Περιεχόμενα

ΚΕΦΑΛΑΙΟ 1

Εισαγωγή στη UML

Γιώργος Κακαρόντζας

1.1 Εισαγωγή

Σκοπός του κεφαλαίου αυτού είναι να γνωρίσει ο αναγνώστης την Ενοποιημένη Γλώσσα Μοντελοποίησης (UML – Unified Modeling Language). Θα περιγράψουμε τι είναι η UML δίνοντας και κάποια ιστορικά στοιχεία. Στη συνέχεια, θα δούμε αναλυτικά τους διάφορους τύπους διαγραμμάτων που υποστηρίζει η UML καθώς και την αντιστοίχησή τους με οικείες στους προγραμματιστές κατασκευές των αντικειμενοστρεφών γλωσσών προγραμματισμού και ειδικότερα της Java. Επίσης για λόγους πληρότητας θα σκιαγραφήσουμε μία μεθοδολογική προσέγγιση για την ανάλυση, σχεδίαση και ανάπτυξη εφαρμογών χρησιμοποιώντας τη UML, αν και τα επόμενα κεφάλαια των παραδειγμάτων εφαρμογής (case studies) θα εμβαθύνουν πολύ περισσότερο στη μεθοδολογία και θα προσφέρουν καλύτερη κατανόηση όσων θα αναφέρουμε εδώ.

Μετά τη μελέτη του κεφαλαίου ο αναγνώστης θα είναι σε θέση:

- Να εξηγήσει τι είναι η UML και για ποιο λόγο χρειάζεται.
- Να χρησιμοποιήσει τις διαγραμματικές τεχνικές της UML για την ανάπτυξη αντικειμενοστρεφών συστημάτων λογισμικού.
- Να εξηγήσει ποια είναι η αντιστοίχηση των εννοιών της UML σε μία τυπική αντικειμενοστρεφή γλώσσα προγραμματισμού.
- Να εξηγήσει τι είναι μια μεθοδολογία ανάπτυξης, ποια είναι η διαφορά της με τη UML και να αρχίσει να προσεγγίζει μεθοδολογικά τη διαδικασία ανάπτυξης αντικειμενοστρεφούς λογισμικού.

1.2 Η γλώσσα UML

Η UML αποτελεί μια γλώσσα απεικόνισης ή μοντελοποίησης ενός πληροφοριακού συστήματος βασισμένου σε αντικείμενα (αντικειμενοστρεφούς συστήματος). Όπως σε όλα τα σύνθετα έργα, έτσι και στα έργα πληροφορικής η ανάγκη της μοντελοποίησης πριν την κατασκευή του συστήματος είναι επιτακτική. Η μοντελοποίηση ενός συστήματος παρέχει τη δυνατότητα της αφαίρεσης των ασήμαντων γι'αυτό λεπτομερειών και της εστίασης στις σημαντικές λεπτομέρειες[1] του συστήματος που είναι απαραίτητο να κατανοηθούν πριν την κατασκευή του. Επίσης, δίνει τη δυνατότητα του πειραματισμού με διαφορετικές λύσεις ή προσεγγίσεις για το ίδιο πρόβλημα. Καθιστά εφικτή τη δυνατότητα ανάλυσης, σχεδιασμού, καταγραφής και παρακολούθησης της προόδου ενός έργου πληροφορικής. Τέλος, προσφέρει μια κοινή γλώσσα για την επικοινωνία όσων εμπλέκονται στην κατασκευή του συστήματος. Χωρίς ένα μοντέλο δεν είναι δυνατόν να προσεγγίσει κανείς την πολυπλοκότητα των σύγχρονων πληροφοριακών συστημάτων.

Φυσικά, η μοντελοποίηση των συστημάτων πληροφορικής δεν είναι κάτι νέο. Μοντέλα συστημάτων πληροφορικής κατασκευάζονται εδώ και δεκαετίες, με διαφορετικές όμως τεχνικές. Ο τρόπος με τον οποίο σχεδιάζουμε ένα σύστημα, είναι άρρηκτα συνδεδεμένος με την τεχνολογία που θα χρησιμοποιήσουμε για να το αναπτύξουμε. Έτσι, από τα διαγράμματα ροής προγράμματος περάσαμε με την εμφάνιση του δομημένου προγραμματισμού σε πιο προχωρημένες τεχνικές αναπαράστασης και σχεδίασης συστημάτων που περιλάμβαναν πιο σύνθετες διαγραμματικές τεχνικές για τον έλεγχο των πιο πολύπλοκων σύγχρονων συστημάτων. Αυτές οι τεχνικές περιλάμβαναν λεξικά δεδομένων, διαγράμματα ροής δεδομένων, πίνακες αποφάσεων κ.λπ. Σήμερα τα συστήματα που αναπτύσσονται είναι σχεδόν στο σύνολό τους αντικειμενοστρεφή— και υπάρχουν καλοί λόγοι γι' αυτό:

- Το αντικειμενοστρεφές λογισμικό είναι ευκολότερο στην αρχική του σύλληψη, μια και τα αντικείμενα είναι — εν μέρει — οντότητες του υπαρκτού κόσμου του πεδίου προβλήματος (π.χ. σε ένα τραπεζικό σύστημα θα υπάρχουν αντικείμενα όπως «Τραπεζικός Λογαριασμός», «Πελάτης» κ.λπ.).
- Το αντικειμενοστρεφές λογισμικό είναι ευκολότερο στην εξέλιξή του. Με τη χρήση μάλιστα των διασυνδέσεων (interfaces), δηλαδή με τη χρήση τύπων — σε αντιδιαστολή με τη χρήση κλάσεων (classes), δηλαδή υλοποίησης, μπορεί πράγματι να επιτρέψει την εξέλιξη του λογισμικού με πρωτοφανή ευκολία για τη βιομηχανία παραγωγής λογισμικού. Η χρήση των διασυνδέσεων δεν δεσμεύει τη χρήση ενός

1 Χρησιμοποιείται εσκεμμένα ο όρος «σημαντικές λεπτομέρειες» για να επισημάνουμε πως με την αφαίρεση δεν εννοούμε πως είμαστε ασαφείς ή ανούσιοι. Αντίθετα είμαστε πολύ σαφείς και ουσιαστικοί χωρίς να προσπαθούμε να μοντελοποιήσουμε ασήμαντα ή τετριμμένα σημεία του συστήματος. Προσπαθούμε να μοντελοποιήσουμε αρκετά ώστε να είμαστε σε θέση αρχικά να κατανοήσουμε και στη συνέχεια να κατασκευάσουμε το σύστημα.

αντικειμένου με τις λεπτομέρειες της υλοποίησής του κάνοντας εφικτή την αντικατάστασή του από ένα άλλο (εξελιγμένο) συμβατό αντικείμενο.

- Η αντικειμενοστρεφής προσέγγιση επιτρέπει τη δημιουργία λογισμικού με βάση τα συστατικά (components) [Szype2002]. Αυτό αποτελεί τη όραμα της βιομηχανίας λογισμικού, μια και στο εγγύς μέλλον θα μπορούμε να κατασκευάσουμε προγράμματα συνθέτοντας συστατικά.
- Σύγχρονες τεχνολογίες κατασκευής κατανεμημένων συστημάτων προσανατολισμένων στη σύνδεση επιχειρήσεων (Business-to-Business – B2B) και στη σύνδεση επιχειρηματικών εφαρμογών (Enterprise Application Integration – EAI), όπως οι υπηρεσίες του παγκόσμιου ιστού (Web Services) [Newco2002], έχουν σαν τεχνολογικό υπόβαθρο αντικειμενοστρεφείς γλώσσες προγραμματισμού (π.χ. Java, C#).

Συμπερασματικά, είναι σχεδόν αδύνατο να ασχοληθεί σήμερα κανείς με την ανάπτυξη λογισμικού χωρίς να γνωρίζει να αναπτύσσει αντικειμενοστρεφή συστήματα. Ως εκ τούτου είναι αναγκαία η γνώση της UML για τη μοντελοποίηση αυτών των συστημάτων, μια και η UML αποτελεί την πρότυπη γλώσσα μοντελοποίησης αντικειμενοστρεφών συστημάτων. Η UML αποτελεί πρότυπο του OMG (Object Management Group)[2]. Ο OMG είναι ένας διεθνής μη κερδοσκοπικός οργανισμός που παράγει και διαχειρίζεται πρότυπα για τη διαλειτουργικότητα των επιχειρηματικών εφαρμογών λογισμικού, μεταξύ των οποίων και η UML. Ο OMG έχει μέλη σχεδόν όλες τις μεγάλες εταιρίες παραγωγής λογισμικού και εκατοντάδες μικρότερες. Η προτυποποίηση της UML από τον OMG, η γενικότερη αποδοχή της από τη βιομηχανία λογισμικού, καθώς και το γεγονός πως έχει περάσει αρκετός χρόνος από το 1997 που υιοθετήθηκε το πρώτο πρότυπο της UML από τον OMG, προσδίδουν στη UML το σημαντικό χαρακτηριστικό της διάρκειας στον χρόνο, κάτι που είναι πολύ σημαντικό για τη βιομηχανία λογισμικού, που εξελίσσεται με ραγδαίους ρυθμούς.

1.2.1 Μεθοδολογίες ανάπτυξης λογισμικού

Η UML είναι μία γλώσσα μοντελοποίησης και όχι μία μεθοδολογία ανάπτυξης έργων λογισμικού. Μια μεθοδολογία ανάπτυξης λογισμικού παρέχει μια συστηματική προσέγγιση στη διαδικασία ανάλυσης, σχεδίασης, κατασκευής και εξέλιξης ενός έργου πληροφορικής. Μια μεθοδολογία είναι ουσιαστικά μια σειρά σταδίων τα οποία περιγράφουν συγκεκριμένες εργασίες. Η UML είναι ουδέτερη σε σχέση με τις μεθοδολογίες, χωρίς να επιβάλλει κάποια συγκεκριμένη μεθοδολογία, και μπορεί να χρησιμοποιηθεί με διάφορους τρόπους. Υπάρχουν πολλές μεθοδολογίες ανάπτυξης λογισμικού (όπως οι Unified Process [Kruch2003], eXtreme Programming [Beck2004], Catalysis [D' Soura 1998],

---

[2] Ο δικτυακός τόπος του OMG είναι http://www.omg.org

Syntropy [Cook1994] κ.ο.κ.). Κάθε μία από αυτές θεωρείται πως είναι πιο κατάλληλη για κάποιο συγκεκριμένο τύπο λογισμικού· για παράδειγμα η Catalysis θεωρείται πως είναι πιο κατάλληλη για συστήματα βασισμένα σε συστατικά.

Ίσως η πιο διαδεδομένη μεθοδολογία είναι η *ενοποιημένη προσέγγιση* (Unified Process ή, εν συντομία, UP). Η UP αναπτύχθηκε αρχικά από την εταιρία Rational[3]. Τα βασικά στάδια της UP είναι τα εξής:

1. Η *σύλληψη* (inception) είναι η πρώτη φάση της ενοποιημένης προσέγγισης, όπου παρουσιάζεται η αρχική ιδέα του συστήματος τουλάχιστον μέχρι του σημείου που είναι αρκετά καλά θεμελιωμένη έτσι ώστε να επιτρέψει την είσοδο στη φάση επεξεργασίας.
2. Η *λεπτομερής επεξεργασία* (elaboration) είναι η δεύτερη φάση, όπου περιγράφεται ο σκοπός του συστήματος καθώς και η υψηλού επιπέδου αρχιτεκτονική του. Σε αυτή τη φάση προσδιορίζονται οι απαιτήσεις του συστήματος.
3. Η *κατασκευή* (construction) είναι η τρίτη φάση, όπου σχεδιάζεται και κατασκευάζεται το λογισμικό.
4. Η *μετάβαση* (transition) είναι η τέταρτη φάση της διαδικασίας, όπου το λογισμικό υπόκειται σε έλεγχο και τελικά παραδίδεται στους χρήστες. Η φάση της μετάβασης σηματοδοτεί την έναρξη της φάσης της συντήρησης λογισμικού και όχι το τέλος της διαδικασίας ανάπτυξης.

Μετά από αυτή τη σύντομη γνωριμία με την UP, ας δούμε τους βασικούς τύπους διαγραμμάτων που παρέχει η UML.

1.2.2 Τύποι διαγραμμάτων της UML

Τα βασικά διαγράμματα που υπάρχουν στη UML είναι τα ακόλουθα:

- *Διάγραμμα περιπτώσεων χρήσης* (Use Case Diagram): Αποτυπώνει τις προδιαγραφές του υπό κατασκευή συστήματος.
- *Διάγραμμα κλάσεων* (Class Diagram): Αποτυπώνει τις κλάσεις του συστήματος, καθώς και τις στατικές σχέσεις μεταξύ αυτών των κλάσεων.

---

[3] Στην εταιρεία Rational Software εργάζονταν οι τρεις πρωτεργάτες της UML, οι Grady Booch, James Rumbaugh, και Ivar Jacobson. Η Rational προώθησε τη UML με την κατασκευή ενός ολοκληρωμένου CASE συστήματος, του Rational Rose. Τα τελευταία χρόνια η Rational εξαγοράσθηκε από την IBM η οποία και συνεχίζει την εξέλιξη των προϊόντων της εταιρίας. Ο δικτυακός τόπος της εταιρείας είναι http:// www.ibm.com/software/rational.

- *Διαγράμματα συμπεριφοράς* (Behavior Diagrams): Αποτυπώνουν τη δυναμική συμπεριφορά του συστήματος. Στα διαγράμματα συμπεριφοράς ανήκουν τα εξής διαγράμματα:
  - *Διάγραμμα καταστάσεων* (Statechart Diagram): Περιγράφει τις καταστάσεις ενός αντικειμένου του συστήματος και τον τρόπο με τον οποίο το αντικείμενο αυτό αλλάζει κατάσταση ως ανάδραση σε συμβάντα
  - *Διάγραμμα δραστηριοτήτων* (Activity Diagram): Παρέχει έναν τρόπο για την αποτύπωση των δραστηριοτήτων που λαμβάνουν χώρα στο σύστημα, συμπεριλαμβανομένων και των παράλληλων δραστηριοτήτων που μπορεί να συμβαίνουν σε αυτό.
  - *Διάγραμμα αλληλεπίδρασης* (Interaction Diagrams): Δύο ισοδύναμοι τύποι διαγραμμάτων που χρησιμοποιούνται για την αποτύπωση της αλληλεπίδρασης των αντικειμένων του συστήματος:
    - *Διάγραμμα ακολουθίας* (Sequence Diagram): Δίνει έμφαση στη χρονική ακολουθία της αλληλεπίδρασης μεταξύ των αντικειμένων και απεικονίζει τις ανταλλαγές των μηνυμάτων που λαμβάνουν χώρα μεταξύ αυτών για την επίτευξη κάποιου συγκεκριμένου στόχου στο πλαίσιο μιας περίπτωσης χρήσης
    - *Διάγραμμα συνεργασίας* (Collaboration Diagram): Το διάγραμμα αυτό είναι ισοδύναμο με το διάγραμμα ακολουθίας με την έννοια ότι αποτυπώνονται σε αυτό οι ίδιες πληροφορίες. Η διαφορά είναι ότι απεικονίζονται οι σύνδεσμοι μεταξύ των αντικειμένων που αλληλεπιδρούν και δεν είναι τόσο ξεκάθαρη η αποτύπωση της χρονικής σειράς των μηνυμάτων
- *Διαγράμματα υλοποίησης* (Implementation Diagrams): Δύο τύποι διαγραμμάτων σχετικοί με την υλοποίηση του συστήματος:
  - *Διάγραμμα συστατικών* (Component Diagram): Δίνει τη δυνατότητα να εμφανίσουμε συστατικά με τις δημόσιες διασυνδέσεις που αυτά παρέχουν.
  - *Διάγραμμα διάταξης* (Deployment Diagram): Δείχνει τη διάταξη των συστατικών ενός συστήματος κατά τον χρόνο εκτέλεσης και τους κόμβους στους οποίους είναι τοποθετημένα αυτά τα συστατικά. Είναι αρκετά σημαντικό για την περιγραφή σύνθετων κατανεμημένων συστημάτων που έχουν διάφορα υποσυστήματα διεσπαρμένα στον χώρο.

Για την περιγραφή ενός συστήματος χρησιμοποιούμε ένα σύνολο διαγραμμάτων διαφορετικού τύπου, το καθένα από τα οποία περιγράφει μια διαφορετική οπτική γωνία του συστήματος. Έτσι χρησιμοποιούμε τα διαγράμματα κλάσεων για την περιγραφή των στατικών σχέσεων μεταξύ των κλάσεων, τα διαγράμματα συμπεριφοράς (κατάστασης, ακολουθίας κ.ο.κ.) για την περιγραφή της δυναμικής συμπεριφοράς του συστήματος και τα

διαγράμματα υλοποίησης (συστατικών, διάταξης) για την καταγραφή των λεπτομερειών υλοποίησης του συστήματος. Τα διαγράμματα περιπτώσεων χρήσης είναι μία ειδική περίπτωση διαγραμμάτων, με την έννοια πως δεν έχουν άμεση σχέση με αντικείμενα ή αντικειμενοστρεφή συστήματα, αλλά αποτελούν ένα μηχανισμό καταγραφής των προδιαγραφών του συστήματος.

1.3 Διαγράμματα της UML

Σε αυτή την ενότητα θα παρουσιάσουμε αναλυτικά τους διάφορους τύπους διαγραμμάτων της UML. Θα ξεκινήσουμε από τις περιπτώσεις χρήσης και θα αναφερθούμε στη διαδικασία εύρεσης των προδιαγραφών. Στη συνέχεια θα δούμε τα διαγράμματα κλάσεων, θα συνεχίσουμε με τα διαγράμματα συμπεριφοράς, και τέλος θα αναφερθούμε στα διαγράμματα υλοποίησης.

1.3.1 Ανάλυση προδιαγραφών και τα διαγράμματα περιπτώσεων χρήσης

Το βασικότερο θέμα στην κατασκευή ενός πληροφοριακού συστήματος είναι η κατασκευή του ορθού συστήματος: του συστήματος που ικανοποιεί τις απαιτήσεις των χρηστών του συστήματος και των άλλων συμμετεχόντων. Πολλά συστήματα πληροφορικής αποτυγχάνουν να ικανοποιήσουν αυτόν το φαινομενικά στοιχειώδη στόχο: δεν ικανοποιούν τις προδιαγραφές του χρήστη.

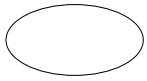
Το μοντέλο περιπτώσεων χρήσης δίνει έμφαση στη λειτουργικότητα ενός συστήματος, όπως αυτή είναι ορατή από τους εξωτερικούς χρήστες του. Μια περίπτωση χρήσης διαμερίζει τη λειτουργικότητα ενός συστήματος σε συναλλαγές (περιπτώσεις χρήσης) που έχουν νόημα για τους χρήστες του συστήματος (χειριστές).

Τα βασικά διαγραμματικά στοιχεία των διαγραμμάτων περιπτώσεων χρήσης είναι τα εξής:

Περίπτωση χρήσης

Αναπαριστά ένα στόχο για έναν εξωτερικό *χειριστή* (actor) του συστήματος. Οι χειριστές ενός συστήματος μπορεί να είναι άνθρωποι (π.χ. γραμματέας, ταμίας) αλλά ενδέχεται να είναι και εξωτερικά συστήματα (π.χ. διατραπεζικό σύστημα συναλλαγών), τα οποία είναι απαραίτητα για τη λειτουργία του υπό ανάπτυξη συστήματος. Το σύμβολο για μία περίπτωση χρήσης είναι η έλλειψη, μέσα στην οποία αναγράφεται το όνομα της περίπτωσης χρήσης, όπως, για παράδειγμα, η περίπτωση χρήσης «Ανάληψη Μετρητών» η οποία φαίνεται στην Εικόνα 1-1. Μία περίπτωση χρήσης αποτελεί ένα στόχο υψηλού επιπέδου για τον χειριστή της· είναι κάτι που του αποδίδει κάποιο απτό αποτέλεσμα που έχει αξία γι' αυτόν [Cockb1997]. Για παράδειγμα, η περίπτωση χρήσης «Ανάληψη Μετρητών» είναι

κάτι που έχει αξία για έναν πελάτη ενός μηχανήματος ATM μιας τράπεζας. Ένα αντιπαράδειγμα θα ήταν η περίπτωση χρήσης «Σύνδεση» για ένα μηχάνημα ATM: η σύνδεση ενός πελάτη με την παροχή του κωδικού του είναι απαραίτητη για τη χρήση του ATM, αλλά δεν είναι κάτι που έχει κάποια αξία για τον πελάτη.



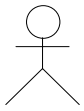
Ανάληψη Μετρητών

Εικόνα 1-1: Σύμβολο περίπτωσης χρήσης

Μία περίπτωση χρήσης περιλαμβάνει συνήθως πολλά εναλλακτικά σενάρια, τα οποία ονομάζονται *επεκτάσεις* (extensions). Η ιδέα είναι ότι μία περίπτωση χρήσης έχει ένα κύριο σενάριο όπου όλα πάνε καλά (happy path) και αρκετές επεκτάσεις στις οποίες κάτι δεν πάει καλά ή προκύπτει μία εξαίρεση. Στο παράδειγμα της ανάληψης μετρητών, θα είχαμε ένα σενάριο όπου όλα θα πήγαιναν καλά (θα υπήρχαν αρκετά χρήματα στο μηχάνημα ATM, ο χρήστης θα έδινε το σωστό PIN, θα ζητούσε ένα ποσό που θα καλυπτόταν από το υπόλοιπο του λογαριασμού του κ.λπ.) και αρκετά σενάρια στα οποία κάτι από τα παραπάνω δεν θα πήγαινε καλά και συνεπώς θα έπρεπε να το χειριστούμε με κάποιο τρόπο (π.χ. μετά από 3 διαδοχικές λάθος προσπάθειες για την εισαγωγή του PIN, το ATM θα μπορούσε να δεσμεύσει την κάρτα).

Χειριστής

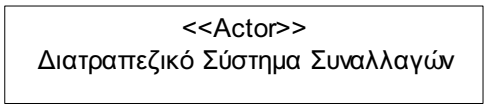
Όπως αναφέραμε και στην προηγούμενη παράγραφο, ο χειριστής ενός συστήματος μπορεί να είναι άνθρωπος ή υποσύστημα. Το σύμβολο που χρησιμοποιείται για τους χειριστές φαίνεται στην Εικόνα 1-2, με ένα όνομα κάτω από τη φιγούρα που επισημαίνει το ρόλο που παίζει στα πλαίσια μίας περίπτωσης χρήσης. Το ίδιο σύμβολο μπορεί να χρησιμοποιηθεί και για τους χειριστές που είναι άνθρωποι αλλά και για τους χειριστές που είναι άλλα εξωτερικά συστήματα. Αν θέλουμε να δώσουμε έμφαση στην περίπτωση όπου ο χειριστής είναι ένα εξωτερικό σύστημα, μπορούμε να χρησιμοποιήσουμε ένα εναλλακτικό σύμβολο —όπως φαίνεται και στην Εικόνα 1-3— όπου το εξωτερικό σύστημα απεικονίζεται σαν μία κλάση με το στερεότυπο «Actor».



Πελάτης

Εικόνα 1-2: Σύμβολο χειριστή

Θα πρέπει να επισημάνουμε το σημαντικό ρόλο που παίζουν οι χειριστές στη διαδικασία εύρεσης των προδιαγραφών: οι χειριστές (χρήστες και συστήματα) αποτελούν την πηγή από την οποία προκύπτουν οι προδιαγραφές του συστήματος. Στην ερώτηση "ποιες λειτουργίες πρέπει να προσφέρει το σύστημα" η απάντηση είναι "αυτές που χρειάζονται για την ικανοποίηση των στόχων των χειριστών του". Επομένως, είναι κρίσιμο να βρούμε τους χειριστές του συστήματος (ανθρώπους, υποσυστήματα και εξωτερικά συστήματα) επειδή οι στόχοι τους θα μας υποδείξουν τις προδιαγραφές του υπό ανάπτυξη συστήματος. Αυτές ακριβώς οι προδιαγραφές αποτυπώνονται στο διάγραμμα περιπτώσεων χρήσης.



Εικόνα 1-3: Εναλλακτικό σύμβολο χειριστή για εξωτερικά συστήματα

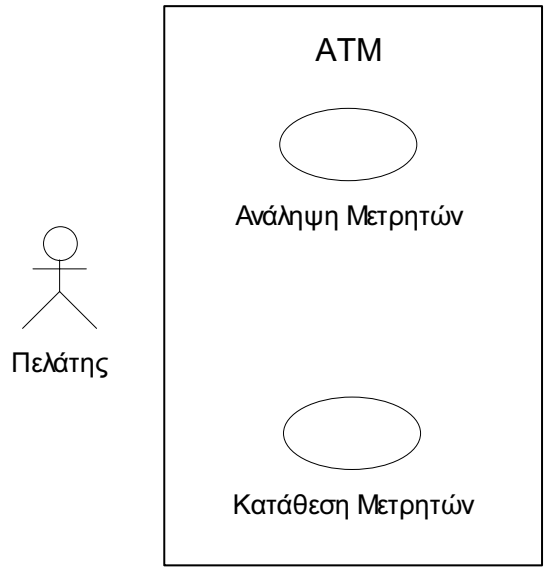
Σύστημα

Για να διακρίνουμε τις προδιαγραφές που βρίσκονται μέσα στα πλαίσια του υπό ανάπτυξη συστήματος από τα πιθανά εξωτερικά συστήματα και τους χρήστες, περιλαμβάνουμε τις περιπτώσεις χρήσης σε ένα πλαίσιο με τίτλο το όνομα του συστήματος όπως φαίνεται και στην Εικόνα 1-4. Μέσα στο πλαίσιο του συστήματος τοποθετούμε τις περιπτώσεις χρήσης και έξω από αυτό τους χειριστές του συστήματος.

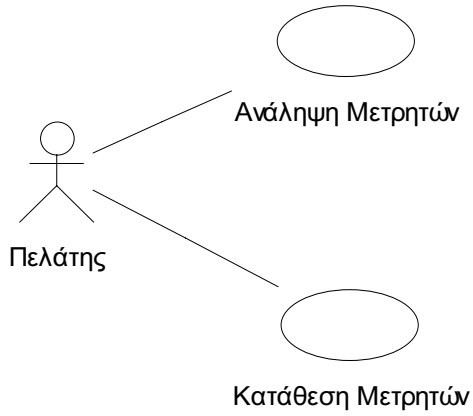
Σχέση

Υποδηλώνει τη σχέση ενός χειριστή με μία περίπτωση χρήσης. Ένα χειριστής, ενδέχεται να είναι ο βασικός (primary) για μία περίπτωση χρήσης, αλλά να σχετίζονται επίσης με αυτήν και άλλοι χειριστές ή εξωτερικά συστήματα. Για παράδειγμα, ένας πελάτης ενός ATM είναι ο βασικός χειριστής της περίπτωσης χρήσης «Ανάληψη Μετρητών». Η σχέση συμβολίζεται με μία γραμμή που συνενώνει τον χειριστή με την περίπτωση χρήσης, όπως δείχνει και η Εικόνα 1-5, στην οποία ένας πελάτης σχετίζεται με τις περιπτώσεις χρήσης «Ανάληψη Μετρητών» και «Κατάθεση Μετρητών».

Αξίζει να σημειωθεί εδώ πως αν μιλούσαμε για ένα ταμείο μίας τράπεζας και όχι για ένα ATM, ο βασικός χειριστής για την ανάληψη μετρητών θα ήταν ο ταμίας της τράπεζας και όχι ο πελάτης, μια και ο ταμίας θα κάνει την ανάληψη για λογαριασμό του πελάτη. Η διάκριση αυτή είναι σημαντική διότι οι προδιαγραφές μπορεί να είναι διαφορετικές για διαφορετικούς χειριστές, ακόμα και αν πρόκειται για την ίδια ουσιαστικά εργασία. Ο ταμίας της τράπεζας, για παράδειγμα, δεν απαιτείται να πιστοποιηθεί από το σύστημα κάθε φορά που κάνει ανάληψη μετρητών, σε αντίθεση με έναν πελάτη ενός ATM που απαιτείται να πιστοποιηθεί από το σύστημα για κάθε συναλλαγή.



Εικόνα 1-4: Πλαίσιο συστήματος στο διάγραμμα περιπτώσεων χρήσης

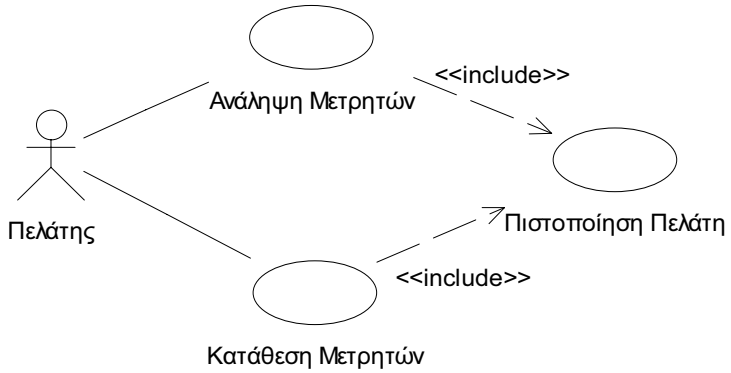


Εικόνα 1-5: Σχέση χειριστή με περιπτώσεις χρήσης

Συμπερίληψη

Η *συμπερίληψη* (include) είναι μία ειδική περίπτωση σχέσης, στην οποία σχετίζουμε δύο περιπτώσεις χρήσης. Η μία περίπτωση χρήσης συμπεριλαμβάνει την άλλη. Η έννοια της συμπερίληψης είναι υποχρεωτική, δηλαδή πάντα η μία περίπτωση χρήσης θα συμπεριλαμβάνει την άλλη. Χρησιμοποιούμε τη συμπερίληψη όταν η ίδια λειτουργικότητα περιλαμβάνεται σε περισσότερες από μία περιπτώσεις χρήσης. Για παράδειγμα, έχοντας τις περιπτώσεις χρήσης «Ανάληψη Μετρητών» και «Κατάθεση Μετρητών» για ένα ATM, θα μπορούσαμε να συμπεριλάβουμε και στις δύο μία περίπτωση χρήσης «Πιστοποίηση Πελάτη» η οποία θα αφορά την πιστοποίηση του πελάτη. Επειδή η πιστοποίηση του πελάτη (εισαγωγή PIN που αντιστοιχεί στην κάρτα, αναγνώριση αν ο πελάτης είναι έγκυρος, χειρισμός λάθους PIN κ.λπ.) είναι υποχρεωτική γι' αυτές και για άλλες περιπτώσεις χρήσης, το προτιμότερο είναι —όπως φαίνεται και στην Εικόνα 1-6— να δείξουμε την πιστοποίηση ως μία ξεχωριστή περίπτωση χρήσης, η οποία συμπεριλαμβάνεται από όλες τις άλλες περιπτώσεις χρήσης που χρειάζονται πιστοποίηση πελάτη.

Η φορά του βέλους στη συμπερίληψη είναι από την περίπτωση χρήσης που συμπεριλαμβάνει προς αυτήν που συμπεριλαμβάνεται.

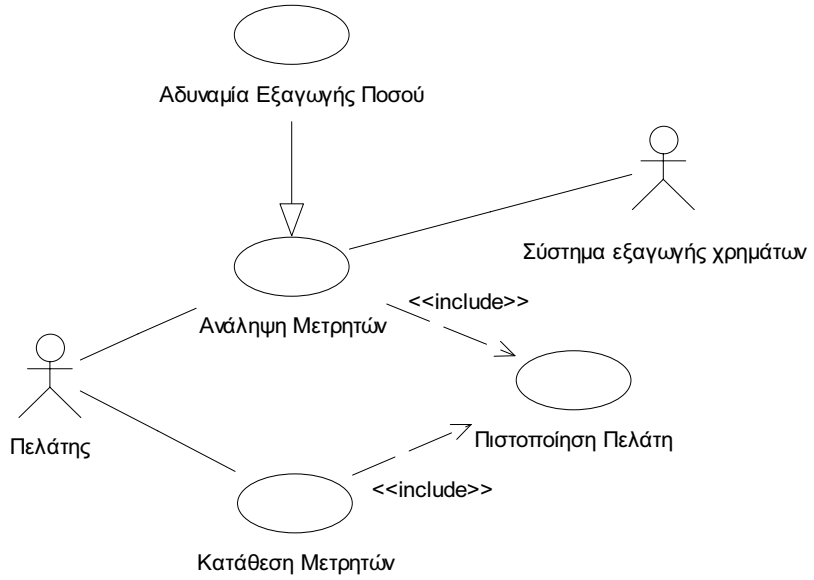


Εικόνα 1-6: Σχέση συμπερίληψης μεταξύ περιπτώσεων χρήσης

Επέκταση

Η *επέκταση* (extend) όπως και η συμπερίληψη, καθορίζει μία σχέση μεταξύ δύο περιπτώσεων χρήσης, στην οποία σχέση μία περίπτωση χρήσης επεκτείνεται προαιρετικά από μία άλλη, ανάλογα με τις επιλογές ή την κατάσταση κάποιου χειριστή (βασικού ή δευτερεύοντος). Η σχέση αυτή μπορεί να προκύψει σε συγκεκριμένα σημεία της λειτουργικότητας της βασικής ροής της περίπτωσης χρήσης τα οποία ονομάζονται *σημεία επέκτασης* (extension points). Έστω, για παράδειγμα, ότι για να γίνει η ανάληψη μετρητών, το σύστημα θα πρέπει να επικοινωνήσει με το εξωτερικό «Σύστημα Χειρισμού Χρημάτων», το οποίο είναι υπεύθυνο για την εξαγωγή των μετρητών. Το σύστημα αυτό είναι ένας από

τους χειριστές της περίπτωσης χρήσης «Ανάληψη Μετρητών» και ενδέχεται να μην έχει τα κατάλληλα χαρτονομίσματα για την εξαγωγή του ποσού που ζήτησε ο πελάτης. Στην περίπτωση αυτή, η περίπτωση χρήσης «Ανάληψη Μετρητών» επεκτείνεται από την περίπτωση χρήσης «Αδυναμία Εξαγωγής Ποσού», η οποία είναι η ίδια περίπτωση χρήσης μέχρι του σημείου όπου ο πελάτης ζητάει το ποσό και ο χειριστής «Σύστημα Εξαγωγής Χρημάτων» διαπιστώνει πως αδυνατεί να εξαγάγει αυτό το ποσό. Η Εικόνα 1-7 παρουσιάζει το παραπάνω παράδειγμα.



Εικόνα 1-7: Σχέση επέκτασης μεταξύ περιπτώσεων χρήσης

1.3.2 Διαγράμματα κλάσεων

Τα διαγράμματα κλάσεων είναι ο πρώτος τύπος διαγράμματος της UML που θα δούμε και ο οποίος έχει άμεση σχέση με τα αντικειμενοστρεφή συστήματα. Τα διαγράμματα περιπτώσεων χρήσης που συζητήσαμε είναι διαγράμματα καταγραφής προδιαγραφών και είναι χρήσιμα για κάθε τύπο συστήματος. Τα αντικειμενοστρεφή συστήματα όμως, λειτουργούν ως μία συλλογή συνεργαζόμενων αντικειμένων. Τα αντικείμενα αποτελούν στιγμιότυπα κλάσεων· η κλάση είναι ένας τύπος από τον οποίο δημιουργούνται κατά τη διάρκεια της εκτέλεσης του προγράμματος αντικείμενα που ανήκουν στον τύπο αυτόν. Είναι λοιπόν χρήσιμο να καταγράψουμε τις κλάσεις που ανήκουν σε ένα σύστημα και τις μεταξύ τους συσχετίσεις. Αυτό ακριβώς επιτυγχάνουμε με ένα διάγραμμα κλάσεων.