούν το προϊόν εργασίας.

5.6.6 Το περιτύλιγμα του GRAPPLE

Αν κοιτάξετε τα τμήματα και τις δράσεις του GRAPPLE, θα δείτε ότι η κίνηση είναι από γενική σε ειδική – από την ασαφή στην αποσαφηνισμένη. Ξεκινάει με μια εννοιολογική κατανόηση του πεδίου ορισμού, προχωράει σε υψηλού επιπέδου λειτουργικότητα, διαπερνάει τα σενάρια, αποσαφηνίζει τα μοντέλα και σχεδιάζει, αναπτύσσει και εξελίσσει το σύστημα.

Θα παρατηρήσετε επίσης ότι περισσότερες δράσεις βρίσκονταν στα τμήματα της Ανάλυσης και του Σχεδιασμού παρά στο τμήμα της Ανάπτυξης. Αυτό είναι – και συγχωρήστε το λογοπαίγνιο – από σχεδιασμού. Η ιδέα είναι να περάσετε όσο το δυνατόν περισσότερο χρόνο μπορείτε σε ανάλυση και σχεδιασμό ώστε οι διαδικασίες κωδικοποίησης να προχωρήσουν ομαλά. Ίσως μοιάζει λίγο με αίρεση, αλλά στον ιδανικό κόσμο, η αποκωδικοποίηση είναι μόνο ένα μικρό μέρος του συστήματος ανάπτυξης. Όσο περισσότερο αναλύετε, τόσο πιο κοντά έρχεστε στο ιδανικό.
Το GRAPPLE, όπως είπα, είναι ένας απλουστευμένος σκελετός της διαδικασίας ανάπτυξης. Δεν έθιξα τις λεπτομέρειες σημαντικών ζητημάτων όπως επίπεδα δοκιμών. Άφησα, επίσης, έξω μερικές σημαντικές πρακτικές εργασίες. Πού και πώς διατηρεί η ομάδα τα προϊόντα εργασίας εν εξελίξει; Πώς αντιμετωπίζει η ομάδα το πολύ σημαντικό ζήτημα της διαχείρισης της σύνδεσης;

Δεν αναφέρθηκα σε αυτά τα θέματα γιατί αποκλίνουν από τη συζήτησή μας για τη UML. Η σύντομη απάντηση για αυτά τα πρακτικά ζητήματα είναι να αγκαλιάσουν την τεχνολογία. Τα προϊόντα εργασίας (ολοκληρωμένα ή εν εξελίξει) μπορούν να εδράσουν σε ένα χώρο φύλαξης LAN της οργάνωσης. Μια επιλογή είναι να υπάρχει ιεραρχία των καταλόγων στους οποίους μπορούν να έχουν πρόσβαση τα μέλη της ομάδας. Μια πιο ασφαλής επιλογή είναι να εγκαταστήσετε ένα κεντρικό πακέτο φύλαξης που εντοπίζει τα checkout και τα check-in των προϊόντων εργασίας και επιτρέπει μόνο σε ένα άτομο τη φορά να κάνει checkout ένα αντίγραφο ενός αντικειμένου που μπορεί να διαμορφοποιηθεί. Αυτή είναι η θεμελίωση μιας λύσης για τη διαχείριση σύνδεσης. Η τεχνολογία φύλαξης προχωράει σταθερά και αρκετές επιλογές είναι διαθέσιμες.

5.6.7 Περίληψη

Μια μεθοδολογία ανάπτυξης δομεί τα τμήματα και τις δραστηριότητες σε μια εργασία ανάπτυξης του συστήματος. Χωρίς μια μεθοδολογία, το χάος θα βασίλευε, οι προγραμματιστές δε θα καταλάβαιναν το πρόβλημα που προσπαθούσαν να λύσουν και τα συστήματα δε θα συναντούσαν τις ανάγκες των χρηστών τους. Πρώιμες μεθοδολογίες επέβαλαν μια συχνότητα «καταρράκτη» ανάλυσης, σχεδιασμού, κώδικα και εξέλιξης.

Αυτό το είδος της μεθοδολογίας μπορεί να διαιρεί την ανάπτυξη έτσι ώστε η ομάδα ανάπτυξης μπορεί να μην εκμεταλλευτεί την αυξημένη κατανόηση που επιδρά στη ζωή μιας εργασίας. Τυπικά, κατανέμει επίσης το μεγαλύτερο μερίδιο του χρόνου της

εργασίας στην κωδικοποίηση και έτσι κλέβει πολύτιμο χρόνο από την ανάλυση και το σχεδιασμό.

Αυτό το κεφάλαιο παρουσίασε το GRAPPLE (Guidelines for Rapid APPLication Engineering), ένας σκελετός μιας διεργασίας ανάπτυξης. Το GRAPPLE περιλαμβάνει πέντε τμήματα: Συγκέντρωση Απαιτήσεων, Ανάλυση, Σχεδιασμός, Ανάπτυξη και Εξέλιξη. Κάθε τμήμα περιλαμβάνει αρκετές δράσεις και κάθε δράση καταλήγει σε ένα προϊόν εργασίας. Τα διαγράμματα της UML είναι προϊόντα εργασίας για πολλές από τις δράσεις.

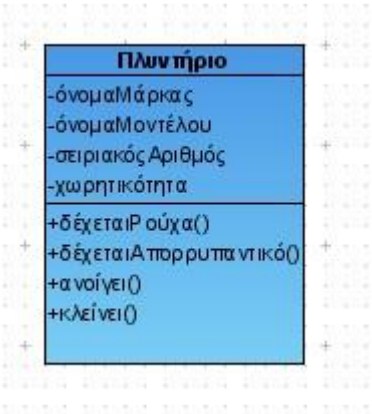
5.7 ΠΑΡΑΔΕΙΓΜΑ ΠΕΡΙΓΡΑΦΗΣ ΕΝΟΣ ΣΥΣΤΗΜΑΤΟΣ

Στο παρόν κεφάλαιο αναλύεται συνοπτικά ένα σύστημα με τη χρήση των βασικών διαγραμμάτων της UML. Η ανάλυση δεν είναι ιδιαίτερα λεπτομερής έτσι ώστε να είναι κατανοητή δίχως την απαραίτητη γνώση εξειδικευμένων εννοιών της UML. Στο παρόν σύστημα περιγράφονται κάποιες βασικές λειτουργίες ενός πλυντηρίου όπως αυτές μοντελοποιούνται και οπτικοποιούνται με τη βοήθεια των διαγραμμάτων της UML.

5.7.1 Διάγραμμα κλάσης

Λεπτομερής περιγραφή των διαγραμμάτων κλάσεων γίνεται στις ενότητες 5.3.1 και 5.4.1 της εργασίας μας.

Στο παρακάτω παράδειγμα, οτιδήποτε στην κλάση των πλυντηρίων έχει ιδιότητες όπως όνομα μάρκας, μοντέλο, σειριακό αριθμό και χωρητικότητα. Συμπεριφορές για πράγματα αυτής της κλάσης περιλαμβάνουν τις λειτουργίες “δέχεται ρούχα”, “δέχεται απορρυπαντικό, “ανοίγει”, ‘κλείνει’.

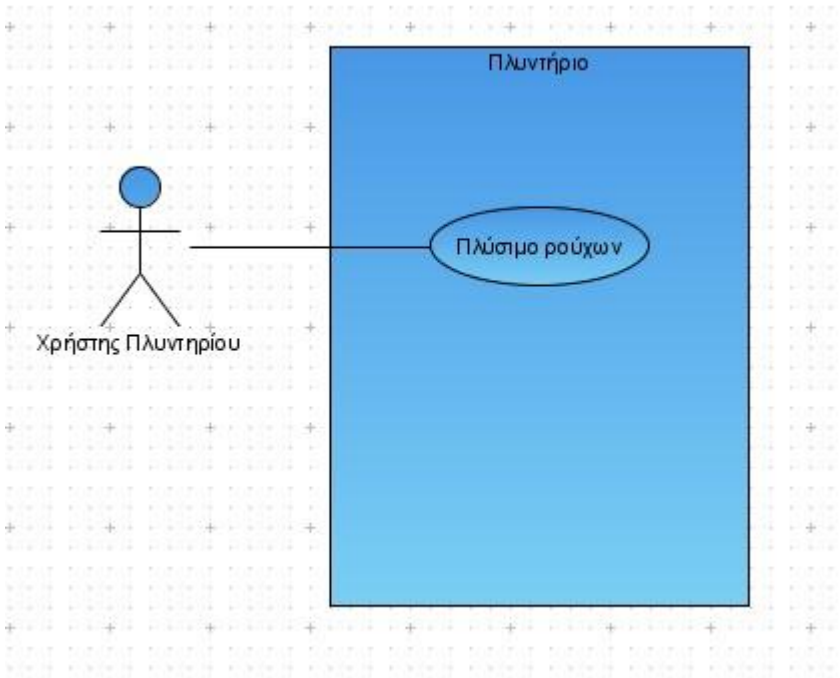


Σχήμα 5.7.1-1: Το διάγραμμα κλάσης ενός Πλυντηρίου

5.7.2 Διάγραμμα σεναρίου

Επεξήγηση των διαγραμμάτων σεναρίου γίνεται στην ενότητα 5.4.2 της εργασίας μας.

Το παράδειγμα που ακολουθεί παριστάνει τη χρήση του πλυντηρίου, που προφανώς είναι να πλένει τα ρούχα, με τη σημειολογία της UML.



Σχήμα 5.7.2-1: Το διάγραμμα σεναρίου για τη χρήση του Πλυντηρίου

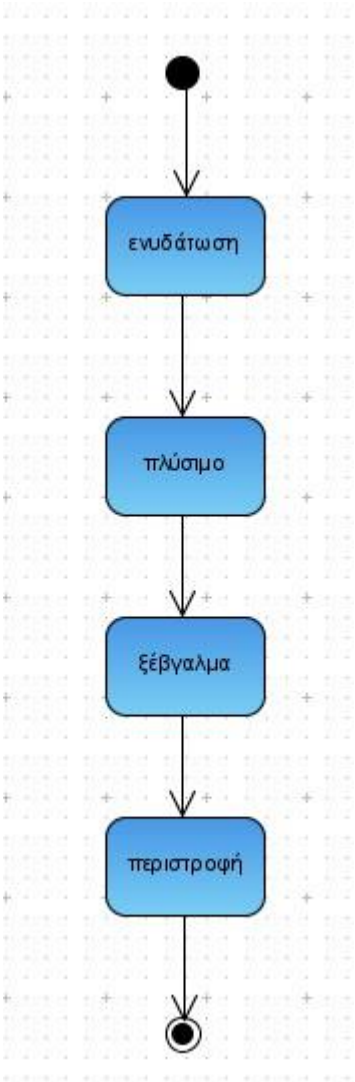
Το σχήμα με το «ανθρωπάκι» αντιστοιχεί στον χρήστη του πλυντηρίου. Η έλλειψη παριστάνει το σενάριο. Ας σημειωθεί πως ο δράστης μπορεί να είναι ένας άνθρωπος ή ένα άλλο σύστημα. Ακόμη, το σενάριο βρίσκεται μέσα σε ένα ορθογώνιο που παριστάνει το σύστημα, και ο δράστης είναι έξω από το ορθογώνιο.

5.7.3 Διάγραμμα κατάστασης

Πλήρης επεξήγηση των διαγραμμάτων κατάστασης γίνεται στην ενότητα 5.4.5 της εργασίας.

Ένα πλυντήριο μπορεί να είναι είτε σε κατάσταση ενυδάτωσης, πλυσίματος, στεγνώματος ή κλειστό.

Το διάγραμμα κατάστασης που παρουσιάζεται στο σχέδιο *5.7.3-1* δείχνει αυτή την πραγματικότητα. Το σχέδιο δείχνει πως το πλυντήριο περνάει από τη μια κατάσταση στην άλλη.



Σχήμα 5.7.3-1: Ένα διάγραμμα κατάστασης για τις καταστάσεις του Πλυντηρίου

Το σύμβολο στο επάνω μέρος του σχήματος παριστάνει την έναρξη της λειτουργίας του και το σύμβολο στο κάτω μέρος τη διακοπή της.

5.7.4 Διάγραμμα ακολουθίας

Ανάλυση των διαγραμμάτων ακολουθίας γίνεται στην ενότητα 5.4.3 της εργασίας μας.

Ακολουθώντας το παράδειγμα του πλυντηρίου, τα συστατικά του πλυντηρίου περιλαμβάνουν ένα χρονοδιακόπτη, μία σωλήνα νερού (για εισαγωγή καθαρού νερού), και ένα κάδο (το εξάρτημα που κρατάει τα ρούχα και το νερό). Αυτά είναι επίσης αντικείμενα.

Υποθέτοντας ότι έχουμε ολοκληρώσει τις λειτουργίες “πρόσθεση ρούχων”, “πρόσθεση απορρυπαντικού” και “έναρξη λειτουργίας” του σεναρίου “πλύσιμο ρούχων”, η ακολουθία βημάτων πηγαίνει κάπως έτσι:

1. Στην αρχή της “ενυδάτωσης”, το νερό εισχωρεί στον κάδο μέσω της σωλήνας νερού.
2. Ο κάδος παραμένει στάσιμος για 5 λεπτά.
3. Στο τέλος της “ενυδάτωσης”, το νερό σταματάει να εισέρχεται στον κάδο.
4. Στην αρχή του “Πλυσίματος”, ο κάδος περιστρέφεται εμπρός και πίσω και συνεχίζει να το κάνει αυτό για 15 λεπτά.
5. Στο τέλος του “Πλυσίματος”, ο κάδος αδειάζει από το νερό με το απορρυπαντικό.
6. Ο κάδος σταματάει να περιστρέφεται.
7. Στην αρχή του “Ξεβγάλματος”, η εισαγωγή νερού ξαναρχίζει.
8. Ο κάδος περιστρέφεται εμπρός και πίσω.
9. Ύστερα από 15 λεπτά η εισαγωγή νερού σταματάει.
10. Στο τέλος του “Ξεβγάλματος”, ο κάδος αδειάζει το νερό του ξεβγάλματος.
11. Ο κάδος σταματάει να περιστρέφεται.
12. Στην αρχή της “Περιστροφής”, ο κάδος περιστρέφεται δεξιά και συνεχίζει για 5 λεπτά.
13. Στο τέλος της “Περιστροφής”, ο περιστροφή του κάδου σταματάει.
14. Το πλύσιμο τελείωσε.

Ας φανταστούμε πως ο χρονοδιακόπτης, ο σωλήνας νερού και ο κάδος είναι αντικείμενα. Ας υποθέσουμε πως το κάθε αντικείμενο έχει μια ή περισσότερες λειτουργίες. Τα αντικείμενα συνεργάζονται μεταξύ τους με το να στέλνουν μηνύματα το ένα στο άλλο. Κάθε μήνυμα είναι μια κλήση από το αντικείμενο που στέλνει προς το αντικείμενο που παραλαμβάνει. Η κλήση ζητάει στον παραλήπτη να ολοκληρώσει μια από τις λειτουργίες του (του παραλήπτη).

Ο χρονοδιακόπτης μπορεί να:

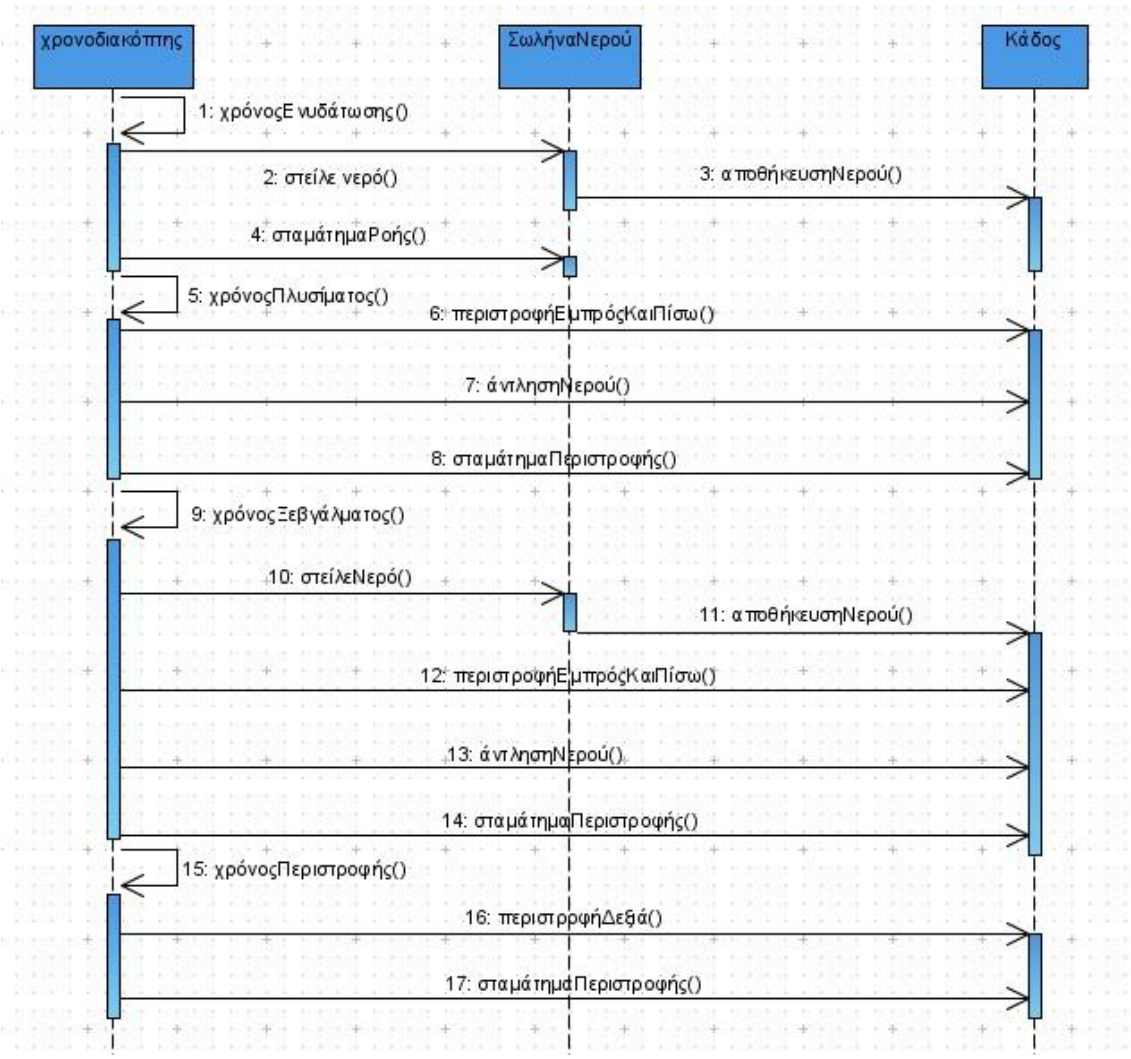
- Μετράει τη διάρκεια της ενυδάτωσης
- Μετράει τη διάρκεια του πλυσίματος
- Μετράει τη διάρκεια του ξεβγάλματος
- Μετράει τη διάρκεια της περιστροφής

Ο σωλήνας του νερού μπορεί να:
- Ξεκινήσει τη ροή
- Σταματήσει τη ροή

Ο κάδος μπορεί να:
- Γεμίσει με νερό
- Περιστραφεί εμπρός και πίσω
- Περιστραφεί προς τα δεξιά
- Σταματήσει να περιστρέφεται
- Αντλήσει νερό

Το σχέδιο *5.7.4-1* δείχνει πώς να χρησιμοποιήσουμε αυτές τις λειτουργίες για να δημιουργήσουμε ένα διάγραμμα ακολουθίας που δείχνει τα μηνύματα μεταξύ του χρονοδιακόπτη, της σωλήνας και του κάδου. Κάθε βέλος παριστάνει ένα μήνυμα που πηγαίνει από το ένα αντικείμενο στο άλλο. Ο χρόνος, σε αυτό το διάγραμμα, παριστάνεται από επάνω προς τα κάτω.

Ας προσέξουμε πως ένα αντικείμενο (σε αυτή την περίπτωση ο χρονοδιακόπτης) μπορεί να στείλει ένα μήνυμα στον εαυτό του.



Σχήμα 5.7.4-1: Ένα διάγραμμα ακολουθίας των μηνυμάτων κατά τη λειτουργία του Πλυντηρίου

5.7.5 Διάγραμμα δραστηριότητας

Λεπτομερής περιγραφή των διαγραμμάτων δραστηριότητας γίνεται στην ενότητα 5.4.6 της εργασίας μας.

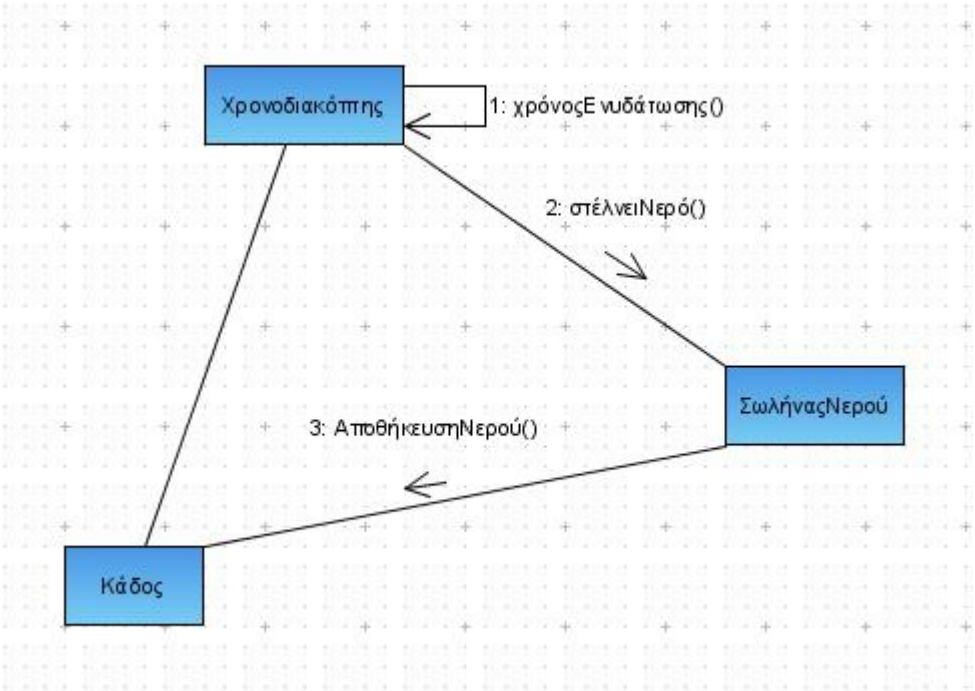
Οι δραστηριότητες που λαμβάνουν χώρα μέσα σε ένα σενάριο ή μέσα στη συμπεριφορά ενός αντικειμένου, τυπικά συμβαίνουν με μια σειρά, όπως στα βήματα που παρατέθηκαν στην πιο πάνω ενότητα. Το σχέδιο *5.7.5-1* δείχνει πώς το διάγραμμα δραστηριότητας της UML παριστάνει τα βήματα 4 έως 6 αυτής της ακολουθίας.

Σχήμα 5.7.5-1: Ένα διάγραμμα δραστηριότητας του Πλυντηρίου

5.7.6 Διάγραμμα επικοινωνίας

Πλήρης επεξήγηση των διαγραμμάτων επικοινωνίας γίνεται στην ενότητα 5.4.4 της εργασίας μας.

Τα συστατικά ενός συστήματος δουλεύουν μαζί για να πραγματοποιήσουν το σκοπό του συστήματος και μια γλώσσα μοντελοποίησης πρέπει να έχει ένα τρόπο να το απεικονίσει. Το προαναφερθέν διάγραμμα ακολουθίας το κάνει αυτό. Το διάγραμμα επικοινωνίας που δείχνεται στο σχέδιο *5.7.6-1* κάνει περίπου το ίδιο με διαφορετικό τρόπο.



Σχήμα 5.7.6-1: Ένα διάγραμμα επικοινωνίας των συστατικών του Πλυντηρίου

Το διάγραμμα ακολουθίας και το διάγραμμα επικοινωνίας δείχνουν αλληλεπιδράσεις μεταξύ των αντικειμένων. Γι' αυτό το λόγο, η UML αναφέρεται σ' αυτά ως διαγράμματα αλληλεπίδρασης.

5.7.7 Διάγραμμα συστατικών

Πλήρης επεξήγηση των διαγραμμάτων κατάστασης γίνεται στην ενότητα 5.4.7 της εργασίας μας.

Το διάγραμμα αυτό καθώς και το επόμενο ξεφεύγουν από τον κόσμο των πλυντηρίων επειδή το διάγραμμα συστατικών και το διάγραμμα ανάπτυξης αναφέρονται σε συστήματα υπολογιστών.

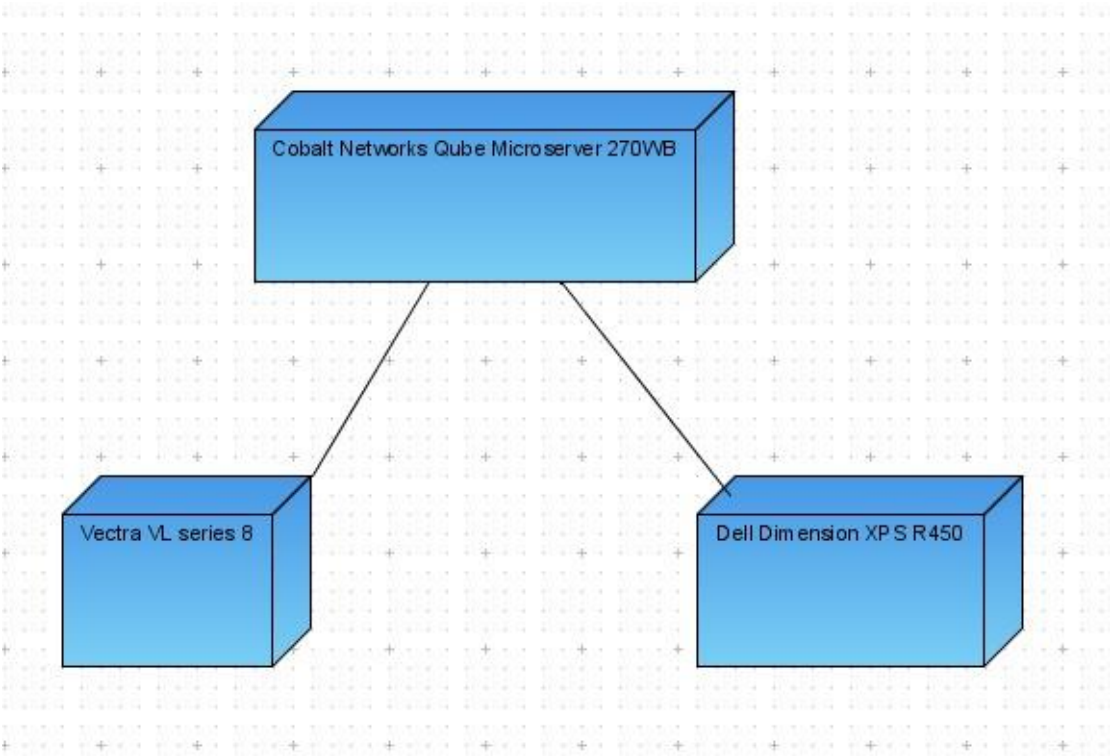
Η μοντέρνα ανάπτυξη λογισμικού βασίζεται σε συστατικά, που είναι απολύτως απαραίτητα σε ομαδικές προσπάθειες. Παρακάτω δείχνεται ένα συστατικό λογισμικού.

Σχήμα 5.7.7-1 Ένα διάγραμμα συστατικού

5.7.8 Διάγραμμα ανάπτυξης

Πλήρης ανάλυση των διαγραμμάτων κατάστασης γίνεται στην ενότητα 5.4.8 της εργασίας.

Το διάγραμμα ανάπτυξης της UML δείχνει τη φυσική αρχιτεκτονική ενός συστήματος υπολογιστών. Δείχνει τους υπολογιστές και τις συνδέσεις τους και το λογισμικό που "τρέχει" σε κάθε μηχάνημα. Κάθε υπολογιστής παριστάνεται ως κύβος, με διασυνδέσεις μεταξύ των υπολογιστών σαν γραμμές που συνδέουν τους κύβους. Το σχέδιο *5.7.8-1* παρουσιάζει ένα παράδειγμα.



Σχήμα 5.7.8-1: Ένα πιθανό διάγραμμα ανάπτυξης

5.8 ΤΟ ΜΕΛΛΟΝ ΤΗΣ UML

5.8.1 UML 2.0

Η UML 2.0 είναι η πρώτη μεγάλη αναθεώρηση της UML 1.0 . Γιατί ήταν αναγκαίο να αναθεωρηθεί η UML ; Το μεγαλύτερο κίνητρο ήταν η καλύτερη υποστήριξη τον εργαλείων και των μεθόδων που υπήρχαν. Την περασμένη δεκαετία πολλοί ανέπτυξαν εργαλεία βασισμένα στην UML που υποστήριζαν σημαντικά μεγαλύτερα επίπεδα αυτοματισμού από τα παραδοσιακά εργαλεία. Για να υποστηριχτούν αυτές οι υψηλότερες μορφές αυτοματισμού ήταν απαραίτητο να οριστεί η UML με ποιο ακριβής ορούς από ότι ορίζονταν μέχρι εκείνη τη στιγμή.

Δυστυχώς οι οροί αυτοί διαφέρανε από προμηθευτή σε προμηθευτή , απειλώντας για μια ακόμα φορά να οδηγήσουν στον διαχωρισμό για τον οποίο η λύση ήταν η αρχική UML. Μια νέα έκδοση μπορούσε να το διόρθωση αυτό.
Τέλος ,κατά την ιδία περίοδο, πολλά είχαν μαθευτεί για καλυτέρους τρόπους χρησιμοποίησης , δόμησης και ορισμού των μοντέλων.

5.8.1.1 Οι σημαντικότερες αλλαγές στη UML

Μπορούμε να ομαδοποιήσουμε τις αλλαγές στην UML 2.0 στις παρακάτω πέντε μεγάλες κατηγορίες :

- Μια σημαντική αύξηση στην ακρίβεια τον ορισμών της γλωσσάς. Αυτό απευθύνεται στην ανάγκη για υποστήριξη των υψηλών επιπέδων αυτοματισμού.
- Μια βελτιωμένη οργάνωση της γλωσσάς ,με κύριο χαρακτηριστικό την νέα διαμόρφωση που επιτρέπει την πιο εύκολη πρόσβαση από νέους χρηστές και την καλύτερη συνεργασία διαφορετικών εργαλείων.
- Σημαντικές αλλαγές στην ικανότητα να μοντελοποιούνται μεγάλης κλίμακας συστήματα λογισμικού.
- Βελτιωμένη υποστήριξη για την εξειδίκευση σε ορισμένες περιοχές. Η πρακτική εμπειρία με την UML κατέδειξε την αξία των αποκαλούμενων μηχανισμών "επέκτασής της". Το OMG παγίωσε αυτούς τους μηχανισμούς για να επιτρέψει τους απλούστερους και ακριβέστερους καθαρισμούς της βασικής γλώσσας.
- Συνολική σταθεροποίηση, εκλογίκευση και διευκρινίσεις στις διαφορές ιδέες των μοντέλων με αποτέλεσμα μια πιο απλή και συμπαγής γλωσσά.

Παρακάτω θα αναφερθουν λίγο πιο αναλυτικά μερικές βασικές αλλαγές από την UML 1.0 στην UML 2.0 .

5.8.1.2 Βαθμός ακρίβειας

Ο βαθμός ακριβείας που χρησιμοποιείται στους ορισμούς της UML 2.0 αυξήθηκε σημαντικά. Αυτό επιτεύχθηκε με τους ακολούθους τρόπους:

- Έγινε μια μεγάλη αλλαγή στην εσωτερική δομή του μεταμοντελου. Η εσωτερική δομή της UML 2.0 αποτελείτε από ένα σετ πρώτου επιπέδου ιδεών μοντελοποίησης και σχεδίων που στις περισσότερες περιπτώσεις είναι πολύ στοιχειώδεις η αφηγημένες για να χρησιμοποιηθούν απευθείας στον σχεδιασμό συστημάτων λογισμικού. Όμως η σχετική απλότητα τους δίνει μια μεγαλύτερη ακρίβεια στην σημειολογία τους. Αυτές οι ιδέες μπορούν να συνδυαστούν με διαφορετικούς τρόπους και να παράγουν πιο πολύπλοκα σε επίπεδο χρηστή μοντέλα. Για παράδειγμα στην UML 1.0 η έννοια της ιδιοκτησίας, η έννοια των ονομαζόμενων συλλογών μοναδικά ονομαζόμενων στοιχειών (namespaces) και η έννοια του ταξινομητή (classifier) , ήταν συνδυασμένα σε μια πολύπλοκη σημειολογικά έννοια. Στην UML 2.0 αυτές οι έννοιες διαχωριστήκαν και οριστήκαν ξεχωριστά.
- Εκτεταμένες και ακριβέστερες περιγραφές σημασιολογίας. Ο καθορισμός σημασιολογίας των εννοιών διαμόρφωσης UML 1 ήταν προβληματικός με διάφορους τρόπους. Το επίπεδο περιγραφής ήταν ιδιαίτερα ανώμαλο, με μερικές περιοχές να έχουν τις εκτενείς και λεπτομερείς περιγραφές (π.χ., διαγράμματα καταστάσεων) ενώ άλλες είχαν ελάχιστες ή καμία εξηγήσεις. Η προδιαγραφή UML 2.0 δίνει περισσότερη έμφαση στη σημασιολογία και, ειδικότερα, στο βασικό τομέα της βασικής συμπεριφοριστικής δυναμικής
- Ένα σαφώς καθορισμένο δυναμικό σημασιολογικό πλαίσιο. Η προδιαγραφή UML 2.0 διευκρινίζει μερικά από τα κρίσιμα σημασιολογικά χάσματα στην αρχική έκδοση.
- Ειδικότερα, αυτό το πλαίσιο αντιμετωπίζει ρητά τα ακόλουθα ζητήματα:
  · Την δομική σημασιολογία των συνδέσεων (links) και των στιγμιότυπα (instances) στο χρόνο εκτέλεσης .
  · Την σχέση μεταξύ της δομής και της συμπεριφοράς.
  · Τα σημασιολογικά σχεδία ή τα μοντέλα αιτιότητας κοινά σε όλους τους υψηλού επιπέδου συμπεριφοριστικούς φορμαλισμούς στην UML (δηλ., διαγράμματα καταστάσεων, δραστηριοτήτων, αλληλεπιδράσεων) καθώς επίσης και πιθανούς μελλοντικοί. Αυτό επίσης εξασφαλίζει ότι τα αντικείμενα των οποίων συμπεριφορές εκφράζονται χρησιμοποιώντας τους διαφορετικούς φορμαλισμούς μπορούν να αλληλεπιδράσουν το ένα με το άλλο.

5.8.1.3 Βελτιωμένη οργάνωση της γλώσσας

Μια άμεση συνέπεια του αυξανόμενου επιπέδου ακρίβειας της UML 2.0 είναι ότι οι οροί της γλωσσάς έχουν αυξηθεί, ακόμη και χωρίς να υπολογίσουμε τις νέες ικανότητες μοντελοποίησης. Αυτό είναι μια ανησυχία, ειδικά δεδομένου ότι η βιομηχανία επέκρινε την αρχική UML ως πάρα πολύ πλούσια και, επομένως, πάρα πολύ δυσκίνητης για να μαθευτεί και να χρησιμοποιηθεί γλωσσάς.

Για να αντιμετωπίσει το πρόβλημα γλώσσα-πολυπλοκότητας, η OMG διαμόρφωσε την UML 2.0 με έναν τρόπο που επιτρέπει στους υπεύθυνους της ανάπτυξης να χρησιμοποιήσουν επιλεκτικά ότι χρειάζονται.
Αυτή η αρχιτεκτονική σημαίνει ότι οι χρήστες μπορούν να μάθουν και να χρησιμοποιήσουν μόνο το υποσύνολο της UML που τους ταιριάζει καλύτερα. Δεν είναι πλέον απαραίτητο να είναι κάποιος εξοικειωμένος με την πλήρη έκταση της UML προκειμένου να την χρησιμοποιήσει αποτελεσματικά όπως και δεν χρειάζεται να μάθει άπταιστα αγγλικά για να τα χρησιμοποιεί αποτελεσματικά. Δεδομένου ότι αποκτάει την εμπειρία, έχει την επιλογή βαθμιαία να μαθαίνει τις ισχυρότερες έννοιες μοντελοποίησης ανάλογα με τις ανάγκες του.

5.8.1.4 Μοντελοποίηση συστημάτων μεγάλης κλίμακας

Σχετικά λίγα χαρακτηριστικά γνωρίσματα προστέθηκαν στην UML 2.0. Αυτό ήταν σκόπιμο για να αποφευχθεί η επίδραση "δεύτερων συστημάτων", με την οποία μια γλώσσα παραφουσκώνει από μια υπερβολή νέων χαρακτηριστικών γνωρισμάτων, που απαιτούνται από την κοινότητα χρηστών. Στην πραγματικότητα, η πλειοψηφία των νέων ικανοτήτων μοντελοποίησης είναι απλά επεκτάσεις των υπαρχόντων χαρακτηριστικών γνωρισμάτων που επιτρέπουν σε σας να τις χρησιμοποιήσετε για να διαμορφώσετε μεγάλης κλίμακας συστήματα λογισμικού.

Επιπλέον, αυτές οι επεκτάσεις επιτεύχθηκαν χρησιμοποιώντας την ίδια βασική προσέγγιση: επαναλαμβανόμενη εφαρμογή του ίδιου βασικού συνόλου εννοιών σε διαφορετικά επίπεδα αφαίρεσης (abstraction layers). Αυτό σημαίνει ότι θα μπορούσατε να συνδυάσετε τα πρότυπα στοιχεία ενός δεδομένου τύπου σε μονάδες που, στη συνέχεια, θα χρησιμοποιούσατε ως δομικές μονάδες για το επόμενο επίπεδο μοντελοποίησης.

5.8.1.5 Βελτιστοποίηση των εξειδικεύσεων της γλώσσας

Ο μηχανισμός περιγραφής της UML 2.0 έχει οργανωθεί ορθολογικά και οι ικανότητές της επεκτάθηκαν. Η εννοιολογική σύνδεση μεταξύ ενός στερεοτύπου και των εννοιών που η UML επεκτείνει έχει διευκρινιστεί. Στην πραγματικότητα, ένα στερεότυπο της UML 2.0 καθορίζεται σαν να ήταν απλά μια υποκατηγορία μιας ήδη υπάρχουσας UML μετακλάσης (metaclass), με τις σχετικές ιδιότητες, τις διαδικασίες, και τους περιορισμούς

5.8.1.6 Περίληψη

Αυτά ήταν ένα μέρος από τις βασικές αλλαγές από την UML 1.0 στην UML 2.0 και οι κυριότεροι λόγοι που έκαναν αυτή την μετάβαση απαραίτητη για την βιωσιμότητα της UML στην σημερινή πραγματικότητα του σχεδιασμού συστημάτων. Με τις αλλαγές αυτές η UML πήρε καινούργια πνοή και μπορεί να σταθεί επάξια σαν το πιο βασικό μέσο για τον σχεδιασμό πολύπλοκων συστημάτων οποιασδήποτε μορφής και όχι μονό λογισμικού .

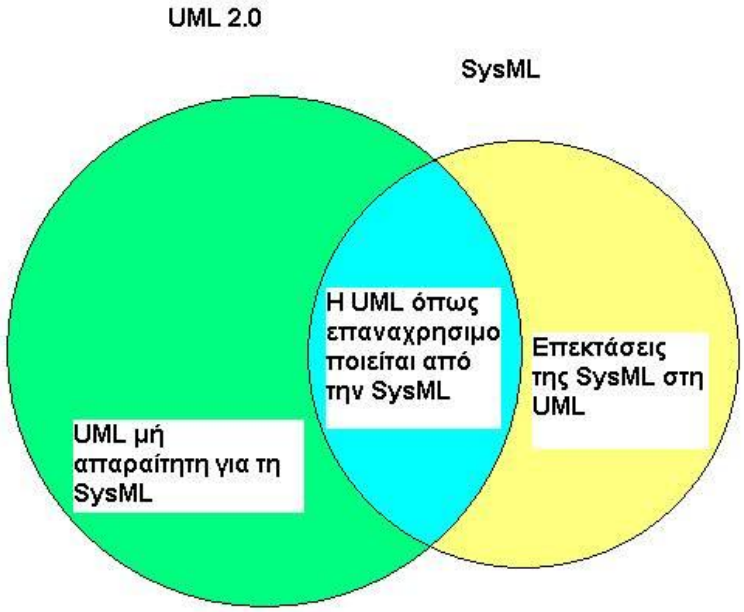
Μια πιο λεπτομερής ανάλυση των αλλαγών στην UML 2.0 μπορεί να βρεθεί στο διαδίκτυο και πιο συγκεκριμένα στην ιστοσελίδα της Rational http://www-306.ibm.com/software/rational/uml

5.8.2 SysML

Η SysML είναι σχεδιασμένη με τέτοιο τρόπο ώστε να παρέχει απλά αλλά ισχυρά εργαλεία για τη μοντελοποίηση μιας ευρείας γκάμας ζητημάτων στον τομέα της εφηρμοσμένης μηχανικής συστημάτων (systems engineering). Είναι ιδιαίτερα αποδοτική στο να αναλύει τις προϋποθέσεις, δομές συστημάτων και λειτουργική συμπεριφορά κατά τη διάρκεια της καταγραφής προδιαγραφών και σχεδίασης συστημάτων. Η πρώτη έκδοση της γλώσσας δεν περιλαμβάνει δέντρα αποφάσεων, τεστάρισμα, εκτενή διαπίστωση και τσεκάρισμα , ή πλήρη εκτελέσιμη λειτουργική συμπεριφορά. Αυτά τα κενά μπορούν να απευθυνθούν σε μελλοντικές εκδόσεις της SysML. Η SysML είναι σύμφωνη με δύο εξελισσόμενα πρότυπα: το ISO AP-223 πρότυπο ανταλλαγής δεδομένων για εργαλεία συστημάτων και το μοντέλο OMG XMI 2.0 για τα εργαλεία μοντελοποίησης της UML 2.0.

Για να οπτικοποιήσουμε τη σχέση μεταξύ των γλωσσών UML και SysML, ας δούμε το διάγραμμα Venn που παρουσιάζεται παρακάτω, όπου παρουσιάζονται τα σετ των στοιχείων των γλωσσών της UML και τη SysML. Η τομή των δύο κύκλων, που δείχνεται από το τμήμα που περιγράφεται ώς “Η UML όπως επαναχρησιμοποιείται από την SysML”, υποδεικνύει τα στοιχεία των μοντέλων της UML που επαναχρησιμοποιούνται από τη SysML. Η περιοχή “Επεκτάσεις της SysML στη UML” υποδεικνύει τα νέα στοιχεία μοντελοποίησης της SysML τα οποία δεν έχουν κοινά στοιχεία με αυτά της UML.

Ας σημειώσουμε πως υπάρχει και ένα μέρος της UML 2.0 που δεν είναι απαραίτητο για να υλοποιηθεί η SysML, που δείχνεται ώς “ UML μή απαραίτητη για τη SysML ”.



Σχήμα 5.8.2-1: Οπτικοποίηση των γλωσσών UML και SysML

5.8.2.1 Σχέσεις με άλλα πρότυπα

Η SysML ορίζεται ώς προέκταση της UML 2.0 Superstructure Specification του οργανισμού OMG (αριθμός κειμένου του OMG: ptc/2004-10-02). Η SysML συμβαδίζει με δύο εξελισσόμενα πρότυπα, όπως αναφέρθηκε παραπάνω, και υποστηρίζει την πρωτοβουλία της Αρχιτεκτονικής Μοντέλων (Model Driven Architecture) του OMG με την επαναχρησιμοποίηση του προτύπου της UML, και του αρχιτεκτονικού ευθυγραμμισμού της με τα πρότυπα OMG XMI 2.0 και ISO AP-233. Οι προϋποθέσεις της SysML ορίζονται χρησιμοποιώντας μια προσέγγιση μεταμοντέλου που προσαρμόζει επίσημες τεχνικές προϋποθέσεων. Αν και η προσέγγιση αυτή στερείται του κύρους μιας επίσημης μεθόδου προϋποθέσεων, προσφέρει τα πλεονεκτήματα του να είναι πιο διαισθητική και “πραγματική” στους περισσότερους υλοποιητές και ασκούμενους.

5.8.2.2 Αρχιτεκτονική

Η γλώσσα SysML επαναχρησιμοποιεί και επεκτείνει πολλά από τα στοιχεία της UML. Πολλαπλοί μηχανισμοί επεκτάσεων χρησιμοποιούνται που συμπεριλαμβάνουν στερεότυπα (stereotypes), μετακλάσεις (metaclasses) και βιβλιοθήκη μοντέλων (model library). Το μοντέλο της SysML δημιουργείται με την τη μετατροπή σε στιγμιότυπα των στερεοτύπων και των μετακλάσεων που ορίζονται από το μεταμοντέλο (metamodel) της SysML και υποκατηγοριοποιώντας τις κλάσεις στο μοντέλο βιβλιοθήκης της SysML. Για να κατανοήσουμε τη δομή του πακέτου της SysML (SysML package), είναι καλό να καταλάβουμε τη δομή του πακέτου της UML, το οποίο και επεκτείνει. Κάθε πακέτο περιέχει μετακλάσεις που ορίζουν τα

βασικά στοιχεία της γλώσσας. Οι εξαρτήσεις μεταξύ των πακέτων δείχνονται ώς διακεκομμένα βέλη. Συγκεντρωτικά, όλα τα πακέτα μαζί με τα διαγράμματα κλάσεων τους, τις περιγραφές των κλάσεων και τους περιορισμούς, αναφέρονται ώς το "μεταμοντέλο της UML". Το μεταμοντέλο της UML ορίζει πακέτα δομικά, συμπεριφοράς και ουδέτερα, καθώς και δίνει τη δυνατότητα για την προσαρμογή της γλώσσας. Τα δομικά πακέτα περιλαμβάνουν Κλάσεις (Classes), Σύνθετες δομές (Composite structures), Συστατικά (Components) και Αναπτύξεις (Deployments). Τα πακέτα συμπεριφοράς περιλαμβάνουν Ενέργειες (Actions), Δραστηριότητες (Activities), Αλληλεπιδράσεις (Interactions), Καταστάσεις (State Machines), και Σενάρια (Use cases).

5.8.2.3 Επισημοποίηση της γλώσσας

Οι προδιαγραφές της SysML ορίζονται χρησιμοποιώντας μια παραλλαγή της προσέγγισης που προσαρμόζει επίσημες τεχνικές προδιαγραφών. Οι επίσημες τεχνικές προδιαγραφών χρησιμοποιούνται για να αυξήσουν την ακρίβεια και την ορθότητα των προδιαγραφών.

Ακολούθως είναι οι στόχοι των τεχνικών προδιαγραφών που χρησιμοποιούνται για να ορίσουν τη SysML:

- Ορθότητα
- Ακρίβεια
- Συνέπεια
- Κατανοητότητα

Η τεχνική προδιαγραφών που χρησιμοποιείται σε αυτές τις προδιαγραφές περιγράφει την SysML ως επέκταση της UML που ορίζεται με τη χρήση στερεοτύπων και μετακλάσεων.

5.8.2.4 Η μετάβαση από την UML 2.0 στην SysML

Ένας από τους στόχους της SysML ήταν να προσπαθήσει να χρησιμοποιήσει όσο το δυνατόν περισσότερο την UML 2.0 και να αποφύγει να κάνει αλλαγές σε αυτή, εκτός κι αν ήταν απολύτως απαραίτητο. Κατά τη διάρκεια της αρχικής φάσης της ανάπτυξης της SysML πολλά νέα διαγράμματα ήταν υποψήφια να ενταχθούν σε αυτήν αλλά τελικά μονάχα δύο (2) από αυτά εντάχθηκαν, το Διάγραμμα Απαιτήσεων (Requirements Diagram) και το Παραμετρικό Διάγραμμα Εξισώσεων (Parametric Equations Diagram).

Συνοψίζοντας, η SysML επέκτεινε την UML 2.0 επιφέροντας κάποιες αλλαγές στα δομικά στοιχεία της, είτε με τη μορφή δανεισμού, επέκτασης και τροποποίησης, είτε με τη μορφή εισαγωγής νέων στοιχείων.

Ως μεγαλύτερη αλλαγή θα μπορούσε να θεωρηθεί η εισαγωγή των δύο (2) νέων διαγραμμάτων που αναφέρθηκαν παραπάνω.

Μια λεπτομερέστερη ανάλυση της SysML μπορεί να βρεθεί στην επίσημη ιστοσελίδα http://www.sysml.org/ .

ΒΙΒΛΙΟΓΡΑΦΙΑ

Fowler, Martin. “UML Distilled: A Brief Guide to the Standard Object Modeling Language”, Third Edition, Addison-Wesley, 1998

Joseph Schmuller. “Teach Yourself UML in 24h”, Third Edition, Sams Publishing, 2004

Jason T. Roff. “UML: A beginner’s guide”, McGraw-Hill/Osborne, 2003

Rational Software, Microsoft, Hewlett-Packard, Oracle Sterling Software, MCI Systemhouse, Unisys, ICON Computing IntelliCorp, i-Logix, IBM, ObjecTime, Platinum Technology, Ptech Taskon, Reich Technologies, Softeam.
“UML Summary, Version 1.1”, 1 September 1997

Rational Software, Microsoft, Hewlett-Packard, Oracle Sterling Software, MCI Systemhouse, Unisys, ICON Computing IntelliCorp, i-Logix, IBM, ObjecTime, Platinum Technology, Ptech Taskon, Reich Technologies, Softeam.
“UML Notation Guide, Version 1.1”, 1 September 1997

IBM Rational Software, http://www-306.ibm.com/software/rational/

Visual Paradigm for UML, http://www.visual-paradigm.com/

Poseidon for UML, http://www.gentleware.com/index.php

Rational Rose, http://www-306.ibm.com/software/rational/

Microsoft Visio, http://office.microsoft.com/en-us/FX010857981033.aspx

ARTiSAN real-time studio, http://www.artisansw.com/