ΚΕΦΑΛΑΙΟ 9

ΕΙΣΑΓΩΓΗ ΣΤΗΝ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΕΦΗ ΣΧΕΔΙΑΣΗ

Σκοπός του κεφαλαίου είναι να εισάγει τον αναγνώστη στην αντικειμενοστρεφή σχεδίαση, συνεχίζοντας την αναφορά στην ενοποιημένη προσέγγιση ανάπτυξης λογισμικού.

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

- να κατασκευάζει UML διαγράμματα κλάσεων σχεδίασης ξεκινώντας από τις κλάσεις του πεδίου ανάλυσης,
- να κατασκευάζει UML διαγράμματα αλληλουχίας ή ακολουθίας ή σειράς (sequence diagrams) ξεκινώντας από τις αντίστοιχες περιγραφές συνεργασίας του πεδίου ανάλυσης,
- να ορίζει υποσυστήματα και να αντιστοιχεί σε αυτά τα συστατικά στοιχεία λογισμικού (components) που θα αποτελέσουν την υπό ανάπτυξη εφαρμογή,
- να κατασκευάζει το μοντέλο διάταξης ή εγκατάστασης (deployment model) μιας εφαρμογής λογισμικού,
- να ακολουθεί συγκεκριμένα βήματα ώστε να μεταβαίνει από το μοντέλο ανάλυσης στο μοντέλο σχεδίασης μιας εφαρμογής λογισμικού σύμφωνα με την ενοποιημένη προσέγγιση ανάπτυξης λογισμικού.

Έννοιες-κλειδιά

- *Κλάση σχεδίασης*
- *Διάγραμμα αλληλουχίας ή ακολουθίας ή σειράς UML (sequence diagram)*
- *Σύστημα – υποσύστημα*
- *Μοντέλο διάταξης ή εγκατάστασης UML (deployment model)*

- *Συστατικό στοιχείο (component)*
- *Μοντέλο σχεδίασης*

Σύνοψη

Η αντικειμενοστρεφής σχεδίαση είναι μια δημιουργική διαδικασία μετάβασης από το πεδίο του προβλήματος στο πεδίο της λύσης του. Σε αντιδιαστολή με τη δομημένη σχεδίαση, χρησιμοποιεί ως επί το πλείστον τα ίδια δομικά συστατικά λογισμικού: τις κλάσεις που έχουν ήδη «αποκαλυφθεί» κατά την ανάλυση, τις οποίες όμως εξειδικεύει σε επίπεδο κατασκευαστικής λεπτομέρειας. Τόσο η λεπτομερής πειθαρχία των διαδικασιών που ακολουθούνται όσο και η περιγραφή των αποτελεσμάτων τους είναι αντικείμενο που σχετίζεται με το εκάστοτε περιβάλλον ανάπτυξης λογισμικού.

Η UML μάς προσφέρει ένα σύνολο από εκφραστικά εργαλεία για την παράσταση των αποτελεσμάτων της σχεδίασης. Εκτός από τα ήδη γνωστά μας διαγράμματα κλάσεων και συνεργασίας, εισάγεται το διάγραμμα αλληλουχίας ή ακολουθίας ή σειράς, το διάγραμμα διάταξης και το διάγραμμα συστατικών. Το επίπεδο λεπτομέρειας κατά τη δημιουργία των διαγραμμάτων αυτών εξαρτάται από το περιβάλλον και τον τρόπο εργασίας κάθε σχεδιαστή. Σε κάθε περίπτωση, ένα διάγραμμα κλάσεων με «μικρή λεπτομέρεια» παραμένει διάγραμμα κλάσεων το οποίο, μολονότι δεν είναι κυριολεκτικά υλοποιήσιμο, εξακολουθεί να καθοδηγεί τον προγραμματιστή στην εργασία του. Όσο πιο πλήρες είναι το μοντέλο σχεδίασης και περιέχει, λόγου χάρη, συστατικά στοιχεία που κάνουν απολύτως αντιληπτή τη συμπεριφορά του λογισμικού (π.χ. διαγράμματα ακολουθίας), τόσο λεπτομερέστερη είναι η σχεδίαση.

Σε κάθε περίπτωση, η προσέγγιση που ακολουθούμε εδώ οριοθετείται από τον εκπαιδευτικό της χαρακτήρα και δεν στοχεύει στην πλήρη κάλυψη όλων των πλευρών του θέματος, για την οποία ο αναγνώστης παραπέμπεται στη βιβλιογραφία.

Εισαγωγικές παρατηρήσεις

Συνεχίζοντας την αναφορά στην αντικειμενοστρεφή ανάπτυξη λογισμικού, φτάνουμε στην αντικειμενοστρεφή σχεδίαση σύμφωνα με την ενοποιημένη προσέγγιση ανάπτυξης λογισμικού στην οποία αναφερόμαστε στο βιβλίο αυτό. Η σχεδίαση

λογισμικού κατά την αντικειμενοστρεφή φιλοσοφία οδηγεί στον καθορισμό κλάσεων οι οποίες θα υλοποιηθούν προγραμματιστικά, αντικείμενα των οποίων θα εκδηλώνουν την επιθυμητή συμπεριφορά κατά την εκτέλεση της εφαρμογής λογισμικού. Λέγοντας «καθορισμός» εννοούμε την περιγραφή των πεδίων και των μεθόδων των κλάσεων, καθώς και όλες τις απαραίτητες κατασκευαστικές λεπτομέρειες για τη συγγραφή του πηγαίου κώδικα που αντιστοιχεί σε αυτές.

Για όσους έχουν ασχοληθεί με την ανάπτυξη λογισμικού με τη δομημένη ανάλυση και σχεδίαση, υπάρχει μια σύγχυση: ενώ στη δομημένη ανάλυση και σχεδίαση στη φάση της ανάλυσης κάνουμε λόγο για άλλες έννοιες απ' ό,τι στη φάση της σχεδίασης, στην αντικειμενοστρεφή προσέγγιση δεν ισχύει το ίδιο. Κατά τη δομημένη ανάλυση μιλάμε για προδιαγραφές, διάγραμμα ροής δεδομένων, λεξικό δεδομένων και, ενδεχομένως, για διάγραμμα καταστάσεων (Κεφάλαια 4 και 5). Ακολούθως, κατά τη δομημένη σχεδίαση μιλάμε για διάγραμμα δομής προγράμματος και για ψευδοκώδικα. Δεν ισχύει το ίδιο για την αντικειμενοστρεφή ανάλυση και σχεδίαση. Στην περίπτωση αυτή, τόσο κατά την ανάλυση όσο και κατά τη σχεδίαση μιλάμε για κλάσεις: «κλάσεις ανάλυσης» και «κλάσεις σχεδίασης», όπως τις ονομάζουμε.

Ωστόσο, δεν είναι πάντα σαφές πότε μια κλάση παύει να είναι κλάση ανάλυσης και γίνεται κλάση σχεδίασης. Μια λεπτομερής αντικειμενοστρεφής ανάλυση οδηγεί σε ένα μοντέλο ανάλυσης που περιέχει κλάσεις οι οποίες θα μπορούσαν κάλλιστα να αποτελούν κλάσεις σχεδίασης μιας λιγότερο λεπτομερούς αντικειμενοστρεφούς σχεδίασης. Αν το καλοσκεφτούμε, η σύγχυση αυτή έχει μικρό νόημα και δεν θα πρέπει να δεσπόζει στην πρακτική ενός κατασκευαστή λογισμικού. Όπως κατά τη δομημένη ανάλυση και σχεδίαση που στην πράξη φτάνουμε μέχρι ενός επιπέδου λεπτομέρειας από το οποίο και μετά «αφήνουμε τη δουλειά στον προγραμματιστή», έτσι και στην περίπτωση της αντικειμενοστρεφούς τεχνολογίας. Το ποιο είναι το επίπεδο αυτό εξαρτάται από πολλούς παράγοντες, μεταξύ των οποίων το κόστος και η «προγραμματιστική κουλτούρα», αλλά και τα πρότυπα που επιβάλλεται κατά περίπτωση να ακολουθήσουμε. Τα πράγματα είναι πιο αυστηρά για εφαρμογές «κρίσιμης αποστολής» (mission critical applications) και λιγότερο για άλλες περιπτώσεις.

Η διάκριση μιας κλάσης σε κλάση ανάλυσης ή σχεδίασης αποτελεί πηγή σύγχυσης και για τους έχοντες εμπειρία στην αντικειμενοστρεφή ανάπτυξη. Ωστόσο, οι τελευταίοι έχουν πιο ισχυρή συνείδηση του ότι ακριβώς αυτή η εκφραστική δύναμη

των αντικειμένων ως εργαλείο μοντελοποίησης του κόσμου, όπως θα τονίσουμε επανειλημμένως στη συνέχεια, είναι που δημιουργεί την όποια σύγχυση η οποία, επαναλαμβάνουμε, έχει μικρό νόημα.

ΕΝΟΤΗΤΑ 9.1. Η ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΕΦΗΣ ΣΧΕΔΙΑΣΗ ΣΤΟ ΜΟΝΤΕΛΟ ΚΥΚΛΟΥ ΖΩΗΣ

Σύμφωνα με το μοντέλο κύκλου ζωής της ενοποιημένης προσέγγισης ανάπτυξης λογισμικού, η αντικειμενοστρεφής σχεδίαση ακολουθεί με τρόπο «φυσικό» την ανάλυση σε μια σειρά διαδικασιών που μας οδηγούν από το αρχικό πρόβλημα στην επίλυσή του. Παρότι η εξέλιξη αυτή είναι αναμενόμενη και κατ' αντιστοιχία με τις διαδικασίες της δομημένης ανάλυσης και σχεδίασης, στην αντικειμενοστρεφή της εκδοχή κρύβει μια σημαντική διαφορά, την οποία τονίσαμε στις εισαγωγικές παρατηρήσεις: η αντικειμενοστρεφής ανάλυση και η αντικειμενοστρεφής σχεδίαση είναι διαδικασίες τόσο κοντινές που πολλοί μηχανικοί λογισμικού τις αντιμετωπίζουν ως μια ενιαία διαδικασία στο πλαίσιο πάντα του εκάστοτε αντικειμένου, της κουλτούρας της ομάδας ανάπτυξης και των λοιπών περιορισμών ή απαιτήσεων της ανάπτυξης. Η εγγύτητα αυτή των δύο διαδικασιών είναι συνέπεια των αυξημένων, σε σχέση με τη δομημένη προσέγγιση, δυνατοτήτων μοντελοποίησης του φυσικού κόσμου από κλάσεις και αντικείμενα. Προκύπτει, δηλαδή, από την κεντρική αξία της αντικειμενοστρεφούς φιλοσοφίας. Σε αυτό το σημείο θα ήταν χρήσιμο να παραθέσουμε τους ορισμούς των διαδικασιών της αντικειμενοστρεφούς ανάλυσης και σχεδίασης, έτσι ώστε ο λόγος της εγγύτητας και της αλληλεξάρτησής τους να γίνει πιο εμφανής:

Αντικειμενοστρεφής ανάλυση:

Η αντικειμενοστρεφής ανάλυση ασχολείται με την ανάπτυξη ενός αντικειμενοστρεφούς μοντέλου του **πεδίου προβλήματος.** Τα αντικείμενα που προκύπτουν από την ανάλυση αντιστοιχούν σε οντότητες και μεθόδους συνυφασμένες με το αρχικό πρόβλημα προς επίλυση.

Αντικειμενοστρεφής σχεδίαση:

Η αντικειμενοστρεφής σχεδίαση ασχολείται με την ανάπτυξη ενός αντικειμενοστρεφούς μοντέλου **ενός συστήματος λογισμικού** το οποίο θα υλοποιήσει τις προκαθορισμένες λειτουργικές απαιτήσεις του προβλήματος.

Ας σημειωθεί ότι δεν πρόκειται για διαφορά διατύπωσης αλλά ουσίας: Κατά τη διαδικασία της αντικειμενοστρεφούς σχεδίασης μπορεί να ανακαλύψουμε πως κάποια αντικείμενα του πεδίου προβλήματος (δηλαδή που προέκυψαν από την ανάλυση) είναι πολύ «συναφή» ή και όμοια με αντίστοιχα αντικείμενα του σχεδίου της επίλυσης (δηλαδή που προκύπτουν κατά τη σχεδίαση). Στον αντίποδα, κάποια άλλα (στην καλύτερη περίπτωση λίγα) αντικείμενα που εντοπίστηκαν από την ανάλυση μπορεί να διαφέρουν με αυτά που επιτελούν τις λειτουργίες τους στο σχεδιαστικό μοντέλο. Σε κάθε περίπτωση πάντως, είναι αναμενόμενο ο σχεδιαστής να χρειαστεί να προσθέσει καινούρια αντικείμενα ή/και να αναπτύξει περαιτέρω τα υπάρχοντα στο σχεδιαστικό μοντέλο. Η πρόσθεση καινούριων αντικειμένων και η προσαρμογή των υπαρχόντων στο σχεδιαστικό μοντέλο γίνεται βάσει των χαρακτηριστικών του εγχειρήματος και της κατανόησης του προβλήματος από τον ίδιο τον σχεδιαστή. Υλικό απαραίτητο για την κατανόηση αυτή έχει κατασκευαστεί ήδη από τη φάση της ανάλυσης. Αφού, λοιπόν, η αρχική κατανόηση και μοντελοποίηση του προβλήματος, όπως και η μετέπειτα σχεδίαση του λογισμικού προς υλοποίηση, γίνεται σε επίπεδο αντικειμένων,

είναι φυσικό ο σχεδιαστής πολλές φορές να χρειάζεται να μεταπηδά από το στάδιο της σχεδίασης στο στάδιο της ανάλυσης και το αντίστροφο.

Από τα παραπάνω συμπεραίνεται η μεγάλη σημασιολογική εγγύτητα της ανάλυσης με τη σχεδίαση στα πλαίσια της αντικειμενοστρεφούς φιλοσοφίας. Αυτή η εγγύτητα προκύπτει κυρίως από τη δυνατότητα που μας παρέχει η ενοποιημένη προσέγγιση ανάπτυξης λογισμικού στο να μεταφέρουμε προβλήματα του φυσικού κόσμου σε λεπτομερή αντικειμενοστρεφή μοντέλα μέσω του εντοπισμού έγκυρων περιπτώσεων χρήσης. Στην περίπτωση της ενοποιημένης προσέγγισης ανάπτυξης λογισμικού, η μετάβαση από την ανάλυση στη σχεδίαση επιτυγχάνεται με τη χρήση κοινών εργαλείων της UML και απαιτεί την περαιτέρω συγκεκριμενοποίηση των υπαρχόντων από την ανάλυση κλάσεων και αντικειμένων αυτών, όπως επίσης και τη δημιουργία νέων, αν αυτό χρειαστεί.

Η αντικειμενοστρεφής προσέγγιση αποδεικνύεται και εδώ επιπλέον ευέλικτη αφού, εξαιτίας της δυνατότητας που παρέχει για αφαίρεση και απόκρυψη δεδομένων, οι λεπτομέρειες κάποιων κλάσεων μπορούν να παραμείνουν «αόριστες» και να αφεθούν έτσι έως και το στάδιο της υλοποίησης. Αυτή η πρακτική όχι μόνο δεν δημιουργεί πρόβλημα στο μοντέλο σχεδίασης αλλά και δεν αποτρέπει τον σχεδιαστή να παλινδρομεί από τη σχεδίαση στην ανάλυση στην προσπάθειά του να καταλήξει στη βέλτιστη λύση. Είναι αναπόφευκτο να αντιπαραβάλλουμε αυτήν τη δυνατότητα με τη δομημένη ανάλυση και σχεδίαση, όπου μια τέτοια παλινδρόμηση είναι δύσκολη και κοστοβόρα. Στην περίπτωση που εξετάζουμε εδώ, ο σχεδιαστής μπορεί να συγκεντρωθεί στην επίλυση του προβλήματος που αντιμετωπίζει χωρίς να περιορίζεται από συγκεκριμένες υλοποιήσεις ή αρχιτεκτονικές υπολογιστών. Έτσι επιτυγχάνεται μέγιστη ευελιξία στη μοντελοποίηση, όπως επίσης και διευκολύνεται η επαναχρησιμοποίηση διαφόρων τμημάτων του σχεδιαστικού μοντέλου.

Σύνοψη ενότητας

Εξαιτίας της εκφραστικής δύναμης της αντικειμενοστρεφούς τεχνολογίας, η αντικειμενοστρεφής ανάλυση δεν είναι εντελώς διακριτή από την αντικειμενοστρεφή σχεδίαση. Στην ανάλυση μοντελοποιούμε τον κόσμο του προβλήματος, ενώ στη σχεδίαση το λογισμικό που θα το επιλύσει, χρησιμοποιώντας και στις δύο περιπτώσεις τα ίδια εργαλεία: τις κλάσεις.

Άσκηση Ι/Κεφάλαιο 9

Προβληματιστείτε γύρω από το εξής θέμα: η οπισθοδρόμηση από τη σχεδίαση στην ανάλυση και αντίστροφα ενδεχομένως (και αυτό είναι το πιθανότερο) να επιφέρει μεταβολές στις κλάσεις και γενικότερα στα στοιχεία λογισμικού και την τεκμηρίωση της ανάλυσης. Οι μεταβολές αυτές πρέπει καταγράφονται ξεχωριστά από τις αρχικές εκδόσεις των στοιχείων του μοντέλου ανάλυσης ή μήπως αρκεί που μετατρέπουν τις αρχικές εκδόσεις των στοιχείων αυτών σε μεταγενέστερες; Αν ισχύει το δεύτερο, δεν χάνεται το σύνορο των φάσεων της ανάπτυξης λογισμικού;

ΕΝΟΤΗΤΑ 9.2. ΤΟ ΜΟΝΤΕΛΟ ΣΧΕΔΙΑΣΗΣ

Το μοντέλο σχεδίασης εξειδικεύει το μοντέλο ανάλυσης με σκοπό να το μεταφέρει από το πεδίο του προβλήματος στο πεδίο της επίλυσης. Κατά το στάδιο αυτό, ο αναλυτής/σχεδιαστής καλείται να αναπτύξει λεπτομερώς τα αντικείμενα που έχουν προηγουμένως αναγνωριστεί και να ορίσει τις μεταξύ τους αλληλεπιδράσεις. Γι' αυτό τον σκοπό, η ενοποιημένη προσέγγιση ανάπτυξης λογισμικού προβλέπει την απευθείας χρήση και εξειδίκευση των υπαρχόντων εργαλείων της UML, εκ των οποίων τα βασικότερα έχουν ήδη χρησιμοποιηθεί κατά την ανάλυση. Έτσι, η μετάβαση αυτή επιτυγχάνεται

με τον ευκολότερο δυνατό και φυσικό τρόπο, στα πλαίσια πάντα της αντικειμενοστρεφούς φιλοσοφίας και προσέγγισης. Αυτό συμβαίνει γιατί κατά τη σχεδίαση δεν δημιουργούμε συστατικά λογισμικού (διαγράμματα, περιγραφές) νέου τύπου, αλλά «χτίζουμε» πάνω σε ό,τι ήδη έχουμε από την ανάλυση.

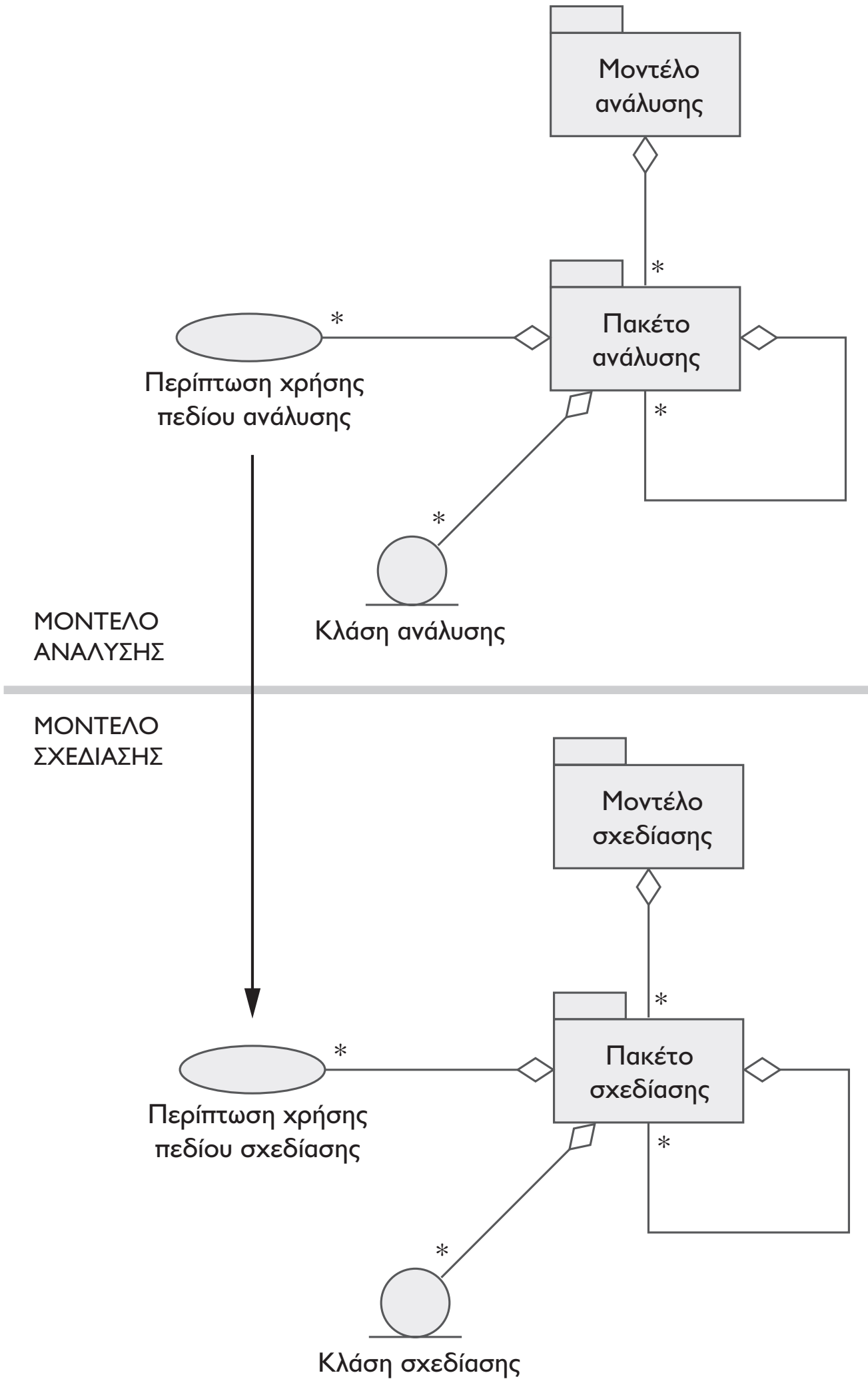
Ο στόχος της αντικειμενοστρεφούς σχεδίασης, λοιπόν, είναι η ακριβής περιγραφή των συνιστάμενων υλοποιήσιμων τμημάτων μιας λύσης λογισμικού. Μια τέτοια περιγραφή περιέχει διάφορα συστατικά λογισμικού ή τεχνουργήματα (artifacts), δηλαδή τεχνητές και υλοποιήσιμες περιγραφές των αντικειμένων του φυσικού κόσμου που εντάσσονται στο πεδίο του προβλήματος που αντιμετωπίζουμε. Η σχεδίαση εμπεριέχει επίσης περιγραφές για όλα τα επιπρόσθετα αντικείμενα που κάποια λύση μπορεί να χρειάζεται για την επιτυχή υλοποίησή της. Σε αυτή την ενότητα θα παρουσιάσουμε πώς περιγράφει η UML τα τεχνουργήματα (artifacts) που εμπλέκονται στη δημιουργία ενός μοντέλου σχεδίασης λογισμικού και πώς συνδέεται η ανάλυση με τη σχεδίαση κατά την ενοποιημένη προσέγγιση ανάπτυξης λογισμικού.

9.2.1. Οι περιπτώσεις χρήσης στο μοντέλο σχεδίασης

Ο εντοπισμός περιπτώσεων χρήσης και η αναλυτική καταγραφή τους είναι από τις πιο βασικές διαδικασίες κατά την ενοποιημένη προσέγγιση ανάπτυξης λογισμικού. Οι περιπτώσεις χρήσης αποτελούν μια πλήρη καταγραφή των λειτουργικών απαιτήσεων του συστήματος μέσα από το πρίσμα των διαφόρων χρηστών του, είτε είναι φυσικά πρόσωπα είτε άλλα περιφερειακά συστήματα. Επιπλέον, καθορίζουν με όσο το δυνατόν σαφή και απόλυτο τρόπο τα όρια του υπό ανάπτυξη συστήματος. Αποτελούν, έτσι, το σημείο αναφοράς του αρχικού προβλήματος προς επίλυση μεταξύ προγραμματιστών, αναλυτών/σχεδιαστών, πελατών και άλλων εμπλεκόμενων μερών, όπως με σαφήνεια φαίνεται στο Σχήμα 8.9. Εξ ορισμού, ο εντοπισμός και η προδιαγραφή των περιπτώσεων χρήσης είναι διαδικασίες που εντάσσονται πρωτίστως στη φάση της ανάλυσης, αφού βοηθούν στο να μεταφερθεί κάποιο φυσικό πρόβλημα σε ένα επίπεδο αφαίρεσης πιο κοντά στην προδιαγραφή της λύσης του λογισμικού. Για όλους αυτούς τους λόγους, οι περιπτώσεις χρήσης είναι απαραίτητες και κατά τη διαδικασία της σχεδίασης. Αποτελούν

μια σχετικά σταθερή στο χρόνο καταγραφή των λειτουργικών απαιτήσεων και των ορίων του συστήματος. Σημειώστε το «σχετικά»: η σχετικότητα εδώ δεν οφείλεται σε αδυναμία έκφρασης του αντικειμενοστρεφούς μοντέλου, αλλά στον ρυθμό μεταβολής των απαιτήσεων από το λογισμικό, ο οποίος, όπως έχουμε αναφέρει, μπορεί να είναι ταχύτερος από την ίδια την ανάπτυξη μιας εφαρμογής.

Σχήμα 9.1 Απεικόνιση περιπτώσεων χρήσης από την ανάλυση στη σχεδίαση.

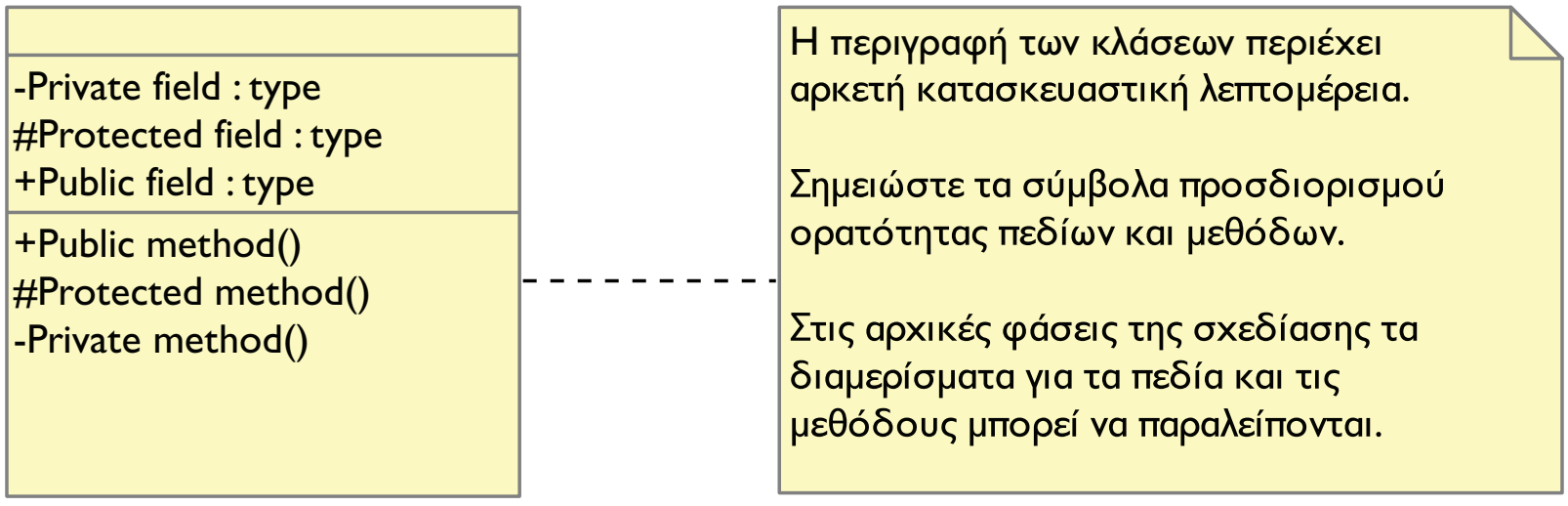


Όπως απεικονίζεται στο Σχήμα 9.1, κατ' αντιστοιχία με την περίπτωση της αντικειμενοστρεφούς ανάλυσης (Σχήμα 8.13), στο επίπεδο της σχεδίασης η κάθε περίπτωση χρήσης μεταφράζεται σε ένα ή περισσότερα πακέτα σχεδίασης. Τα πακέτα αυτά, με τη σειρά τους, αποτελούνται από μία ή περισσότερες κλάσεις σχεδίασης κ.ο.κ. Αξίζει να αναφερθεί ξανά πως τα δομικά στοιχεία του μοντέλου ανάλυσης, δηλαδή οι κλάσεις αντικειμένων και τα πακέτα ανάλυσης, πολλές φορές αντιστοιχούν αυτούσια στα δομικά στοιχεία της σχεδίασης. Με άλλα λόγια, είναι συνηθισμένο ένα πακέτο ή μία κλάση του μοντέλου ανάλυσης να αντικατοπτρίζεται ακριβώς σε ένα πακέτο ή μία κλάση αντιστοίχως του μοντέλου σχεδίασης. Είναι, βέβαια, αντιληπτό πως ο αναλυτής/σχεδιαστής τείνει να μετακινείται από το αφαιρετικό επίπεδο της ανάλυσης στο πιο πρακτικό επίπεδο της σχεδίασης. Κατά τη μετακίνηση αυτή, ο στόχος του πρέπει να είναι η όσο το δυνατόν λεπτομερής και υλοποιήσιμη καταγραφή των υπαρχόντων κλάσεων και η αλλαγή ή/και η δημιουργία καινούριων, αν αυτό κριθεί σκόπιμο. Όπως θα δούμε παρακάτω, ένας επιπλέον στόχος που θα πρέπει να επιτευχθεί είναι και η λεπτομερής περιγραφή της συνεργασίας μεταξύ κλάσεων, έτσι ώστε να ικανοποιηθούν οι περιγραφόμενες από τις περιπτώσεις χρήσης λειτουργικές απαιτήσεις του συστήματος λογισμικού.

Συμβολισμοί UML

Στο Σχήμα 9.2 αναπαράγονται τα βασικά σύμβολα της UML για την αναπαράσταση κλάσεων και διαγραμμάτων κλάσεων. Όπως παρατηρούμε, ο συμβολισμός είναι πιο λεπτομερής στο στάδιο της σχεδίασης, μιας και σκοπός είναι να παραστήσουμε ένα επίπεδο αφαίρεσης περιπτώσεων χρήσης –και άρα ένα τμήμα του αρχικού προβλήματος– με τρόπο σαφή και υλοποιήσιμο. Έμφαση δίνεται τόσο στα πεδία και στις μεθόδους των κλάσεων όσο και στην εμβέλεια (scope) και στους τύπους τους, δηλαδή στοιχεία που κατεξοχήν αφορούν την υλοποίηση. Στο Σχήμα 9.2 φαίνονται οι τρόποι σύνδεσης και συσχετισμού στο διάγραμμα κλάσεων. Είναι κατανοητό πως ένα διάγραμμα κλάσεων μπορεί να απεικονίσει μόνο τα δομικά χαρακτηριστικά μιας περίπτωσης χρήσης και δεν περιλαμβάνει άλλες πληροφορίες, όπως χρονισμός ενεργειών.

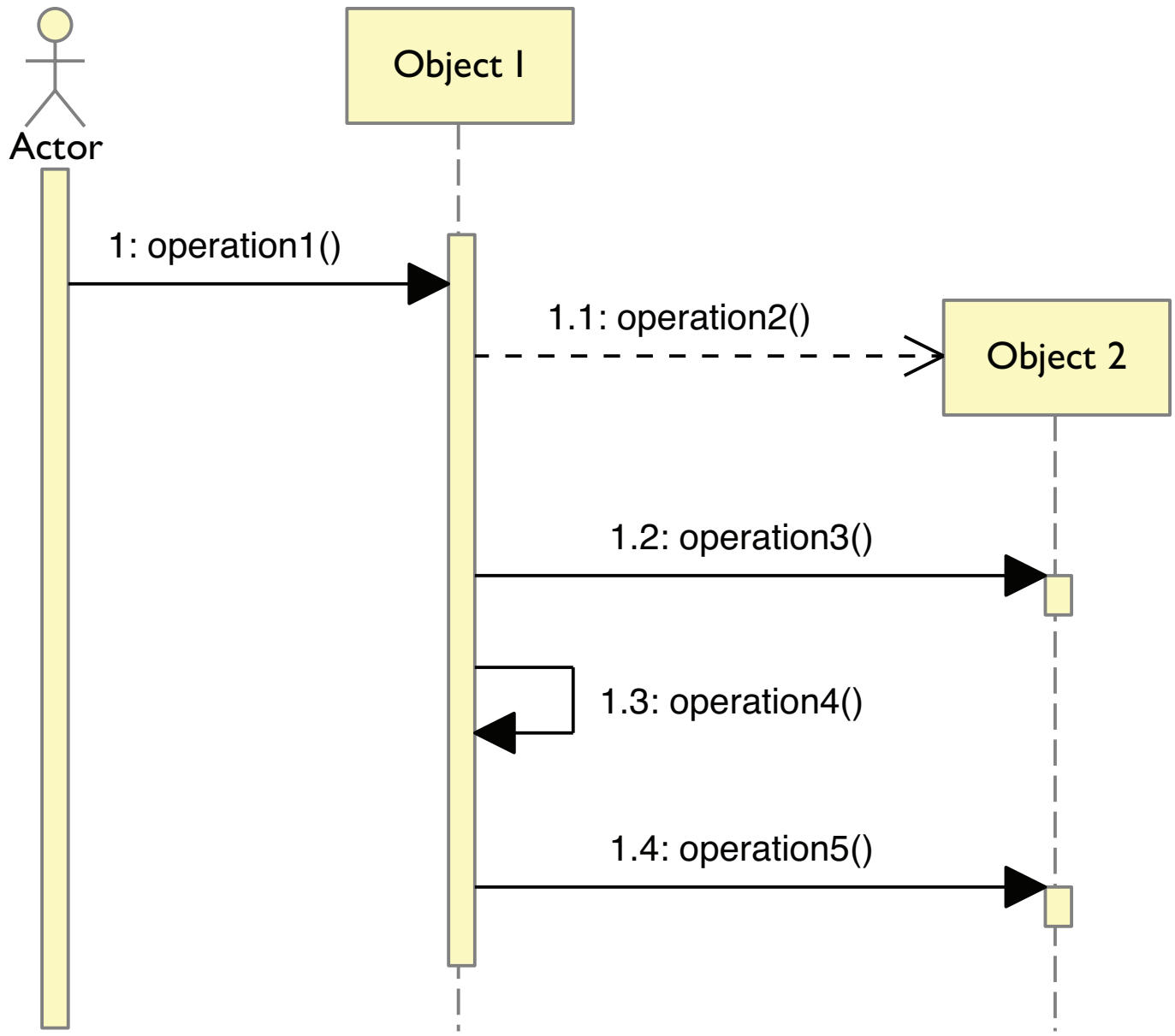
Σχήμα 9.2 Συμβολισμοί UML για κλάσεις.



9.2.2. Πώς σχεδιάζεται η περίπτωση χρήσης

Όπως φαίνεται στο Σχήμα 9.1, η κάθε περίπτωση χρήσης, αφού αναλυθεί, μπορεί να περιγραφεί από μια σειρά διεργασιών μεταξύ ενός συνόλου αντικειμένων. Αφού τα αντικείμενα ανήκουν στις αντίστοιχες προδιαγεγραμμένες κλάσεις τους, η UML μάς δίνει τη δυνατότητα να περιγράψουμε τον τρόπο με τον οποίο αυτά επικοινωνούν με δύο συμπληρωματικούς μηχανισμούς: τα διαγράμματα κλάσεων και τα διαγράμματα αλληλεπιδράσεων. Τα διαγράμματα κλάσεων αποτελούν μία αρχιτεκτονική όψη για τις διάφορες περιπτώσεις χρήσης. Τα διαγράμματα αλληλεπιδράσεων χωρίζονται σε διαγράμματα αλληλουχίας ή σειράς ή ακολουθίας (sequence) και σε διαγράμματα συνεργασίας (collaboration), και απεικονίζουν με ποιο τρόπο και μέσω ποιων μεθόδων επικοινωνούν τα διάφορα αντικείμενα για να φέρουν εις πέρας την περίπτωση χρήσης. Με τη χρήση των δύο αυτών μηχανισμών της UML, ο αναλυτής/σχεδιαστής επιτυγχάνει, πλέον, την πλήρη προδιαγραφή μιας περίπτωσης χρήσης με τρόπο που την καθιστά, θεωρητικά τουλάχιστον, κατανοητή και συγχρόνως ευθέως υλοποιήσιμη. Τα διαγράμματα συνεργασίας αναφέρθηκαν στο Κεφάλαιο 8, ενώ τα διαγράμματα ακολουθίας ακολουθούν τον συμβολισμό όπως στο Σχήμα 9.3.

Σχήμα 9.3 Συμβολισμοί UML



Στα διαγράμματα σειράς ή ακολουθίας (sequence) ο χειριστής αντιπροσωπεύεται από ένα αντικείμενο.

Κάθε περίπτωση χρήσης περιγράφεται από μια ακολουθία κλήσεων μεταξύ αντικειμένων που συμμετέχουν στην υλοποίησή της.

Κάθε αντικείμενο έχει μια "γραμμή ζωής" (lifeline) η οποία παριστάνεται με τη διακεκομμένη γραμμή στο σχήμα. Οταν το αντικείμενο είναι ενεργό, τότε η διακεκομμένη γραμμή μετατρέπεται σε πλαίσιο. Οι κλήσεις μεθόδων δηλώνονται με ένα βέλος, στο οποίο σημειώνεται το όνομα και η σειρά κλήσης της μεθόδου που καλείται.

9.2.3. Διεπαφές στο μοντέλο σχεδίασης

Οι διεπαφές (interfaces) αποτελούν έναν κεντρικό μηχανισμό στην αντικειμενοστρεφή φιλοσοφία. Μέσω αυτών ο σχεδιαστής ορίζει τρόπους επικοινωνίας μεταξύ των διαφόρων αντικειμένων και τεχνουργημάτων του μοντέλου σχεδίασης. Ήδη από το Κεφάλαιο 8 έχουμε δει ότι οι κλάσεις μπορούν να διαχωριστούν σε συνοριακές, οντοτήτων και ελέγχου. Από αυτές, οι συνοριακές κλάσεις αποτελούν, εξ ορισμού, το πρωτόκολλο επικοινωνίας μεταξύ του υπό σχεδίαση συστήματος και του έξω κόσμου, δηλαδή των περιφερειακών συστημάτων, συσκευών κ.λπ. Υπάρχουν όμως και άλλες εκφάνσεις διεπαφών που μπορούν να οριστούν ρητά τόσο στη UML όσο και σε διάφορες αντικειμενοστρεφείς γλώσσες προγραμματισμού και αφορούν αντικείμενα του ίδιου συστήματος. Όμως, πριν προχωρήσουμε παρακάτω, θα ήταν χρήσιμο να ορίσουμε πώς εννοούμε τη διεπαφή στην περίπτωση της αντικειμενοστρεφούς τεχνολογίας.

Διεπαφή (interface):

Η διεπαφή μιας κλάσης αντικειμένων είναι το σύνολο των πεδίων και μεθόδων της κλάσης που μπορούν να αναγνωστούν, να μεταβληθούν ή να κληθούν από κάποιο αντικείμενο της ίδιας ή άλλης κλάσης.

Με άλλα λόγια, η διεπαφή αποτελεί ένα «δημόσια κοινοποιημένο» πρωτόκολλο επικοινωνίας, με την ευρεία έννοια του όρου, ή αλλιώς ένα σύνολο επιθυμητών τρόπων συναλλαγής των αντικειμένων μιας κλάσης με το περιβάλλον τους.

Από τον ορισμό αυτό συμπεραίνουμε πως ο ακριβής ρόλος και έκφανση μιας διεπαφής θα εξαρτηθεί τόσο από τον ορισμό της προγραμματιστικά όσο και από τη σχέση μεταξύ των εμπλεκόμενων κλάσεων. Για παράδειγμα, είναι κατανοητό πως η διεπαφή μιας κλάσης, όσον αφορά τα αντικείμενα της ίδιας της κλάσης, είναι το σύνολο των πεδίων και των μεθόδων της ανεξαρτήτως από το αν έχουν οριστεί ως κοινοποιημένα ή κρυφά. Μία

μέθοδος της κλάσης θα έχει πρόσβαση σε όλα τα πεδία και τις μεθόδους που έχουν οριστεί μέσα σε αυτή. Για ένα αντικείμενο μιας δεύτερης κλάσης όμως, αναλόγως με τη σχεδίαση που έχουμε, όπως αυτή αποτυπώνεται στο διάγραμμα κλάσεων (βλ. Σχήμα 9.2), η επικοινωνία θα περιοριστεί στα κοινοποιημένα (public) πεδία και μεθόδους της πρώτης κλάσης.

Παράδειγμα

Ας επιστρέψουμε στην εφαρμογή «Επίκουρος». Σε αυτή την εφαρμογή, διάφορες κλάσεις οντοτήτων περιέχουν ευαίσθητα δεδομένα όπως, για παράδειγμα, τον αριθμό ταυτότητας των καθηγητών στην κλάση «καθηγητής» κ.ο.κ. Αν απαγορεύσουμε την απευθείας αλλαγή των αντίστοιχων πεδίων και αντ' αυτού ορίσουμε πως οι όποιες αλλαγές θα γίνονται μόνο μέσα από συγκεκριμένες μεθόδους, επιτυγχάνουμε μεγαλύτερο έλεγχο σχετικά με τον τρόπο που μπορούν τέτοια στοιχεία να αλλαχθούν. Το σύνολο αυτών των μεθόδων στην ουσία θα αποτελέσουν μία διεπαφή μεταξύ της κλάσης «καθηγητής» και του υπόλοιπου συστήματος.

Οι μέθοδοι αυτές, αν και ίσως ακατανόητες στην αρχή της καριέρας ενός σχεδιαστή λογισμικού, είναι πολύ χρήσιμες: επιτρέπουν την πραγματοποίηση μεταβολών στην υλοποίηση της κλάσης χωρίς να αλλάξει τίποτε στον τρόπο που «οι άλλοι» βλέπουν την κλάση, πράγμα που οδηγεί στην κατασκευή θωρακισμένου και επαναχρησιμοποιήσιμου λογισμικού. Σε αρκετά βιβλία προγραμματισμού αναφέρονται ως «accessors», από το ρήμα access, που σημαίνει «προσπελαύνω, αποκτώ πρόσβαση σε κάτι».

Δραστηριότητα Ι/Κεφάλαιο 9

Υλοποιήστε ένα interface για τη λειτουργία «εκτύπωση βαθμολογίας σπουδαστή» του λογισμικού «Επίκουρος», σχολιάζοντας κατάλληλα τη χρησιμότητα και τον ρόλο του.

9.2.4. Κλάσεις στο μοντέλο σχεδίασης

Όπως έχουμε δει, οι κλάσεις κατέχουν κεντρικό ρόλο στη σχεδίαση, και προέρχονται από τις κλάσεις που έχουν οριστεί στο μοντέλο ανάλυσης. Κατά τη σχεδίαση, οι κλάσεις αυτές πρέπει να καθοριστούν με λεπτομερή τρόπο, έτσι ώστε το μοντέλο να μπορέσει να ικανοποιήσει τις λειτουργικές ανάγκες του προς ανάπτυξη συστήματος. Γι' αυτό τον σκοπό, ο σχεδιαστής έχει στη διάθεσή του τα εργαλεία που έχουμε προαναφέρει όσον αφορά τις κλάσεις του συστήματος:

- Τα διαγράμματα κλάσεων.
- Τα διαγράμματα αλληλεπιδράσεων στα οποία ανήκουν:
  - τα διαγράμματα αλληλουχίας ή ακολουθίας ή σειράς (sequence) και
  - τα διαγράμματα συνεργασίας (collaboration).

Τα μεν διαγράμματα κλάσεων παρέχουν μια στατική απεικόνιση της αρχιτεκτονικής που θα πρέπει να υλοποιηθεί, τα δε διαγράμματα αλληλεπιδράσεων βοηθούν στην συγκεκριμενοποίηση και περαιτέρω κατανόηση των λειτουργικών απαιτήσεων του προβλήματος μέσα από τη δυναμική απεικόνιση των περιπτώσεων χρήσης, η οποία περιέχει και τη διάσταση του χρόνου. Τα διαγράμματα κλάσεων αποτυπώνουν τις εμπλεκόμενες κλάσεις και τις συσχετίσεις τους σε αρχιτεκτονικό επίπεδο. Από την άλλη, τα διαγράμματα ακολουθίας επικεντρώνονται στη σειρά με την οποία ανταλλάσσονται τα μηνύματα μεταξύ των κλάσεων (δηλαδή οι κλήσεις των μεθόδων των διεπαφών τους από άλλες κλάσεις), ενώ τα διαγράμματα συνεργασίας επικεντρώνονται στις σχέσεις μεταξύ των αντικειμένων και είναι ικανά να αποτυπώσουν τη συνεργασία μεταξύ πολλών κλάσεων.

Ενώ τα δύο αυτά εργαλεία (τα διαγράμματα κλάσεων και τα διαγράμματα αλληλεπιδράσεων) είναι εξίσου σημαντικά για την υλοποίηση του συστήματος, πολλές φορές οι σχεδιαστές δίνουν μεγαλύτερη βαρύτητα στα διαγράμματα κλάσεων εις βάρος των διαγραμμάτων αλληλεπιδράσεων. Αυτό μπορεί να συμβαίνει για δύο λόγους: για ένα σύστημα λογισμικού, τα διαγράμματα κλάσεων είναι λίγα –συνήθως ένα για κάθε υποσύστημα. Από την

άλλη, τα διαγράμματα αλληλεπιδράσεων είναι πάρα πολλά, αναλόγως με τον αριθμό των περιπτώσεων χρήσης και των λειτουργικών απαιτήσεων του συστήματος. Μάλιστα, για κάθε περίπτωση χρήσης μπορούν να υπάρχουν εναλλακτικές ροές (βλ. Ενότητα 8.5.2) οι οποίες πολλαπλασιάζουν τον αριθμό των διαγραμμάτων ακολουθίας. Συνεπώς, είναι πολύ πιο εύκολο να δημιουργήσει και να συντηρήσει κάποιος στατικά διαγράμματα κλάσεων παρά διαγράμματα αλληλεπιδράσεων.

Ένας επιπρόσθετος λόγος για αυτή την άτυπη προτίμηση είναι ότι τα διαγράμματα κλάσεων μπορούν να αναγνωστούν και να χρησιμοποιηθούν ευκολότερα από προγραμματιστές και έτσι βοηθούν στην υλοποίηση πρωτότυπων συστημάτων σε σύντομο χρόνο. Αυτά τα πλεονεκτήματα όμως είναι πλασματικά και οι σχεδιαστές που κλίνουν προς τη χρήση μόνο διαγραμμάτων κλάσεων δεν λαμβάνουν υπόψη τους την επαναληπτική και επαυξητική φύση της αντικειμενοστρεφούς προσέγγισης και αφαιρούν από την εκφραστική της δύναμη. Δεν αρκούν μόνο τα διαγράμματα κλάσεων για να γίνει μια επιτυχής υλοποίηση που να πληροί τις προδιαγραφές της εφαρμογής. Πρέπει τα στατικά και τα δυναμικά διαγράμματα να χρησιμοποιηθούν επαναλαμβανόμενα, τροφοδοτώντας τα μεν τα δε μέχρις ότου συγκλίνουν σε μία αποδεκτή λύση. Στο παράδειγμα που ακολουθεί γίνεται φανερό πώς τα διαγράμματα κλάσεων και τα διαγράμματα ακολουθίας και συνεργασίας αλληλοσυμπληρώνονται και πώς η επαναληπτική, εκ περιτροπής εξειδίκευσή τους μπορεί να μας οδηγήσει σε μια πιο ορθολογική και υλοποιήσιμη λύση.

Δραστηριότητα 2/Κεφάλαιο 9

Υλοποιήστε ένα interface για τη λειτουργία «εκτύπωση βαθμολογίας σπουδαστή» του λογισμικού «Επίκουρος», σχολιάζοντας κατάλληλα τη χρησιμότητα και τον ρόλο του.

Ας αφήσουμε προσωρινά την εφαρμογή «Επίκουρος» και ας ξαναδούμε την εφαρμογή συναρμολόγησης του αεροσκάφους του παραδείγματος του Κεφαλαίου 7. Ας υποθέσουμε πως θέλουμε να αναπτύξουμε ένα λογισμικό εξομοιωτή πτήσεων και, πιο συγκεκριμένα, τον αυτόματο πιλότο. Από τις διάφορες λειτουργίες που μπορεί να κάνει ένας αυτόματος πιλότος, ας επικεντρωθούμε στην περίπτωση χρήσης της απογείωσης. Τροποποιήστε κατάλληλα το διάγραμμα κλάσεων του ίδιου παραδείγματος και κατασκευάστε ένα διάγραμμα συνεργασίας.

9.2.5. Αρχιτεκτονική όψη και υποσυστήματα

Είναι σύνηθες και απολύτως απαραίτητο για τα μεγάλα συστήματα λογισμικού να χωρίζονται σε μικρότερα συνιστώμενα τμήματα ή υποσυστήματα. Αυτό πρακτικά εφαρμόζει την αρχή του «διαίρει και βασίλευε» στην ανάπτυξη λογισμικού, με όλα τα πλεονεκτήματα μιας τέτοιας μεθοδολογίας, όπως έχει ήδη αναφερθεί.

Μερικά από αυτά τα πλεονεκτήματα είναι τα παρακάτω:

- Το πρόβλημα επιλύεται ταχύτερα και πιο οργανωμένα όταν επικεντρωνόμαστε σε μικρότερα προβλήματα παρά σε μεγαλύτερα.
- Κάνει δυνατό τον διαχωρισμό της εργασίας σε ανεξάρτητες ομάδες αναλυτών, σχεδιαστών, προγραμματιστών κ.λπ., ανάλογα με το υποσύστημα και τις λειτουργικές και μη απαιτήσεις του.
- Ενθαρρύνει την επαναχρησιμοποίηση υποσυστημάτων σε άλλα εγχειρήματα που μπορεί να προκύψουν.
- Μπορεί να προκύψει φυσικά από το πρόβλημα και την έκφανσή του στον πραγματικό κόσμο.

Αυτά είναι μερικά μόνο από τα πλεονεκτήματα αυτής της τακτικής και η επίτευξή τους προϋποθέτει την εφαρμογή κάποιων παραδοχών. Αυτές οι παραδοχές θα γίνουν ευδιάκριτες μέσα από τον ορισμό του υποσυστήματος και του συστατικού στοιχείου.

> **Υποσύστημα** είναι ένα λειτουργικό τμήμα ενός ολοκληρωμένου συστήματος λογισμικού το οποίο, μέσα από μια σειρά εννοιολογικά συναφών λειτουργιών που περιέχει και υλοποιεί, παρέχει υπηρεσίες στη συνολική λύση που υλοποιεί το ολοκληρωμένο σύστημα.

Το συστατικό στοιχείο (component) της UML είναι πολύ σχετικό με το υποσύστημα. Η πλέον συνηθισμένη προσέγγιση είναι πως το συστατικό στοιχείο αποτελεί την υλοποίηση ενός υποσυστήματος. Άρα, ενώ και τα δύο αποτελούν μέρη της αρχιτεκτονικής, είναι συνηθέστερο η αρχιτεκτονική κατά την ανάλυση να παρουσιάζεται βάσει υποσυστημάτων, ενώ η αρχιτεκτονική κατά την υλοποίηση να παρουσιάζεται βάσει συστατικών στοιχείων του λογισμικού. Η πιο πρόσφατη έκδοση της UML, η UML 2, δεν είναι σαφής ως προς τον διαχωρισμό των συστατικών στοιχείων και των υποσυστημάτων καθώς δεν διαχωρίζει τις δύο αυτές δομικές οντότητες σε σχέση με τη χρήση τους κατά τη μοντελοποίηση. Μάλιστα, υποστηρίζει πως η χρήση του ενός έναντι του άλλου επαφίεται στη μεθοδολογία που επιλέγει να ακολουθήσει ο αναλυτής/σχεδιαστής. Προκειμένου να διατηρήσουμε τα δύο αυτά στοιχεία διακριτά, θα ακολουθήσουμε την προσέγγιση που αναφέρουμε και πιο πάνω: κατά τη σχεδίαση θα χρησιμοποιούμε υποσυστήματα και κατά την υλοποίηση θα χρησιμοποιούμε συστατικά στοιχεία λογισμικού, αποδεχόμενοι την παραδοχή ότι στη γενική περίπτωση ένα υποσύστημα αποτελείται από περισσότερα του ενός συστατικά στοιχεία.

Η αρχιτεκτονική όψη ή σχεδίαση συνήθως καθορίζεται νωρίς κατά την ανάλυση και σχεδίαση ενός συστήματος. Όπως και οι υπόλοιπες διαδικασίες της ενοποιημένης προσέγγισης ανάπτυξης λογισμικού, έχει επαναληπτικό και επαυξητικό χαρακτήρα. Έτσι, είναι σύνηθες να ξεκινούμε από την αρχιτεκτονική όψη, να συνεχίζουμε με τη σχεδίαση των επιμέρους υποσυστημάτων βάσει της οποίας στη συνέχεια αναπροσδιορίζουμε την αρχιτεκτονική

όψη κ.ο.κ. Ο ακριβής τρόπος και η σειρά με την οποία θα χρειαστεί να ακολουθήσουμε αυτά τα βήματα δεν είναι δυνατό να είναι γνωστά εκ των προτέρων. Θα εξαρτηθούν από τη φύση του προβλήματος, από το μέγεθός του, από τη μεθοδολογία που προτιμά να ακολουθεί ο εκάστοτε αναλυτής/σχεδιαστής, όπως και από πολλές άλλες παραμέτρους. Σε κάθε περίπτωση όμως, πρέπει να τηρούνται τα παρακάτω:

- Το κάθε υποσύστημα θα πρέπει να έχει διακριτό ρόλο στη συνολική αρχιτεκτονική του συστήματος, δηλαδή τα επιμέρους τμήματά του θα πρέπει να εξυπηρετούν κάποιο συγκεκριμένο σκοπό, να καλύπτουν έναν αριθμό συναφών περιπτώσεων χρήσης ή λειτουργικών απαιτήσεων με ακρίβεια και συνοχή.
- Η επικοινωνία και η διαλειτουργικότητα μεταξύ των υποσυστημάτων θα πρέπει να είναι ορισμένη με τρόπο όσο το δυνατόν πιο ακριβή. Αυτό επιτυγχάνεται με την εισαγωγή διεπαφών μεταξύ των υποσυστημάτων, οι οποίες, όπως έχουμε δει, είναι ανεξάρτητες από τις λεπτομέρειες της οποιασδήποτε υλοποίησης.
- Το μοντέλο ελέγχου μεταξύ των υποσυστημάτων του λογισμικού πρέπει να οριστεί. Με αυτό εννοούμε τον τρόπο με τον οποίο θα συντονίζονται τα υποσυστήματα. Για παράδειγμα, μπορεί να έχουμε κεντρικό έλεγχο ή ασύγχρονο έλεγχο. Η επιλογή του μοντέλου ελέγχου θα εξαρτηθεί από τις λειτουργικές απαιτήσεις του συστήματος ή θα είναι στις μη λειτουργικές απαιτήσεις βάσει συμβολαίου.
- Το κάθε υποσύστημα θα πρέπει να αναλυθεί ανεξάρτητα στα δομικά του μέρη και αυτά με τη σειρά τους στα δικά τους, με τα εργαλεία που έχουμε περιγράψει στις προηγούμενες ενότητες. Όπως έχουμε τονίσει επανειλημμένως, οι παλινδρομήσεις είναι αναμενόμενες και αναγκαίες μεταξύ των διαφόρων επιπέδων της σχεδίασης. Η περαιτέρω ανάλυση και σχεδίαση είναι αναμενόμενο να ρίξει φως σε πτυχές του προβλήματος που ίσως επηρεάσουν την αρχική αρχιτεκτονική ή τις περιπτώσεις χρήσης ή τις διεπαφές μεταξύ κλάσεων ή υποσυστημάτων που έχουν οριστεί κ.ο.κ.

Δραστηριότητα 3/Κεφάλαιο 9

Ας επιστρέψουμε στην εφαρμογή «Επίκουρος». Οι λειτουργικές απαιτήσεις της εφαρμογής περιγράφονται στο Σχήμα 8.11 και οι περιπτώσεις χρήσης περιγράφονται στο Σχήμα 8.17 του προηγούμενου κεφαλαίου. Μπορείτε να ορίσετε υποσυστήματα;

9.2.6. Μοντέλο διάταξης (Deployment)

Όπως είδαμε στην προηγούμενη ενότητα, η αρχιτεκτονική όψη μέσω της μοντελοποίησης υποσυστημάτων και συστατικών στοιχείων μπορεί να μας δώσει μια ολοκληρωμένη εικόνα των εμπλεκόμενων, λογικών μερών ενός συστήματος. Ενώ η αρχιτεκτονική όψη είναι πολύ χρήσιμη τόσο σε αναλυτές/σχεδιαστές όσο και σε προγραμματιστές, δεν μας βοηθά να κατανοήσουμε και να περιγράψουμε τις μη λειτουργικές ανάγκες του συστήματος και τις αλληλεξαρτήσεις των διαφόρων υποσυστημάτων βάσει αυτών. Μια τέτοια απεικόνιση επιτυγχάνεται μέσω της δημιουργίας και συντήρησης ενός μοντέλου διάταξης ή, ισοδύναμα, εγκατάστασης. Το μοντέλο διάταξης απεικονίζει τον τρόπο που διατάσσονται τα διάφορα λογικά μέρη στο υλικό που έχουμε στη διάθεσή μας. Απεικονίζει, επίσης, και τις διάφορες αλληλεξαρτήσεις τους σε επίπεδο σχεδίασης, υλοποίησης και εγκατάστασης –εξού και ο όρος «μοντέλο εγκατάστασης» που συναντάται συχνά στη βιβλιογραφία και αναφέρεται στο ίδιο μοντέλο. Μιας και όταν κληθούμε να παραδώσουμε το σύστημα, αυτό δεν θα γίνει με βάση τα διαγράμματα κλάσεων και τα άλλα αφαιρετικά σχήματα που ενδεχομένως έχουμε χρησιμοποιήσει αλλά με βάση τα παράγωγα της υλοποίησης, είναι αναμενόμενο πως το μοντέλο διάταξης θα αποτελείται από αναπαραστάσεις αυτών. Έτσι, σε ένα τέτοιο μοντέλο βρίσκουμε συστατικά στοιχεία (όπως τα έχουμε παρουσιάσει και στην προηγούμενη ενότητα), στοιχεία υλικού και τους μεταξύ τους συνδέσμους.

Σημείωση: Τα στοιχεία που απαρτίζουν το μοντέλο διάταξης ή εγκατάστασης ανήκουν στη φάση της υλοποίησης, μιας και συναντάμε βιβλιοθήκες, εκτελέσιμα αρχεία και υλικό, όπως σταθμούς εργασίας, εξυπηρετητές (servers), δικτυακές

συνδεσμολογίες κ.λπ. Όμως, όπως και άλλα μοντέλα της ενοποιημένης προσέγγισης ανάπτυξης λογισμικού, έτσι και το μοντέλο διάταξης δεν έχει στενά όρια χρήσης. Μιας και σε κάποιο δεδομένο έργο οι μη λειτουργικές απαιτήσεις είναι γνωστές από τα πρώτα στάδια της ανάλυσής του, μοντέλα διάταξης είναι καλό να δημιουργούνται από το στάδιο της σχεδίασης (ή ακόμα νωρίτερα), έτσι ώστε να αποτελούν ένα σημείο αναφοράς μεταξύ της πιο αφαιρετικής σχεδίασης και της απολύτως πρακτικής υλοποίησης. Αυτός είναι ο λόγος για τον οποίο η περιγραφή αυτού του μοντέλου βρίσκεται σε αυτό το κεφάλαιο και όχι στο επόμενο που πραγματεύεται την υλοποίηση.

Στο μοντέλο διάταξης τα συστατικά στοιχεία είναι συνήθως συνδεδεμένα με αντίστοιχα υποσυστήματα. Για παράδειγμα, ένα υποσύστημα Α, το οποίο περιγράφεται από μία σειρά διαγραμμάτων κλάσεων και άλλων σχεδιαστικών περιγραφικών κειμένων, στην υλοποιημένη του μορφή θα αποτελέσει ένα συστατικό στοιχείο. Ανάλογα με τις προδιαγεγραμμένες λειτουργίες που θα πρέπει να προσφέρει στο σύστημα, μπορεί να είναι κάποιο εκτελέσιμο, μία βιβλιοθήκη, μία φόρμα εισόδου/εξόδου για χρήστες κ.λπ. Όταν εισάγουμε ένα συστατικό στοιχείο στο διάγραμμα διάταξης, η UML επιτρέπει τη χρήση στερεοτύπων (stereotypes) για τον προσδιορισμό του. Επίσης, οι σχέσεις μεταξύ των συστατικών στοιχείων αποδίδονται με συνδέσμους οι οποίοι μπορούν επίσης να φέρουν στερεότυπα, υποδηλώνοντας τους τρόπους εξάρτησης των συστατικών στοιχείων. Ένα τελευταίο χαρακτηριστικό των συστατικών στοιχείων είναι πως, όπως και οι κλάσεις, έτσι και αυτά, μπορούν να υλοποιούν διεπαφές. Στην περίπτωση αυτή, αν η σχέση κάποιου άλλου συστατικού στοιχείου εξαρτάται από τις λειτουργίες που προδιαγράφονται από τη διεπαφή, αυτή καταλήγει στη διεπαφή και όχι απευθείας στο αρχικό συστατικό στοιχείο. Έτσι, μπορούμε να ξεχωρίσουμε ποια συστατικά στοιχεία τηρούν μία διεπαφή (δηλαδή ακολουθούν το πρωτόκολλο επικοινωνίας που αυτή περιγράφει) και ποια όχι. Ο αναγνώστης μπορεί να βρει περισσότερες πληροφορίες στους συμβολισμούς UML που ακολουθούν.

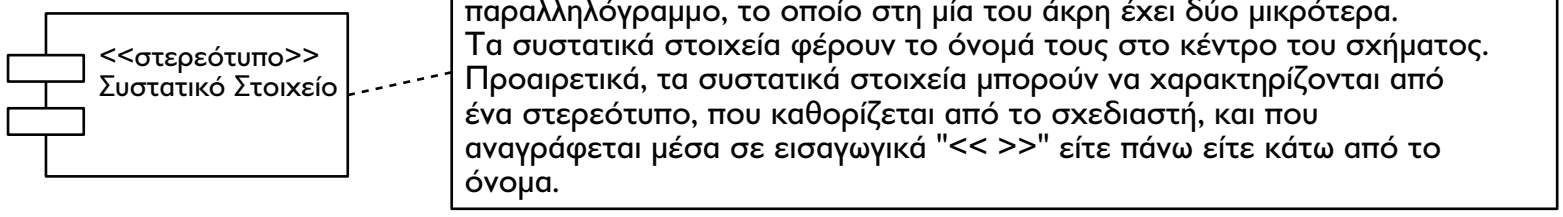
Έχοντας παρουσιάσει τη χρησιμότητα και τους συμβολισμούς των συστατικών στοιχείων, μένει να περάσουμε στη διάταξη των συστατικών στοιχείων στα διάφορα μέρη του υλικού που έχουμε στη διάθεσή μας, δηλαδή να

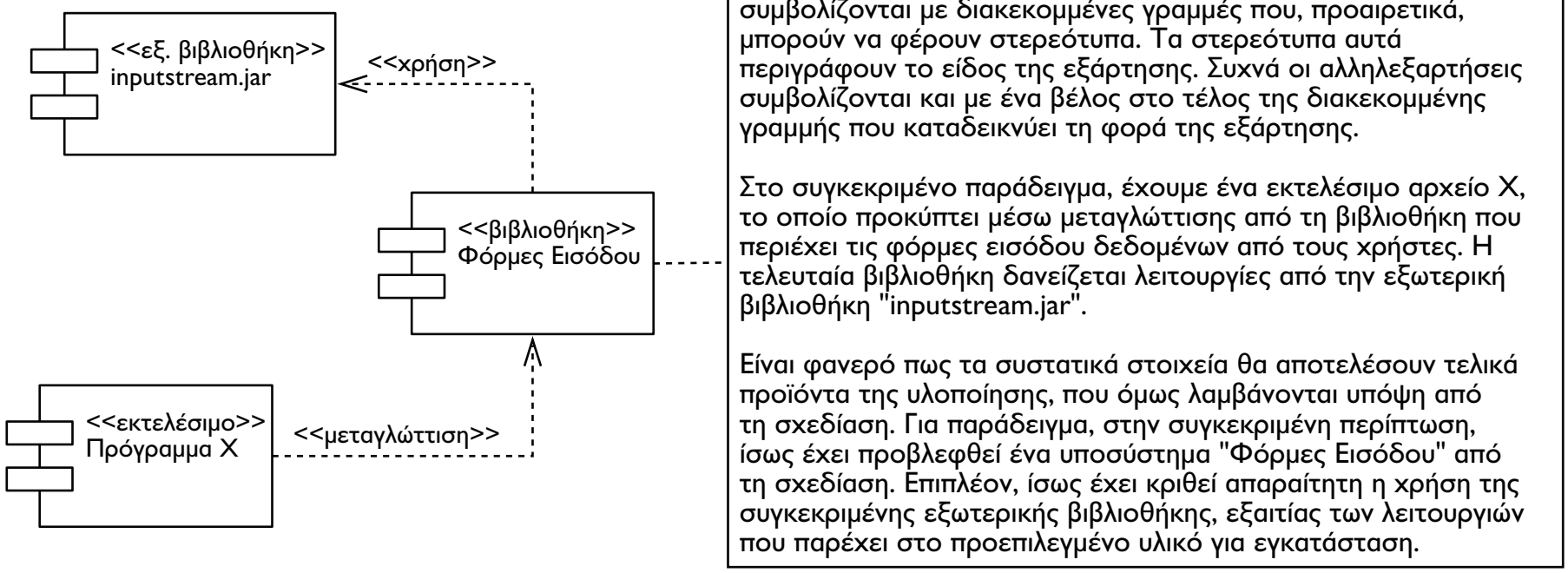
δημιουργήσουμε το μοντέλο διάταξης ή εγκατάστασης για το σύστημά μας. Πιο συγκεκριμένα, το διάγραμμα διάταξης ή εγκατάστασης απεικονίζει τα εξής:

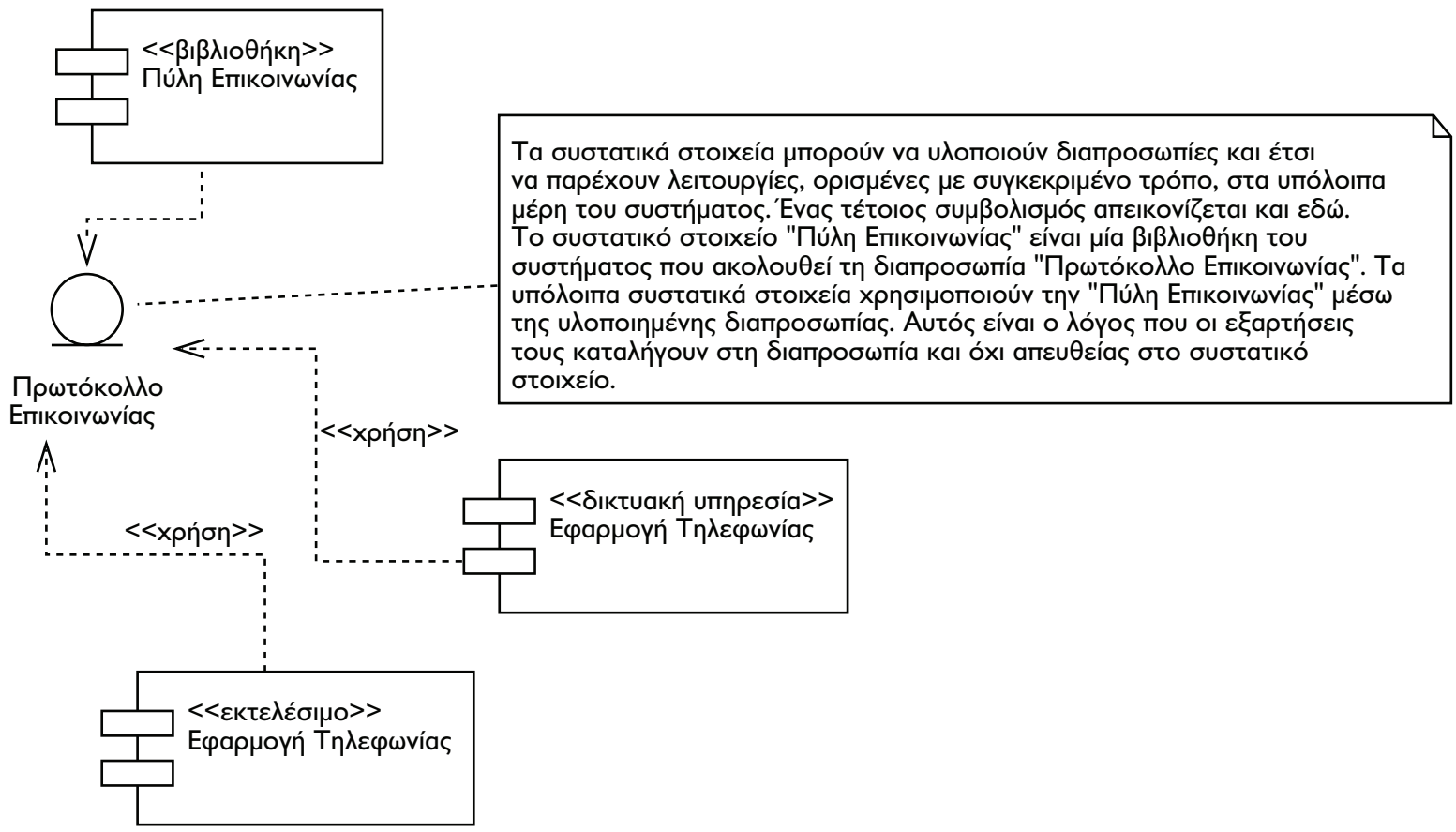
- Τους συνδέσμους επικοινωνίας μεταξύ των διαφόρων μερών του υλικού, όπως μηχανήματα, διακομιστές, εκτυπωτές κ.λπ.
- Τις σχέσεις των συστατικών στοιχείων, όπως τα έχουμε ορίσει παραπάνω, και των διαφόρων μερών του υλικού. Με άλλα λόγια, σε ποιο τμήμα του υλικού υπάρχει και εκτελείται κάθε συστατικό στοιχείο.

Στη συνέχεια οι όροι «μοντέλο διάταξης» και «μοντέλο εγκατάστασης» θα χρησιμοποιούνται ισοδύναμα. Στο μοντέλο διάταξης απεικονίζονται τόσο τα μέρη του υλικού όσο και τα συστατικά στοιχεία της τελικής εφαρμογής λογισμικού. Έχοντας ήδη καλύψει τον συμβολισμό και τη σημασία των συστατικών στοιχείων, μένει να περάσουμε και στη μοντελοποίηση του υλικού. Στο συγκεκριμένο μοντέλο, η διάταξη του υλικού έχει τη μορφή κόμβων και συνδέσμων. Οι κόμβοι συμβολίζουν τα διάφορα εμπλεκόμενα μέρη του υλικού, ενώ οι σύνδεσμοι συμβολίζουν τα διάφορα κανάλια επικοινωνίας μεταξύ των μερών του υλικού. Μέσα στους κόμβους του υλικού υπάρχουν και εκτελούνται (τρέχουν) τα συστατικά στοιχεία που έχουμε προκαθορίσει. Άρα, το μοντέλο διάταξης μας δίνει μια εικόνα του συστήματος κατά την εκτέλεση ή τη χρήση του. Ο ακριβής συμβολισμός UML δίνεται παρακάτω.

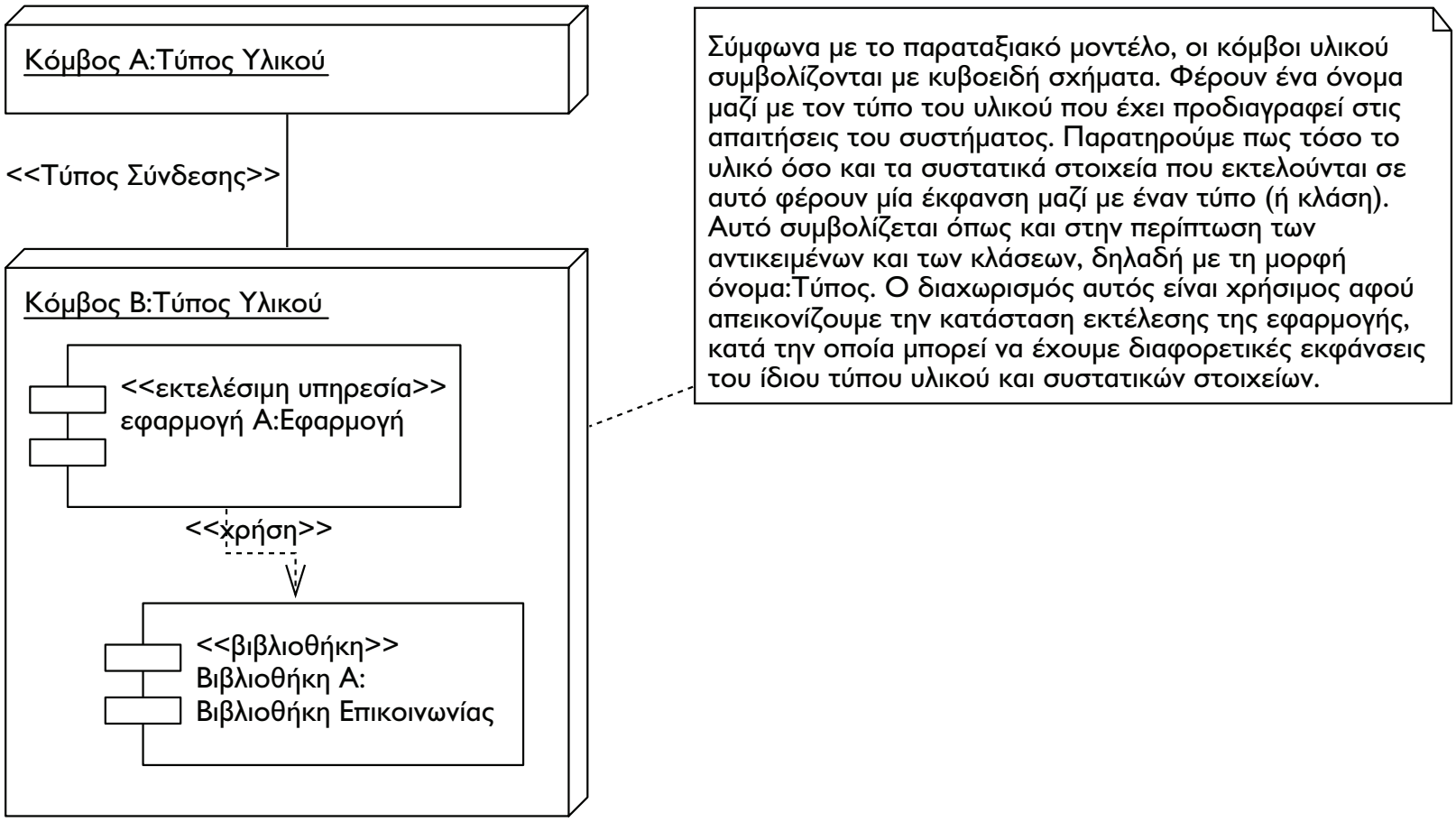
ΣΥΜΒΟΛΙΣΜΟΙ UML (ΣΥΣΤΑΤΙΚΩΝ ΣΤΟΙΧΕΙΩΝ)







ΣΥΜΒΟΛΙΣΜΟΙ UML (ΣΥΜΒΟΛΙΣΜΟΙ ΜΟΝΤΕΛΟΥ ΔΙΑΤΑΞΗΣ)



Το γεγονός ότι το μοντέλο διάταξης απεικονίζει την αρχιτεκτονική του συστήματος κατά την εκτέλεσή του μας αναγκάζει να προσέξουμε ένα επιπλέον χαρακτηριστικό των συστατικών στοιχείων και των κόμβων υλικού. Στα δύο αυτά σύμβολα είμαστε, πλέον, υποχρεωμένοι να εισάγουμε την έννοια της έκφανσης κατά την εκτέλεση (instantiation) σε αντιπαράθεση με τους τύπους (ή τα είδη) των τμημάτων υλικού ή των συστατικών στοιχείων. Αυτές οι έννοιες βρίσκονται σε ευθεία αναλογία (εννοιολογικά και συμβολικά) με τις έννοιες των αντικειμένων και των κλάσεων που έχουμε συναντήσει σε διάφορα σημεία του βιβλίου. Για να κατανοήσουμε τη διαφορά μεταξύ ενός είδους και μιας έκφανσής του ίσως βοηθήσει το να σκεφτόμαστε την έκφανση ως κάτι συγκεκριμένο και απτό και το είδος ως κάτι περιγραφικό και αφαιρετικό. Για παράδειγμα, η έννοια του ακέραιου αριθμού μπορεί να είναι μία κλάση, ενώ ο αριθμός 1 είναι μία έκφανσή της. Σε αντιστοιχία με το μοντέλο διάταξης, μπορεί να έχουμε μία κλάση (ή ένα είδος) «προσωπικός υπολογιστής» που να περιγράφει όλους τους προσωπικούς υπολογιστές, αλλά να χρειαστεί να εγκαταστήσουμε διαφορετικά συστατικά στοιχεία (για την ακρίβεια εκφάνσεις αυτών) σε συγκεκριμένους, διακριτούς υπολογιστές κ.ο.κ.

Δραστηριότητα 4/Κεφάλαιο 9

Ας αφήσουμε τη φαντασία μας ελεύθερη και ας υποθέσουμε πως χρειάζεται να αναπτύξουμε μία εφαρμογή διαδικτυακής τηλεφωνίας (VoIP) όπως, λόγου χάρη, το Skype. Τέτοιου είδους εφαρμογές επιτρέπουν στους χρήστες τους να συνομιλούν όχι μέσω του τηλεφωνικού δικτύου αλλά μέσω του διαδικτύου, μέσω πακέτων δεδομένων. Η χρήση τέτοιων προγραμμάτων απαιτεί έναν υπολογιστή με σύνδεση στο διαδίκτυο, ακουστικά και μικρόφωνο. Βιωματικά όλοι γνωρίζουμε ότι απαιτείται και η ύπαρξη κάποιου κεντρικού σημείου αναφοράς της υπηρεσίας. Υποθέστε ότι, βάσει της ανάλυσης και της σχεδίασης που έχουμε ήδη ολοκληρώσει, έχουμε καταλήξει πως η εφαρμογή μας θα εκτελείται σε προσωπικούς υπολογιστές και, όσον αφορά τους τελικούς χρήστες, θα χωρίζεται σε δύο υποσυστήματα: το υποσύστημα επικοινωνίας και το υποσύστημα διεπαφής (interface). Σχεδιάστε ένα διάγραμμα διάταξης για την εφαρμογή.

ΕΝΟΤΗΤΑ 9.3. ΒΗΜΑΤΑ ΣΤΗ ΣΧΕΔΙΑΣΗ

Έχοντας παρουσιάσει τις έννοιες και τα εκφραστικά εργαλεία που μας παρέχει η ενοποιημένη προσέγγιση ανάπτυξης λογισμικού για τη διαδικασία της σχεδίασης, μπορούμε να περάσουμε στα βήματα που μας επιτρέπουν να μεταβούμε από την ανάλυση στη σχεδίαση. Στη διαδικασία αυτή, δεδομένα εισόδου είναι το μοντέλο ανάλυσης και τα στοιχεία που το αποτελούν, όπως αυτά έχουν παρουσιαστεί στο Κεφάλαιο 3, ενώ κατά την έξοδο της διαδικασίας λαμβάνουμε το μοντέλο σχεδίασης, όπως αυτό περιγράφεται στο παρόν κεφάλαιο. Έχοντας στα χέρια μας το μοντέλο σχεδίασης, μπορούμε να προχωρήσουμε στην υλοποίηση, η οποία με τη σειρά της θα μας δώσει μια πρώτη λειτουργική έκδοση του συστήματος προς ανάπτυξη.

Ακολουθώντας αυτά τα βήματα και ενώ προσπαθούμε να δημιουργήσουμε το μοντέλο της σχεδίασης, οφείλουμε να κρατάμε κατά νου δύο δεδομένα που ισχύουν και στις άλλες μεταβατικές φάσεις από μία διαδικασία σε κάποια άλλη:

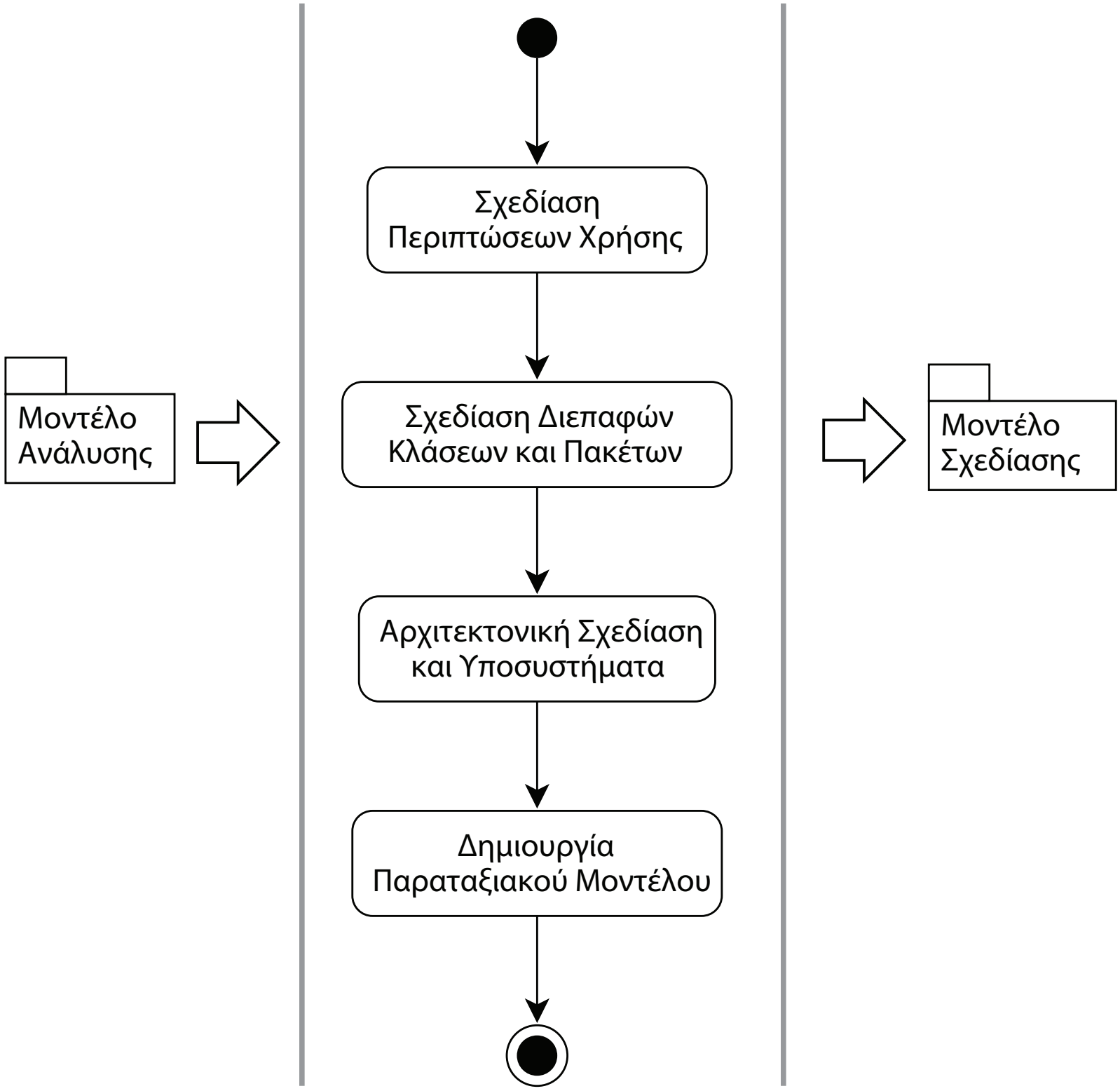
I) Δεν υπάρχει μία μοναδική ορθή μετάβαση ούτε ένα μοναδικό ορθό μοντέλο σχεδίασης δεδομένων ενός προβλήματος προς επίλυση και ενός μοντέλου ανάλυσης. Η ακριβής μεθοδολογία και τα βήματα που θα ακολουθηθούν θα εξαρτηθούν τόσο από την εφαρμογή που αναπτύσσεται όσο και από την ιδιοσυγκρασία του σχεδιαστή (μεταξύ πολλών άλλων παραμέτρων). Εδώ παρουσιάζεται μια μεθοδολογία που πιστεύουμε πως μπορεί να αποβεί χρήσιμη και διδακτική, χωρίς να είναι απαραίτητα και η βέλτιστη.

2) Όπως έχουμε αναφέρει σε διάφορα σημεία του βιβλίου, η ανάπτυξη λογισμικού είναι μια διαδικασία επαναληπτική και επαυξητική. Έτσι, οι διάφορες διαδικασίες, όπως αυτή της ανάλυσης, της σχεδίασης κ.λπ., δεν είναι πρακτικό να αντιμετωπιστούν απομονωμένα. Όπως έχουμε ήδη αναφέρει, το σενάριο κατά το οποίο ολοκληρώνουμε την ανάλυση και τη χρησιμοποιούμε αυτούσια σε όλα τα μεταγενέστερα στάδια της ανάπτυξης πρακτικά δεν υπάρχει. Σε μια ρεαλιστική κατάσταση, η σχεδίαση θα εγείρει ζητήματα τα οποία, προκειμένου να επιλυθούν, θα πρέπει να επαναπροσδιοριστούν και σημεία της ανάλυσης. Το ίδιο συμβαίνει σε όλα τα στάδια της

ανάπτυξης, αφού αποτελεί μια συνεχόμενη και επίπονη προσπάθεια και αλλαγές ή προβλήματα σε κάποιο στάδιο είναι φυσικό να επηρεάσουν προηγούμενα στάδια κ.ο.κ.

Κατά την αντικειμενοστρεφή φιλοσοφία, και όπως έχουμε δει στο τρέχον κεφάλαιο, αναγνωρίζονται τέσσερα διακριτά βήματα στη σχεδίαση, όπως φαίνονται στο Σχήμα 9.4.

Σχήμα 9.4 Βήματα στη σχεδίαση κατά την ενοποιημένη προσέγγιση.



Κατά τη σχεδίαση περιπτώσεων χρήσης, τα πακέτα ανάλυσης της κάθε περίπτωσης χρήσης μεταφέρονται στο μοντέλο σχεδίασης. Πρακτικά, αυτό σημαίνει πως το κάθε πακέτο ελέγχεται πως πληροί τις προϋποθέσεις της μελλοντικής υλοποίησης και συγκεκριμενοποιείται περαιτέρω. Αυτός ο έλεγχος και η συγκεκριμενοποίηση συμπεριλαμβάνει και τις υπάρχουσες κλάσεις και διεπαφές. Άρα, η σχεδίαση κλάσεων και διεπαφών ουσιαστικά ανήκει στη σχεδίαση περιπτώσεων χρήσης. Εδώ επιλέγουμε να παρουσιάσουμε τα βήματα αυτά ως διακριτά εξαιτίας της σημαντικότητάς τους. Η σχεδίαση περιπτώσεων χρήσης μάς αφήνει ένα σημείο αναφοράς με την αρχική ανάλυση του προβλήματος, ενώ η σχεδίαση των διεπαφών, των κλάσεων και τελικά των πακέτων μάς φέρνει πιο κοντά στην υλοποίηση. Για τον σκοπό αυτής της εισαγωγής θα κάνουμε την παραδοχή ότι το μοντέλο περιπτώσεων χρήσης παραμένει αμετάβλητο κατά τη σχεδίαση και θα συνεχίσουμε στη σχεδίαση των αντίστοιχων πακέτων. Αυτή η παραδοχή είναι ρεαλιστική, αφού αν κατά τη διάρκεια της σχεδίασης χρειαστεί να προβούμε στην αλλαγή της προδιαγραφής κάποιου πακέτου που αντιστοιχεί σε κάποια περίπτωση χρήσης, τότε, κατά πάσα πιθανότητα, οφείλουμε να διακόψουμε τη σχεδίαση του εν λόγω πακέτου και να επανεξετάσουμε την περίπτωση χρήσης από το επίπεδο της ανάλυσης, δηλαδή αναδρομικά. Βάσει των προηγούμενων βημάτων, η αρχιτεκτονική σχεδίαση μας βοηθά να διαχωρίσουμε τις διάφορες λογικές οντότητες της σχεδίασης σε υποσυστήματα με καλά ορισμένες λειτουργίες, που συνολικά θα μας δώσουν το θεμιτό αποτέλεσμα. Τέλος, η δημιουργία του μοντέλου εγκατάστασης θα μας δώσει μια πρακτική απεικόνιση της διάταξης των –μελλοντικά– υλοποιημένων τμημάτων λογισμικού που θα κληθούμε να εφαρμόσουμε στον εξοπλισμό που προβλέπεται από το έργο. Αυτή η σειρά βημάτων θα μας αφήσει σε μια θέση όπου η σταδιακή υλοποίηση του συστήματος να μπορεί να γίνει οργανωμένα και ελεγχόμενα.

9.3.1. Σχεδίαση περιπτώσεων χρήσης

Ήδη από το στάδιο της ανάλυσης οι περιπτώσεις χρήσεις έχουν αναλυθεί σε πακέτα αποτελούμενα από κλάσεις. Το καθένα από αυτά τα πακέτα αντιστοιχεί συνήθως σε μία περίπτωση χρήσης. Επιπλέον, οι κλάσεις της

ανάλυσης έχουν κατηγοριοποιηθεί σε κλάσεις οντοτήτων, ελέγχου και συνοριακές, και οι περιπτώσεις χρήσεις έχουν περιγραφεί βάσει διαγραμμάτων συνεργασίας. Αυτές οι κλάσεις όμως βρίσκονται ακόμα σε αρχικό επίπεδο και η περιγραφή τους είναι αρκετά αφηρημένη. Είναι προφανές πως η ανάλυση των κλάσεων βρίσκεται προσκολλημένη στο πεδίο του προβλήματος και όχι στη λύση που προσπαθούμε να επιτύχουμε. Ο σκοπός της σχεδίασης είναι η μετάβαση από το πεδίο του προβλήματος στο πεδίο της λύσης. Έτσι, με βάση τις κλάσεις της ανάλυσης θα προχωρήσουμε στις κλάσεις της σχεδίασης, ούτως ώστε η κάθε περίπτωση χρήσης να προδιαγραφεί με τέτοιο τρόπο που να καταστεί υλοποιήσιμη. Προφανώς, υπάρχουν διάφοροι τρόποι να μεταφέρουμε μία περίπτωση χρήσης στη σχεδίαση, και η διαδικασία είναι, όπως έχουμε τονίσει πολλές φορές, επαυξητική και επαναληπτική. Έχοντας ως εργαλεία κλάσεις και διεπαφές, μια αρχική προσέγγιση θα μπορούσε να είναι η ακόλουθη:

i. Αρχικά διατηρούμε τις περιπτώσεις χρήσης και τα πακέτα τους ως έχουν.

ii. Για κάθε περίπτωση χρήσης αναλογιζόμαστε τις κλάσεις ανάλυσης που έχουμε δημιουργήσει. Αντιμετωπίζουμε τους διαφορετικούς τύπους κλάσεων ως εξής:

   α. Οι συνοριακές κλάσεις θα μετατραπούν σε κλάσεις σχεδίασης, οι οποίες όμως θα διέπονται από διεπαφές. Αναλόγως με τις ανάγκες της εφαρμογής, μπορούμε να δημιουργήσουμε πολλές κλάσεις που θα υλοποιούν την κάθε διεπαφή, όμως θα υπάρχει μία διεπαφή για καθεμία συνοριακή κλάση.

   β. Οι κλάσεις ελέγχου θα μετατραπούν σε κλάσεις οι οποίες, αρχικά, δεν είναι απαραίτητο να έχουν εσωτερική κατάσταση, δηλαδή δεν θα είναι παρά συλλογές από μεθόδους οι οποίες θα έχουν, θεωρητικά τουλάχιστον, σημασιολογική συνοχή. Αυτό σημαίνει πως κάθε τέτοια κλάση θα μπορεί να δημιουργεί ένα μοναδικό αντικείμενο κατά την εκτέλεση της εφαρμογής. Αυτό μπορεί να αλλάξει μέχρι να κατασταλάξουμε στο τελικό σύνολο κλάσεων που θα χρειαστούμε και στη συμβολή τους στην εφαρμογή. Σε κάποιες μεθοδολογίες οι μέθοδοι των κλάσεων αυτών (και τελικά οι κλάσεις) μπορεί να ενσωματώνονται ως μέθοδοι σε άλλες κλάσεις.

γ. Οι κλάσεις οντοτήτων θα μετατραπούν σε ολοκληρωμένες κλάσεις με εσωτερική κατάσταση και μεθόδους.

iii. Με βάση την περίπτωση χρήσης που εξετάζουμε, «γεμίζουμε» την κάθε κλάση σχεδίασης με τα απολύτως απαραίτητα πεδία και μεθόδους που θα μπορέσουν να ικανοποιήσουν τις ανάγκες της περίπτωσης χρήσης. Για τον σκοπό αυτό είναι χρήσιμο να συμβουλευόμαστε και τα διαγράμματα συνεργασίας που έχουμε από την ανάλυση, καθώς και τα διαγράμματα ακολουθίας που δημιουργούμε κατά τη σχεδίαση.

iv. Για κάθε περίπτωση χρήσης δημιουργούμε ένα διάγραμμα κλάσεων και ένα διάγραμμα σειράς (sequence diagram). Αυτό θα μας βοηθήσει να εξακριβώσουμε την ορθότητα των μεθόδων που έχουμε εισάγει στις κλάσεις και να κατανοήσουμε περισσότερο τις ανάγκες της περίπτωσης χρήσης με βάση τα αντικείμενα που απαιτεί για την υλοποίησή της. Όταν έχουμε δημιουργήσει τα διαγράμματα κλάσεων για όλες τις περιπτώσεις χρήσης, μπορούμε να τα ενώσουμε σε ένα για όλη την εφαρμογή ή, στην περίπτωση που η εφαρμογή είναι μεγάλη, μπορούμε να περιοριστούμε σε ένα διάγραμμα κλάσης ανά υποσύστημα.

v. Εάν η περίπτωση χρήσης ικανοποιείται, μπορούμε να συνεχίσουμε με την επόμενη.

Όταν έχουμε δημιουργήσει κλάσεις και διεπαφές σχεδίασης για όλες τις περιπτώσεις χρήσης, ελέγχουμε πως δεν έχουμε πανομοιότυπες κλάσεις ή διεπαφές σε πολλαπλά μέρη. Αν αυτό συμβαίνει, τότε πρέπει να τα συμπτύξουμε με τέτοιο τρόπο ώστε οι εμπλεκόμενες περιπτώσεις χρήσης να συνεχίσουν να ικανοποιούνται. Τέτοιες συμπτύξεις θα μας καθοδηγήσουν και αργότερα, όταν θα καταλήξουμε στα υποσυστήματα της εφαρμογής μας. Η λογική πίσω από αυτή την κίνηση είναι απλή: αν δύο πακέτα έχουν κοινές κλάσεις, τότε ίσως σε συνδυασμό καλύπτουν μια σειρά συναφών λειτουργιών και, άρα, ίσως να έπρεπε να ανήκουν στο ίδιο υποσύστημα.

Εάν διαπιστώσουμε πως οι κλάσεις ή τα πακέτα μας δεν επαρκούν για να ικανοποιήσουν μια περίπτωση χρήσης ή πως έχουμε συχνά αλληλεπικαλυπτόμενες κλάσεις ή πακέτα που ικανοποιούν διαφορετικές περιπτώσεις

χρήσης, ίσως είναι χρήσιμο να επανεξετάσουμε τις περιπτώσεις χρήσης, να προβούμε σε αλλαγές και να επαναλάβουμε την ανάλυση.

Μελέτη περίπτωσης

Θα εφαρμόσουμε την παραπάνω μεθοδολογία στο πρόγραμμα «Επίκουρος» για την περίπτωση χρήσης πεδίου ανάλυσης «διαγραφή σπουδαστή». Η ανάλυση αυτής της περίπτωσης χρήσης έχει δοθεί στην Ενότητα 8.5.2. Το αντίστοιχο διάγραμμα συνεργασίας φαίνεται στο Σχήμα 8.22 του προηγούμενου κεφαλαίου.

Από τις συνοριακές κλάσεις της μελέτης προκύπτουν οι διεπαφές «Δ-Εγγραφές», από την κλάση «Class Interface εγγραφής», και «Δ-Σπουδαστές», από την κλάση «Class Interface σπουδαστή». Οι υπόλοιπες συνοριακές κλάσεις που έχουμε αναγνωρίσει αποτελούν διεπαφικά εργαλεία για τον χρήστη (τον χειριστή στη γραμματεία) και δεν θα μας απασχολήσουν στην παρούσα μελέτη.

Από την κλάση ελέγχου «διαγραφή σπουδαστή» προκύπτει η κλάση σχεδίασης «έλεγχος σπουδαστών». Βλέποντας το διάγραμμα συνεργασίας, σκεφτόμαστε πως αυτή θα περιέχει μεθόδους που θα μας επιτρέψουν να διαγράφουμε σπουδαστές και που θα συντονίζουν διάφορες αναζητήσεις σε λίστες εγγραφών, σπουδαστών κ.λπ. Δεν περιορίζουμε το όνομά της στην πράξη της διαγραφής, αφού η γραμματεία θα χρειαστεί να διαχειρίζεται επιπλέον παρόμοιες εντολές και σκεφτόμαστε πως μια τέτοια κλάση μπορεί να μας φανεί χρήσιμη και για άλλες περιπτώσεις χρήσης.

Τέλος, από τις κλάσεις οντοτήτων προκύπτουν οι κλάσεις σχεδίασης «εγγραφές» και «σπουδαστές». Αυτές οι κλάσεις θα αντικατοπτρίζουν κάποιας μορφής λίστα δεδομένων, της οποίας η υλοποίηση δεν μας αφορά σε αυτή τη φάση. Όμως, όπως προκύπτει από το διάγραμμα συνεργασίας, θα πρέπει να υποστηρίζουν ενέργειες όπως εύρεση, διαγραφή κ.λπ. Με άλλα λόγια, θα πρέπει να υπόκεινται στις διεπαφές που έχουμε αντιστοιχίσει, τις «Δ-Εγγραφές» και «Δ-Σπουδαστές».

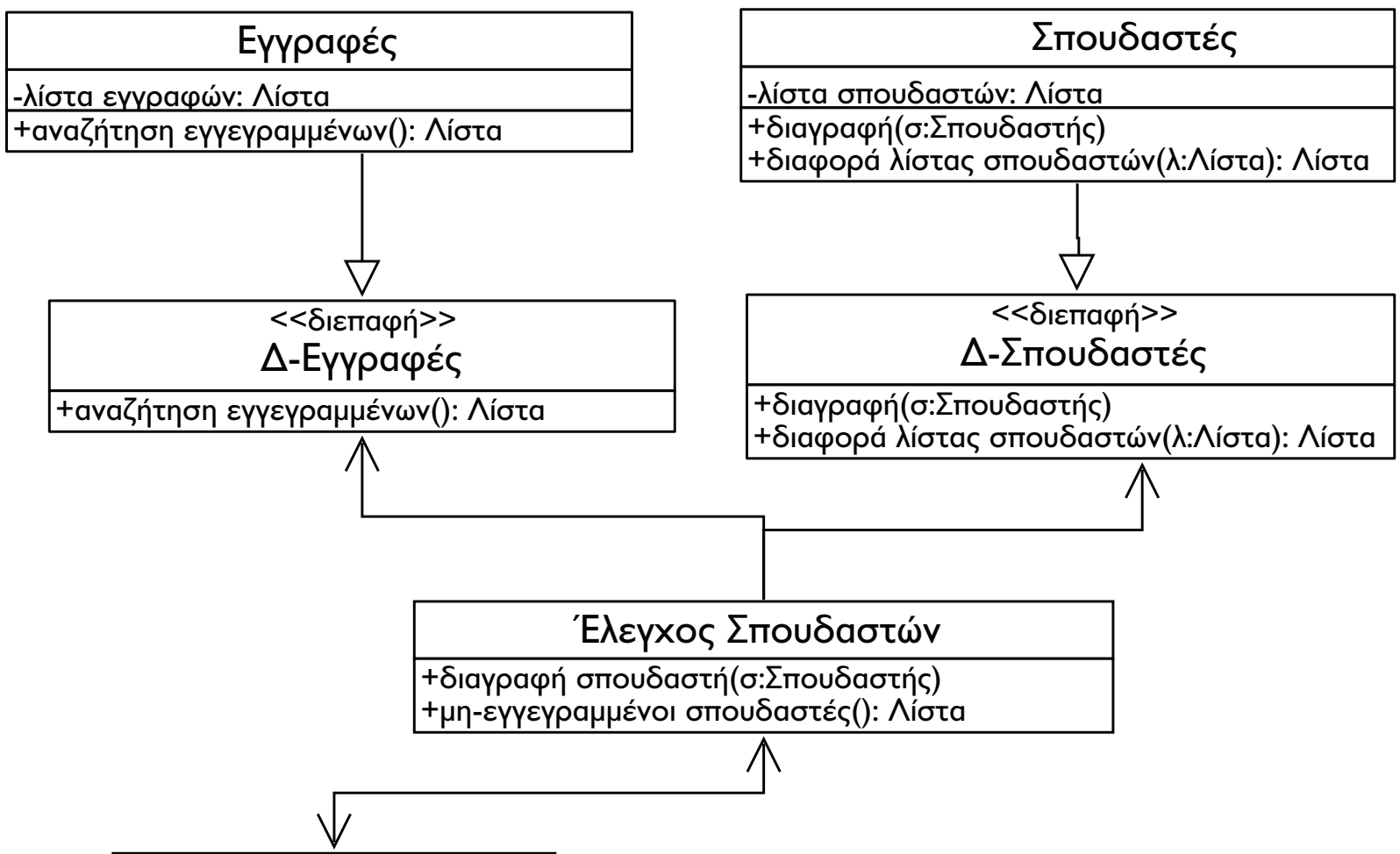
Έχοντας αναγνωρίσει τις κλάσεις που μας ενδιαφέρουν και έχοντάς τες μεταφέρει στο μοντέλο σχεδίασης, μπορούμε να περάσουμε στο επόμενο

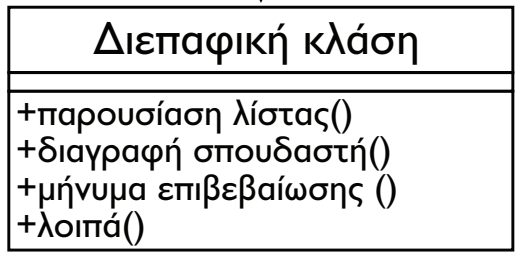
βήμα, όπου θα δώσουμε στις κλάσεις μεθόδους και, αν κρίνουμε σκόπιμο, πεδία. Ας ξεκινήσουμε με την ίδια σειρά, δηλαδή από τις διεπαφές. Για τη διεπαφή «Δ-Εγγραφές» αναγνωρίζουμε τη μέθοδο «αναζήτηση εγγεγραμμένων σπουδαστών», ενώ για τη διεπαφή «Δ-Σπουδαστές» αναγνωρίζουμε τις μεθόδους «διαγραφή» και «διαφορά λίστας σπουδαστών». Αυτές οι μέθοδοι θα πρέπει να υποστηρίζονται και από τις αντίστοιχες υλοποιήσιμες κλάσεις «εγγραφές» και «σπουδαστές», που τελικά θα μας εφοδιάσουν και με τα αντικείμενα που θα φέρουν εις πέρας το έργο.

Η κλάση ελέγχου «έλεγχος σπουδαστών» θα έχει τις μεθόδους «διαγραφή σπουδαστή», που θα διευθύνει την ενέργεια της διαγραφής, και τη μέθοδο «μη εγγεγραμμένοι σπουδαστές», που θα επιστρέφει μια λίστα σπουδαστών την οποία θα μπορεί το σύστημα να διαγράψει.

Έτσι, καταλήγουμε στο παρακάτω διάγραμμα κλάσεων:

Σχήμα 9.5 Διάγραμμα κλάσεων που προέκυψε από τη σχεδίαση της περίπτωσης χρήσης «διαγραφή σπουδαστή» του προγράμματος «Επίκουρος».

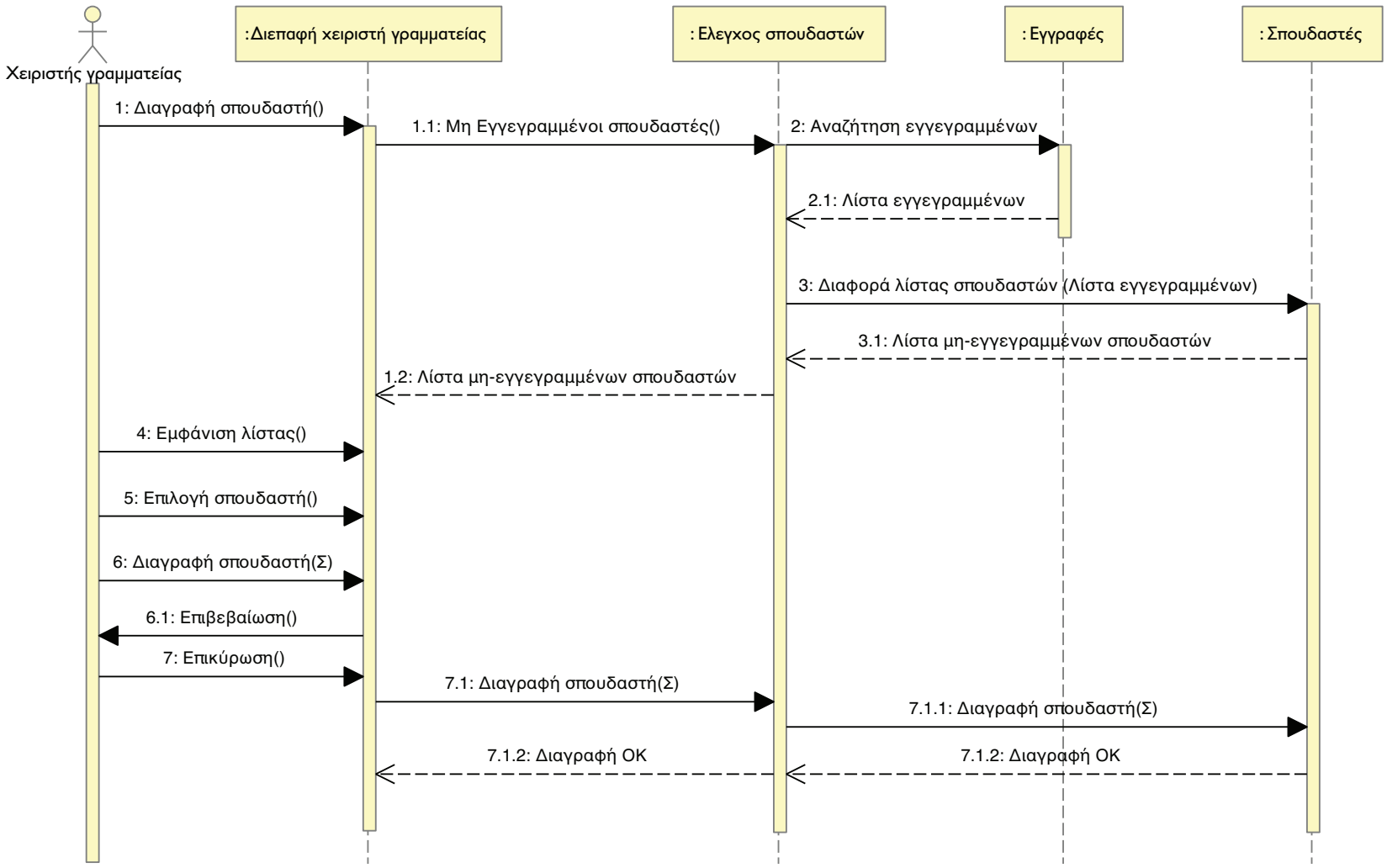




Σε αυτό το διάγραμμα, και για το σκοπό της διαφοροποίησης της περίπτωσης χρήσης από τη διεπαφή με τους τελικούς χρήστες, έχουμε εισάγει μία γενική κλάση «διεπαφική κλάση», η οποία έχει κάποιες μεθόδους για εισαγωγή και έξοδο δεδομένων. Αυτές οι μέθοδοι δεν έχουν προδιαγραφεί πλήρως, απλώς υπάρχουν για να μας βοηθήσουν να μοντελοποιήσουμε την περίπτωση χρήσης.

Έχοντας σχεδιάσει το διάγραμμα κλάσεων, θα περάσουμε στη δημιουργία ενός διαγράμματος ακολουθίας, σημειώνοντας ότι δεν είναι πάντα αυτή η σειρά με την οποία είναι απαραίτητο να κατασκευάζονται τα διαγράμματα αυτά. Ενδέχεται, κατά τη σχεδίαση της περίπτωσης χρήσης στην οποία βρισκόμαστε να έχουμε ήδη ένα διάγραμμα ακολουθίας το οποίο σε κάθε περίπτωση θα είναι συνεπές με το διάγραμμα κλάσεων, δηλαδή θα χρησιμοποιήσει τις μεθόδους που εισάγαμε στις κλάσεις. Το παρακάτω σχήμα απεικονίζει ένα διάγραμμα ακολουθίας που καλύπτει την περίπτωση που η διαγραφή του σπουδαστή γίνει επιτυχώς.

Σχήμα 9.6 Διάγραμμα ακολουθίας που περιγράφει την περίπτωση χρήσης «διαγραφή σπουδαστή» του προγράμματος «Επίκουρος».



Σε αυτό το σημείο, η περίπτωση χρήσης ικανοποιείται, άρα μπορούμε να πούμε πως η σχεδίασή της ολοκληρώθηκε.

Δραστηριότητα 5/Κεφάλαιο 9

Κατασκευάστε διαγράμματα κλάσεων και ακολουθίας για την περίπτωση χρήσης «διαγραφή μαθήματος» του λογισμικού «Επίκουρος».

Άσκηση 2/Κεφάλαιο 9

Για την εφαρμογή «Επίκουρος» και την περίπτωση χρήσης «εκτύπωση βαθμολογίας μαθήματος», παράγετε τα σχετικά διαγράμματα κλάσεων και ακολουθίας. Στη λύση σας υποθέστε πως η προηγούμενη ανάλυση της συγκεκριμένης περίπτωσης χρήσης μάς παρέχει την παρακάτω ροή:

1. Ο χειριστής «χειριστής γραμματείας» επιλέγει από το μενού την εντολή «προβολή μαθημάτων».
2. Ο χειριστής «χειριστής γραμματείας» επιλέγει ένα από τα μαθήματα.
3. Ο χειριστής «χειριστής γραμματείας» επιλέγει από το μενού την εντολή «εκτύπωση βαθμολογίας».
4. Ο «Επίκουρος» συλλέγει και εκτυπώνει όλους τους εγγεγραμμένους σπουδαστές και τις βαθμολογίες τους για το επιλεγμένο μάθημα.
5. Ο «Επίκουρος» ενημερώνει τον χειριστή πως η εκτύπωση ολοκληρώθηκε.

9.3.2. Αρχιτεκτονική σχεδίαση και υποσυστήματα

Η αρχιτεκτονική σχεδίαση και ο διαχωρισμός του συστήματος σε υποσυστήματα μπορεί να γίνει πριν ή μετά τη σχεδίαση των περιπτώσεων χρήσης. Όπως έχει προαναφερθεί, η απόφαση αυτή ανήκει κυρίως στον σχεδιαστή και εξαρτάται από διάφορους παράγοντες. Εδώ προτιμούμε να προβούμε στην αρχιτεκτονική σχεδίαση μετά, γιατί κατά τη σχεδίαση των περιπτώσεων χρήσης υπάρχει περίπτωση να διορθώσουμε και να προσθέσουμε κλάσεις στο μοντέλο σχεδίασης. Πολλές φορές αυτό έχει ως αποτέλεσμα την αλλαγή και των πακέτων που έχουμε, τα οποία θα μας υπαγορεύσουν σε μεγάλο βαθμό και τα υποσυστήματα που θα χρειαστούμε. Όπως και στη σχεδίαση των περιπτώσεων χρήσης, έτσι και στην αρχιτεκτονική σχεδίαση, δεν υπάρχει μία μοναδική ορθή λύση.

Η μεθοδολογία που θα ακολουθήσουμε για την αναγνώριση των υποσυστημάτων είναι απλή. Θα ξεκινήσουμε θεωρώντας ως υποσυστήματα τα πακέτα της σχεδίασης και θα προχωρήσουμε περαιτέρω με γνώμονα την εφαρμογή που αναπτύσσουμε και τα ειδικά χαρακτηριστικά της. Πολλές φορές τέτοια χαρακτηριστικά προέρχονται και από τις μη λειτουργικές ανάγκες. Για παράδειγμα, αν η εφαρμογή μας βασίζεται σε μία εξωτερική βάση δεδομένων, ίσως πρέπει να δημιουργήσουμε ένα υποσύστημα που θα ελέγχει και θα χρησιμοποιεί αυτή τη βάση δεδομένων. Μια άλλη προσέγγιση βασίζεται στους χρήστες της εφαρμογής και τις διαφορετικές όψεις που έχουν. Για παράδειγμα, σε μία διαδικτυακή πύλη που υποστηρίζει διαφορετικά επίπεδα χρηστών, όπως ελεγκτών, μελών, επισκεπτών κ.λπ., μπορεί να αναγνωρίσουμε υποσυστήματα που παρέχουν λειτουργίες στις διαφορετικές ομάδες χρηστών κ.ο.κ.

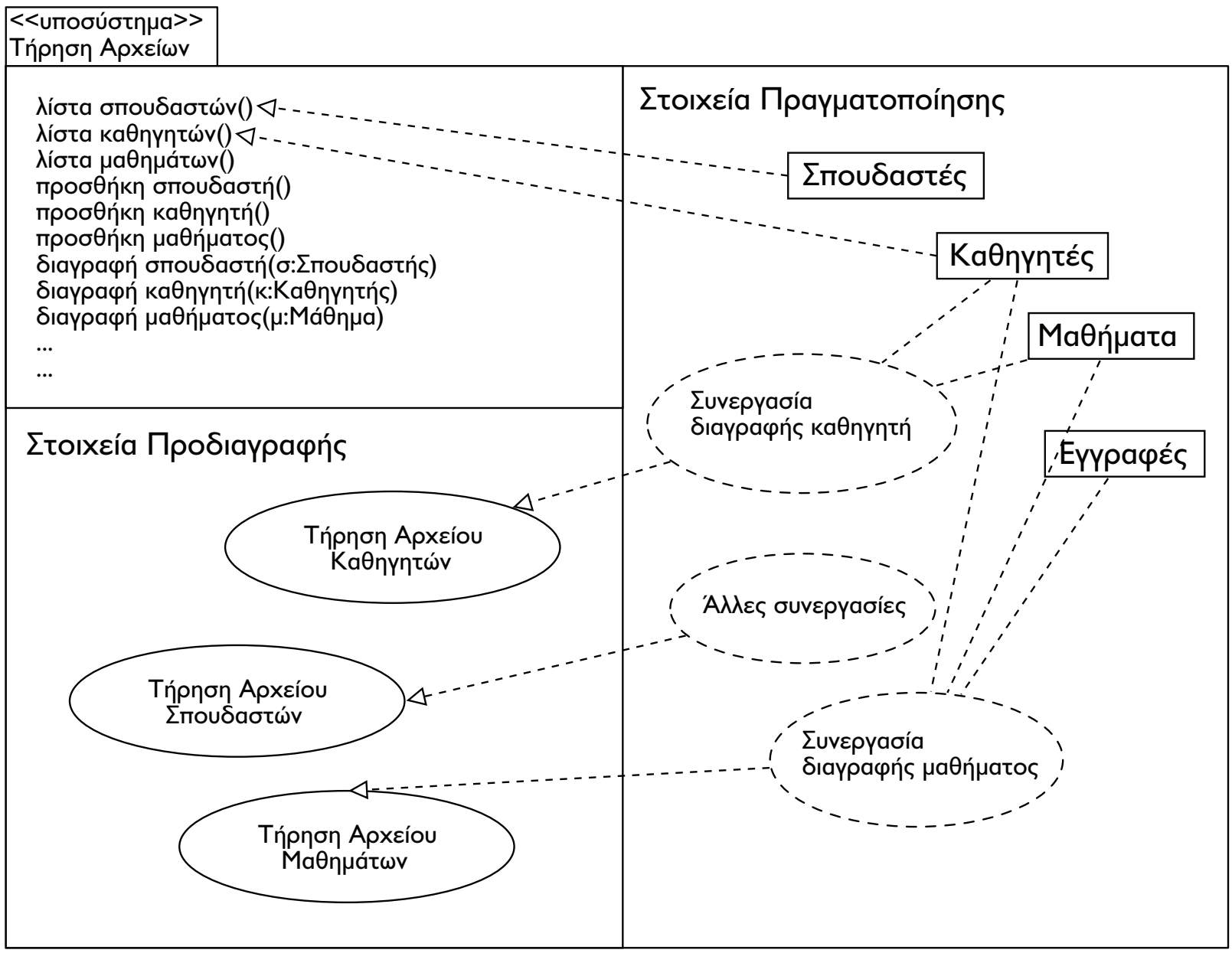
Σε κάθε περίπτωση, τα υποσυστήματα που αναγνωρίζουμε στη σχεδίαση πρέπει να παρέχουν κάτι συγκεκριμένο στο συνολικό σύστημα και οι λειτουργίες του από τα άλλα υποσυστήματα να είναι απολύτως διακριτές. Γενικά, κάθε υποσύστημα χρειάζεται να έχει τρία χαρακτηριστικά: την περιγραφή κάποιας διεπαφής για να επικοινωνεί με τα άλλα υποσυστήματα, έναν αριθμό κλάσεων που συντελούν στην εργασία που επιτελεί το υποσύστημα και έναν αριθμό περιπτώσεων χρήσης που ικανοποιεί. Επειδή η έννοια του

υποσυστήματος είναι συνυφασμένη με την έννοια του πακέτου κλάσεων, θα χρησιμοποιήσουμε τον ίδιο συμβολισμό με τα πακέτα, συγκεκριμενοποιώντας το όμως με τη χρήση του στερεότυπου «υποσύστημα».

Μελέτη περίπτωσης

Επιστρέφοντας στην εφαρμογή «Επίκουρος» και, συγκεκριμένα, στη δομή της ανάλυσής της (Σχήμα 8.17), ας υποθέσουμε πως έχει προηγηθεί η σχεδίαση του πακέτου «τήρηση αρχείων» και των συμπεριλαμβανομένων περιπτώσεων χρήσης. Τώρα, ας υποθέσουμε πως στο πακέτο αυτό συμπεριλαμβάνονται οι κλάσεις «εγγραφές», «μαθήματα», «καθηγητές» και «σπουδαστές», όπως επίσης και οι συναφείς τους μέθοδοι για την τήρηση του αρχείου, όπως «προσθήκη», «διαγραφή» κ.λπ. Εάν δεχτούμε πως η τήρηση αρχείου μπορεί να αποτελέσει ένα υποσύστημα, τότε αυτό προδιαγράφεται όπως στο παρακάτω σχήμα.

Σχήμα 9.7 Το υποσύστημα «τήρηση αρχείων» της εφαρμογής «Επίκουρος».



Το σχήμα αυτό είναι προφανώς ημιτελές, όμως η σημασία του είναι εμφανής. Το υποσύστημα χωρίζεται στη διεπαφή (είναι το τμήμα πάνω αριστερά) μέσω της οποίας θα επιτυγχάνεται η επικοινωνία με άλλα υποσυστήματα. Τα στοιχεία πραγματοποίησης περιέχουν τις κλάσεις που συνεργάζονται, ενώ τα στοιχεία προδιαγραφής είναι οι περιπτώσεις χρήσης που ικανοποιούνται. Στα στοιχεία πραγματοποίησης ανήκουν επίσης και τα διαγράμματα συνεργασίας μεταξύ των κλάσεων για τα οποία δίνονται παραπομπές μέσα στις διακεκομμένες ελλείψεις. Όπως είναι φανερό, ένα τέτοιο σχήμα γίνεται σύντομα δυσανάγνωστο. Για τον λόγο αυτό προτείνουμε τη χρήση απλού κειμένου ή ενός πίνακα για την περιγραφή του.

Είναι καλή πρακτική το κάθε στοιχείο της ανάλυσης και της σχεδίασης να φέρει και έναν μοναδικό κωδικό. Έτσι, οι παραπομπές στα διάφορα εμπλεκόμενα στοιχεία (όπως κλάσεις, διαγράμματα συνεργασίας κ.λπ.) γίνονται πιο εύκολα και σε μικρότερο χώρο. Αν θέλετε να πάτε λίγο παραπέρα, μπορείτε να δοκιμάσετε να ολοκληρώσετε το παραπάνω σχήμα του υποσυστήματος. Μια σκέψη είναι να χρησιμοποιήσετε κάποιο ευρετήριο κλάσεων, διαγραμμάτων συνεργασίας και περιπτώσεων χρήσης, το οποίο θα συνοδεύσει το σχήμα.

9.3.3. Δημιουργία μοντέλου διάταξης

Το τελευταίο βήμα της σχεδίασης είναι η αρχική δημιουργία ενός μοντέλου διάταξης ή εγκατάστασης (deployment) το οποίο θα απεικονίζει την «ανάθεση» των συστατικών στοιχείων του λογισμικού στα διάφορα μέρη του εξοπλισμού. Η διάταξη αυτή πιθανώς θα αλλάξει καθώς θα προχωρά η υλοποίηση, αλλά είναι σημαντικό να την έχουμε δημιουργήσει λίγο πριν ξεκινήσει η υλοποίηση, αφού θα αποτελέσει έναν χρήσιμο οδηγό για τους προγραμματιστές του συστήματος. Σε αυτήν τη φάση ακολουθούμε την απλή μέθοδο, σύμφωνα με την οποία το κάθε υποσύστημα αντιστοιχεί σε κάποιο συστατικό στοιχείο. Προσοχή πρέπει να δοθεί στην επικοινωνία των κόμβων λογισμικού, καθώς και στο είδος του κάθε υποσυστήματος.

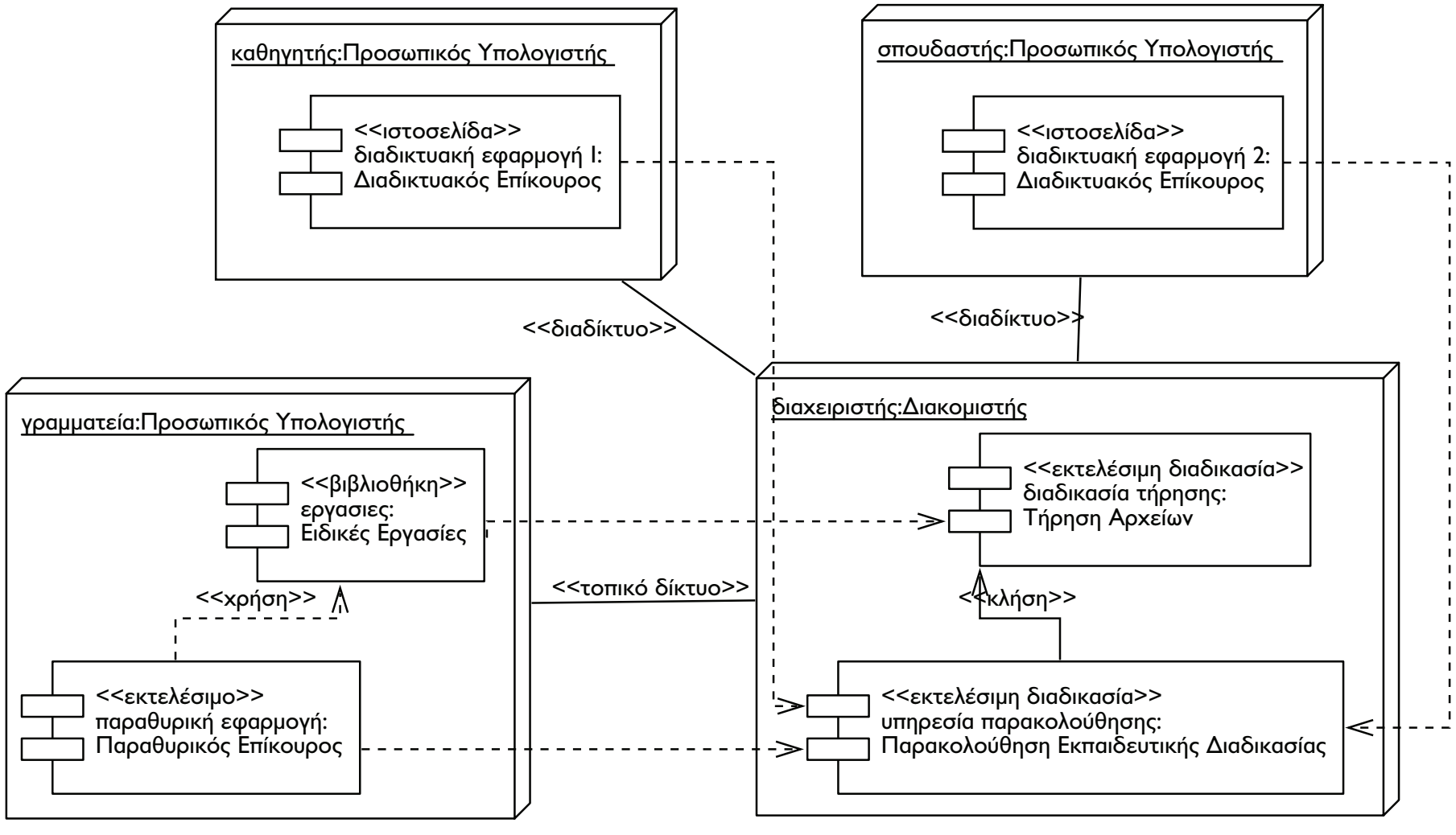
Μελέτη περίπτωσης

Θα επιστρέψουμε στην εφαρμογή «Επίκουρος» και θα χρησιμοποιήσουμε τα υπάρχοντα πακέτα που έχουν αναγνωριστεί κατά την ανάλυση ως συστατικά στοιχεία της εφαρμογής. Θα κάνουμε επίσης τις εξής παραδοχές:

I. Η τήρηση αρχείων θα γίνεται απομακρυσμένα, δηλαδή σε κάποιον διακομιστή και όχι τοπικά στον υπολογιστή του χρήστη της γραμματείας.
2. Η παρακολούθηση της εκπαιδευτικής διαδικασίας θα γίνεται και από καθηγητές και από σπουδαστές μέσω μιας δικτυακής πύλης. Αυτό σημαίνει πως θα υπάρχουν δύο καινούρια υποσυστήματα που θα παρέχουν διεπαφικές δυνατότητες στους χρήστες: ένα μέσω συμβατικού παραθυρικού περιβάλλοντος και ένα μέσω διαδικτύου.
3. Οι ειδικές εργασίες θα γίνονται μόνο από τη γραμματεία.

Αφού έχουμε κάνει αυτές τις παραδοχές, το διάγραμμα διάταξης της εφαρμογής μας θα μπορούσε να είναι το παρακάτω:

Σχήμα 9.8 Το διάγραμμα διάταξης για την εφαρμογή «Επίκουρος».



Άσκηση 3/Κεφάλαιο 9

Αν προσθέσουμε ένα ακόμα κανάλι χρήσης της εφαρμογής «Επίκουρος», για παράδειγμα, από κινητό τηλέφωνο, μπορείτε να περιγράψετε πώς θα άλλαζε το παραπάνω σχήμα; Αν το συστατικό στοιχείο «παρακολούθηση εκπαιδευτικής διαδικασίας» βρισκόταν στους επιμέρους υπολογιστές, πώς θα επηρεαζόταν η διαδικασία της εγκατάστασης;

ΕΝΟΤΗΤΑ 9.4. ΑΠΑΝΤΗΣΕΙΣ ΔΡΑΣΤΗΡΙΟΤΗΤΩΝ ΚΑΙ ΑΣΚΗΣΕΩΝ

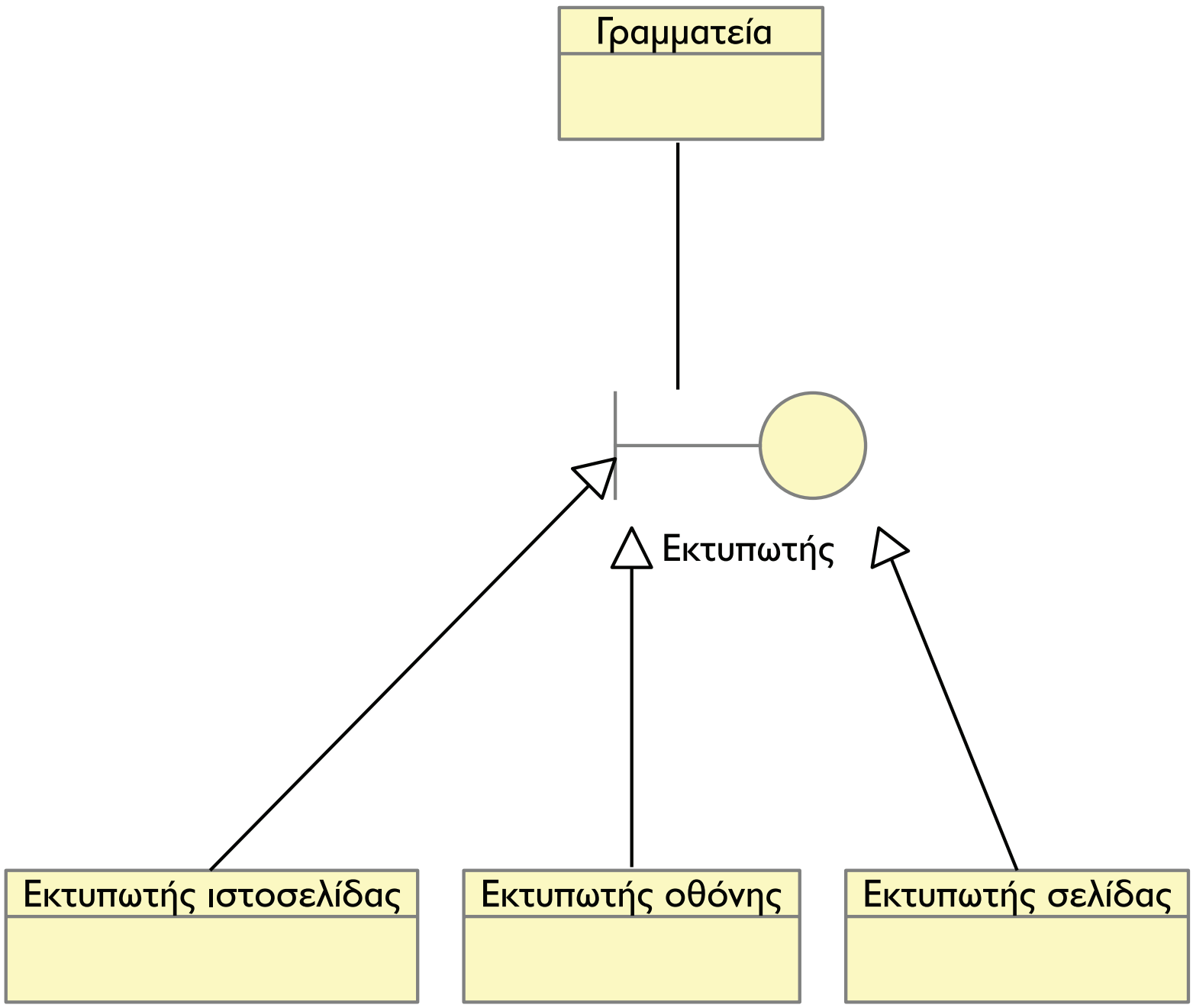
Δραστηριότητα I/Κεφάλαιο 9

Από την περιγραφή της εφαρμογής προκύπτει πως η γραμματεία θα χρειάζεται να εκτυπώνει βαθμολογίες σπουδαστών, όπως επίσης και να τις παρουσιάζει στην οθόνη. Σε κάποια ενδεχόμενη επέκταση θα μπορούσαν να απαιτηθούν και επιπλέον εναλλακτικά μέσα για την εκτύπωση βαθμολογιών, όπως εμφάνιση σε ιστοσελίδες κ.ά. Επομένως, αν η εκτύπωση της βαθμολογίας (ή και άλλων στοιχείων) ενός σπουδαστή γίνεται αποκλειστικά μέσα από την εφαρμογή της γραμματείας, θα δυσκολέψουμε την επεκτασιμότητα και θα περιορίσουμε την ευελιξία της εφαρμογής. Αυτό γιατί, ενώ η εφαρμογή της γραμματείας απλώς χρειάζεται να στείλει κάποια βαθμολογία για εκτύπωση, θα πρέπει και να διαχειρίζεται τους ακριβείς τρόπους των διαφόρων εκτυπώσεων. Επιπλέον, η παραμικρή αλλαγή στον τρόπο εκτύπωσης βαθμολογιών θα επισύρει και μια σειρά αλλαγών στην εφαρμογή της γραμματείας, που θα είναι ήδη επιφορτισμένη με μια σειρά επιπρόσθετων λειτουργιών.

Η χρήση διεπαφών μάς επιτρέπει να προσεγγίσουμε αυτό το πρόβλημα με έναν πιο αποτελεσματικό τρόπο, ο οποίος μάλιστα είναι πιο κοντά στον φυσικό κόσμο του προβλήματος. Αν χρησιμοποιήσουμε τη διεπαφή «εκτυπωτής» να ορίζει μια μέθοδο «εκτύπωση σπουδαστή (σ: Σπουδαστής)», θα είμαστε σε θέση να έχουμε την εφαρμογή της γραμματείας να καλεί οποιονδήποτε εκτυπωτή χρειάζεται μέσω αυτής της διεπαφής. Έπειτα, θα

μπορούμε να δημιουργήσουμε διάφορες κλάσεις που θα υλοποιούν διαφορετικού τύπου εκτυπωτές μέσω της προκείμενης διεπαφής. Τέτοιες κλάσεις, για παράδειγμα, θα μπορούσαν να είναι οι εξής: «εκτυπωτής οθόνης», «εκτυπωτής χαρτιού», «εκτυπωτής ιστοσελίδας» κ.ο.κ. Η σχεδίαση αυτού του εναλλακτικού τρόπου εκτύπωσης φαίνεται στο Σχήμα 9.9. Με αυτόν τον τρόπο θα επιτυγχάναμε μεγαλύτερη ευελιξία και επεκτασιμότητα.

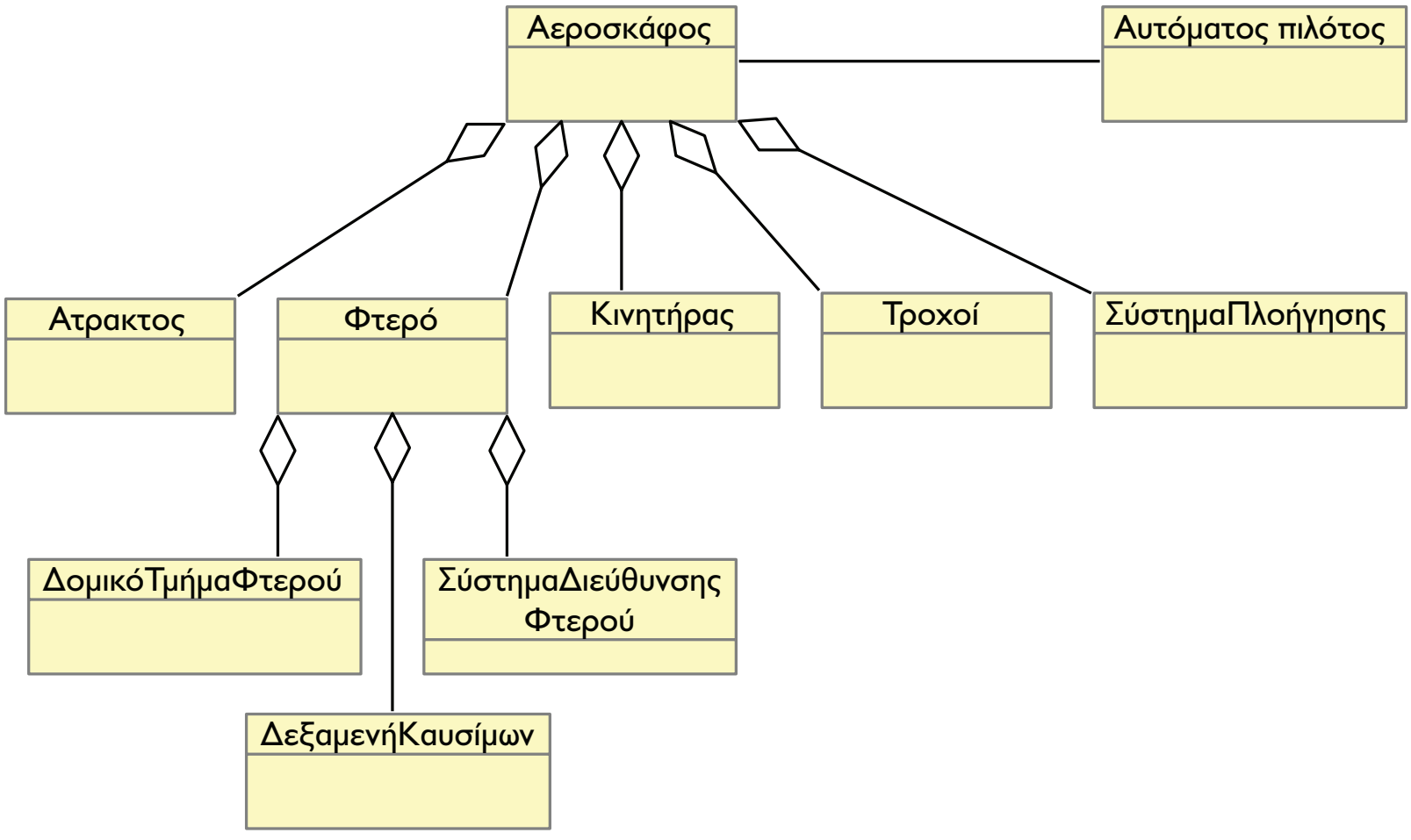
Σχήμα 9.9 Η εναλλακτική σχεδίαση της εκτύπωσης βαθμολογίας σπουδαστών για την εφαρμογή «Επίκουρος».



Δραστηριότητα 2/Κεφάλαιο 9

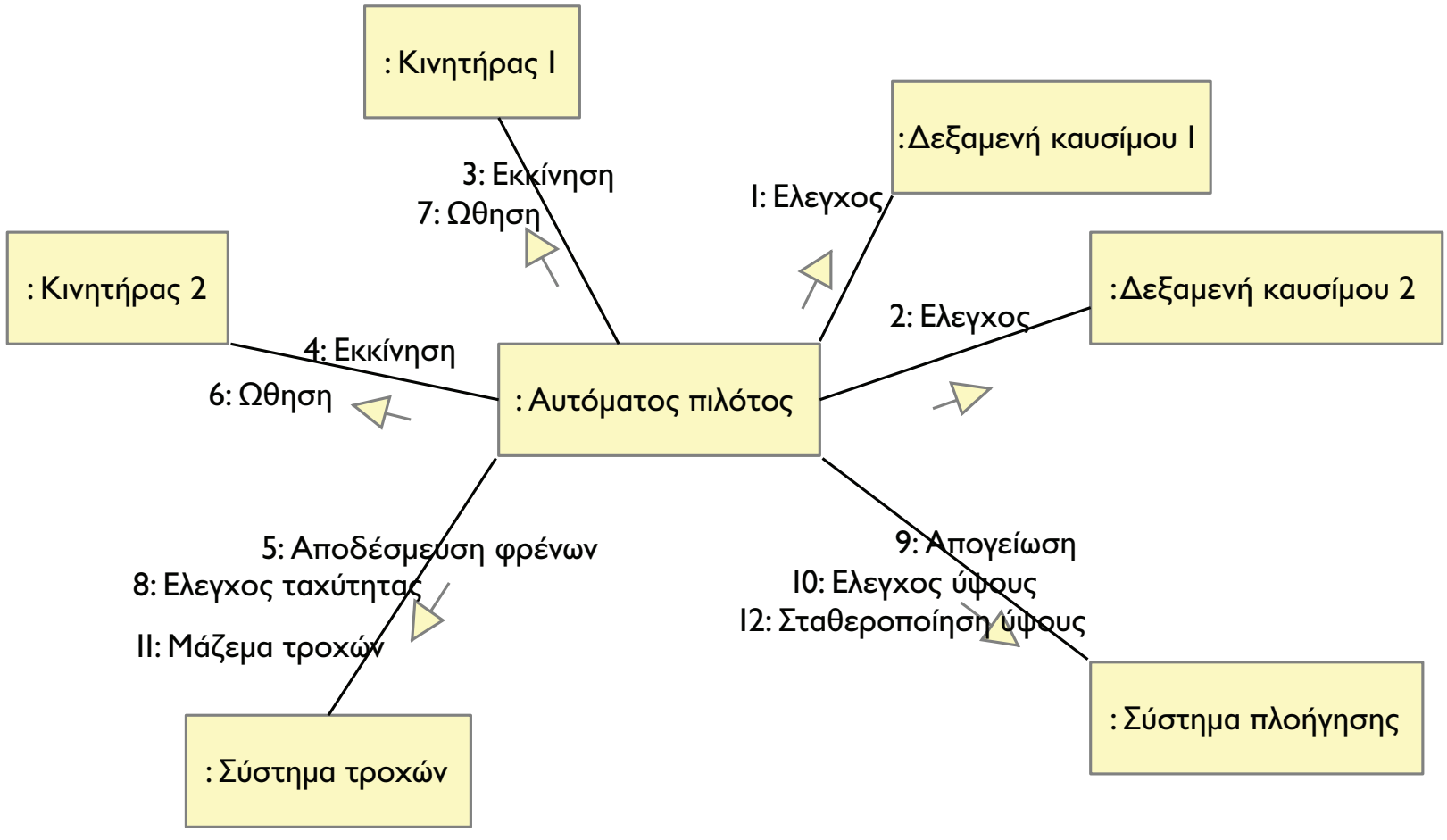
Το συγκεκριμένο διάγραμμα κλάσεων είναι ένα σωστό διάγραμμα μιας και μεταφέρει με άμεσο τρόπο τη φυσική σχέση συναρμολόγησης των διαφόρων εξαρτημάτων ενός αεροσκάφους στην αντικειμενοστρεφή λογική. Το διάγραμμα κλάσεων του παραδείγματος θα μπορούσε να χρησιμοποιηθεί αυτούσιο. Αποτελώντας μία καινούρια κλάση, ο αυτόματος πιλότος θα μπορούσε να επικοινωνεί με ένα αντικείμενο της κλάσης «αεροσκάφος», έτσι ώστε να χειριστεί τα διάφορα εξαρτήματα του αεροσκάφους ως εξής:

Σχήμα 9.10 Παράδειγμα της σχέσης συναρμολόγησης.



Μοντελοποιώντας μια υποθετική περίπτωση χρήσης που θα επιτρέψει στον αυτόματο πιλότο να απογειώσει ο αεροσκάφος, είναι εύκολο να καταλήξουμε στο παρακάτω διάγραμμα συνεργασίας:

Σχήμα 9.11 Ένα διάγραμμα συνεργασίας.

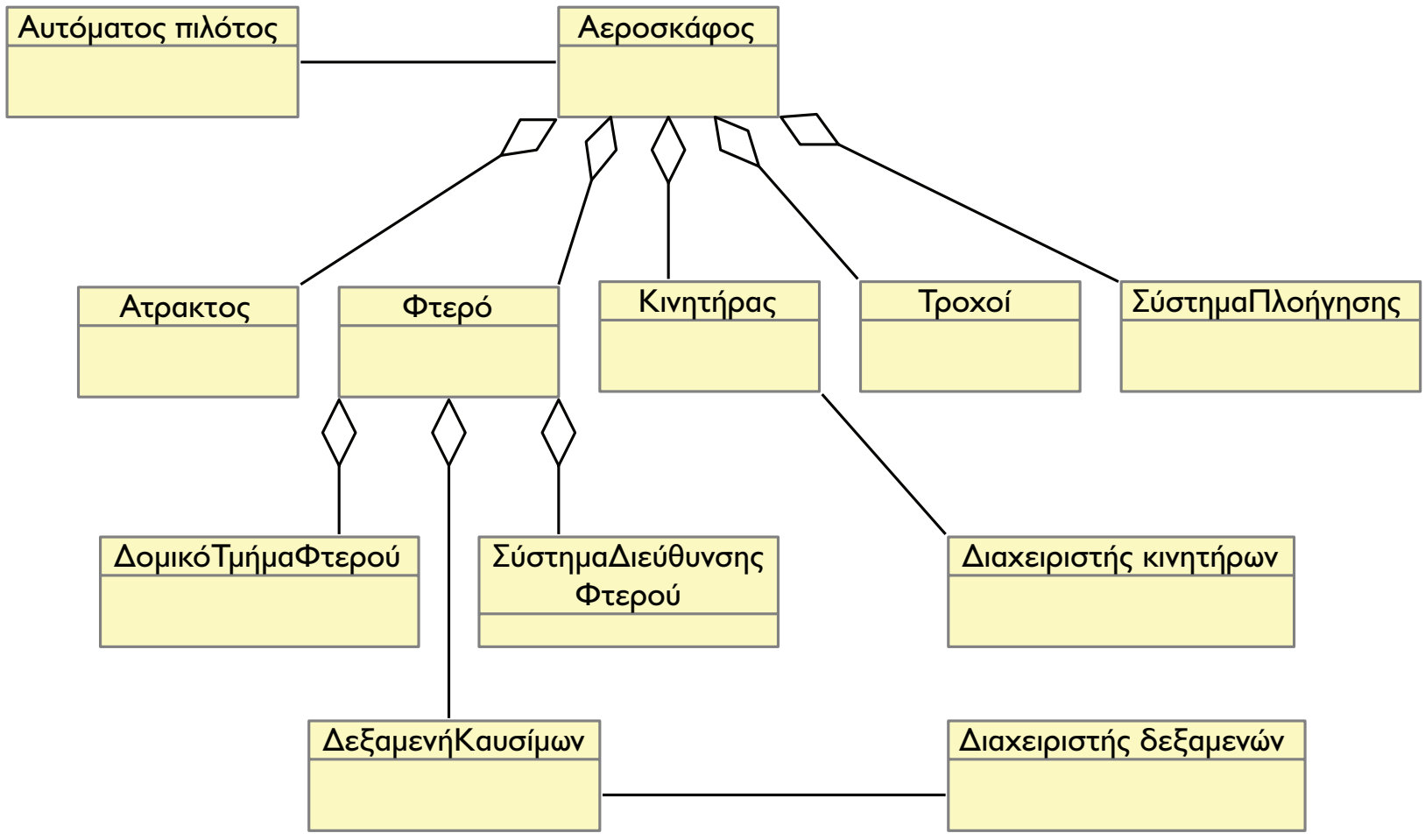


Έχοντας αυτό το διάγραμμα συνεργασίας, βλέπουμε πως το παραπάνω διάγραμμα κλάσεων θα μπορούσε, δεδομένων των απαραίτητων μεθόδων, να χρησιμοποιηθεί για την υλοποίηση της περίπτωσης χρήσης της απογείωσης. Όμως, παρατηρούμε πως επιδέχεται βελτιώσεων σε τουλάχιστον δύο σημεία:

1. Δεν θα ήταν λογικό για τον αυτόματο πιλότο να εκκινεί τους δύο (ή σε άλλο αεροσκάφος παραπάνω) κινητήρες ξεχωριστά για τον σκοπό της απογείωσης. Άρα, θα μπορούσε να βρεθεί ένας τρόπος, έτσι ώστε να αποφευχθεί αυτή η επανάληψη, αν είναι δυνατό. Το ίδιο θα έπρεπε να συμβεί και για τις δεξαμενές καυσίμου.
2. Από τη μεριά του διαγράμματος κλάσεων βλέπουμε πως η δεξαμενή καυσίμου ανήκει στην κλάση «φτερό». Όμως, όσον αφορά την περίπτωση χρήσης μας, αυτή η σχέση δεν είναι χρήσιμη. Με άλλα λόγια, ενώ αποτυπώνει τη δομή του αεροσκάφους, δεν αποτυπώνει τις ανάγκες μας για την επίλυση του προβλήματος.

Με αυτή την επιπλέον γνώση, που προέκυψε από την τριβή μας με το πρόβλημα, μπορούμε να καταλήξουμε στο εξής εναλλακτικό διάγραμμα κλάσεων:

Σχήμα 9.12 Το τελικό διάγραμμα κλάσεων.

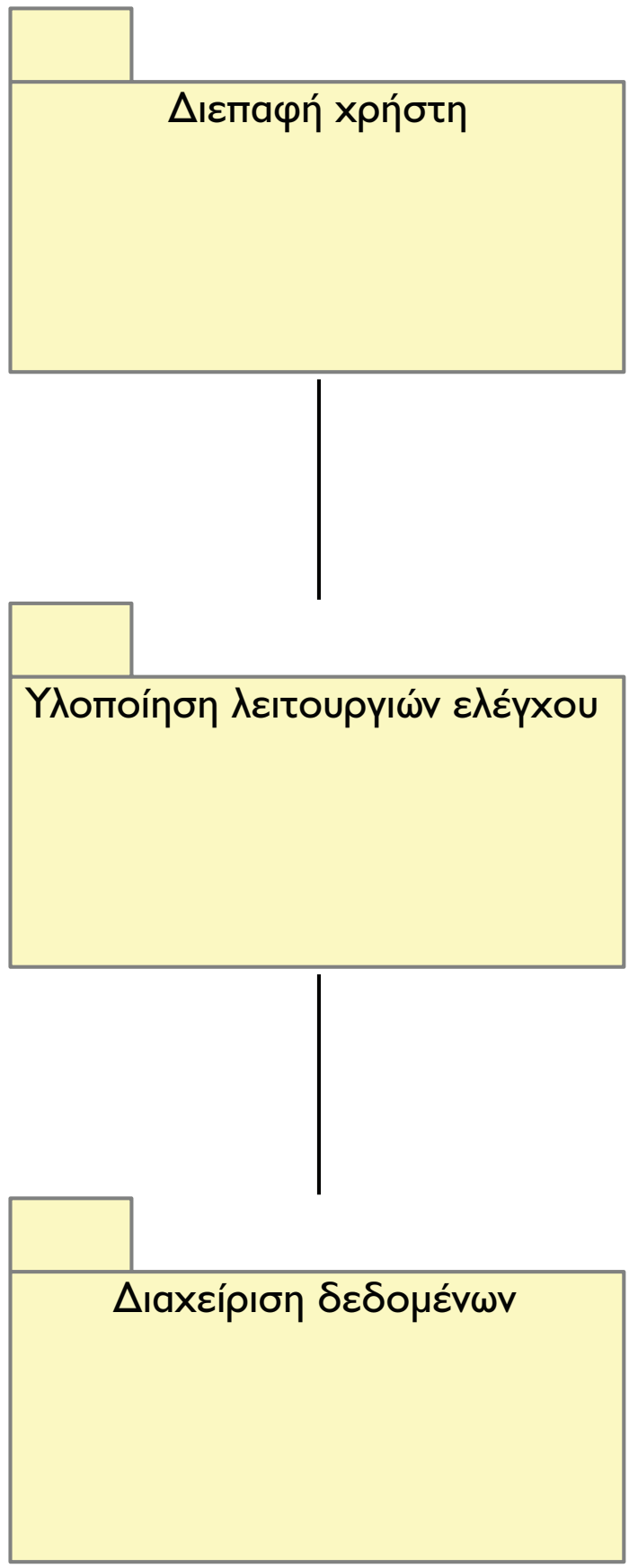


Δραστηριότητα 3/Κεφάλαιο 9

Βάσει της περιγραφής που φαίνεται στο σχήμα 8.11, μπορούμε να χωρίσουμε το τελικό σύστημα σε τρία υποσυστήματα:

1. Υποσύστημα γραφικής διεπαφής χρηστών (Graphical User Interface – GUI).
2. Υποσύστημα ελέγχου και διεκπεραίωσης των διαδικασιών.
3. Βάση δεδομένων σπουδαστών και καθηγητών.

Σχήμα 9.13 Ορισμός υποσυστημάτων.

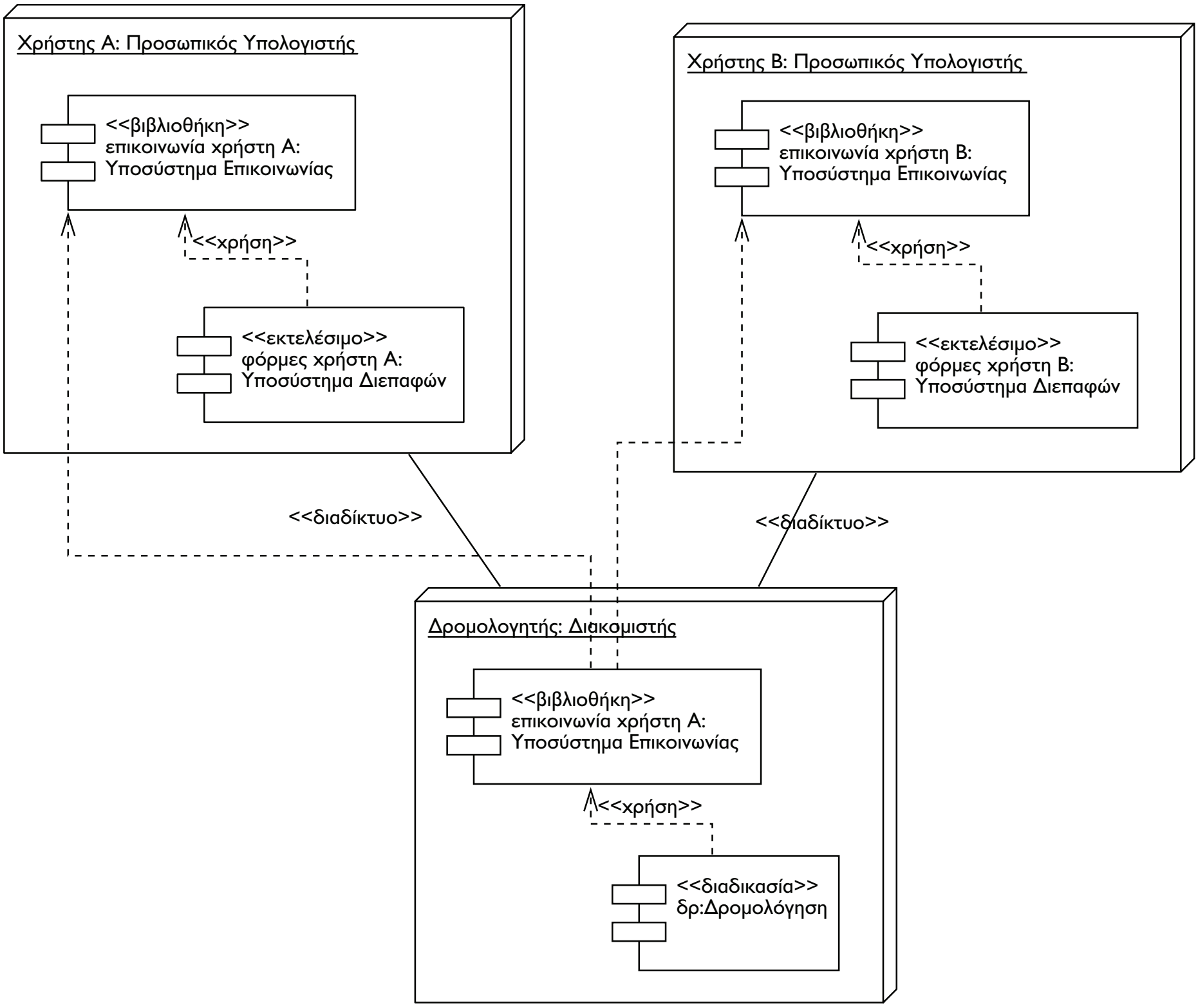


Το υποσύστημα GUI θα είναι υπεύθυνο για την παρουσίαση της εφαρμογής στους χρήστες/χειριστές της, δηλαδή στον χειριστή της γραμματείας. Θα παρέχει ένα παραθυρικό περιβάλλον μέσα από το οποίο ο χρήστης θα μπορεί να διεκπεραιώσει ό,τι προβλέπεται από τις σχετικές περιπτώσεις χρήσης.

Δραστηριότητα 4/Κεφάλαιο 9

Το υποσύστημα επικοινωνίας θα αναλαμβάνει τη μετάδοση και λήψη δεδομένων φωνής, ενώ το υποσύστημα διεπαφής θα παρέχει στον χρήστη ένα περιβάλλον λειτουργίας της εφαρμογής. Στην αρχιτεκτονική μας, τα δεδομένα φωνής θα ταξιδεύουν μεταξύ των ομιλητών μέσω ενός κεντρικού διακομιστή που θα αναλαμβάνει να τα προωθεί στους σωστούς τελικούς χρήστες βάσει της υπάρχουσας σύνδεσης. Το μοντέλο διάταξης ενός τέτοιου σεναρίου φαίνεται στο επόμενο σχήμα.:

Σχήμα 9.14 . Το μοντέλο διάταξης του προβλήματός μας.



Για τη διεκπεραίωση μιας υποτιθέμενης συνομιλίας μεταξύ δύο χρηστών Α και Β συμβάλλουν τρείς κόμβοι: οι δύο προσωπικοί υπολογιστές των χρηστών και ο διακομιστής που αναλαμβάνει τη δρομολόγηση των δεδομένων φωνής. Στους δύο προσωπικούς υπολογιστές θα πρέπει να εκτελείται το υποσύστημα διεπαφής, ούτως ώστε οι χρήστες να μπορούν να χρησιμοποιήσουν τις λειτουργίες του προγράμματος. Όμως, μιας και πρόκειται για δύο διαφορετικούς υπολογιστές, μιλάμε για δύο διαφορετικές εκδοχές (στιγμιότυπα, εκφάνσεις) του υποσυστήματος αυτού. Ακόμα πιο χαρακτηριστική είναι η περίπτωση του υποσυστήματος επικοινωνίας. Μιας και όλοι ανεξαιρέτως οι κόμβοι θα πρέπει να δέχονται και να στέλνουν φωνητικά δεδομένα, εκφάνσεις του υποσυστήματος αυτού θα βρίσκονται σε λειτουργία τόσο στους προσωπικούς υπολογιστές των τελικών χρηστών όσο και στον ενδιάμεσο διακομιστή παρά τις διαφορές που μπορεί να έχουν από άποψη υλικού ή άλλων εκτελούμενων συστατικών στοιχείων, υποσυστημάτων ή λειτουργιών.

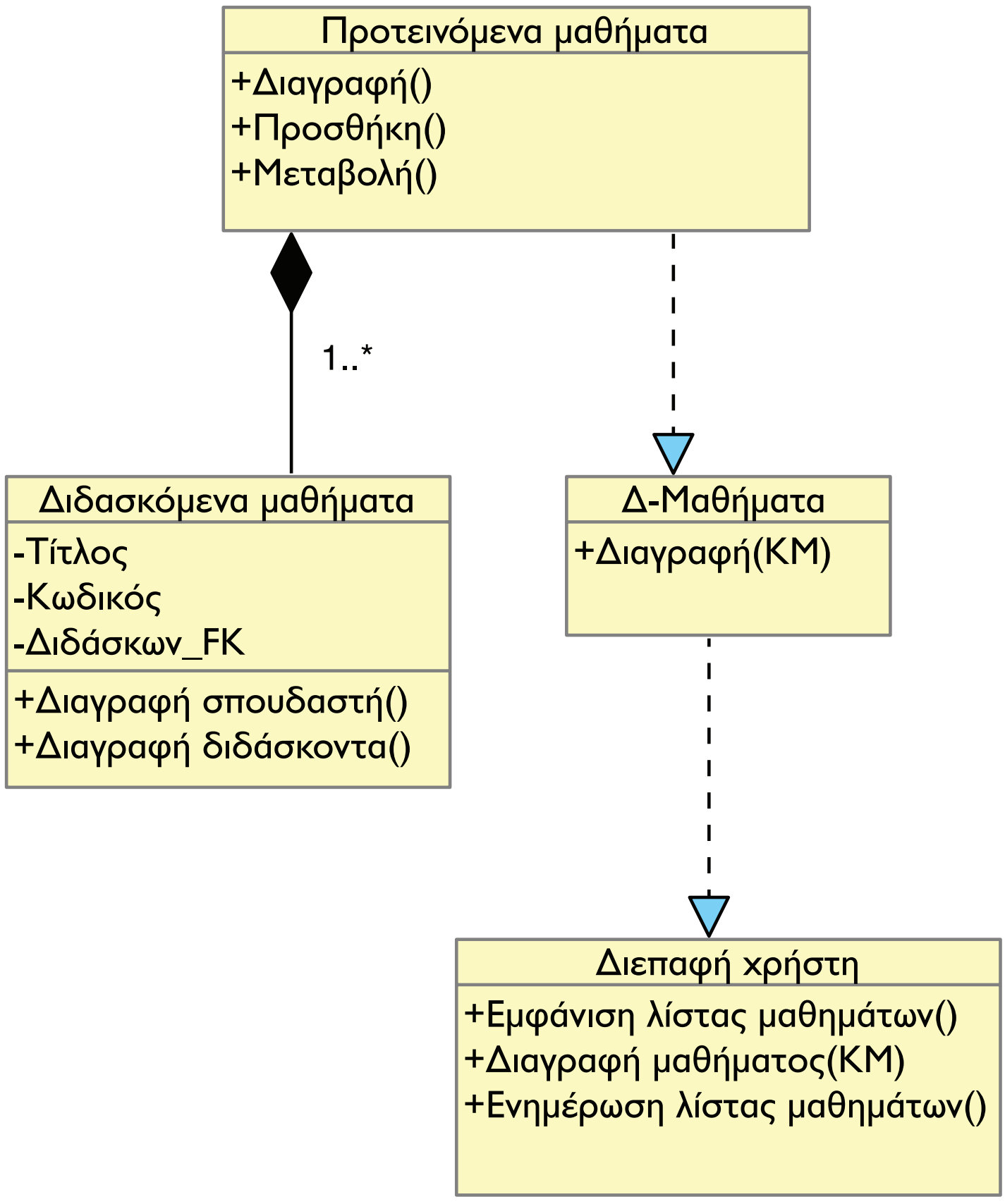
Δραστηριότητα 5/Κεφάλαιο 9

Υποθέτουμε πως η περίπτωση χρήσης «διαγραφή μαθήματος» έχει την ακόλουθη βασική ροή γεγονότων:

1. Ο χειριστής «χειριστής γραμματείας» επιλέγει από το μενού την εντολή «προβολή μαθημάτων».
2. Ο χειριστής επιλέγει ένα μάθημα από τη λίστα.
3. Ο χειριστής επιλέγει από το μενού την εντολή «διαγραφή μαθήματος».
4. Ο «Επίκουρος» ελέγχει αν υπάρχουν εγγεγραμμένοι σπουδαστές στο συγκεκριμένο μάθημα. Αν υπάρχουν, τους αποσυνδέει από τη λίστα σπουδαστών του μαθήματος.
5. Ο «Επίκουρος» ελέγχει αν υπάρχουν εγγεγραμμένοι καθηγητές που διδάσκουν το επιλεγμένο μάθημα. Αν υπάρχουν, τους αποσυνδέει από τη λίστα καθηγητών του μαθήματος.
6. Ο «Επίκουρος» διαγράφει μόνιμα το μάθημα.
7. Ο «Επίκουρος» ενημερώνει τη λίστα μαθημάτων του χρήστη.

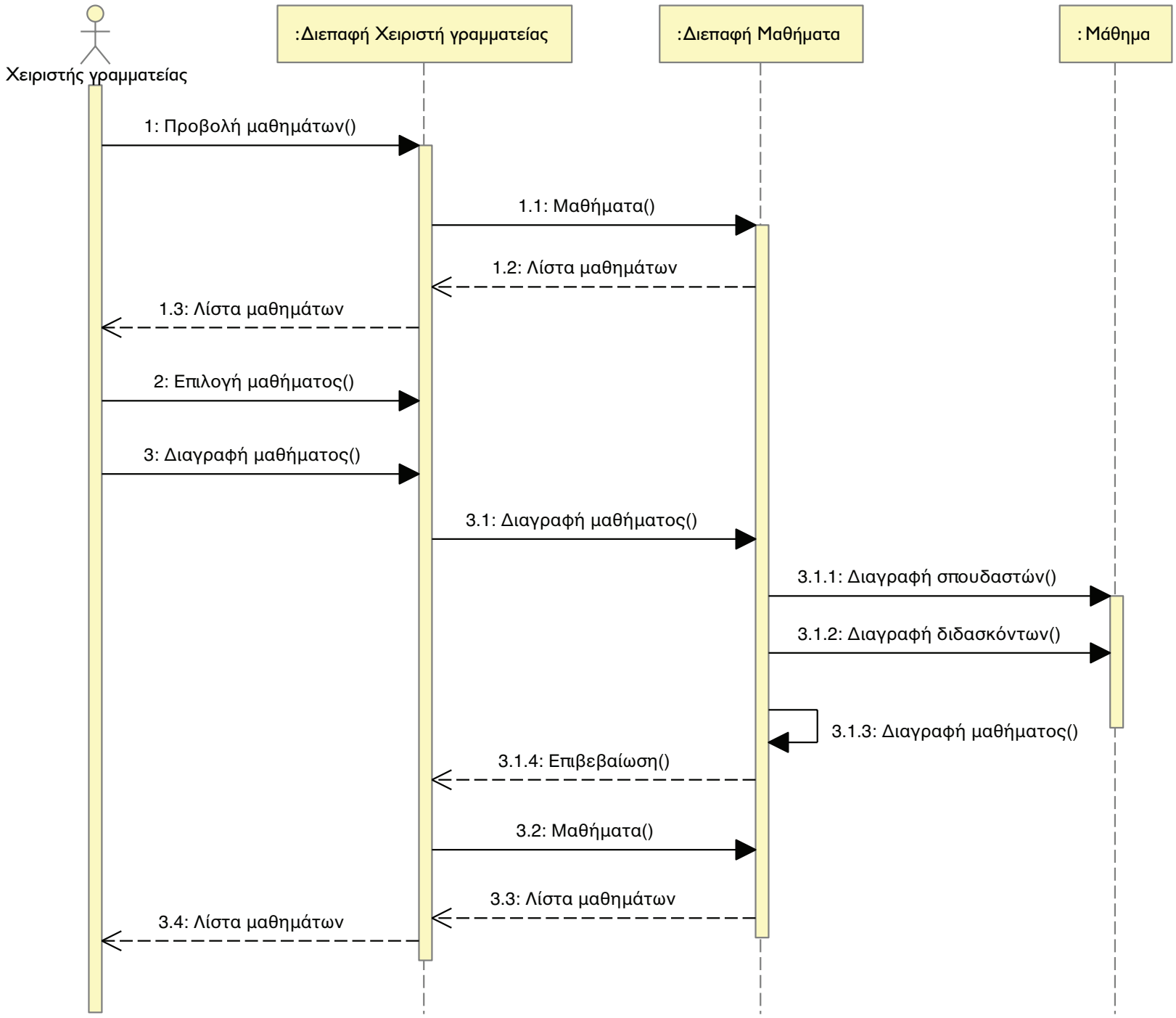
Βάσει αυτής της ροής και του διαγράμματος κλάσεων της περίπτωσης που μελετήσαμε πιο πάνω, μπορούμε να καταλήξουμε στο παρακάτω διάγραμμα κλάσεων.

Σχήμα 9.15 Το ζητούμενο διάγραμμα κλάσεων.



Βέβαια, είναι κατανοητό πως αυτό το διάγραμμα θα τεθεί υπό αίρεση αν δούμε πως δεν είναι συνεπές με το αντίστοιχο διάγραμμα ακολουθίας. Οπότε, περνάμε στο διάγραμμα ακολουθίας για την προαναφερθείσα περίπτωση χρήσης:

Σχήμα 9.16 Το ζητούμενο διάγραμμα ακολουθίας.



Αυτό το διάγραμμα ακολουθίας είναι επαρκές για τη συγκεκριμένη περίπτωση χρήσης. Επιστρέφοντας στο σχετικό διάγραμμα κλάσεων, παρατηρούμε ότι είναι συνεπές, με μόνη εξαίρεση την απουσία μιας μεθόδου στην κλάση «μαθήματα» η οποία να επιστρέφει μια λίστα μαθημάτων για παρουσίαση στον χειριστή, την οποία μπορείτε να προσθέσετε μόνοι σας.

Άσκηση 1/Κεφάλαιο 9

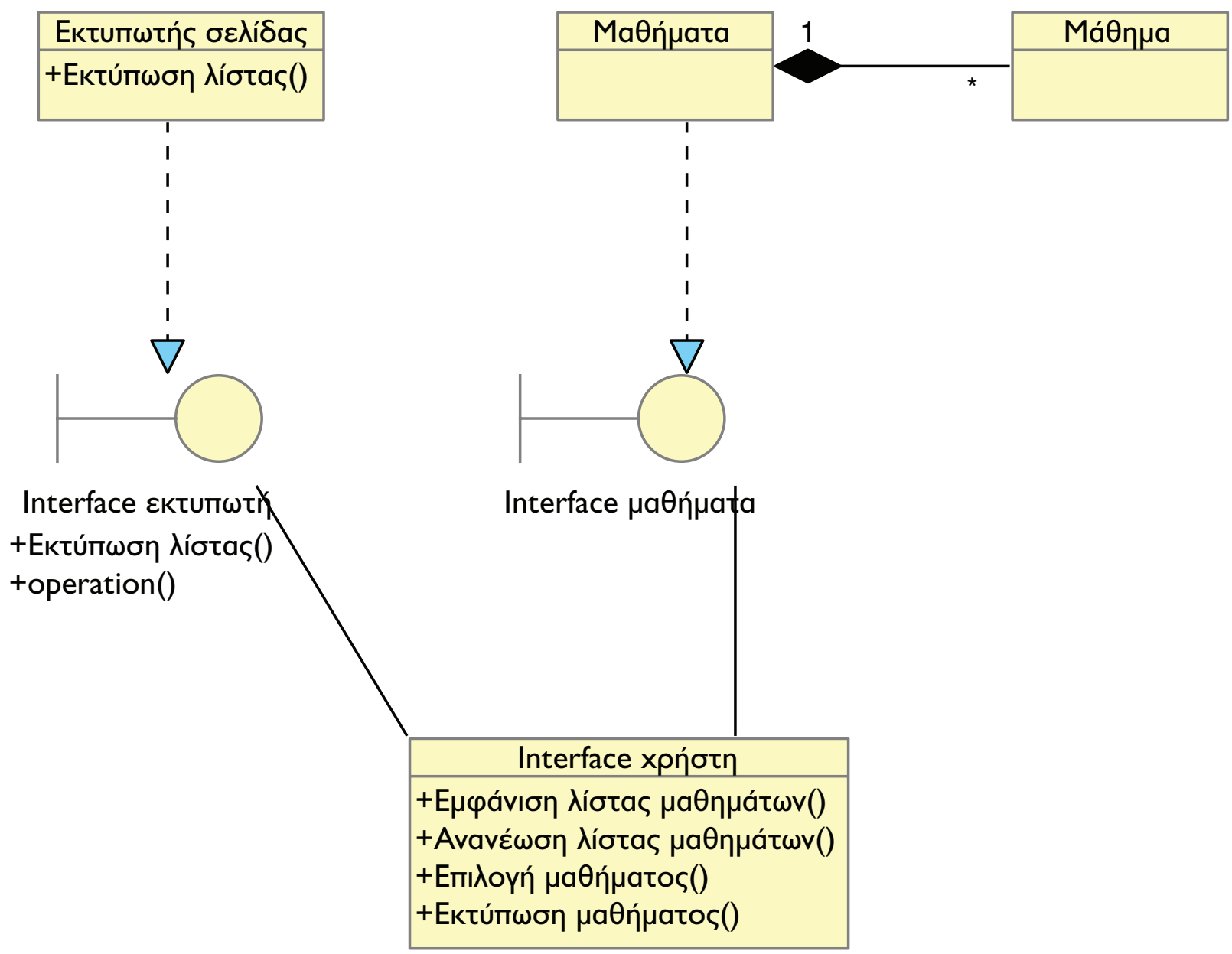
Όπως αναφέρθηκε, τόσο στο μοντέλο ανάλυσης όσο και στο μοντέλο σχεδίασης τα δομικά στοιχεία είναι οι κλάσεις. Με αυτό κατά νου, δεν έχει νόημα να ισχυριστούμε ότι «η ανάλυση τελείωσε εδώ και αυτά είναι τα αποτελέσματά της», διότι αυτό απλά θα ήταν η καταγραφή ενός ανεπίκαιρου στιγμιότυπου κάποιων κλάσεων οι οποίες στη συνέχεια θα υποστούν μεταβολές. Η αντικειμενοστρεφής λογική χαρακτηρίζεται από συνέχεια και οι φάσεις είναι διακριτές όχι γιατί τα αποτελέσματά τους χαρακτηρίζονται κάποια στιγμή τετελεσμένα, αλλά γιατί καθεμία προσθέτει στο οικοδόμημα και από κάτι, μέχρι αυτό να χαρακτηριστεί πλήρες.

Οι φάσεις, δηλαδή, δεν οριοθετούνται πρωτίστως από τα αποτελέσματά τους, αλλά από την επίδρασή τους στα ίδια, κατ' ουσίαν, συστατικά στοιχεία λογισμικού τα οποία εμπλουτίζουν, εξειδικεύουν, κάνουν πιο συγκεκριμένα κ.λπ. Η ίδια η φύση των συστατικών στοιχείων λογισμικού επί των οποίων εργαζόμαστε επιτρέπει και ενθαρρύνει τη συμπεριφορά αυτή. Αυτή είναι η σημαντικότερη διαφορά από τη δομημένη ανάλυση και σχεδίαση, και ένας από τους κύριους λόγους διάδοσης της αντικειμενοστρεφούς τεχνολογίας.

Άσκηση 2/Κεφάλαιο 9

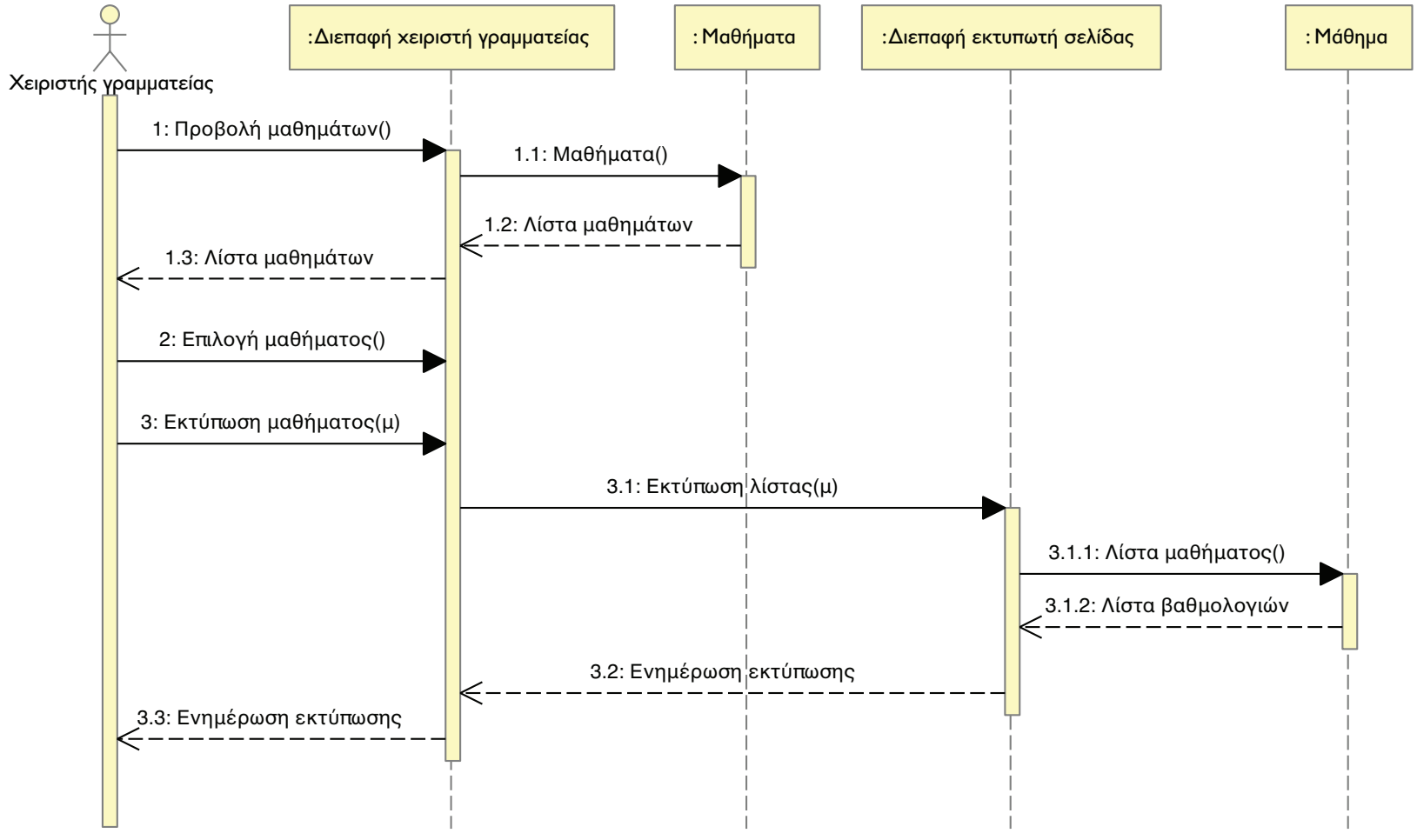
Το διάγραμμα κλάσεων φαίνεται στο σχήμα που ακολουθεί.

Σχήμα 9.17 Το ζητούμενο διάγραμμα κλάσεων.



Το ζητούμενο διάγραμμα ακολουθίας φαίνεται στο σχήμα που ακολουθεί.

Σχήμα 9.18 Το ζητούμενο διάγραμμα ακολουθίας.



Άσκηση 3/Κεφάλαιο 9

Προκειμένου να προσθέσουμε άλλο ένα κανάλι χρήσης του «Επίκουρου» μέσω κινητού τηλεφώνου, θα χρειαζόμασταν μια διαδικτυακή εφαρμογή προσαρμοσμένη σε κινητά τηλέφωνα, παρεμφερή με τον «διαδικτυακό Επίκουρο», η οποία να εκτελείται μέσα σε έναν καινούριο κόμβο, έκφανση του κινητού τηλεφώνου. Ο νέος κόμβος θα συνδέεται στον διακομιστή μέσω δικτύου δεδομένων κινητής τηλεφωνίας, για παράδειγμα, και το νέο υποσύστημα «κινητός Επίκουρος» θα επικοινωνεί όπως και ο «διαδικτυακός Επίκουρος» με την υπηρεσία « παρακολούθηση εκπαιδευτικής διαδικασίας».

Εάν η υπηρεσία «παρακολούθηση εκπαιδευτικής διαδικασίας» βρισκόταν στους επιμέρους υπολογιστές, θα είχαμε μια διαφορετική διάταξη. Σε αυτούς τους κόμβους, τα υποσυστήματα διεπαφής θα επικοινωνούσαν με την «παρακολούθηση εκπαιδευτικής διαδικασίας» και αυτό το υποσύστημα, με τη σειρά του, με τον κεντρικό διακομιστή.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Applying Use Case Driven Object Modeling with UML: An Annotated e-Commerce Example, Doug Rosenberg και Kendall Scott, Addison-Wesley, ISBN 978-0201730395

Booch G., Rumbaugh J., Jacobson I., *The Unified Modeling Language User Guide,* Addison-Wesley.

Booch, G., *Object-Oriented Analysis and Design with Applications,* Addison-Wesley.

Fowler M., Scott K., *UML Distilled,* Addison-Wesley.

Jacobson I., Booch G., Rumbaugh J., *The Unified Software Development Process,* Addison-Wesley.

Jacobson I., Christerson M., Johnson P., Overgaad G., *Object-Oriented Software Engineering: A Use Case Driven Approach,* Addison-Wesley.

Pressman, R. S., *Software Engineering-A Practitioners Approach,* McGraw-Hill.

Quadrani T., *Visual Modeling with Rational Rose and UML,* Addison-Wesley.

Rumbaugh J., Jacobson I., Booch G., *The Unified Modeling Language Reference Manual,* Addison-Wesley.

Rumbaugh J., *Object-Oriented Modeling and Design,* Prentice Hall.

Sommerville, I. *Software Engineering,* London: Addison-Wesley.