ΑΛΕΞΑΝΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΩΝ ΕΦΑΡΜΟΦΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πτυχιακή Εργασία

Ανάπτυξη εκπαιδευτικής εφαρμογής «Ταξιδιωτικό γραφείο» με τη μεθοδολογία RUP

Oneiro Travel

Ναταλία Παρατσικίδου

Επιβλέπων καθηγητής: κος Δεληγιάννης Ιγνάτιος

Θεσσαλονίκη 2011

Πρόλογος

Το βιβλίο αυτό παρουσιάζει την ανάπτυξη μίας εκπαιδευτικής εφαρμογής «Online Τουριστικό Γραφείο» με τη βοήθεια της μεθοδολογίας ICONIX. Παρουσιάζεται βήμα-βήμα η πορεία που ακολουθήθηκε για την ανάπτυξη της εκπαιδευτικής αυτής εφαρμογής ώστε να αποτελέσει ένα εγχειρίδιο για τους ενδιαφερόμενους φοιτητές που θέλουν να ασχοληθούν με την ανάπτυξη λογισμικού. Μέσα στις ενότητες του βιβλίου θα βρει κανείς και χρήσιμες πληροφορίες για τις μεθοδολογίες και τις τεχνολογίες που χρησιμοποιήθηκαν.

Η συμμετοχή στην ανάπτυξη μίας πλήρης εφαρμογής λογισμικού με γοήτευε πάντα, όσο δύσκολο και αν φαινόταν. Η ιδέα της δημιουργίας ενός έργου και η ατομική προσπάθεια για την ολοκλήρωσή του σου δίνει τη δυνατότητα να δεις, να δοκιμάσεις και να απορρίψεις διαφορετικές μεθόδους μέχρι να καταλήξεις στις κατάλληλες που θα συνδυαστούν για την επίτευξη του έργου. Το έργο «Τουριστικό Γραφείο» ήταν η τέλεια επιλογή ως θέμα για την εκπόνηση της πτυχιακής μου εργασίας, ένα σημείο ενεργοποίησης που μου έδειξε πόσο μακριά μπορώ να φτάσω, στηριζόμενη στις δικές μου δυνάμεις και γνώσεις που απέκτησα καθ’ όλη τη διάρκεια της τριτοβάθμιας εκπαίδευσής μου. Ήταν ένα προσωπικό στοίχημα και την ίδια στιγμή μία ευχαρίστηση να βλέπω τα πρόχειρα σχέδιά μου να βελτιώνονται μετά από κάθε επανάληψη τελικά να καταλήγουν σε γραμμές κώδικα που υλοποιεί το σύστημα «Online Τουριστικό Γραφείο».

Περίληψη

Η πτυχιακή εργασία «Ανάπτυξη εκπαιδευτικής εφαρμογής Τουριστικό Γραφείο με τη χρήση της μεθοδολογία RUP» έχει ως στόχο να αναπαραστήσει τη διαδικασία ανάπτυξης λογισμικού μίας εφαρμογής και να προσομοιάσει τη διαδικασία αυτή με τις ανάλογες ενέργειες που λαμβάνουν χώρα μέσα σε μία ομάδα ανάπτυξης λογισμικού.

Είναι ένα πλήρες εγχειρίδιο που παρουσιάζει μία ασαφής περιγραφή συστήματος κάτω από τις κατάλληλες ενέργειες και το συντονισμό συγκεκριμένων βημάτων μπορεί να μεταμορφωθεί σε ένα πλήρες και λειτουργικό λογισμικό που περιλαμβάνει τεχνολογίες αιχμής και είναι παράλληλα αποδοτικό από πλευράς χρόνου και κόστους.

Μέσω της εκπόνησης της συγκεκριμένης πτυχιακής εργασίας δείχνουμε ότι είναι εφικτό στηριζόμενη στις δικές μας δυνάμεις και γνώσεις να φέρουμε εις πέρας το δύσκολο έργο της ανάπτυξης ενός λογισμικού για ένα σύστημα.

Ο αναγνώστης δε χρειάζεται να είναι πλήρης γνώστης της μεθοδολογίας και των τεχνολογιών που αναπτύσσονται σε αυτό το βιβλίο καθώς γίνεται εκτενής περιγραφή και των μεθόδων αλλά και των τεχνολογιών αυτών. Πρέπει όμως να είναι γνώστης του γενικού αντικειμένου της Πληροφορικής και να γνωρίζει της βασικές αρχές της ανάλυσης και του προγραμματισμού.

Abstract

The dissertation "Development of an Educational application Travel Agent using the methodology RUP» aims to simulate the process of developing a software application and to simulate this process with similar processes which take place in a software development team.

It is a complete guide presenting how an ambiguous system description under the appropriate action and the coordination of specific steps can be transformed into fully functional software that includes advanced technologies and is also cost and time efficient.

The development of this dissertation shows that it is possible to complete with success the difficult task of implementing a software development system based on our own strengths and knowledge.

The reader might not be fully aware of the methodology and technologies developed in this book, while an extensive description of the methods and technologies are presented. But he/she must be aware of the general subject of IT and have appropriate knowledge on the basic principles of analysis and programming.

Ευχαριστίες

Πρωτίστως θα ήθελα να ευχαριστήσω όλους του καθηγητές του Τμήματος Πληροφορικής, του Αλεξάνδρειου Τεχνολογικού Εκπαιδευτικού Ιδρύματος Θεσσαλονίκης που προσπάθησαν αν μεταλαμπαδεύσουν τις γνώσεις τους σε εμάς τους φοιτητές και μου δίδαξαν να αγαπώ τον τομέα της Πληροφορικής και να συνεχίσω να δουλεύω πάνω σε αυτόν. Κυρίως θα ήθελα να ευχαριστήσω τον επιβλέπων καθηγητή της πτυχιακής μου, τον κύριο Δεληγιάννη Ιγνάτιο, που μου έδωσε την ευκαιρία να εργαστώ στον τομέα ανάπτυξης λογισμικού μέσω της συγκεκριμένης πτυχιακής εργασίας και τον κύριο Διαμαντάρα που υπήρξε πολύτιμος δάσκαλος στην πορεία μου στο τμήμα και που με την πολύτιμη βοήθειά του θα μπορέσω να συνεχίσω τις σπουδές στην Πληροφορική.

Τέλος θα ήθελα να ευχαριστήσω την οικογένειά μου και τους κοντινούς μου φίλους που στηρίζουν κάθε μου επιλογή και είναι πάντοτε αρωγή κάθε μου προσπάθειας.

Ένα μεγάλο ευχαριστώ σε όλους.

Ευρετήριο Περιεχομένων

Ευρετήριο Πινάκων και Σχημάτων

Εισαγωγή

Η συγκεκριμένη πτυχιακή εργασία ασχολείται με την ανάπτυξη λογισμικού, χρησιμοποιώντας μεθόδους, τεχνικές και τεχνολογίες για την επίτευξη του στόχου. Η ανάπτυξη λογισμικού είναι μία δύσκολη και αρκετές φορές επίπονη διαδικασία που επιβάλλει το συντονισμό ενεργειών και τη χρήση μεθοδολογιών ανάπτυξης για την επίτευξη του επιθυμητού αποτελέσματος. Αυτές ακριβώς τις ενέργειες και τις αποφάσεις που πάρθηκαν, κατά τη διάρκεια της εκπόνησης του έργου, παρουσιάζουμε στη συγκεκριμένη πτυχιακή εργασία.

Το «Online Τουριστικό Γραφείο» είναι ένα αντιπροσωπευτικό παράδειγμα ανάπτυξης λογισμικού και περικλείει όλες τις φάσεις και τα προβλήματα παρόμοιων έργων. Είναι ένα παράδειγμα που εύκολα χρησιμοποιείται σαν εκπαιδευτική εφαρμογή στα πλαίσια της εκμάθησης ανάπτυξης λογισμικού, αποτελεί όμως και από μόνο του μία δυνατή και ισχυρή εφαρμογή του σύγχρονου κόσμου των λογισμικών, με πολλές παρόμοιες εφαρμογές να κατακλύζουν το διαδικτυακό χώρο, όπου οι απαιτήσεις είναι αυξημένες.

Η πτυχιακή αυτή παρουσιάζει μία προσπάθεια ανάπτυξης λογισμικού από έναν φοιτητή, με τη βοήθεια των μεθόδων και των γνώσεων των τεχνολογιών που αποκτήθηκαν κατά τη τετραετή φοίτησή του στο Τμήμα Πληροφορικής, του Α.Τ.Ε.Ι.Θ. Αναπτύχθηκαν και σχολιάστηκαν μεθοδολογίες που διδάχθηκαν στο τμήμα όπως η RUP και η ICONIX μεθοδολογίες, HTML και CSS ανάπτυξη κώδικα, αλλά συνάμα προστέθηκαν και τεκμηριώθηκαν νέες γνώσεις πάνω στην ανάπτυξη λογισμικού, όπως σελίδες JSP και servlets. Απώτερος στόχος είναι να μετρήσουμε τις δυνάμεις μας σε ένα έργο, παρόμοιο με αυτά που υπάρχουν και αναπτύσσονται στην σημερινή αγορά εργασίας και να κερδίσουμε γνώσεις και εμπειρίες πάνω σε ένα μεγάλο τμήμα της Πληροφορικής, την ανάπτυξη λογισμικού.

Τα κεφάλαια που ακολουθούν περιγράφουν με τη σειρά την μεθοδολογία RUP, τα βασικά στοιχεία και τα βήματα που ορίζει. Τη μεθοδολογία ICONIX, τα βασικά της μέρη και τα βήματα που ορίζει, σύγκριση μεταξύ των δύο παραπάνω τεχνολογιών και κυριάρχηση της ICONIX για την υλοποίηση της εκπαιδευτικής μας

εφαρμογής. Αναλυτικά βήματα υλοποίησης της εφαρμογής με τη βοήθεια της μεθόδου ICONIX από την ανάλυση των περιπτώσεων χρήσης μέχρι το σχεδιασμό και την υλοποίηση του κώδικα. Η ανάπτυξη της μεθοδολογίας ICONIX γίνεται με τη βοήθεια προσδιορισμού συγκεκριμένων οδηγιών, όπου η χρήση τους μας παρέχει σωστή, ολοκληρωμένη και εμπεριστατωμένη υλοποίηση του έργου. Όλα τα παραπάνω βήματα παρουσιάζονται με αναλυτικές οδηγίες και περιγραφικές εικόνες, πολλές από τις οποίες είναι εικόνες και διαγράμματα του ίδιου του συστήματος που αναπτύχθηκε.

Τέλος παρέχονται οδηγίες χρήσης του λογισμικού που συνοδεύει την πτυχιακή εργασία. Στο λογισμικό, που βρίσκεται σε cd στο τέλος του βιβλίου, παρέχεται πλήρης ανάπτυξη των διαγραμμάτων του έργου μας, καθώς και τα κατάλληλα αρχεία που περιέχουν την υλοποίηση της εφαρμογής μας σε JSP και servlets.

Κεφάλαιο 1: Εισαγωγή στην πλατφόρμα J2EE

Το κεφάλαιο αυτό παρέχει μία εισαγωγή στην πλατφόρμα Java 2 Platform, Enterprise Edition (J2EE πλατφόρμα) στοιχεία της οποίας χρησιμοποιήθηκαν για την ανάπτυξη αυτής της εφαρμογής. Η πλατφόρμα J2EE περιλαμβάνει αρκετά στοιχεία και τεχνολογίες και εκτείνεται σε ένα ευρύ φάσμα επιχειρησιακών συστημάτων, που δεν αφορούν τους σκοπούς αυτού του βιβλίου. Εμείς θα θέσουμε τα βασικά χαρακτηριστικά και θα αναλύσουμε μόνο τα στοιχεία που χρησιμοποιήθηκαν για την ανάπτυξη του συστήματός μας, το «Online Τουριστικό Γραφείο».

1.1 Αρχιτεκτονική πολλαπλών βαθμίδων και η πλατφόρμα J2EE

Είναι γνωστό εδώ και αρκετό καιρό ότι η καλύτερη προσέγγιση για την ανάπτυξη των συστημάτων είναι η κατανομή των ευθυνών τους σε διάφορες βαθμίδες[1], με αποτέλεσμα κοινές αρχιτεκτονικές μορφές γνωστές ως αρχιτεκτονικές πολλαπλών βαθμίδων (multitier architectures). Ένα παράδειγμα μιας αρχιτεκτονικής πολλαπλών βαθμίδων είναι η 3^{ων} βαθμίδων δομή που φαίνεται στο Σχήμα 1.1. Η βαθμίδα της παρουσίασης (presentation tier) είναι υπεύθυνη για τη διαχείριση των αλληλεπιδράσεων με τον τελικό χρήστη. Η επιχειρησιακή βαθμίδα (business tier) είναι υπεύθυνη για την εκτέλεση οποιασδήποτε επιχειρηματικής επεξεργασίας. Η βαθμίδα της ενσωμάτωσης είναι υπεύθυνη για την πρόσβαση στη διαχείριση των πόρων, συμπεριλαμβανομένων των βάσεων δεδομένων και των εξωτερικών συστημάτων. Οι εν λόγω διαίρεση επιτρέπει στο περιεχόμενο της κάθε βαθμίδας να αναπτυχθεί και να αλλάξει ανεξάρτητα.

Σχήμα 1.1 Αρχιτεκτονική 3ων βαθμίδων

[1] Μία βαθμίδα είναι ένα αρχιτεκτονικό στρώμα που έχει συγκεκριμένη ευθύνη.

Από ιστορική άποψη, η τεχνολογία των βάσεων δεδομένων της βαθμίδας ενσωμάτωσης, ήταν η πρώτη που ωρίμασε, με αποτέλεσμα τις ισχυρές σχεσιακές βάσεις δεδομένων[2]. Οι τεχνολογίες στη βαθμίδα της παρουσίασης και την επιχειρησιακή βαθμίδα ωρίμασαν αργότερα, με αποτέλεσμα τα πλαίσια διεπαφής χρήστη και τις οθόνες συναλλαγής και επεξεργασίας (transaction-processing monitors), αντίστοιχα.

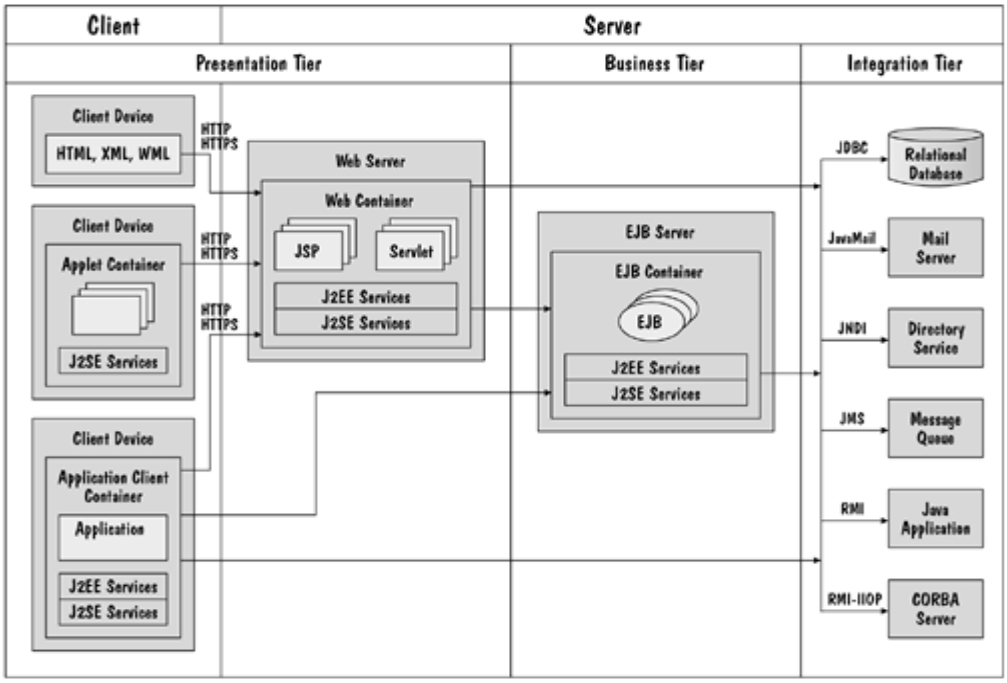
Έτσι, συνοπτικά, η J2EE πλατφόρμα θεωρείται καλύτερα ως ένα σύνολο τεχνολογιών για την ανάπτυξη και την εγκατάσταση εταιρικών συστημάτων πολλαπλών βαθμίδων. Ως εκ τούτου, περιέχει ένα αναμενόμενο σύνολο υπηρεσιών (τόσο αναμενόμενο, στην πραγματικότητα, που η πρωτοβουλία της Microsoft NET, παρέχει ένα σχεδόν πανομοιότυπο σύνολο). Έχουμε εισαγάγει τις τεχνολογίες αυτές στην επόμενη ενότητα.

Η πλατφόρμα J2EE έχει επίσης το επιθυμητό χαρακτηριστικό ότι είναι μια ανοικτή προδιαγραφή. Υπάρχουν πολλές εμπορικές και ανοικτού κώδικα υλοποιήσεις της προδιαγραφής. Μια δωρεάν εφαρμογή της J2EE πλατφόρμας είναι επίσης διαθέσιμη.

1.2 Επισκόπηση Πλατφόρμας J2EE

Ας ξεκινήσουμε θέτοντας τις τεχνολογίες J2EE μαζί ώστε να δείξουμε το πλαίσιο μέσα στο οποίο λειτουργούν, καθώς και τις μεταξύ τους σχέσεις. Το Σχήμα 1.2 τοποθετεί κάθε μία από τις τεχνολογίες όσον αφορά τις βαθμίδες που συζητήσαμε νωρίτερα. Δείχνει επίσης τη φυσική θέση του καθενός από τα στοιχεία από την άποψη του πελάτη και του διακομιστή. Οι λεπτομέρειες που περιέχονται στο Σχήμα είναι 1.2 αναφέρονται αργότερα σε αυτό το κεφάλαιο. Αν και δεν φαίνεται στο Σχήμα 1.2, αξίζει επίσης να αναφερθεί ότι όλα τα κοντέινερ χρησιμοποιούν ρητά μια Java Virtual Machine (JVM) κατά την εκτέλεση οποιουσδήποτε μεταγλωττισμένου κώδικα σε Java.

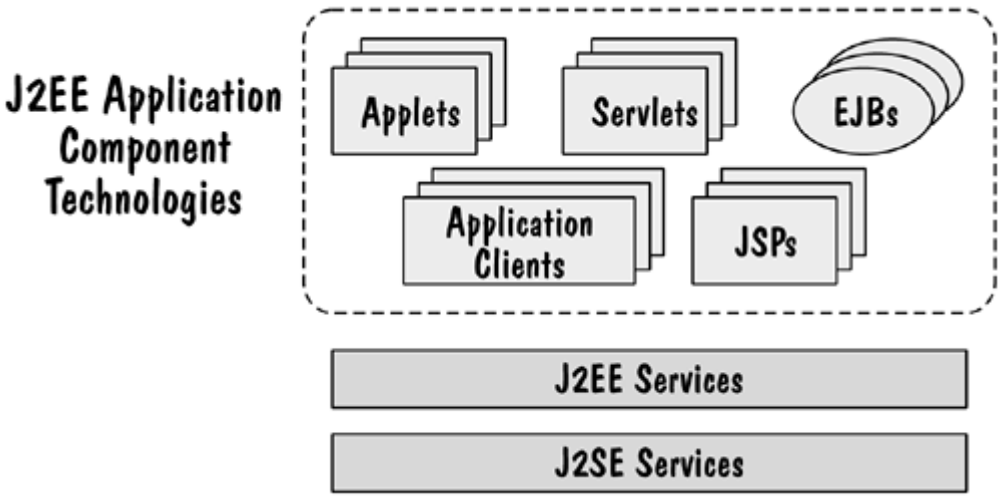
[2] Η ίδρυση της τεχνολογίας των σχεσιακών βάσεων δεδομένων δημιουργήθηκε πάνω από δύο δεκαετίες πριν.



Σχήμα 1.2 Η πλατφόρμα J2EE και οι τεχνολογίες

1.2.1 Επισκόπηση Τεχνολογιών J2EE

Μία περίληψη των διάφορων τεχνολογιών J2EE φαίνεται στο Σχήμα 1.3.



Σχήμα 1.3 Περίληψη των τεχνολογιών J2EE

1.2.1.1 J2EE Τεχνολογίες Στοιχείων Εφαρμογών

Οι τεχνολογίες στοιχείων εφαρμογών είναι αυτές που χρησιμοποιούμε για να οικοδομήσουμε τα στοιχεία της λύσης. Οι τεχνολογίες αυτές περιλαμβάνουν τα Applets, τους Application clients, τα Java servlets (servlets), τα JavaServer Pages (JSPs), τα Enterprise JavaBeans (EJBs). Ενδεικτικά αναλύουμε τις τρεις τελευταίες τεχνολογίες οι οποίες χρησιμοποιήθηκαν για τη δημιουργία του συστήματός μας.

- **Java servlets ("servlets").** Ένα servlet ορίζει πώς ένα αίτημα (request) από τον πελάτη θα επεξεργαστεί και πώς δημιουργείται μία απάντηση (response).
- **JavaServer Pages (JSPs).** Ένα JSP είναι ένα έγγραφο κειμένου που, όπως ένα servlet, περιγράφει πώς μια αίτηση υποβάλλεται σε επεξεργασία και μια απάντηση δημιουργείται. Τα JSPs παρέχουν μια εναλλακτική λύση για servlets όταν απαιτείται η δημιουργία δηλώσεων σε μία γλώσσα σήμανσης (όπως HTML).
- **Enterprise JavaBeans (EJBs).** Ένα EJB είναι υπεύθυνο για την εφαρμογή μιας πτυχής της επιχειρηματικής λογικής μιας εφαρμογής J2EE.

1.2.1.2 Υπηρεσίες J2EE

Οι υπηρεσίες J2EE, όπως δείχνει και το όνομά τους, είναι υπηρεσίες που διατίθενται από την πλατφόρμα J2EE. Οι υπηρεσίες αυτές επιγραμματικά είναι οι εξής: Βιβλιοθήκη Java (Java Api) για ανάλυση (parsing) XML (JAXP), Java Message Services (JMS), Java Authentication και Service Authorization (JAA), Java Transaction API (JTA), J2EE Connector Architecture (JCA), Java DataBase Connectivity (JDBC) και Βιβλιοθήκη JavaMail (JavaMail API). Οι δύο τελευταίες παρουσιάζονται παρακάτω καθώς χρησιμοποιήθηκαν εκτενώς για την ανάπτυξη της εφαρμογής μας.

- **Java DataBase Connectivity (JDBC).** Η JDBC παρέχει πρόσβαση μέσω προγραμματισμού σε μια σχεσιακή βάση δεδομένων.

- **Βιβλιοθήκη JavaMail (JavaMail API).** Η βιβλιοθήκη JavaMail επιτρέπει στα στοιχεία της εφαρμογής να στέλνουν mail χρησιμοποιώντας ένα τυποποιημένο interface. Τυπικές εφαρμογές της διεπαφής JavaMail API σε μια σειρά πρωτοκόλλων και προδιαγραφών, όπως το Simple Mail Transfer Protocol (SMTP), Multipurpose Internet Mail Extensions (MIME) και το Post Office Protocol 3 (POP3).

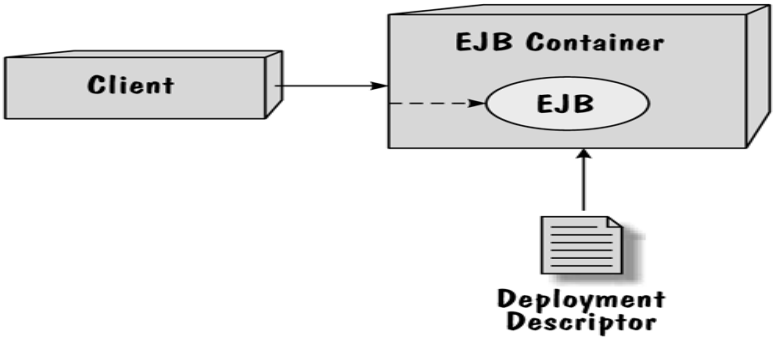
1.2.1.3 Υπηρεσίες J2SE

Η πλατφόρμα J2EE εξαρτάται από τις υπηρεσίες που παρέχονται από την Java 2 Platform, Standard Edition (J2SE). Οι J2SE υπηρεσίες περιλαμβάνουν υποστήριξη για τις συλλογές, τη διεθνοποίηση (υποστήριξη για πολλές ανθρώπινες γλώσσες), την είσοδο / έξοδο, τα Java Archive (JAR) αρχεία, τις διεπαφές χρηστών, τα μαθηματικά, τη δικτύωση, τη κωδικοποίηση αντικειμένων, την απομακρυσμένη μέθοδο επίκλησης (Remote Method Invocation, RMI), την ασφάλεια και τον ήχο. Οι J2SE τεχνολογίες είναι επιγραμματικά οι ακόλουθες: Βιβλιοθήκη Hypertext Transfer Protocol (HTTP API), Βιβλιοθήκη HTTPS (HTTPS API), Remote Method Invocation over Internet Inter-Orb Protocol (RMI-IIOP), Java Naming and Directory Interface (JNDI).

1.2.2 Κοντέινερ (Containers)

Η έννοια του κοντέινερ έχει κεντρική σημασία στην πλατφόρμα J2EE. Ένα κοντέινερ παρέχει υποστήριξη κατά το χρόνο εκτέλεσης (runtime) για τα στοιχεία της εφαρμογής (όπως JSPs, servlets, ή EJBs) που εκτελούνται μέσα σε αυτήν. Για παράδειγμα, ένα κοντέινερ EJB παρέχει τη διαχείριση του κύκλου ζωής του στοιχείου (με τη δημιουργία και την απομάκρυνση των στοιχείων της αίτησης, όπως απαιτείται), τη διαχείριση συναλλαγής, την ασφάλεια και τη συνεχής υποστήριξη των EJBs που εκτελούνται εντός αυτού.

Το γεγονός ότι ένα στοιχείο εκτελείται μέσα σε ένα κοντέινερ χαρακτηρίζεται από διαφάνεια στους πελάτες της. Για παράδειγμα, στο Σχήμα 1.4 υπάρχει ένας πελάτης που αλληλεπιδρά με ένα EJB. Το αίτημα του πελάτη χειρίζεται από το κοντέινερ EJB που περιέχει το EJB, και όχι άμεσα από το ίδιο το EJB.



Σχήμα 1.4 Κοντέινερ EJB

Υπάρχει μία σύμβαση διπλής κατεύθυνσης μεταξύ ενός κοντέινερ και ενός στοιχείου εφαρμογής. Από την πλευρά του κοντέινερ, το στοιχείο της εφαρμογής πρέπει να ανταποκρίνονται σε ορισμένες διασυνδέσεις, έτσι ώστε το κοντέινερ να μπορεί να διαχειριστεί το στοιχείο κατάλληλα. Για παράδειγμα, ένα EJB πρέπει να παρέχει ενέργειες για τη στήριξη της απομάκρυνσής του από τη μνήμη (γνωστή ως «παθητικοποίηση» (“passivation”)). Από την πλευρά του στοιχείου της εφαρμογής, το κοντέινερ πρέπει να παρέχει ορισμένες υπηρεσίες διαθέσιμες στο στοιχείο. Για παράδειγμα, ένα κοντέινερ EJB πρέπει να παρέχει τη βιβλιοθήκη Java DataBase Connectivity (JDBC).

Όπως φαίνεται και στο Σχήμα 1.2, υπάρχουν τέσσερις τύποι κοντέινερ.

- Κοντέινερ μίας μικροεφαρμογής (applet), το οποίο παρέχει τις υπηρεσίες που απαιτούνται από τις μικροεφαρμογές της Java.
- Κοντέινερ ενός application client, το οποίο παρέχει υπηρεσίες που απαιτούνται από τους application client της Java.
- Κοντέινερ διαδικτύου (Web), το οποίο παρέχει υπηρεσίες σε JSPs και servlets.
- Κοντέινερ EJB, το οποίο παρέχει υπηρεσίες σε EJBs.

1.2.3 Βαθμίδα Παρουσίασης (Presentation Tier)

Η βαθμίδα της παρουσίασης περιλαμβάνει στοιχεία που βρίσκονται τόσο στον πελάτη όσο και στον διακομιστή.

Τα στοιχεία προγράμματος-πελάτη είναι υπεύθυνα για την απόδοση της διεπαφής του χρήστη και για το χειρισμό των αλληλεπιδράσεων των χρηστών. Στο Σχήμα 1.2, βλέπουμε τρεις πελάτες (clients), όπου καθένας εκτελείται στη δική του μηχανή. Ο πρώτος πελάτης επεξεργάζεται μια γλώσσα σήμανσης. Παραδείγματα τέτοιων πελατών περιλαμβάνουν ένα πρόγραμμα περιήγησης Web που επεξεργάζεται HTML, ένα ενημερωμένο για XML μηχάνημα που επεξεργάζεται XML, και μία συσκευή Wireless Access Protocol (WAP) όπως ένα κινητό τηλέφωνο, που χειρίζεται Wireless Markup Language (WML). Ο δεύτερος πελάτης στεγάζει ένα applet κοντέινερ που υποστηρίζει την εκτέλεση μικροεφαρμογών. Μια μικροεφαρμογή είναι ένα πρόγραμμα σε Java που περιέχει κατά κανόνα κάποια μορφή παροχής υψηλής απόδοσης διεπαφής χρήστη. Ο τρίτος πελάτης στεγάζει ένα application client κοντέινερ που υποστηρίζει την εκτέλεση ενός J2EE application client. Ένα J2EE application client είναι μια αυτόνομη εφαρμογή της Java η οποία τυπικά παρέχει πρόσβαση σε στοιχεία στην επιχειρησιακή βαθμίδα και τη βαθμίδα ένταξης. Για παράδειγμα, ένα application client μπορεί να χρησιμοποιηθεί ώστε να παρέχει μία διοικητική διεπαφή (administrative interface) σε εφαρμογή J2EE.

Τα στοιχεία στην πλευρά του διακομιστή είναι υπεύθυνα για την επεξεργασία αιτήσεων (requests) στην πλευρά του πελάτη και την παροχή κατάλληλων απαντήσεων (responses). Η απάντηση συνήθως παραδίδεται στον πελάτη με τη μορφή μιας γλώσσας σήμανσης, όπως HTML, XML ή WML. Η απάντηση είναι συχνά εξαρτώμενη από τα δεδομένα που κατέχει ένα EJB ή ένα πληροφοριακό σύστημα επιχείρησης (Enterprise Information System, EIS), όπως ένα κεντρικό σύστημα επεξεργασίας συναλλαγών ή μια βάση δεδομένων που κληρονομείτε. Ως εκ τούτου, τα στοιχεία εφαρμογής της βαθμίδας παρουσίαση στον διακομιστή (τα JSPs και servlets) αλληλεπιδρούν με τα στοιχεία στην επιχειρησιακή βαθμίδα ή άμεσα με την βαθμίδα ένταξης. Τα στοιχεία αυτά μπορούν επίσης να είναι υπεύθυνα για μέρος της διαχείρισης συνεδριών χρήστη, της επικύρωσης δεδομένων και των εφαρμογών λογικού ελέγχου.

1.2.4 Επιχειρησιακή Βαθμίδα (Business Tier)

Η επιχειρησιακή βαθμίδα είναι υπεύθυνη για την επιχειρηματική λογική της εφαρμογής. Στην πιο συνηθισμένη περίπτωση, τα στοιχεία της επιχειρησιακής βαθμίδας (EJBs) παρέχουν υπηρεσίες επιχειρηματικής λογικής στα στοιχεία εφαρμογής της βαθμίδας παρουσίασης στην πλευρά του διακομιστή. Ωστόσο, μπορούν επίσης να παρέχουν υπηρεσίες σε αυτόνομες εφαρμογές πελάτη Java. Τα EJBs και EJB κοντέινερ έχουν σχεδιαστεί για να απλοποιήσουν την επικοινωνία μεταξύ της βαθμίδας παρουσίαση και της βαθμίδας ένταξη.

1.2.5 Βαθμίδα Ένταξη (Integration Tier)

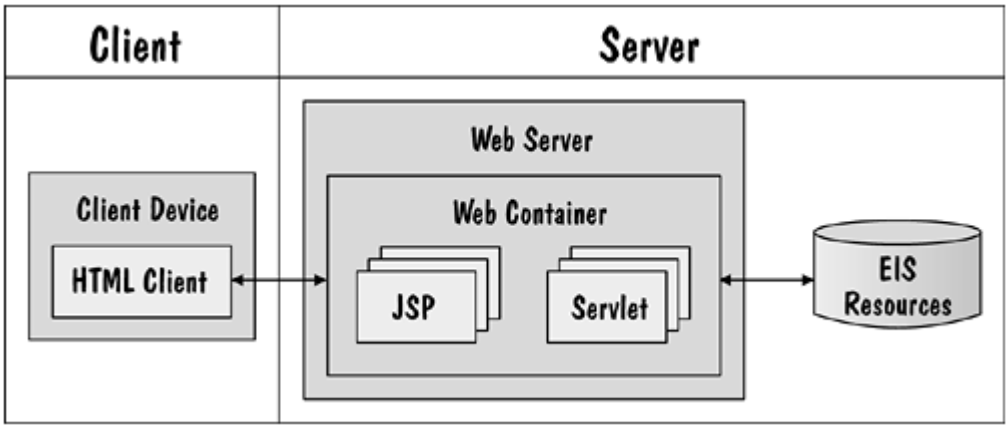
Η βαθμίδα ένταξη είναι υπεύθυνη για την παροχή πρόσβασης σε EIS πόρους. Το Σχήμα 1.2 προσδιορίζει συγκεκριμένους τύπους πόρων EIS, συμπεριλαμβανομένης μιας σχεσιακής βάσης δεδομένων, ενός διακομιστή ηλεκτρονικού ταχυδρομείου, μίας υπηρεσίας καταλόγου, μίας ουράς μηνυμάτων, μίας εφαρμογής σε Java και ενός διακομιστή CORBA. Έχουμε επισημάνει κάθε σύνδεση σε ένα πόρο EIS με την τεχνολογία που χρησιμοποιείται για την πρόσβαση αυτού του πόρου. Για παράδειγμα, το JDBC χρησιμοποιείται για την πρόσβαση σε μια σχεσιακή βάση δεδομένων.

1.2.6 Διαμορφώσεις εγκατάστασης J2EE

Μία διαμόρφωση εγκατάστασης είναι μια χαρτογράφηση των λειτουργιών της εφαρμογής στα στοιχεία της και στη συνέχεια στα κοντέινερ J2EE και στις υπηρεσίες. Με άλλα λόγια, είναι ένας τρόπος για την οργάνωση και τη διανομή της λειτουργικότητας της εφαρμογή μεταξύ των βαθμίδων, των κοντέινερ και των στοιχείων. Αν και υπάρχουν διάφορες διαμορφώσεις εγκατάστασης, υπάρχουν μερικές κοινές δομές. Εμείς εν συντομία θα εξετάσουμε τη διαμόρφωση εγκατάστασης των web εφαρμογών και θα παραλείψουμε τις υπόλοιπες καθώς ο σκοπός αυτού του βιβλίου δεν είναι να παρουσιάσει εκτενώς τις διαμορφώσεις εγκατάστασης της πλατφόρμας J2EE αλλά τα βασικά της στοιχεία και τα στοιχεία που χρησιμοποιήθηκαν για την ανάπτυξη της διαδικτυακής εφαρμογής μας.

1.2.6.1 Web-κεντρική διαμόρφωση εγκατάστασης

Στη Web-κεντρική διαμόρφωση εγκατάστασης, που φαίνεται στο Σχήμα 1.5, ένα Web κοντέινερ κάθεται μεταξύ του πελάτη και των πόρων EIS, και δεν υπάρχει κοντέινερ EJB. Τόσο η λογική της παρουσίασης και της επιχειρηματικής λογικής αντιμετωπίζονται από στοιχεία του Web κοντέινερ. Μια Web-κεντρική διαμόρφωση εγκατάστασης συνήθως καταλήγει σε μια έμφαση στην εμφάνιση και την αίσθηση της εφαρμογής (look and feel), με λιγότερη έμφαση στη στήριξη της επιχειρηματικής λογικής.



Σχήμα 1.5 Web-κεντρική διαμόρφωση εγκατάστασης

Αυτή η διαμόρφωση παρέχει μια σειρά από οφέλη. Οι πελάτες δεν επηρεάζονται από αλλαγές στους πόρους EIS, δεδομένου ότι οι πελάτες δεν έχουν άμεση πρόσβαση σε αυτούς τους πόρους (εκτός και αν, για παράδειγμα, η έκταση των αλλαγών απαιτεί πρόσθετες πληροφορίες που πρέπει να παρέχονται από τον πελάτη). Είναι επίσης ευκολότερο να αναδιατάξει το σύνολο της εφαρμογής, δεδομένου ότι όλη η λογική της εφαρμογής βρίσκεται στο διακομιστή.

Ωστόσο, αν και η χρήση των EJBs μερικές φορές θεωρείται ότι είναι υπερβολική, η παράλειψη των EJBs έχει ως αποτέλεσμα σε ορισμένα μειονεκτήματα. Συγκεκριμένα, αυτή η ρύθμιση δεν ενθαρρύνει μια σαφή κατανομή των αρμοδιοτήτων μεταξύ της λογικής της παρουσίασης και της επιχειρηματικής λογικής, με αποτέλεσμα στενά συνδεδεμένα στοιχεία τα οποία εμποδίζουν την εξέλιξη της εφαρμογής και τη διατήρησή της. Επίσης, από την άποψη της κλιμάκωσης, είναι ευθύνη του προγραμματιστή να εξασφαλίσει την

αποτελεσματική χρήση των περιορισμένων πόρων, όπως οι συνδέσεις με βάσεις δεδομένων.

Για να συνοψίσουμε, υπάρχει μια σειρά από ρυθμίσεις εγκατάστασης, η καθεμία με τα πλεονεκτήματα και τα μειονεκτήματά της. Ένας από τους στόχους της J2EE πλατφόρμας είναι να είναι αρκετά ευέλικτη ώστε να υποστηρίζει τις οποιεσδήποτε ρυθμίσεις που ταιριάζουν καλύτερα σε έναν οργανισμό, καθώς αντιμετωπίζει τα προβλήματα της εγκατάστασης.

1.3 Συνιστάμενες Τεχνολογίες J2EE

Ας ρίξουμε τώρα μία πιο προσεκτική ματιά στις συνιστάμενες τεχνολογίες εφαρμογών J2EE που χρησιμοποιούνται διεξοδικά στο σύστημά μας.

1.3.1 Java servlets

Ένα servlet είναι μια κλάση Java που χρησιμοποιείται για την εφαρμογή της λογικής της παρουσίασης στο διακομιστή. Ένα servlet καθορίζει τον τρόπο με τον οποίο η αίτηση (request) του πελάτη επεξεργάζεται και τον τρόπο με τον οποίο μια απάντηση δημιουργείται. Τα Servlets συχνά είναι απευθείας προσβάσιμα από ένα μηχάνημα πελάτη, όπως ένα πρόγραμμα περιήγησης Web, είτε χρησιμοποιώντας ένα URL ή μέσω της χρήσης μίας φόρμας HTML, όπως φαίνεται στο παρακάτω απόσπασμα HTML. Όταν η φόρμα που εκπροσωπείται από τον κώδικα HTML, υποβάλλεται στο διακομιστή Web για την επεξεργασία, ο διακομιστής Web εντοπίζει τα servlets, με βάση το όνομα που καθορίζεται στο χαρακτηριστικό “action” της φόρμας, προσδιορίζει την κατάλληλη μέθοδο servlet, με βάση το χαρακτηριστικό “method” της φόρμας, κατασκευάζει τη σχετική αίτηση και επικαλείται τη μέθοδο servlet, περνώντας την αίτηση ως όρισμα.

```
<html>
  <body>
    <form method=post action="/reservation/main">
      ....
    </form>
  </body>
</html>
```

Το κομμάτι κώδικα που βρίσκεται παρακάτω είναι από τον πηγαίο κώδικα του servlet που χρησιμοποιήθηκε στο παραπάνω παράδειγμα. Ο κώδικας αυτός δείχνει πτυχές της υλοποίησης της μεθόδου doPost που επικαλείται ο διακομιστής Web όταν η φόρμα HTML έχει υποβληθεί. Αυτή η μέθοδος παίρνει δύο παραμέτρους. Η πρώτη παράμετρος είναι ένα HttpServletRequest, το οποίο παρέχει το περιεχόμενο της αίτησης. Η δεύτερη παράμετρος είναι ένα HttpServletResponse, το οποίο χρησιμοποιείται για να επιστρέψει την απάντηση. Κατά την εφαρμογή της απαιτούμενης λογικής της παρουσίασης, ένα servlet αλληλεπιδρά συνήθως με άλλα servlets, EJBs και JSPs.

```
import java.io.*;

import javax.servlet.*;

import javax.servlet.http.*;


public class PresentationRequestController extends HttpServlet

{

  public void doPost(HttpServletRequest req, HttpServletResponse
resp)
           throws ServletException, IOException

  {
    ....

  }

}
```

Συχνά είναι απαραίτητο για ένα servlet να παράγει έξοδο που μπορεί να ερμηνευθεί στο μηχάνημα του πελάτη, όπως η HTML (εάν αυτή είναι η αναμενόμενη παραγωγή). Αυτό απαιτεί τη σύνταξη της doPost μεθόδου, ώστε να τοποθετεί την HTML στο αντικείμενο HttpServletResponse. Ωστόσο, όταν πρόκειται για την εμφάνιση μίας ιστοσελίδας, η πλατφόρμα J2EE προσφέρει μια εναλλακτική τεχνολογία, τα JavaServer Pages (JSP), για να ερμηνευθεί η απαιτούμενη η γλώσσα σήμανσης.

1.3.2 JavaServer Pages

Ένα JSP είναι ένα έγγραφο κειμένου που, όπως ένα servlet, περιγράφει πώς μια αίτηση επεξεργάζεται και μια απάντηση δημιουργείται. Ένα JSP συχνά

προσπελαύνεται απευθείας από το μηχάνημα του πελάτη, όπως ένα πρόγραμμα περιήγησης Web, χρησιμοποιώντας μια διεύθυνση URL. Για παράδειγμα, προσπελάζοντας το URL http://www.OneiroTravel.gr/PassengerInfo.js έχει ως αποτέλεσμα στο διακομιστή Web να εκτελείται το αρχείο PassengerInfo.jsp και να επιστρέφεται η απάντηση που παράγεται από αυτό το JSP. Ένας τρόπος για να καταλάβουμε τα JSPs είναι ότι παρέχουν μια εναλλακτική λύση για servlets για τη δημιουργία δηλώσεων σε μια γλώσσα σήμανσης. Εσωτερικά, ο διακομιστής Web αυτόματα μεταγλωττίζει τα JSPs σε servlets πριν εκτελεσθούν. Αυτό θέτει το ερώτημα του πότε να χρησιμοποιούνται τα servlets και πότε να χρησιμοποιούνται τα JSPs, αφού μπορούν να παρέχουν παρόμοιες λειτουργικές δυνατότητες. Αυτή είναι μία απόφαση σχεδιασμού και εξαρτάται από την εκάστοτε εφαρμογή.

Το περιεχόμενο ενός απλού αρχείου JSP που επιστρέφει την ημερομηνία που σχετίζεται με το μηχάνημα στο οποίο ο διακομιστής Web εκτελείται, φαίνεται παρακάτω. Μπορούμε να δούμε ότι η JSP περιέχει δύο τύπους δηλώσεων. Ο πρώτος τύπος δήλωσης είναι μια γλώσσα σήμανσης που επιστρέφεται στην απάντηση. Στο παράδειγμα, η γλώσσα σήμανσης είναι η HTML. Ο δεύτερος τύπος δήλωσης είναι μια γλώσσα εντολών που υποστηρίζει τη δημιουργία δυναμικού περιεχομένου, όταν το αρχείο JSP εκτελείται. Στο παράδειγμα, το κείμενο newDate().toString() είναι μια εντολή που θα δημιουργήσει ένα νέο αντικείμενο Java Date, και θα επιστρέψει την τρέχουσα τιμή ως string. Όλες οι δηλώσεις εντολών εσωκλείονται μέσα σε "<%...%>" ζευγάρια. Η δήλωση @page import="java.util.Date" συμπεριλαμβάνεται για να δηλώσει τη θέση της κλάσης Date.

```
<%@page import="java.util.Date"%>
<html>
  <body>
    <h2>Web server information</h2>
    <table border=1>
      <tr>
        <td>Date:</td>
    <td><%= new Date().toString()%></td>
```

```
      </tr>
    </table>
  </body>
</html>
```

Όταν αυτό το αρχείο JSP εκτελείται από το διακομιστή Web, παράγει έξοδο παρόμοια με αυτό που φαίνεται παρακάτω. Όλες οι δηλώσεις HTML τοποθετούνται στην απάντηση ως έχει. Ωστόσο, οι δηλώσεις εντολών στο αρχικό αρχείο JSP εκτελέστηκαν και τα αποτελέσματα περιλαμβάνονται στην απάντηση. Πιο συγκεκριμένα, μπορούμε να δούμε ότι η εκτέλεση της δήλωσης newDate().toString() παράγει την τιμή "Τετ Μαρ 03 17:02:50 GMT +00:00 2002 ".

```
<html>
  <body>
    <h2>Web server information</h2>
    <table border=1>
      <tr>
        <td>Date:</td>
    <td>Wed Mar 03 17:02:50 GMT+00:00 2002</td>
      </tr>
    </table>
  </body>
</html>
```

Όταν ερμηνευθεί, αυτό το HTML παράγει το αποτέλεσμα που φαίνεται στο Σχήμα 1.6.

Σχήμα 1.6 Έξοδος από ένα JSP σε έναν browser

Η τεχνολογία JSP ενθαρρύνει μία σύμβαση για περιβάλλοντα εργασίας που βασίζεται μεταξύ του παρόχου των σελίδων JSP και οποιονδήποτε στοιχείων εφαρμογής που χρησιμοποιούνται από τις σελίδες JSP (που μπορεί να είναι EJBs, καθώς και απλές κλάσεις Java, όπως η κλάση της ημερομηνίας που χρησιμοποιείται στο παραπάνω παράδειγμα).

1.3.3 Κλάση Bean

Ένα κομμάτι κώδικα μιας κλάσης bean για το UserAccount EJB παρουσιάζεται παρακάτω.

```
import javax.util.*;
import javax.ejb.*;
....

public abstract class UserAccountBean implements EntityBean
{
  // Instance variables
```

```
  private String password;

  ....

  // Business operations

  public void setPassword(String password) {  this.password =
password; }

  public String getPassword() {  return password; }

  ....

  // Container operations

  public void ejbCreate() throws CreateException {  .... }

  public void ejbRemove()                          {  .... }

  public void ejbActivate()                        {  .... }

  public void ejbPassivate()                       {  .... }

  ....

}
```

Το παράδειγμα αυτό έχει διατηρηθεί σκοπίμως απλό για να μας βοηθήσει να επικεντρωθούμε σε συγκεκριμένα χαρακτηριστικά EJB. Μπορούμε να δούμε ότι η κλάση bean περιέχει μεταβλητές στιγμιότυπων που αντιπροσωπεύουν την κατάσταση των αντικειμένων που υλοποιούνται από το EJB. Μπορούμε επίσης να δούμε ότι η κλάση bean υλοποιεί τις επιχειρηματικές λειτουργίες που ορίζονται στο remote interface (ή local interface), όπως setPassword. Τέλος, μπορούμε να δούμε ότι η κλάση bean υλοποιεί λειτουργίες που απαιτούνται ως μέρος της αμφίδρομης σύμβασης μεταξύ των bean και του κοντέινερ. Για παράδειγμα, όταν ένας πελάτης επικαλείται τη λειτουργία δημιουργίας στο home interface (ή local home interface), αυτό τελικά έχει ως αποτέλεσμα να καλεί ο κοντέινερ την ejbCreate μέθοδο της κλάσης bean.

1.4 Περίληψη

Σε αυτό το κεφάλαιο παρείχαμε μία σύντομη περιγραφή της πλατφόρμας J2EE.

Αν και η J2EE πλατφόρμα παρέχει ένα καλό σημείο εκκίνησης για την ανάπτυξη συστημάτων επιχειρήσεων, παρέχει μόνο ένα μέρος της λύσης που απαιτείται. Ειδικότερα, η πλατφόρμα J2EE δεν παρέχει μια ολοκληρωμένη λύση από τη στιγμή που χρειάζεται ακόμα να καθορίσουμε τη λογική της εφαρμογής που εκτελεί τη χρήση της J2EE πλατφόρμας. Η J2EE είναι επίσης μια περίπλοκη πλατφόρμα και πρέπει να αναζητήσουμε τρόπους για την απλοποίηση της αντίληψης των προγραμματιστών για την αυτές τις πολυπλοκότητες.

Προτού αναλύσουμε τα στάδια υλοποίησης της εφαρμογής μας, στο επόμενο κεφάλαιο θα αναλύσουμε την μεθοδολογία RUP, τα οφέλη της και στην συνέχεια θα παρουσιάσουμε την μεθοδολογία ICONIX και την σύγκρισή τους ώστε να καταλήξουμε στη μεθοδολογία ICONIX. Είναι σημαντικό να μελετήσουμε τη μεθοδολογία RUP καθώς μας παρέχει ένα πλήρες πλαίσιο στήριξης και βοήθειας για να ξεκινήσουμε ένα μεγάλο σύστημα λογισμικού καθώς και η σύγκρισή της με τη μεθοδολογία ICONIX, που τελικά χρησιμοποιήθηκε, είναι πολύτιμη.

Κεφάλαιο 2: Μεθοδολογία RUP

Ο σκοπός αυτού του κεφαλαίου είναι να παρουσιάσει την Επαναληπτική Ενοποιημένη Διαδικασία RUP (Rational Unified Process), ένα πλαίσιο (framework) μεθόδου για την ανάπτυξη λογισμικού. Ξεκινάμε αναλύοντας τις βέλτιστες πρακτικές που αποτελούν το θεμέλιο της RUP. Έπειτα συνεχίζουμε περιγράφοντας τις βασικές έννοιες και την γενική αρχιτεκτονική της RUP και κλείνουμε με την χρήση της RUP ως πλαίσιο μεθόδου για την ανάπτυξη μίας εξατομικευμένης διαδικασίας.

2.1 Βέλτιστες πρακτικές – Τα θεμέλια της RUP

Η RUP (Rational Unified Process) είναι ένα επαναληπτικό πλαίσιο ανάπτυξης λογισμικού που δημιουργήθηκε από τη Rational Software Corporations, τμήμα της IBM από το 2003. Η RUP δεν είναι μία απλή, απτή, κανονιστική μέθοδος, αλλά πιο πολύ ένα προσαρμόσιμο πλαίσιο μεθόδου, που προορίζεται για προσαρμογή από εταιρίες και ομάδες ανάπτυξης λογισμικού, που θα επιλέξουν τα κατάλληλα στοιχεία της μεθοδολογίας που ικανοποιούν τις ανάγκες τους. Ο στόχος της είναι να διασφαλίσει την παραγωγή υψηλής ποιότητας λογισμικού που ανταποκρίνεται στις ανάγκες των τελικών χρηστών της, μέσα στα πλαίσια ενός προβλέψιμου χρονοδιαγράμματος και προϋπολογισμού («καλύτερο λογισμικό γρηγορότερα», “better software faster”).

Οι βέλτιστες πρακτικές είναι ένα σύνολο εμπορικά δοκιμασμένων προσεγγίσεων για την ανάπτυξη λογισμικού. Όταν χρησιμοποιούνται σε συνδυασμό, εξασφαλίζουν την επιτυχία ενός έργου ανάπτυξης λογισμικού χτυπώντας στα γενεσιουργά αίτια των τυπικών προβλημάτων ανάπτυξης λογισμικού. Αυτές οι πρακτικές είναι οι παρακάτω:

- **Επαναληπτική ανάπτυξη.** Η λειτουργικότητα του συστήματος πρέπει να δίνετε σε μία διαδοχική σειρά εκδόσεων (releases) του προγράμματος στις οποίες αυξάνεται η πληρότητα του, όπου κάθε έκδοση είναι το αποτέλεσμα μιας επανάληψης. Η επιλογή των απαιτήσεων (requirements) που απευθύνονται σε κάθε επανάληψη

είναι συνυφασμένη με το μετριασμό των κινδύνων του έργου, με τους πιο σημαντικούς κινδύνους να αντιμετωπίζονται πρώτοι.

- **Διαχείριση απαιτήσεων.** Χρησιμοποίηση μίας συστηματικής προσέγγισης για την εκμαίευση και την καταγραφή των απαιτήσεων του συστήματος και στη συνέχεια τη διαχείριση αλλαγών στις απαιτήσεις, κυρίως αξιολογώντας τον αντίκτυπο των αλλαγών αυτών στο υπόλοιπο σύστημα. Η αποτελεσματική διαχείριση των απαιτήσεων προϋποθέτει τη διατήρηση σαφούς δήλωσης των απαιτήσεων, καθώς και της ιχνηλασιμότητας από τις απαιτήσεις στα άλλα αντικείμενα (artifacts) του έργου.

- **Χρήση του κατακερματισμού.** Δόμηση της αρχιτεκτονικής του λογισμικού χρησιμοποιώντας επιμέρους τμήματα (κατακερματισμός προβλήματος σε επιμέρους μικρότερα και ευκολότερα). Η χρήση επιμέρους τμημάτων στην ανάπτυξη λογισμικού συμβάλει στη μείωση της πολυπλοκότητας του προβλήματος και οδηγεί σε μία αρχιτεκτονική πιο ισχυρή και ανθεκτική, που παρέχει τη δυνατότητα της επαναχρησιμοποίησης των επιμέρους τμημάτων που συνθέτουν την αρχιτεκτονική. Επίσης προωθεί την δυνατότητα να δοκιμάζονται τα επιμέρους τμήματα του λογισμικού για την αποτελεσματικότητά τους πριν ενταχθούν στο κυρίως σύστημα.

- **Οπτική μοντελοποίηση.** Δημιουργία μιας σειράς οπτικοποιημένων μοντέλων, καθένα από τα οποία υπογραμμίζει συγκεκριμένες λεπτομέρειες και στοιχεία του συστήματος και αγνοεί-φιλτράρει άλλα. Τα μοντέλα αυτά βοηθούν στην καλύτερη κατανόηση του συστήματος που αναπτύσσεται και παρέχουν ένα μηχανισμό επικοινωνίας μεταξύ των διαφορετικών μελών της ομάδας (σχεδιαστές, προγραμματιστές, testers κτλ.). Η χρησιμοποίηση διαγραμμάτων για την αναπαράσταση όλων των επιμέρους τμημάτων του συστήματος, των χρηστών και την αλληλεπίδρασή τους σε γλώσσα UML (Unified Modeling Language), αποτελεί ένα χρήσιμο εργαλείο για την επίτευξη της πρακτικής αυτής.

- **Συνεχής έλεγχος της ποιότητας.** Συνεχής αξιολόγηση της ποιότητας του συστήματος με τις λειτουργικές (functional) και μη λειτουργικές (non-functional) απαιτήσεις του. Εκτέλεση δοκιμών (testing) ως μέρος της κάθε επανάληψης. Είναι λιγότερο δαπανηρή η διόρθωση ελλείψεων που διαπιστώθηκαν στα πρώτα στάδια της ανάπτυξης του κύκλου ζωής του λογισμικού από την διόρθωσή τους σε επόμενα στάδια.

- **Διαχείριση αλλαγών.** Καθιέρωση μίας πειθαρχημένης και ελεγχόμενης προσέγγισης στη διαχείριση αλλαγών σε μεταβαλλόμενες απαιτήσεις, τεχνολογία, πόρους, προϊόντα, πλατφόρμες και ούτω καθεξής. Έλεγχος στο πως εισάγονται οι αλλαγές στα αντικείμενα (artifacts) του έργου, ποιος εισάγει τις αλλαγές και πότε πρέπει να εισάγονται. Παροχή ενός μέσου για τον αποτελεσματικό συγχρονισμό των αλλαγών μεταξύ των επιμέρους ομάδων ανάπτυξης του λογισμικού, των διαφορετικών εκδόσεων του λογισμικού, των πλατφόρμων και ούτω καθεξής.

Αυτές οι πρακτικές είναι αποτέλεσμα εμπειριών της Rational Software, η οποία εξαγοράστηκε από την IBM το 2003, στην ανάπτυξη προϊόντων λογισμικού μαζί με τις εμπειρίες των πελατών τους. Η εφαρμογή αυτών των βέλτιστων πρακτικών θέτει μία εταιρία ανάπτυξης λογισμικού σε θέση να παρέχει υψηλής ποιότητας λογισμικό με τη βοήθεια ενός επαναλαμβανόμενου και προβλέψιμου τρόπου.

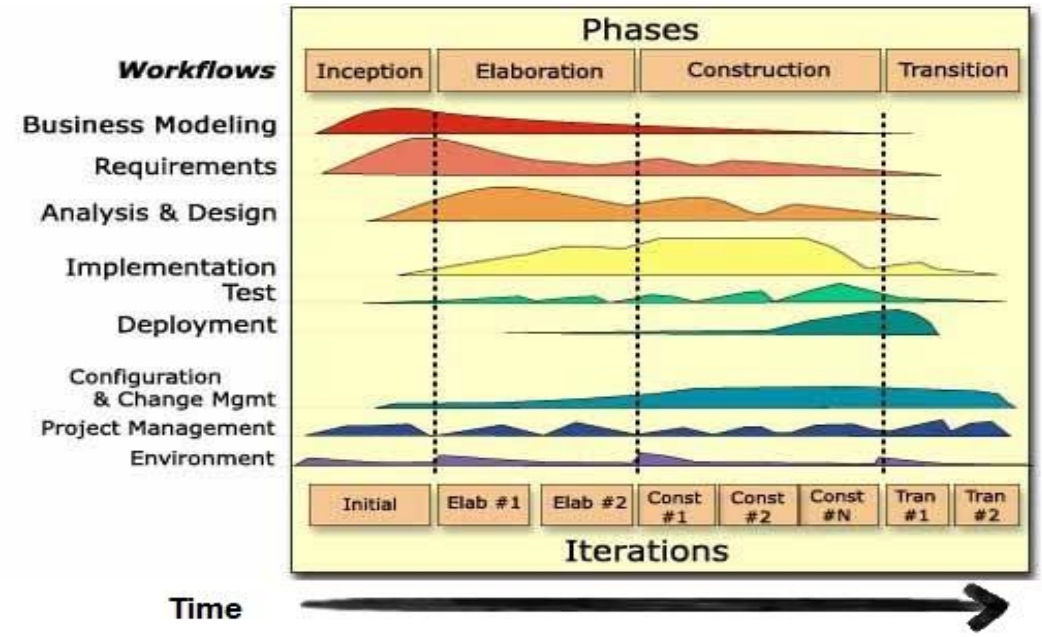
2.2 Δομικά στοιχεία RUP

Η RUP βασίζεται σε ένα σύνολο δομικών στοιχείων τα οποία περιγράφουν τι θα παραχθεί, τις απαραίτητες δεξιότητες που απαιτούνται και εξήγηση βήμα προς βήμα πως θα επιτευχθούν οι συγκεκριμένοι στόχοι ανάπτυξης της εφαρμογής. Τα κύρια δομικά στοιχεία είναι τα εξής:

- Ρόλοι (ποιός) – Ένας ρόλος καθορίζει ένα σύνολο σχετικών αρμοδιοτήτων, δεξιοτήτων και ευθυνών.

- Αντικείμενα (artifacts) (τι) – Τα αντικείμενα αντιπροσωπεύουν κάτι που προκύπτει από μία εργασία, συμπεριλαμβανομένων και όλων των εγγράφων και των μοντέλων που παράγονται κατά τη διάρκεια της εργασίας μέσω της διαδικασίας RUP.
- Καθήκοντα (πως) – Ένα καθήκον περιγράφει μία μονάδα εργασίας που ανατίθεται σε ένα ρόλο και παρέχει ένα ουσιαστικό αποτέλεσμα.
- Φάσεις, επαναλήψεις, κλάδοι-τομείς και λεπτομέρειες ροής εργασίας (workflow details) (πότε)

Η RUP μπορεί να περιγραφεί και αναπαρασταθεί με τη βοήθεια δύο διαστάσεων: του **χρόνου** και του **περιεχομένου**. Το Σχήμα 2.1 μας παρουσιάζει μία γραφική αναπαράσταση των διαστάσεων αυτών. Ο οριζόντιος άξονας αναπαριστά τον χρόνο και παρουσιάζει τις πτυχές του κύκλου ζωής της διαδικασίας ανάπτυξης λογισμικού. Η διάσταση αυτή περιγράφεται σε σχέση με τις φάσεις (phases) και τις επαναλήψεις (iterations). Ο κάθετος άξονας παρουσιάζει το περιεχόμενο και δείχνει τους κλάδους-τομείς (disciplines) ,που ομαδοποιούν λογικά τη διαδικασία του περιεχομένου.



Σχήμα 2.1 Διαστάσεις χρόνου και περιεχομένου της RUP

Όπως δείχνουν τα «σαμαράκια» στο Σχήμα 2.1, η σχετική έμφαση των κλάδων (disciplines) αλλάζει κατά την διάρκεια ζωής του έργου. Για παράδειγμα, στις αρχικές επαναλήψεις ο περισσότερος χρόνος δαπανάται για τις απαιτήσεις του συστήματος, καθώς σε μεταγενέστερες επαναλήψεις περισσότερος χρόνος δαπανάται στην υλοποίηση της εφαρμογής. Πρέπει να θυμόμαστε, ωστόσο, ότι κάθε κλάδος πρέπει να λαμβάνεται υπόψιν σε κάθε επανάληψη.

Σε κάθε επανάληψη τα καθήκοντα κατηγοριοποιούνται σε εννέα κλάδους-τομείς (disciplines). Ένας κλάδος είναι μία συλλογή από δραστηριότητες που σχετίζονται με ένα μεγάλο «τομέα ενδιαφέροντος» στο πλαίσιο του συνολικού έργου. Οι κλάδοι ομαδοποιούν τις δραστηριότητες λογικά. Όπως δείχνουμε στην εικόνα 2.1 η RUP είναι οργανωμένη γύρω από 9 κλάδους. Ο πίνακας 2.1 παρέχει μία σύντομη περιγραφή αυτών των κλάδων.

Κλάδοι RUP	Σύντομη περιγραφή
Μοντελοποίηση Επιχείρησης (Business Modeling)	Ο σκοπός της Μοντελοποίησης Επιχείρησης είναι: • Κατανόηση της δομής και της δυναμικής της εταιρίας για την οποία αναπτύσσεται το σύστημα. • Κατανόηση των υπαρχόντων προβλημάτων της εταιρίας και εντοπισμός πιθανής βελτίωσης. • Επιβεβαίωση ότι οι πελάτες, οι τελικοί χρήστες και οι προγραμματιστές έχουν κοινή αντίληψη του στόχου. • Άντληση των απαιτήσεων του συστήματος.
Απαιτήσεις (Requirements)	Ο σκοπός των Απαιτήσεων είναι: • Καθιέρωση και διατήρηση μίας συμφωνίας μεταξύ των πελατών και των άλλων ενδιαφερόμενων μελών σχετικά με το σύστημα που θα υλοποιήσουν. • Παροχή μίας καλύτερης κατανόησης του συστήματος στους προγραμματιστές. • Καθορισμός των ορίων που οριοθετούν το σύστημα.

	• Παροχή μίας βάσης για τον καθορισμό των τεχνικών περιεχομένων των επαναλήψεων. • Παροχή μίας βάσης για την εκτίμηση του κόστους και του χρόνου για την ανάπτυξη του συστήματος.
Ανάλυση και Σχεδιασμός (Analysis and Design)	Ο σκοπός της Ανάλυσης και του Σχεδιασμού είναι: • Μετατροπή των απαιτήσεων σε σχεδιασμό του μελλοντικού συστήματος. • Ανάπτυξη μιας εύρωστης αρχιτεκτονικής για το σύστημα. • Προσαρμογή του σχεδιασμού ώστε να ταιριάζει με το περιβάλλον της εφαρμογής.
Υλοποίηση (Implementation)	Ο σκοπός της Υλοποίησης είναι: • Ορισμός της οργάνωσης της υλοποίησης. • Υλοποίηση των στοιχείων σχεδίασης. • Έλεγχος κάθε ομάδας υλοποίησης. • Ενσωμάτωση των αποτελεσμάτων που παράγονται από μεμονωμένες ομάδες υλοποίησης, στο τελικό εκτελέσιμο σύστημα.
Έλεγχος (Test)	Ο σκοπός του Ελέγχου είναι: • Εύρεση και καταγραφή των ελαττωμάτων στην ποιότητα λογισμικού. • Παροχή γενικών συμβουλών σχετικά με την θεωρούμενη ποιότητα λογισμικού. • Απόδειξη της εγκυρότητας των υποθέσεων που έγιναν κατά τον σχεδιασμό και την καταγραφή απαιτήσεων μέσω επίδειξης. • Επικύρωση ότι το λογισμικό λειτουργεί όπως είχε σχεδιαστεί. • Επικύρωση ότι οι λειτουργίες των προϊόντων λογισμικού λειτουργούν όπως απαιτείται (δηλαδή οι απαιτήσεις έχουν υλοποιηθεί σωστά).
Εγκατάσταση (Deployment)	Ο σκοπός της Εγκατάστασης είναι: • Επιβεβαίωση ότι το προϊόν λογισμικού είναι διαθέσιμο σε όλους τους τελικούς χρήστες.

Διαμόρφωση και Διαχείριση Αλλαγών (Configuration and Change Management)	Ο σκοπός της Διαμόρφωσης και Διαχείρισης Αλλαγών είναι: • Προσδιορισμός των στοιχείων διαμόρφωσης[3]. • Περιορισμός των αλλαγών σε αυτά τα στοιχεία. • Έλεγχος των αλλαγών που έγιναν σε αυτά τα στοιχεία. • Καθορισμός και διαχείριση των διαμορφώσεων[4] των εν λόγω αντικειμένων.
Διαχείριση Έργου (Project Management)	Ο σκοπός της Διαχείρισης έργου είναι: • Διαχείριση ενός εντατικού έργου • Σχεδίαση, στελέχωση, εκτέλεση και παρακολούθηση του έργου. • Διαχείριση κινδύνων.
Περιβάλλον (Environment)	Ο σκοπός του Περιβάλλοντος είναι: • Παροχή στην οργάνωση ανάπτυξης του λογισμικού ενός περιβάλλοντος ανάπτυξης λογισμικού, που περιλαμβάνει διαδικασίες και εργαλεία, που θα στηρίξουν την ομάδα οργάνωσης. Αυτό περιλαμβάνει τη διαμόρφωση της διαδικασίας για ένα συγκεκριμένο έργο, καθώς και την ανάπτυξη κατευθυντήριων γραμμών για την υποστήριξη του έργου.

Πίνακας 1. 1 Κλάδοι RUP

2.2.1 Επαναλήψεις (Iterations)

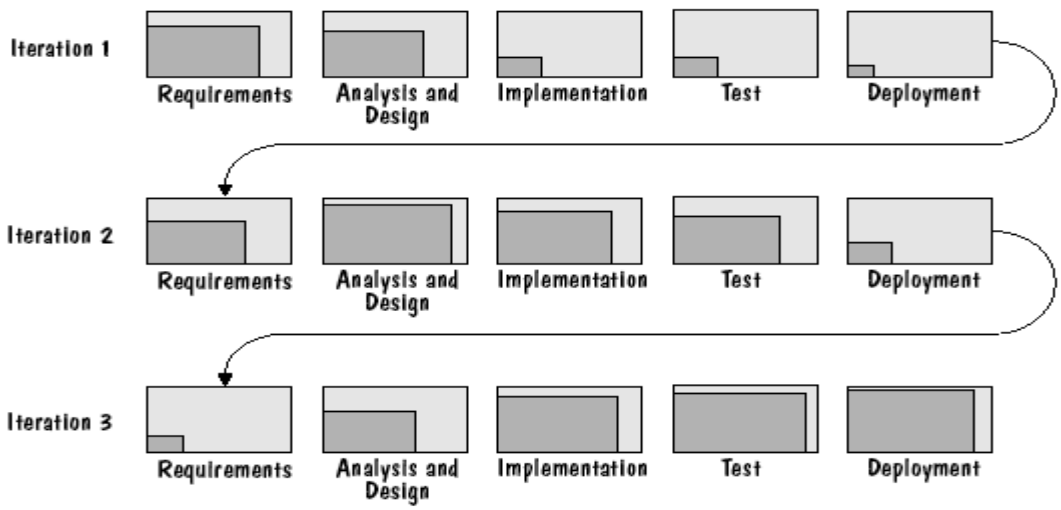
Η επαναληπτική ανάπτυξη αποτελεί ένα βασικό χαρακτηριστικό των επιτυχημένων έργων ανάπτυξης λογισμικού. Μέσα σε μια επαναληπτική ανάπτυξη του κύκλου ζωής του λογισμικού, αρκετά περάσματα πραγματοποιούνται σε καθέναν από τους κλάδους. Κάθε πέρασμα ονομάζεται επανάληψη. Μια επανάληψη είναι μια ευδιάκριτη, χρονικά υπολογισμένη ακολουθία

[3] Στοιχεία διαμόρφωσης (configuration items) είναι τα στοιχεία που έχουν τεθεί υπό τον έλεγχο της διαχείρισης διαμορφώσεων. Τα στοιχεία διαμόρφωσης αποδίδονται ατομικά και ως εκ τούτου μπορούν να προσδιοριστούν μονοσήμαντα σε μία δεδομένη χρονική στιγμή. Τα στοιχεία διαμόρφωσης είναι μέρος μιας Διαμόρφωσης.

[4] Διαμόρφωση είναι ένα σύνολο από Στοιχεία διαμορφώσεως που καθορίζουν μία συγκεκριμένη έκδοση ενός συστήματος ή μέρος του συστήματος.

δραστηριοτήτων που καταλήγει στην έκδοση (εσωτερικά ή εξωτερικά) ενός εκτελέσιμου προϊόντος. Καθώς το έργο προχωρά, οι εκδόσεις εξελίσσονται από ένα υποσύνολο του τελικού προϊόντος στο τελικό σύστημα. Μια επαναληπτική διαδικασία ανάπτυξης είναι παρόμοια με το «αναπτυσσόμενο» λογισμικό, όπου το τελικό προϊόν ωριμάζει με το χρόνο. Κάθε επανάληψη καταλήγει σε μια καλύτερη κατανόηση των απαιτήσεων, μια πιο εύρωστη αρχιτεκτονική, ένα πιο έμπειρο οργανισμό ανάπτυξης και μια πιο πλήρη εφαρμογή.

Το Σχήμα 2.2 δείχνει πως το επίκεντρο του έργου μετατοπίζεται σε όλη τη διάρκεια των διαδοχικών επαναλήψεων. Το μέγεθος των κουτιών σε καθεμία από τις δραστηριότητες δείχνει το σχετικό χρόνου που δαπανάται για να εκτελεστούν οι δραστηριότητες σε κάθε κλάδο. Κάθε κλάδος εμφανίζεται σε κάθε επανάληψη, αλλά η σχετική έμφαση μετατοπίζεται καθώς το έργο προχωράει από τις Απαιτήσεις, στην Ανάλυση και το Σχεδιασμό, στην Υλοποίηση, στον Έλεγχο και τέλος στην Εγκατάσταση.



Σχήμα 2.2 Ένας επαναληπτικός κύκλος ζωής

Ο παρακάτω κατάλογος παρουσιάζει μερικά σημαντικά χαρακτηριστικά μία επιτυχημένης επανάληψης.

- Η επανάληψη πρέπει να έχει σαφή κριτήρια αξιολόγησης.
- Η επανάληψη έχει μία σχεδιασμένη δυνατότητα που μπορεί να εξακριβωθεί.

- Η επανάληψη ολοκληρώνεται με ένα μικρό ορόσημο, όπου το αποτέλεσμα της επανάληψης εκτιμάται σε σχέση με τα αντικειμενικά κριτήρια επιτυχίας αυτής της συγκεκριμένης επανάληψης.
- Κατά τη διάρκεια της επανάληψης, τα αντικείμενα ενημερώνονται (αντικείμενα εξελίσσονται με το σύστημα).
- Κατά τη διάρκεια της επανάληψης, το σύστημα είναι ολοκληρωμένο και δοκιμασμένο.

2.3 Τέσσερις φάσεις του κύκλου ζωής του έργου

Η RUP έχει καθορίσει τον κύκλο ζωής του έργου που αποτελείται από τέσσερις φάσεις. Οι φάσεις αυτές επιτρέπουν στην διαδικασία να παρουσιαστεί σε υψηλό επίπεδο με παρόμοιο τρόπο με ένα έργο τύπου «καταρράκτη» (“waterfall”), αν και στην ουσία το κλειδί της διαδικασίας έγκειται στις επαναλήψεις της ανάπτυξης που βρίσκονται μέσα σε όλες τις φάσεις . Επίσης, κάθε φάση έχει βασικό στόχο και ορόσημο στο τέλος που υποδηλώνει ότι στόχος έχει επιτευχθεί.

2.3.1 Φάση Έναρξης (Inception)

Ο πρωταρχικός στόχος είναι να εξετάσουμε το σύστημα κατάλληλα ως βάση για την επικύρωση της αρχικής κοστολόγησης και του προϋπολογισμού. Σε αυτή τη φάση εγκαθίσταται το επιχειρηματικό ενδιαφέρον που περιλαμβάνει το πλαίσιο των επιχειρήσεων, παράγοντες επιτυχίας (αναμενόμενα έσοδα, η αναγνώριση της αγοράς, κ.λπ.), καθώς και οικονομικές προβλέψεις. Για την ολοκλήρωση του επιχειρηματικού ενδιαφέροντος δημιουργούνται ένα βασικό μοντέλο περίπτωσης χρήσης, το σχέδιο του έργου, η αρχική εκτίμηση των κινδύνων και η περιγραφή του έργου (οι κεντρικές απαιτήσεις του έργου, οι περιορισμοί και τα βασικά χαρακτηριστικά του). Μετά την ολοκλήρωση των παραπάνω, το έργο ελέγχεται με βάση τα ακόλουθα κριτήρια:

- Σύμπτωση των ενδιαφερομένων μερών σχετικά με τον ορισμό του πεδίου εφαρμογής και τις εκτιμήσεις του κόστους και του χρονοδιαγράμματος.
- Κατανόηση απαιτήσεων, όπως αποδεικνύεται από την πιστότητα των πρωτογενών περιπτώσεων χρήσης.

- Αξιοπιστία των εκτιμήσεων του κόστους και του χρονοδιαγράμματος, των προτεραιοτήτων, των κινδύνων και της αναπτυξιακής διαδικασίας.
- Βάθος και εύρος κάθε αρχιτεκτονικού προτύπου που αναπτύχθηκε.
- Καθιέρωση μιας βάσης από την οποία συγκρίνουμε τις πραγματικές δαπάνες σε σχέση με τις προγραμματισμένες δαπάνες.

Εάν το σχέδιο δεν περάσει το ορόσημο αυτό, το οποίο ονομάζεται αντικειμενικό ορόσημο του κύκλου ζωής, μπορεί είτε να ακυρωθεί ή να επαναληφθεί αφού επανασχεδιαστεί για να ανταποκριθεί καλύτερα στα κριτήρια.

2.3.2 Φάση Επιμέλειας (Elaboration)

Ο πρωταρχικός στόχος είναι η άμβλυνση των κύριων στοιχείων κινδύνου που προσδιορίζονται από την ανάλυση έως το τέλος της φάσης αυτής. Η φάση της επιμέλειας είναι, όταν το έργο αρχίζει να παίρνει μορφή. Σε αυτή τη φάση έχουμε τη domain ανάλυση και η αρχιτεκτονική του σχεδίου παίρνει τη βασική μορφή της.

Η φάση αυτή πρέπει να περάσει το ορόσημο του κύκλου ζωής της αρχιτεκτονικής που ανταποκρίνεται στα παρακάτω παραδοτέα:

- Ένα μοντέλο περίπτωσης χρήσης (use case model) στο οποίο οι περιπτώσεις χρήσης και οι χαρακτήρες (actors) έχουν προσδιοριστεί και οι περισσότερες περιγραφές περιπτώσεων χρήσης έχουν αναπτυχθεί. Το μοντέλο της περίπτωσης χρήσης θα πρέπει να ολοκληρωθεί κατά 80%.
- Μία περιγραφή της αρχιτεκτονικής του λογισμικού σε μία διαδικασία ανάπτυξης συστημάτων λογισμικού.
- Μία εκτελέσιμη αρχιτεκτονική που πραγματοποιεί αρχιτεκτονικά σημαντικές περιπτώσεις χρήσης.
- Επιχειρησιακές υποθέσεις και κατάλογος κινδύνων που αναθεωρούνται.
- Ένα σχέδιο ανάπτυξης του συνόλου του έργου.
- Πρωτότυπα που μετριάζουν εμφανώς κάθε προσδιορισμένο τεχνικό κίνδυνο.

Εάν το σχέδιο δεν μπορεί να περάσει αυτό το ορόσημο, υπάρχει ακόμη χρόνος για να μπορέσει να ακυρωθεί ή να επανασχεδιαστεί. Ωστόσο, μετά τη λήξη αυτής της φάσης, το έργο μεταβαίνει σε μία υψηλού κινδύνου λειτουργία όπου οι αλλαγές είναι πολύ πιο δύσκολες και επιζήμιες όταν γίνονται.

Η βασική domain ανάλυση για την επιμέλεια είναι η αρχιτεκτονική του συστήματος.

2.3.3 Φάση Κατασκευής (Construction)

Ο πρωταρχικός στόχος είναι η δημιουργία του συστήματος λογισμικού. Σε αυτή τη φάση, το κύριο ενδιαφέρον επικεντρώνεται στην ανάπτυξη των στοιχείων (components) και των άλλων χαρακτηριστικών του συστήματος. Αυτή είναι η φάση στην οποία λαμβάνει χώρα το μεγαλύτερο μέρος της κωδικοποίησης. Σε μεγάλα έργα, αρκετές επαναλήψεις κατασκευής μπορεί να αναπτυχθούν σε μια προσπάθεια να διαιρεθούν οι περιπτώσεις χρήσης σε διαχειρίσιμα τμήματα που παράγουν αποδεδειγμένα πρωτότυπα.

Η φάση αυτή παράγει την πρώτη εξωτερική έκδοση του λογισμικού. Το συμπέρασμα της χαρακτηρίζεται από το ορόσημο της αρχικής επιχειρησιακής δυνατότητας.

2.3.4 Φάση Μετάβασης (Transition)

Ο πρωταρχικός στόχος είναι «διέλευση» του συστήματος από την ανάπτυξη στην παραγωγή, τη διάθεσή του και την κατανόηση του από τον τελικό χρήστη. Οι δραστηριότητες της φάσης αυτής περιλαμβάνουν την κατάρτιση των τελικών χρηστών και συντηρητών και δοκιμές beta του συστήματος για να επικυρωθεί

2.4 Η RUP ως πλαίσιο διαδικασίας

Η RUP είναι περιεκτική και πλήρης. Παρέχει λεπτομερή βήματα δραστηριοτήτων, αντικειμένων, προτύπων, κατευθυντήριων γραμμών, ελέγχων, και παραδειγμάτων. Ωστόσο, η σκοπιά «ένα μέγεθος για όλους» δεν ισχύει σε μία διαδικασία ανάπτυξης λογισμικού. Έτσι, η RUP σχεδιάστηκε σαν πλαίσιο

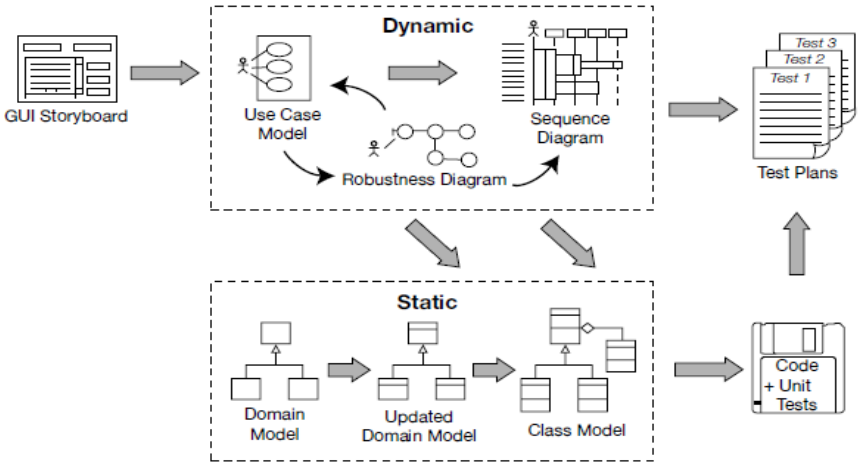
διαδικασίας από το οποίο θα μπορούσαν να προκύψουν προσαρμοσμένες διαδικασίες σύμφωνα με τις εκάστοτε απαιτήσεις του συστήματος υλοποίησης. Στην πραγματικότητα, εκτός από καθοδήγηση ανάπτυξης λογισμικού, η RUP περιέχει διαδικασία καθοδήγησης προσαρμογής. Με άλλα λόγια, η RUP περιέχει λεπτομερείς πληροφορίες για το πώς πρέπει να προσαρμοστεί η RUP για ένα συγκεκριμένο έργο, είδος λύσης ή οργανισμό. Η διαδικασία καθοδήγησης προσαρμογής παρέχεται από τον κλάδο Περιβάλλον της RUP.

2.5 Περίληψη

Στο κεφάλαιο αυτό παρείχαμε μια σύντομη εισαγωγή για την Επαναληπτική Ενοποιημένη Διαδικασία (Rational Unified Process, RUP), συμπεριλαμβανομένης της μηχανικής λογισμικού βέλτιστων πρακτικών από τις οποίες προέρχεται, τις βασικές έννοιες που χρησιμοποιούνται για την περιγραφή της και την υποστήριξη που παρέχεται για την προσαρμογή σε αυτή. Εν ολίγοις, η RUP συλλαμβάνει την ανάπτυξη λογισμικού βέλτιστων πρακτικών σε μια μορφή που μπορεί να προσαρμοστεί για ένα ευρύ φάσμα σκοπών.

Στο επόμενο κεφάλαιο αναλύουμε τη μεθοδολογία ICONIX και τα πλεονεκτήματα που παρουσιάζει έναντι της RUP, καθώς και το λόγω που τελικά επιλέχτηκε για την ανάπτυξη της εκπαιδευτικής μας εφαρμογής «Online Τουριστικό Γραφείο».

Κεφάλαιο 3: Εισαγωγή στη μεθοδολογία ICONIX



Σχήμα 3.1 Μέθοδος ICONIX Μέθοδος ICONIX σε λεπτομερειακή ροή εργασίας

Θεωρητικά, κάθε πτυχή της UML είναι δυνητικά χρήσιμη, αλλά στην πράξη, ποτέ δεν υπάρχει αρκετός χρόνος για να γίνει η μοντελοποίηση, η ανάλυση, και ο σχεδιασμός. Πάντα υπάρχει πίεση από τη διοίκηση για μετάβαση σε κώδικα, για να ξεκινήσει η κωδικοποίηση πρόωρα επειδή η πρόοδος των έργων λογισμικού τείνει να μετράται με το πόσο κώδικας έχει γραφτεί. Η μεθοδολογία ICONIX, όπως φαίνεται και στο Σχήμα 3.1 παραπάνω, είναι μια μινιμαλιστική, εξελιγμένη προσέγγιση που εστιάζει σε εκείνη την περιοχή που βρίσκεται ανάμεσα στις περιπτώσεις χρήσης και τον κώδικα. Έμφαση δίνεται στο τι πρέπει να συμβεί σε εκείνο το σημείο του κύκλου ζωής όπου ξεκινάμε: έχουμε ξεκινήσει με ορισμένες περιπτώσεις χρήσης και τώρα θα πρέπει να κάνουμε καλή ανάλυση και σχεδιασμό.

Στο κεφάλαιο αυτό περιγράφουμε τις βασικές αρχές της μεθοδολογίας ICONIX, τη μεθοδολογία που τελικά χρησιμοποιήθηκε για την ανάπτυξη της εφαρμογής «Online Τουριστικό Γραφείο».

3.1 Λίγα λόγια για τη μέθοδο ICONIX

Η ICONIX είναι μία μεθοδολογία ανάπτυξης λογισμικού που προϋπάρχει της RUP, του Extreme Programming (XP) και της Agile ανάπτυξης λογισμικού. Όπως

και η RUP, η μεθοδολογία ICONIX είναι UML, Use Case-driven αλλά πιο ελαφριά από την RUP.

Αυτό που ξεχωρίζει τη μεθοδολογία ICONIX είναι ότι χρησιμοποιεί τα διαγράμματα ευρωστίας (robustness diagrams) της UML, μία μέθοδο γεφύρωσης του κενού μεταξύ της ανάλυσης και του σχεδιασμού. Η ανάλυση ευρωστίας περιορίζει την ασάφεια στις περιγραφές περιπτώσεων χρήστη, εξασφαλίζοντας ότι έχουν συνταχθεί στο πλαίσιο ενός συνοδευτικού domain model. Η διαδικασία αυτή καθιστά τις περιπτώσεις χρήσης πολύ πιο εύκολες στο σχεδιασμό, τον έλεγχο και την εκτίμηση.

Ουσιαστικά η μεθοδολογία ICONIX χρησιμοποιεί τον «λογικό» πυρήνα της ανάλυσης και του σχεδιασμού της διαδικασίας μοντελοποίησης. Ωστόσο, η μεθοδολογία μπορεί να χρησιμοποιηθεί χωρίς μεγάλη προσαρμογή σε έργα που ακολουθούν διαφορετική πορεία διαχείρισης.

3.2 Μέθοδος ICONIX στη Θεωρία

Στην ενότητα αυτή παρέχουμε μια επισκόπηση της μεθοδολογίας ICONIX, δείχνοντας πως όλες οι δραστηριότητες ταιριάζουν μεταξύ τους. Θα ξεκινήσουμε με μια άποψη υψηλού επιπέδου - μια συνολική εικόνα της επισκόπησης - και στη συνέχεια θα εξετάσουμε κάθε δραστηριότητα με περισσότερες λεπτομέρειες. Καθώς περπατάτε μέσα από την επισκόπηση, μπορούμε να αναπέμπουμε στο Σχήμα 3.1, ώστε να βλέπουμε πώς το κάθε μέρος εντάσσεται στη συνολική διαδικασία.

3.2.1 Επισκόπηση: Μετάβαση από τις περιπτώσεις χρήσης στον κώδικα

Το διάγραμμα στην αρχή αυτού του κεφαλαίου παρέχει μια επισκόπηση της μεθοδολογίας ICONIX. Όπως βλέπουμε από το διάγραμμα, η μεθοδολογία ICONIX χωρίζεται σε δυναμικές και στατικές ροές εργασίας, οι οποίες είναι ιδιαίτερα επαναληπτικές: μπορεί να χρειαστεί να γίνει χρήση της μεθοδολογίας ICONIX, για μια μικρή παρτίδα των περιπτώσεων χρήσης (ίσως χρήση σε μερικά πακέτα όπου δεν περιλαμβάνουν μεγάλη λειτουργικότητα δεδομένου ότι κάθε περίπτωση χρήσης αποτελείται από δύο παραγράφους), περνώντας από όλη τη

διαδρομή μέχρι τον πηγαίο κώδικα και τις μονάδες ελέγχου. Για το λόγο αυτό, η μεθοδολογία ICONIX είναι κατάλληλη για ευέλικτα (agile) έργα, όπου η άμεση ανατροφοδότηση είναι αναγκαία σε παράγοντες όπως οι απαιτήσεις, ο σχεδιασμός και οι εκτιμήσεις. Ας δούμε τα βήματα από τα οποία απαρτίζεται η υλοποίηση του προγράμματος μας. (Αναλυτική περιγραφή και παραδείγματα δίνονται στα επόμενα κεφάλαια).

Όπως και με κάθε έργο, σε κάποιο στάδιο στην αρχή ξεκινάμε την εξερεύνηση και τον καθορισμό των απαιτήσεων. Σημειώστε ότι σε κάθε στάδιο υπάρχει ένας βαθμός παραλληλισμού, έτσι ώστε όλες οι δραστηριότητες κατά τη φάση καθορισμού απαιτήσεων επικαλύπτονται και στρώνονται μέχρι να είναι πλήρως έτοιμες.

3.2.1.1 Απαιτήσεις (Requirements)

Υπάρχουν τέσσερα βήματα που ακολουθούνται κατά τη διάρκεια της ενότητας απαιτήσεις.

i. **Λειτουργικές απαιτήσεις(functional requirements).** Ορίζουμε τι θα πρέπει να κάνει το σύστημα μας. Ανάλογα με τον τρόπο που είναι οργανωμένο το σύστημά μας είτε θα πρέπει να συμμετέχουμε στη δημιουργία-εξεύρεση των λειτουργικών απαιτήσεων ή οι απαιτήσεις θα εκδίδονται από τον πελάτη ή μία ομάδα αναλυτών επιχειρήσεων.
ii. **Domain μοντελοποίηση (Domain modeling).** Κατανόηση των προβλημάτων με ξεκάθαρο τρόπο.
iii. **Απαιτήσεις συμπεριφοράς (behavioral requirements).** Καθορίζουμε το πώς ο χρήστης και το σύστημα θα αλληλεπιδρούν (δηλαδή γράφουμε τις πρώτες πρόχειρες περιπτώσεις χρήσης). Επίσης ξεκινάμε με ένα πρωτότυπο του γραφικού περιβάλλοντος και εντοπίζουμε όλες τις περιπτώσεις χρήσης που θα υλοποιήσουμε, ή τουλάχιστον καταλήγουμε σε μία πρώτη λίστα με περιπτώσεις χρήσης, όπου λογικά θα

περιμένουμε να αλλάξουν όσο αναλύουμε τις απαιτήσεις σε μεγαλύτερο βάθος.

iv. **Ορόσημο 1: Επανεξέταση απαιτήσεων (Requirements Review).** Βεβαιωνόμαστε ότι το κείμενο που αφορά τις περιπτώσεις χρήσης ταιριάζει με τις προσδοκίες του πελάτη. Σημειώνετε, ότι ενδέχεται να χρειαστεί επανεξέταση των περιπτώσεων χρήσης σε μικρότερες παρτίδες.

Στη συνέχεια σε κάθε επανάληψη (των μικρότερων παρτίδων περιπτώσεων χρήσης) κάνουμε τα ακόλουθα.

3.2.1.2 Ανάλυση/ Προκαταρκτική Μελέτη (Preliminary Design)

i. **Ανάλυση Ευρωστίας(Robustness Analysis).** Σχεδιασμός ενός διαγράμματος ευρωστίας (robustness diagram) (μια «εικόνα αντικειμένου» από τα βήματα μίας περίπτωσης χρήσης), ανασυντάσσοντας το κείμενο της περίπτωσης χρήσης καθώς προχωράμε.

ii. Ενημέρωση του μοντέλου domain καθώς γράφουμε την περίπτωση χρήσης και με βάση το διάγραμμα ευρωστίας. Εδώ ανακαλύπτουμε κλάσεις που λείπουν, διορθώνουμε ασάφειες, και προσθέτουμε χαρακτηριστικά στα domain αντικείμενα (objects) (π.χ., διαπιστώνουμε ότι ένα αντικείμενο Ξενοδοχείο έχει ένα όνομα, διεύθυνση, περιγραφή, κ.λπ.).

iii. Ονομάζουμε όλες τις λογικές λειτουργίες του λογισμικού (ελεγκτές, controllers) που χρειάζονται για τη λειτουργία της περίπτωσης χρήσης.

iv. Ανασύνταξη των πρώτων πρόχειρων περιπτώσεων χρήσης.

v. **Ορόσημο 2: Προκαταρκτική Επισκόπηση Σχεδιασμού (Preliminary Design Review, PDR).**

3.2.1.3 Λεπτομερής Σχεδιασμός (Detailed Design)

i. **Διαγράμματα Ακολουθίας (Sequence Diagrams).** Σχεδιασμός ενός διαγράμματος ακολουθίας (ένα διάγραμμα ακολουθίας για κάθε περίπτωση χρήσης) για να δείξουμε με λεπτομέρεια πώς θα

υλοποιήσουμε την περίπτωση χρήσης. Η πρωταρχική λειτουργία των διαγραμμάτων ακολουθίας είναι να κατανεμηθεί η συμπεριφορά στις κλάσεις μας.

ii. Ενημέρωση του domain model καθώς σχεδιάζουμε το διάγραμμα ακολουθίας και προσθέτουμε μεθόδους στα domain objects. Από αυτό το στάδιο, τα domain objects είναι πραγματικές domain κλάσεις, ή οντότητες, και το domain model πρέπει να γίνει γρήγορα ένα στατικό μοντέλο, ή διάγραμμα κλάσης - ένα κρίσιμο μέρος του λεπτομερούς σχεδιασμού σας.

iii. **Καθαρισμός του στατικού μοντέλου.**

iv. **Ορόσημο 3: Κρίσιμη Επισκόπηση Σχεδιασμού (Critical Design Review, CDR).**

3.2.1.4 Υλοποίηση (Implementation)

i. **Κωδικοποίηση/έλεγχος μονάδων (unit testing).** Γράφουμε τον κώδικα και τις μονάδες ελέγχου. (Ή, ανάλογα με τις προτιμήσεις μας, γράφουμε τις μονάδες ελέγχου και έπειτα τον κώδικα).

ii. **Ολοκλήρωση και έλεγχος σεναρίου.** Βασίζουμε τους ελέγχους ολοκλήρωσης πάνω στις περιπτώσεις χρήσης, ώστε να ελέγχουμε τόσο το βασικό σενάριο όσο και τα εναλλακτικά.

iii. Εκτελούμε επανεξέταση του κώδικα και μία ενημέρωση του μοντέλου για να ετοιμαστούμε για τον επόμενο κύκλο ανάπτυξης.

3.3 Χρήση της ICONIX για την ανάπτυξη της εφαρμογής μας

Στην εκπαιδευτική εφαρμογή «Online Τουριστικό Γραφείο» που αναπτύξαμε χρησιμοποιήσαμε τη μέθοδο ICONIX έναντι της RUP για τους κάτωθι λόγους:

- Η μεθοδολογία ICONIX είναι πιο «ελαφριά» σαν μέθοδος απ' ότι την RUP.
- Η ICONIX χρησιμοποιεί μόνο τέσσερα διαγράμματα της UML σε μία σύντομη διαδικασία που φτάνει αμέσως από τις απαιτήσεις στον κώδικα. Επομένως κλιμακώνεται γρηγορότερα από την RUP.

- Η ICONIX μεθοδολογία χρησιμοποιεί ισχυρούς κανόνες στη διάρκεια της συλλογής και ανακάλυψης απαιτήσεων, ώστε το σύστημα να βασίζεται από την αρχή σε σαφής και ξεκάθαρες απαιτήσεις που είναι υλοποιήσιμες και δεν θέτουν σε κίνδυνο την μετέπειτα πορεία της εφαρμογής (π.χ. η διαπίστωση ότι τελικά οι πελάτες θέλανε κάτι άλλο και οι προγραμματιστές λόγω λανθασμένων και ελλιπών απαιτήσεων απέδωσαν λανθασμένη λειτουργικότητα στο σύστημα).
- Η μεθοδολογία ICONIX παρέχει επαρκή τεκμηρίωση απαιτήσεων και σχεδιασμού ώστε να αποφεύγεται η γνωστή περίπτωση της παράλυσης από ανάλυση (analysis paralysis) (αναφέρεται λεπτομερώς παρακάτω).
- Τα διαγράμματα ευρωστίας που χρησιμοποιεί η μεθοδολογία ICONIX για την μετάβαση από την ανάλυση στο σχεδιασμό είναι τα πιο κατάλληλα από οποιαδήποτε άλλα διαγράμματα.
- Η ICONIX είναι πιο ευέλικτη και αποτελεσματική σε μικρές εφαρμογές, που οι απαιτήσεις είναι σχετικά ξεκάθαρες από την αρχή και είναι πλήρως αποτελεσματική και ιδανική σε εκπαιδευτικές εφαρμογές όπως η δική μας. Η μεθοδολογία RUP από την άλλη πλευρά, είναι ιδανική σε μεγάλες και πολύπλοκες εφαρμογές όπου δίνει τη δυνατότητα της προσαρμογής ανάλογα με το εκάστοτε σύστημα. Επίσης η RUP αν και περισσότερο πολύπλοκη σαν διαδικασία εγγυάται ένα σωστό σχεδιασμό της εφαρμογής μας, όπου είναι πιο εύκολα να αναδιαμορφωθεί ανά πάσα στιγμή.

3.4 Ανάλυση Παράλυση (Analysis Paralysis)

Ο όρος «ανάλυση παράλυση» ή «παράλυση της ανάλυσης" αναφέρεται σε υπερ-ανάλυση (ή παραπάνω σκέψεις) μιας κατάστασης, έτσι ώστε μια απόφαση ή ενέργεια δεν λαμβάνεται ποτέ, παραλύοντας το αποτέλεσμα. Μία απόφαση μπορεί να θεωρηθεί ως υπερβολικά περίπλοκη, με πάρα πολλές λεπτομερείς επιλογές, έτσι ώστε η επιλογή δεν πραγματοποιείται ποτέ, αντί να δοκιμαστεί κάτι και να αλλάξει αν το πρόβλημα είναι σημαντικό. Ένα άτομο μπορεί να επιδιώκει τη βέλτιστη ή "τέλεια" αρχική λύση, και να φοβάται να αναλάβει οποιαδήποτε

απόφαση που θα μπορούσε να οδηγήσει σε λανθασμένα αποτελέσματα, όσο βρίσκεται στο δρόμο για μία καλύτερη λύση.

Στην ανάπτυξη λογισμικού, η ανάλυση παράλυση εκδηλώνεται συνήθως με υπερβολικά μεγάλες φάσεις σχεδιασμού του έργου, της συλλογής απαιτήσεων, του σχεδιασμού του προγράμματος και της μοντελοποίησης δεδομένων, με τη δημιουργία μικρής ή καθόλου πρόσθετης αξίας από αυτά τα βήματα. Όταν επεκταθούν υπερβολικά στο χρονικό πλαίσιο, τέτοιες διαδικασίες έχουν την τάση να τονίζουν την οργανωτική (δηλαδή, γραφειοκρατική) πτυχή του λογισμικού, ενώ αποκλίνουν από τη λειτουργική πτυχή του (δημιουργία προστιθέμενης αξίας).

Η ανάλυση παράλυση συχνά συμβαίνει λόγω της έλλειψης εμπειρίας εκ μέρους των αναλυτών επιχειρηματικών συστημάτων, των διαχειριστών έργων ή των προγραμματιστών, καθώς και της έλλειψης μιας αυστηρής και τυπικής οργανωτικής κουλτούρας.

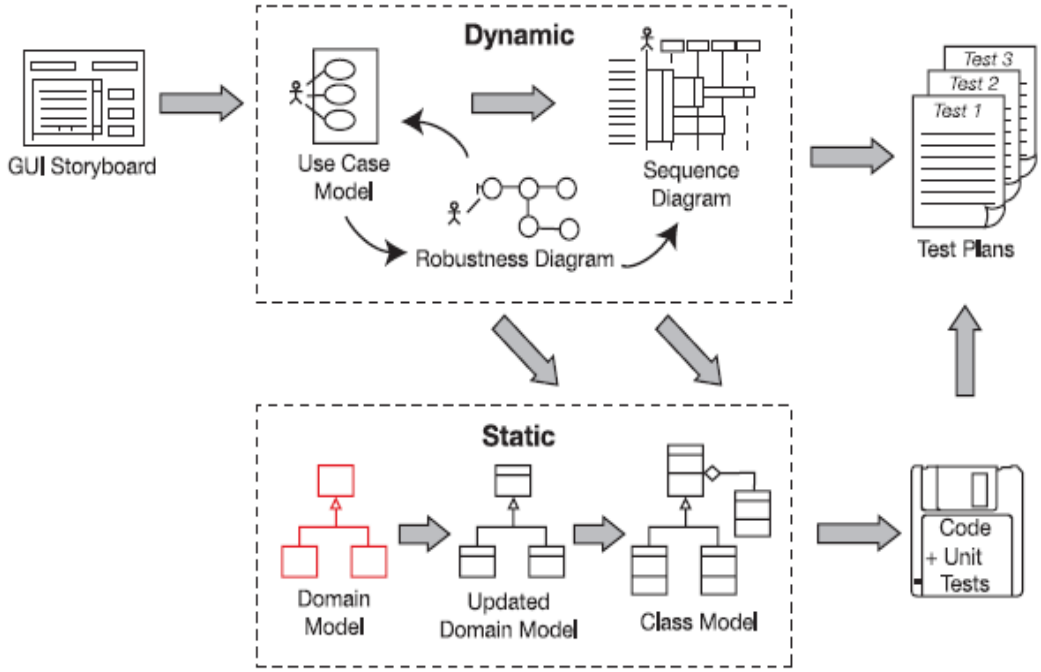
Η ανάλυση παράλυση είναι ένα παράδειγμα ενός αντι-πρότυπο. Οι ευέλικτες και γρήγορες (agile) μεθοδολογίες ανάπτυξης λογισμικού επιδιώκουν να αποφευχθεί η παράλυση της ανάλυση με την προώθηση ενός επαναληπτικού κύκλου εργασιών που δίνει έμφαση στην εργασία των προϊόντων πάνω από τις προδιαγραφές του προϊόντος.

3.5 Περίληψη

Σε αυτό το κεφάλαιο αναφερθήκαμε περιληπτικά στη μέθοδο ICONIX, στα βήματα που την απαρτίζουν και αναφέραμε γιατί επιλέξαμε τελικά αυτή τη μέθοδο για την ανάπτυξη του προγράμματός μας συγκριτικά με την RUP μεθοδολογία.

Στα επόμενα κεφάλαια ακολουθεί αναλυτική αναφορά των βημάτων της μεθόδου ICONIX και παραδείγματα από την εφαρμογή «Online Τουριστικό Γραφείο» που αναπτύξαμε.

Κεφάλαιο 4: Domain Modeling



Σχήμα 4.1 Μέθοδος ICONIX σε λεπτομερειακή ροή εργασίας (Domain Modeling)

Σε όλα σχεδόν τα έργα (projects) της πληροφορικής, το πρόβλημα της προβληματικής επικοινωνίας είναι ανεξέλεγκτο, αλλά σπάνια παρατηρείται, επειδή όλοι νομίζουν ότι μιλούν την ίδια γλώσσα. Αυτό όμως δεν ισχύει. Το ένα άτομο λέει "κριτική βιβλίου" και άλλα άτομα το ερμηνεύουν αυτό ως έννοια «συντακτική ανασκόπηση» (μια κριτική που γράφτηκε από μια εκδοτική ομάδα), ενώ άλλοι μπορεί να το ερμηνεύσουν υπό την έννοια της «αξιολόγησης του πελάτη» (μια κριτική που γράφτηκε από έναν πελάτη και δημοσιεύτηκε στο site). Τα αποτελέσματα μπορεί να είναι - και συχνά είναι - καταστροφικά δεδομένου ότι το σύστημα αναπτύσσεται ανάλογα με την ερμηνεία που δίνει ο καθένας στις διαφορετικές απαιτήσεις και το σχεδιασμό.

Το domain model είναι ένα ζωντανό αντικείμενο και ένα αντικείμενο συνεργασίας. Βελτιώνεται και αναπροσαρμόζεται σε όλη τη διάρκεια του έργου, ώστε να αντικατοπτρίζει πάντα την τρέχουσα αντίληψη του χώρου του προβλήματος.

Σε αυτό το κεφάλαιο θα εξετάσουμε το domain modeling, το οποίο αποσκοπεί στην επίλυση του προβλήματος της προβληματικής επικοινωνίας στα έργα με τη θέσπιση ενός κοινού λεξιλογίου που χαρτογραφεί το χώρο του προβλήματος.

4.1 Τι είναι το domain modeling;

Όπως μόλις αναφέραμε, ένα domain model είναι, στην ουσία, ένα γλωσσάρι του έργου: ένα «ζωντανό» λεξικό όλων των όρων που χρησιμοποιούνται στο έργο μας. Όμως, ένα domain model είναι καλύτερο από ένα γλωσσάρι του έργου, γιατί δείχνει γραφικά πώς όλοι αυτοί οι διαφορετικοί όροι σχετίζονται μεταξύ τους. Στην πράξη πρόκειται για ένα απλοποιημένο διάγραμμα κλάσης (class diagram), με γραμμές που σύρονται μεταξύ των διαφόρων κατηγοριών (αντικείμενα domain) για να δείξει τη μεταξύ τους συσχέτιση. Το μοντέλο domain δείχνει τις σχέσεις της συνάθροισης (aggregation) και της γενίκευσης (generalization) (έχει-ένα και είναι-ένα σχέσεις) μεταξύ των domain κλάσεων.

4.2 Γιατί να ξεκινήσουμε από το domain model αντί των περιπτώσεων χρήσης

Βοηθάει να κάνουμε ένα γρήγορο πέρασμα από το domain model στην αρχή του έργου μας. Όταν γράφουμε περιπτώσεις χρήσης, είναι δελεαστικό να τις γράψουμε αφηρημένες, υψηλού επιπέδου, αόριστες και ασαφείς. Στην πραγματικότητα, το κείμενό των περιπτώσεων χρήσης θα πρέπει να στηρίζεται στην πραγματικότητα, και θα πρέπει να είναι πολύ κοντά στο σύστημα που θα σχεδιάσουμε. Με άλλα λόγια, οι περιπτώσεις χρήσης θα πρέπει να γράφονται στα πλαίσια του μοντέλου αντικειμένων (object model) (δηλαδή, το κείμενο των περιπτώσεων χρήσης πρέπει να αναφέρεται στα domain objects με το όνομά τους). Με τον τρόπο αυτό, θα είμαστε σε θέση να συνδέσουμε μαζί τα στατικά και δυναμικά μέρη του μοντέλου, το οποίο είναι ζωτικής σημασίας εάν θέλουμε η προσπάθεια της ανάλυσης και του σχεδιασμού να προωθείτε από τις περιπτώσεις χρήσης μας.

Έτσι προτού να γράψουμε τις περιπτώσεις χρήσης μας, πρέπει να καταλήξουμε σε μια πρώτη προσπάθεια δημιουργίας του domain model. Το

domain model αποτελεί το θεμέλιο του στατικού μέρους του μοντέλου μας, ενώ οι περιπτώσεις χρήσης είναι το θεμέλιο του δυναμικού μέρους. Το στατικό τμήμα περιγράφει τη διάρθρωση, ενώ το δυναμικό μέρος περιγράφει τη συμπεριφορά.

4.3 Οδηγίες Μοντελοποίησης

Οι αρχές που αναπτύχθηκαν σε αυτό το κεφάλαιο συνοψίζονται σε μία λίστα με οδηγίες.

1. **Εστίαση σε πραγματικά Αντικείμενα.** Κατά τη δημιουργία ενός domain model, πρέπει να επικεντρωθούμε στον πραγματικό κόσμο των αντικειμένων μέσα στα πλαίσια του χώρου προβλήματος. Προσπαθούμε να οργανώσουμε την αρχιτεκτονική λογισμικού μας γύρω από το πώς μοιάζει ο πραγματικός κόσμος. Ο πραγματικός κόσμος έχει την τάση να αλλάζει λιγότερο συχνά από τις απαιτήσεις του λογισμικού.
2. **Χρησιμοποίηση των Σχέσεων Γενίκευσης (Είναι-ένα) και Συνάθροισης (Έχει-ένα).** Κατά τη διάρκεια του χρόνου, θα συμπληρώνουμε το domain μοντέλο μας με νέες κλάσεις domain, όπως και όταν τις εντοπίζουμε. Θα παρατηρήσουμε επίσης ότι υπάρχουν σχέσεις (ή τις ενώσεις) μεταξύ τους-για παράδειγμα, μια κριτική Ξενοδοχείου ανήκει σε ένα Ξενοδοχείο και μια Εντολή Αγοράς και οι Πιστωτικές Κάρτες είναι και οι δύο από το ίδιο είδος, αφού και οι δύο είναι Τρόποι Πληρωμών. Η πρώτη σχέση (Κριτική Ξενοδοχείου – Ξενοδοχείο) ονομάζεται συνάθροιση (aggregation) (έχει-ένα, γιατί ένα Ξενοδοχείο έχει μία Κριτική Ξενοδοχείου). Η δεύτερη σχέση ονομάζεται γενίκευση (generalization) (είναι-ένα, γιατί η Εντολή Αγοράς και η Πιστωτική Κάρτα είναι και οι δύο Τύποι Πληρωμής). Αυτές οι δύο και η απλή σχέση είναι οι πιο κοινές σχέσεις που θα χρησιμοποιήσουμε στο domain μοντέλο μας.
3. **Θέτουμε χρονικό όριο μερικών ορών στη μοντελοποίηση του αρχικού domain model.** Συνίσταται η καθιέρωση χρονικού ορίου για την κατασκευή του αρχικού domain model. Μερικές ώρες είναι όσες χρειαζόμαστε. Δεν χρειάζεται να γίνει τέλειο έτσι και αλλιώς, γι' αυτό

συνίσταται να γίνεται γρήγορα και περιμένουμε ότι θα το διορθώσουμε στη συνέχεια. Θα ανακαλύψουμε αντικείμενα που λείπουν καθώς θα δουλεύουμε τις περιπτώσεις χρήσης και τα διαγράμματα ευρωστίας.

4. **Οργανώνουμε τις κλάσεις μας γύρω από κύριες Αφαιρέσεις (Abstractions) στο χώρο του προβλήματος.** Είναι γενικά σωστή πρακτική να οργανώνουμε τις κλάσεις μας γύρω από κύριες αφαιρετικές ενότητες στο χώρο του προβλήματος. Πρέπει να θυμόμαστε ότι το domain model είναι το πρώτο διάγραμμα κλάσης που γίνεται το θεμέλιο της αρχιτεκτονικής λογισμικού μας. Αυτό κάνει το μοντέλο πιο ανθεκτικό στην αντιμετώπιση των αλλαγών. Οργανώνοντας την αρχιτεκτονική γύρω από αφαιρέσεις του πραγματικού κόσμου κάνει το μοντέλο πιο ανθεκτικό στην αντιμετώπιση των μεταβαλλόμενων απαιτήσεων, καθώς οι απαιτήσεις συνήθως αλλάζουν συχνότερα από ό, τι στον πραγματικό κόσμο.
5. **Δεν πρέπει να μπερδεύουμε το Domain Model με το Data Model.** Ακόμα κι αν τα διαγράμματα μπορεί να μοιάζουν, πρέπει να θυμόμαστε ότι αυτό που μπορεί να είναι μία καλή πρακτική σε ένα μοντέλο δεδομένων (data model) είναι πιθανό να μην είναι καλή πρακτική σε ένα διάγραμμα κλάσης (class diagram) (και αντιστρόφως). Οι κλάσεις είναι μικρές και οι πίνακες μεγάλοι. Ένας πίνακας σε μια σχεσιακή βάση δεδομένων σχετίζεται συχνά με μια σειρά από πράγματα. Αντίθετα, οι κλάσεις είναι καλύτερα σχεδιασμένες αν αποτελούν σχετικά μικρά πακέτα δεδομένων και συμπεριφοράς. Σε ένα διάγραμμα κλάσης, είναι πιθανό ότι θα έχουμε μια κλάση που διαχειρίζεται έναν πίνακα βάσης δεδομένων και μπορεί να βρούμε και κάποια κλάση TableManager. Ο σκοπός αυτών των κλάσεων τύπου TableManager είναι να αποκρύπτουν τις λεπτομέρειες του συστήματος διαχείρισης βάσης δεδομένων (DBMS) από τον υπόλοιπο κώδικα.
6. **Δεν πρέπει να μπερδεύουμε ένα αντικείμενο με ένα πίνακα βάσης.** Ένα αντικείμενο αντιπροσωπεύει ένα μόνο παράδειγμα από κάτι. Ένας πίνακας δεδομένων αποτελεί μια συλλογή από τα

πράγματα. Δεν χρειάζεται να είμαστε τόσο κυριολεκτικοί όπως στον κόσμο του Enterprise JavaBeans (EJB), όπου μία οντότητα bean αντιπροσωπεύει, συνήθως, μια ολόκληρη γραμμή σε ένα πίνακα. Οι κλάσεις domain είναι παρόμοιες. Αν πούμε μία κλάση Ξενοδοχείο, τότε δεν σημαίνει ένας πίνακας Ξενοδοχείο - εννοείτε ένα μόνο Ξενοδοχείο. Οι στήλες σε ένα πίνακα γενικά παρουσιάζονται σαν ορίσματα (attributes) σε μία κλάση. Όμως, συνήθως οι πίνακες σε μία βάση περιλαμβάνουν περισσότερες στήλες από τα ορίσματα μία κλάσης (για παράδειγμα, οι πίνακες συνήθως έχουν ξένα κλειδιά). Έτσι υπάρχει περίπτωση να μην υπάρχει ένα-προς-ένα σχέση μεταξύ των στηλών ενός πίνακα και της αντίστοιχης κλάσης στον κώδικα.

7. **Χρησιμοποίηση του domain μοντέλου ως γλωσσάρι για το έργο μας.** Αν οι ασαφείς απαιτήσεις είναι ο εχθρός, το domain μοντέλο είναι η πρώτη γραμμή άμυνας. Ασαφής χρήση των ονομάτων από "ειδικούς θεμάτων" είναι πολύ συχνό και επιβλαβές φαινόμενο. Το domain μοντέλο θα πρέπει να χρησιμεύσει ως ένα γλωσσάρι του έργου που βοηθά να εξασφαλιστεί η συνεπής χρήση των όρων κατά την περιγραφή του χώρου προβλημάτων. Χρησιμοποιώντας το domain model ως ένα γλωσσάριο του έργου είναι το πρώτο βήμα προς την αποσαφήνιση του μοντέλου μας.
8. **Γράφουμε το domain model πριν τις περιπτώσεις χρήσης.** Επειδή χρησιμοποιούμε το domain model για να αποσαφηνίσουμε τα προβλήματα των domain αφαιρέσεων θα ήταν κουτό να γράψουμε πρώτα τις λεπτομερείς περιπτώσεις χρήσης χρησιμοποιώντας ασαφείς όρους για να περιγράψουμε τις domain κλάσεις. Έτσι είναι καλύτερο να ξοδέψουμε μερικές ώρες ξεκινώντας από το domain model. Το να γράψουμε τις περιπτώσεις χρήσης χωρίς ένα μοντέλο domain να ενώνει τα πάντα μεταξύ τους αποθηκεύει πολλά προβλήματα για αργότερα.
9. **Δεν περιμένουμε το τελικό Διάγραμμα Κλάσης (Class Diagram) να μοιάζει με το Domain Μοντέλο.** Τα διαγράμματα κλάσης θα γίνουν πολύ πιο λεπτομερή από το μοντέλο domain, όπως ο

σχεδιασμός προχωράει, το domain μοντέλο σκόπιμα διατηρείται αρκετά απλό. Καθώς σχεδιάζουμε (χρησιμοποιώντας διαγράμματα ακολουθίας), ο λεπτομερής σχεδιασμός προσθέτει για παράδειγμα βοηθούς GUI, constructor κλάσεις και υποδομής προστίθεται στο διάγραμμα κλάσης, καθώς το διάγραμμα του domain μοντέλου θα είναι σχεδόν σίγουρα χωρισμένο σε πολλά λεπτομερή διαγράμματα κλάσεων. Ωστόσο, θα πρέπει ακόμη να είναι δυνατόν να εντοπίσουμε τις περισσότερες κλάσεις του διαγράμματος κλάσης στο domain μοντέλο.

10. **Δεν τοποθετούμε οθόνες και άλλες GUI κλάσεις στο Domain μοντέλο.** Κάνοντάς το μπορεί αν ανοίξουν οι ασκοί του Αιόλου σε ένα κορεσμένο domain model που περιλαμβάνει πολλές λεπτομέρειες υλοποίησης. Κλάσεις απόδοσης βελτιστοποίησης και βοηθητικές κλάσεις θα πρέπει να λείπουν επίσης από το domain model. Πρέπει να περιλαμβάνει μόνο κλάσεις που αναφέρονται στο χώρο του προβλήματος.

4.4 Domain Model στην πράξη

Στη συνέχεια θα δούμε τα βήματα που ακολουθήσαμε, σύμφωνα με τις παραπάνω οδηγίες, για την ανάπτυξη της εφαρμογής μας «Online Τουριστικό Γραφείο».

Καθώς σχεδιάζουμε το domain model, μία καλή πηγή domain κλάσεων είναι το κείμενο των απαιτήσεων του συστήματος – απαιτήσεις που είναι συνήθως γραμμένες ως εξής «το σύστημα πρέπει να κάνει αυτό, το σύστημα δεν πρέπει να κάνει εκείνο». Είναι χρήσιμο να ανιχνεύσουμε στο κείμενο αυτό τα αντικείμενα και τις φράσεις αντικειμένων. Αργότερα θα ραφινάρουμε αυτές τις κλάσεις δημιουργώντας το domain model.

Έχοντας αυτά στο νου μας προχωράμε στην εκμαίευση αντικειμένων από ένα απόσπασμα της περιγραφής του συστήματος Όνειρο. Το σύστημα Όνειρο περιγράφει την εκπαιδευτική μας εφαρμογή «Online Τουριστικό Γραφείο» και το απόσπασμα που βλέπουμε παρακάτω είναι η περιγραφή της κράτησης δωματίου

σε ξενοδοχειακή μονάδα. Ολόκληρη η περιγραφή συστήματος μπορείτε να την βρείτε στο Παράρτημα Α.

- Ο χρήστης-πελάτης εισάγει τον προορισμό που θέλει να κάνει κράτηση δωματίου καθώς και την ημερομηνία άφιξης και αναχώρησης. Κατόπιν επιλέγεται ο αριθμός δωματίων που θέλει να κάνει κράτηση καθώς και πόσοι ενήλικες, αλλά και παιδιά, αν υπάρχουν, θα φιλοξενηθούν σε κάθε ένα δωμάτιο.
- Το σύστημα θα δώσει αποτελέσματα σύμφωνα με τις προτιμήσεις του χρήστη και θα εμφανίσει τα αντίστοιχα ξενοδοχεία ταξινομημένα με βάση την τιμή, γράφοντας τον τύπο του/ων δωματίου/ων και την τιμή. Επιπλέον ο χρήστης μπορεί να επιλέξει αν θέλει παροχή μόνο πρωινού ή ημιδιατροφής(παροχή πρωινού και μεσημεριανού) ή πλήρης διατροφή (πρωινό, μεσημεριανό και βραδινό) με διαφορά στη τιμή που θα αναγράφεται. Ο χρήστης-πελάτης επιλέγει το/τα επιθυμητό/τά δωμάτιο/α και κάνει κλικ στο κουμπί Κράτηση.
- Στη συνέχεια ο χρήστης-πελάτης θα πρέπει να εισάγει τα στοιχεία του κάθε επισκέπτη σε κάθε δωμάτιο, όπως αναγράφονται στα επίσημα έγγραφα του καθενός, ώστε να μπορέσουν να εκδοθεί η ακριβής κράτηση του ξενοδοχείου.
- Στη συνέχεια, εμφανίζεται η καρτέλα κράτηση στην οποία ο χρήστης-πελάτης πρέπει να συμπληρώσει κάποια προσωπικά του στοιχεία για να θεωρείται έγκυρη η κράτηση. Τα στοιχεία που πρέπει να συμπληρώσει είναι το ονοματεπώνυμο, με λατινικούς χαρακτήρες, του ατόμου που συμμετέχει στην κράτηση και το γένος του, άνδρας ή γυναίκα. Επιπλέον το ονοματεπώνυμο και τα στοιχεία της διεύθυνσης του ατόμου που διεξάγει την κράτηση καθώς και ένας αριθμός τηλεφώνου και μία διεύθυνση e-mail. Τέλος τα στοιχεία της πιστωτικής κάρτας με την οποία θα γίνει πληρωμή όπως τύπος, ονοματεπώνυμο κατόχου, αριθμός κάρτας και CVC/CVV και ημερομηνία λήξης της κάρτας. Συλλέγονται ακόμα και κάποια πρόσθετα στοιχεία (όπως αν ο πελάτης επιθυμεί τιμολόγιο και επίσης ενημερώνεται για την πολιτική της ακύρωσης της

κράτησης), τα οποία όμως δεν είναι υποχρεωτικά για την διεξαγωγή μίας επιτυχής κράτησης.

- Τέλος επιβεβαιώνονται τα στοιχεία από το σύστημα της τράπεζας, καταχωρείται η κράτηση στο σύστημά μας, αν είναι έγκυρη, και εμφανίζουμε το πιστοποιητικό της κράτησης και ταυτόχρονα στέλνεται και στον πελάτη με mail στη διεύθυνση του ηλεκτρονικού ταχυδρομείου που έχει ορίσει. Οι κρατήσεις καταγράφονται σε μία λίστα κρατήσεων στη βάση δεδομένων του συστήματος.

Επιπλέον περιορισμοί (Requirement constraints):

- Ο αριθμός των δωματίων που μπορεί να εισάγει ο κάθε χρήστης για κάθε κράτηση είναι το πολύ 4. Επίσης ο αριθμός των ατόμων που μπορεί να φιλοξενηθούν σε ένα δωμάτιο είναι το πολύ 4.
- Τα υποχρεωτικά πεδία συμπλήρωσης στην online φόρμα προσωπικών στοιχείων του πελάτη πρέπει να είναι ευδιάκριτα.
- Ο χρήστης-πελάτης δεν μπορεί να εισάγει μη έγκυρα στοιχεία στις φόρμες. Πρέπει να γίνεται έλεγχος των πεδίων πριν τη καταχώρησή τους.

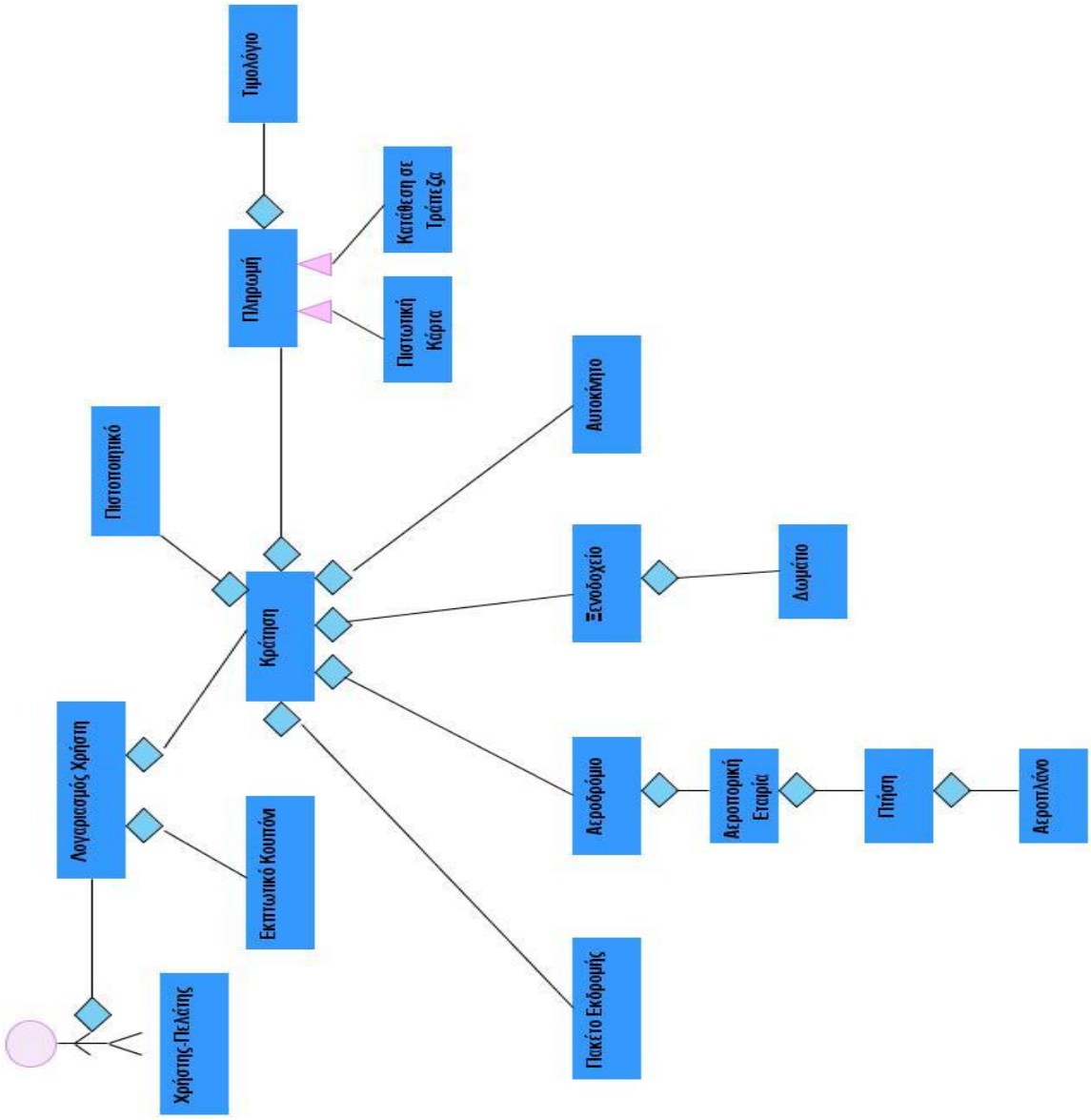
Αυτές οι απαιτήσεις είναι πλούσια πηγή domain κλάσεων. Τα στοιχεία που έχουν επισημανθεί τα τοποθετούμε σε μία λίστα όπου **διαγράφουμε τις διπλοεγγραφές:** υπάρχει αρκετό κομμάτι επικάλυψης στη λίστα που δημιουργήθηκε. Παρόμοιοι όροι χρησιμοποιούνται για το ίδιο πράγμα. Αλλά αυτό αποτελεί και το κυρίως όφελος του domain modeling, εντοπίζουμε και εξαλείφουμε αυτές τις διπλοεγγραφές στην αρχή του έργου. Η ολοκληρωμένη λίστα των αντικειμένων βρίσκεται στο Παράρτημα Β.

Αφού ολοκληρώσουμε το ραφινάρισμα της λίστας μας, προχωράμε στο σχεδιασμό του domain model μας, τοποθετούμε όλες μας τις κλάσεις και σχεδιάζουμε τις μεταξύ τους σχέσεις, στις οποίες πρέπει να καθορίσουμε τις σχέσεις γενίκευσης και τη συνάθροισης. Μία χρήσιμη τεχνική είναι να διαβάσουμε τις απαιτήσεις από την αρχή χρησιμοποιώντας τις λέξεις έχει ένα και είναι ένα. Για παράδειγμα το Αεροδρόμιο έχει μία Αεροπορική Εταιρία ενώ η Πληρωμή με Κάρτα είναι μία Πληρωμή.

Στο σχήμα 4.2 φαίνεται το domain model του συστήματος Όνειρο.

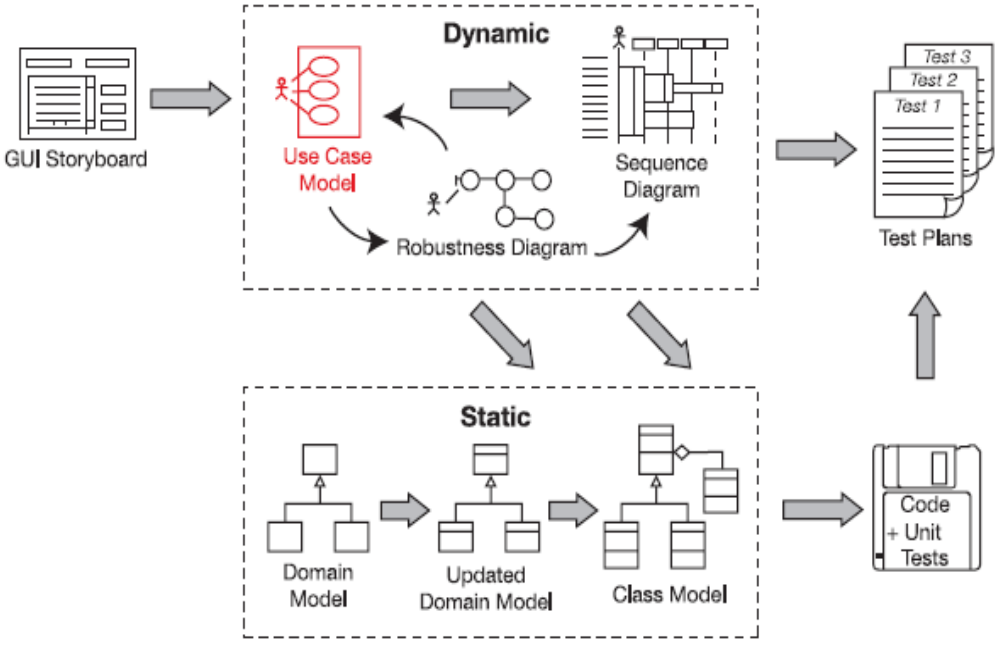
4.5 Περίληψη

Στο κεφάλαιο αυτό περιγράφεται αναλυτικά η πρώτη μεγάλη φάση της ICONIX διαδικασίας. Η Domain μοντελοποίηση αποτελεί τη βάση για τη συνολική δραστηριότητα μοντελοποίησης αντικειμένων. Όπως θα δούμε στο επόμενο κεφάλαιο, οι περιπτώσεις χρήσης είναι γραμμένες στο πλαίσιο του domain μοντέλου, και (όπως θα δούμε στο Κεφάλαιο 7) η ανάλυση ευρωστίας βοηθάει στο δέσιμο του domain model και των περιπτώσεων χρήσης ώστε να έρθουν πιο κοντά.



Σχήμα 4.2 Domain Model του συστήματος Όνειρο

Κεφάλαιο 5: Μοντελοποίηση Περιπτώσεων Χρήσης



Σχήμα 5.1 Μέθοδος ICONIX σε λεπτομερειακή ροή εργασίας (Use Case Modeling)

Έχοντας ένα αρχικό domain μοντέλο, είναι καιρός να αρχίσουμε τις περιπτώσεις χρήσης. Οι περιπτώσεις χρήσης μας δίνουν ένα δομημένο τρόπο να αντιληφθούμε τις απαιτήσεις συμπεριφοράς (behavioral requirements) του συστήματος, έτσι ώστε να μπορέσουμε να δημιουργήσουμε λογικά ένα σχέδιο από αυτές. Θα μας βοηθήσει να απαντήσουμε σε ορισμένα θεμελιώδη ερωτήματα: Τι προσπαθούν να κάνουν οι χρήστες του συστήματος; Ποια είναι η εμπειρία του χρήστη; Ένα εκπληκτικά μεγάλο μέρος του τι κάνει το λογισμικό υπαγορεύεται από τον τρόπο με τον οποίο οι χρήστες πρέπει να αλληλεπιδρούν με αυτό.

5.1 Γιατί χρειαζόμαστε τις περιπτώσεις χρήσεις πέραν των λειτουργικών απαιτήσεων;

Οι λειτουργικές απαιτήσεις (functional requirements) τείνουν να είναι ένα μείγμα υψηλού και χαμηλού επιπέδου απαιτήσεων, ουσιαστικά ένα ρεύμα της συνειδητοποίησης από τους διαχειριστές, πελάτες και την ομάδα μάρκετινγκ, που

συλλαμβάνεται σε σειριακή μορφή και τοποθετείται σε ένα έγγραφο του Word. Όχι ότι υπάρχει κάτι λάθος με αυτό, είναι απλώς το πρώτο, πρόωρο βήμα στο δρόμο για να πάρουμε ένα οριστικοποιημένο, σαφή, ξεκάθαρο σύνολο απαιτήσεων συμπεριφοράς από το οποίο μπορούμε να δημιουργήσουμε ρεαλιστικά ένα σχέδιο.

Οι Λειτουργικές προδιαγραφές είναι μεν σημαντικές, φυσικά. Αλλά το να σχεδιάζουμε, κωδικοποιούμε, ή εκτιμούμε, απευθείας από μια λειτουργική προδιαγραφή είναι σαν να παίζουν ένα συναρπαστικό παιχνίδι "διάλεξε έναν τυχαίο αριθμό." Θα χρειαστεί να κάνουμε κάποια διερευνητική εργασία για να εξομαλύνουμε τους όρους. Η μοντελοποίηση των περιπτώσεων χρήσης και η προκαταρκτική μελέτη μετά, μας οδηγούν εκεί.

5.2 Οδηγίες Μοντελοποίησης περιπτώσεων χρήσης

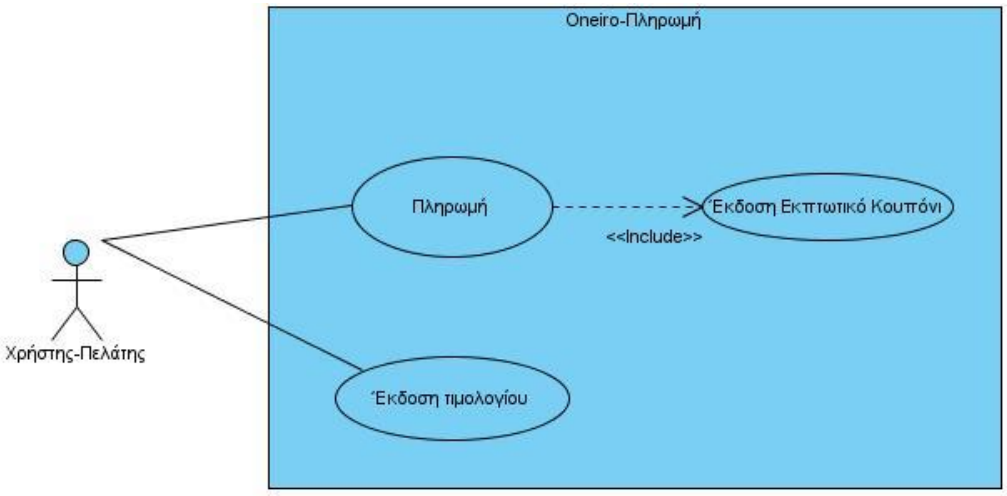
Οι αρχές που αναλύονται σε αυτό το κεφάλαιο συνοψίζονται σε μία λίστα με οδηγίες. Αυτές οι κατευθυντήριες γραμμές μπορούν με την σειρά τους να συνοψιστούν σε μία φράση:

Περιγράφουμε τη χρήση του συστήματος στο πλαίσιο του μοντέλου αντικειμένου (object model).

Τα σημεία από το 1 έως το 6 περιγράφουν τη χρήση του συστήματος, ενώ από το 7 έως το 10 ασχολούνται με την τοποθέτηση της χρήσης του συστήματος στο πλαίσιο του μοντέλου αντικειμένου.

1. **Ακολουθούμε τον κανόνα των δύο παραγράφων.** Κάθε περίπτωση χρήσης θα πρέπει να χωρά άνετα σε δύο παραγράφους, συμπεριλαμβανομένων τόσο του βασικού σεναρίου και των εναλλακτικών σεναρίων. Οτιδήποτε πολύ περισσότερο από δύο παραγράφους, θα οδηγήσει σε ακατανόητα διαγράμματα ακολουθίας. Εάν η περίπτωση χρήσης πηγαίνει πάνω από δύο παραγράφους, θα πρέπει πιθανώς να διαιρεθεί σε δύο ή περισσότερες ξεχωριστές περιπτώσεις χρήσης.

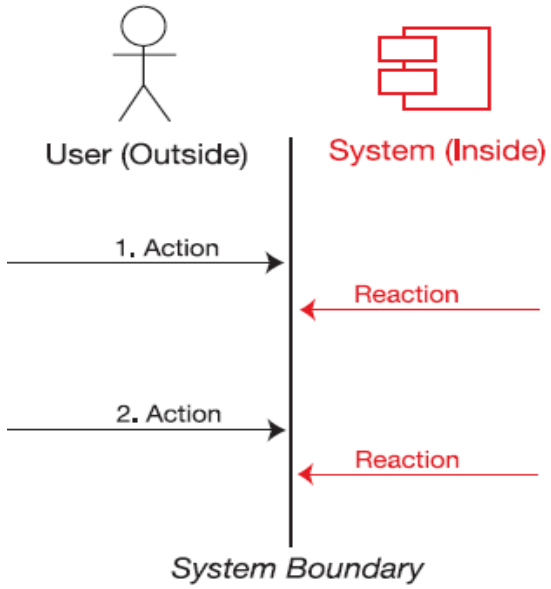
2. **Οργανώνουμε τις περιπτώσεις χρήσης μας με ρόλους (actors) και διαγράμματα περιπτώσεων χρήσης.** Ένα διάγραμμα περιπτώσεως χρήσης δείχνει πλήθος περιπτώσεων χρήσης. Πρόκειται για μια επισκόπηση μιας συσχετιζόμενης ομάδας περιπτώσεων χρήσης. Το κείμενο σε κάθε οβάλ είναι ο τίτλος της περίπτωσης χρήσης. Για παράδειγμα το σχήμα 5.2 δείχνει ένα παράδειγμα διαγράμματος περιπτώσεως χρήσης που αναπαριστά την λογική ομάδα Πληρωμή του συστήματος «Online Τουριστικό Γραφείο». Ένας χαρακτήρας (actor) αντιπροσωπεύεται στο διάγραμμα ως ανθρωπάκι σκαρίφημα (stick figure) και είναι ανάλογος με ένα «ρόλο» που οι χρήστες μπορούν να παίξουν. Μερικές φορές ο χαρακτήρας ονομάζεται απλώς "Χρήστης", αλλά δίνεται συχνά ένα συγκεκριμένο όνομα ρόλου. Ο χρήστης (χαρακτήρας) είναι εκτός του συστήματος, αυτός ή αυτή είναι στην "εξωτερική" πλευρά, ενώ το σύστημα βρίσκεται στο «εσωτερικό». Οι χαρακτήρες μπορεί να αντιπροσωπεύουν μη ανθρώπινα εξωτερικά συστήματα καθώς και ανθρώπους. Μια ένωση από τον χαρακτήρα σε μία περίπτωση χρήσης σημαίνει ότι αυτός ο χαρακτήρας είναι που εκτελεί αυτή την περίπτωση χρήσης. Η ένωση μπορεί να σημαίνει επίσης ευθύνες. Μπορούμε να έχουμε περισσότερους από έναν χαρακτήρες να δείχνουν σε μία περίπτωση χρήσης που σημαίνει ότι η περίπτωση χρήσης συνδέεται με περισσότερους από έναν ρόλους. Ομοίως ένας χρήστης μπορεί να διαθέτει πολλαπλούς ρόλους, για παράδειγμα ο ίδιος χρήστης μπορεί να είναι Ξενοδοχοϋπάλληλος και Διαχειριστής του συστήματος.



Σχήμα 5.2 Διάγραμμα περιπτώσεως χρήσης Πληρωμή

3. **Γράφουμε τις περιπτώσεις χρήσης σε ενεργητική φωνή.** Ο όρος έχει να κάνει με την σκοπιά από την οποία γράφουμε την πρόταση. Όταν γράφουμε σε παθητική φωνή, περιγράφουμε το σύστημα χωρίς να δίνουμε έμφαση – συνήθως δεν αναφέρεται καθόλου – στο ποιός ή τι εκτελεί τις ενέργειες. Για παράδειγμα : *Η δυνατότητα αυτή παρέχεται στους χρήστες για εισαγωγή στο σύστημα, χρησιμοποιώντας ένα κωδικό πρόσβασης.* Από την πρόταση φαίνεται ότι η δυνατότητα έχει ήδη δοθεί στους χρήστες, οπότε δε χρειάζεται να ανησυχούμε για αυτήν. Δυστυχώς αυτή μπορεί να είναι ή να μην είναι η υπόθεση (μπορεί να χρειάζεται να δημιουργήσουμε αυτή τη δυνατότητα). Το πρόβλημα με την παθητική φωνή είναι ότι κρύβει τους χαρακτήρες και, πιο σημαντικό, τις λειτουργίες του λογισμικού. Η φράση που χρησιμοποιήσαμε στο παράδειγμα ακούγεται επίσης σαν μία λειτουργική απαίτηση. Η παθητική φωνή είναι συχνά ασαφής και στερείται ενέργειας. Η ενεργητική φωνή καθιστά σαφές το ποιος κάνει τι. Εδώ είναι ένα παράδειγμα περιπτώσεως χρήσης σε ενεργητική φωνή: *Ο χρήστης εισάγει το όνομα και τον κωδικό πρόσβασης και πατάει το κουμπί Σύνδεση.*
4. **Γράφουμε τις περιπτώσεις χρήσης χρησιμοποιώντας μία ροή Γεγονός/Απάντηση.** Μια περίπτωση χρήσης συχνά ενεργοποιείται από το γεγονός που ξεκίνησε ένας χρήστης και στο οποίο το σύστημα

πρέπει να ανταποκριθεί. Ωστόσο, μπορεί επίσης να προκληθεί από ένα σύστημα που έχει ξεκινήσει ένα γεγονός με το οποίο ο χρήστης αλληλεπιδρά. Αλλά σε κάθε περίπτωση, η περίπτωση χρήσης ακολουθεί την ροή γεγονός/απάντηση (βλέπε σχήμα 5.3).



Σχήμα 5.3 Ανατομία ενός σεναρίου περιπτώσεως χρήσης

5. **Χρησιμοποιούμε Storyboards, Πρωτότυπα GUI και μακέτες Οθονών.** Τα Storyboards, τα GUI πρωτότυπα, και οι μακέτες οθονών είναι συχνά πολύ χρήσιμα οπτικά βοηθήματα όταν γράφουμε μία περίπτωση χρήσης. Αν βασίσουμε την περίπτωση χρήσης σε ένα πρωτότυπο GUI, για παράδειγμα, είναι σημαντικό να συμπεριληφθούν όλα τα κουμπιά και τα μενού που ο χρήστης μπορεί να αγγίξει ώστε να δημιουργήσει γεγονότα, μέσα στην περίπτωση χρήσης.
6. **Θυμόμαστε ότι η περίπτωση χρήσης είναι πραγματικά μία προδιαγραφή συμπεριφοράς κατά τον χρόνο εκτέλεσης.** Με τη μεθοδολογία ICONIX, ο σχεδιασμός οδηγείται από τις περιπτώσεις χρήσης. Στην πράξη, αυτό σημαίνει ότι σχεδιάζουμε ένα διάγραμμα ακολουθίας για κάθε περίπτωση χρήσης στην τρέχουσα έκδοση. Το διάγραμμα ακολουθίας δείχνει με μεγάλη λεπτομέρεια πώς παραδείγματα αντικειμένων συνεργάζονται από κοινού κατά το χρόνο εκτέλεσης ώστε να ολοκληρωθεί η συμπεριφορά της περίπτωσης χρήσης. Ως εκ τούτου, το κείμενο της περίπτωσης χρήσης θα

χρησιμεύσει ως προδιαγραφή σχετικά με τη συμπεριφορά χρόνου εκτέλεσης που φαίνεται στα διαγράμματα ακολουθίας.

7. **Γράφουμε τις περιπτώσεις χρήσης μας στο πλαίσιο του Μοντέλου Αντικειμένων (Object Model).** Δεν μπορούμε να δημιουργήσουμε αντικειμενοστραφή σχέδια από τις περιπτώσεις χρήσης αν δεν σχετίσουμε τις περιπτώσεις χρήσεις με τα αντικείμενα. Στην πράξη, αυτό σημαίνει ότι θα πρέπει να αναφέρουμε τις domain κλάσεις που συμμετέχουν στην περίπτωση χρήσης, και θα πρέπει να ονομάσουμε τις οθόνες μας και τα άλλα boundary objects ρητά στο κείμενο των περιπτώσεων χρήσης. Διαφορετικά, οι απαιτήσεις συμπεριφοράς μας θα είναι πλήρως αποσυνδεδεμένες από το μοντέλο αντικειμένου μας, και δεν θα είμαστε σε θέση να πάρουμε τα σχέδια από τις περιπτώσεις χρήσης.
8. **Γράφουμε τις περιπτώσεις χρήσης χρησιμοποιώντας προτάσεις που έχουν τη δομή Αντικείμενο-Ρήμα-Αντικείμενο:** είναι πολύ πιο εύκολο να σχεδιάσουμε έναν αντικειμενοστραφή σχεδιασμό από προτάσεις που παρουσιάζουν αυτή τη δομή. Το κείμενο των περιπτώσεων χρήσης μας τελικά θα βρίσκεται στο πλαίσιο του διαγράμματος ακολουθίας. Και τα διαγράμματα ακολουθίας προσανατολίζονται θεμελιωδώς γύρω από τα ουσιαστικά και τα ρήματα:
    - Τα ουσιαστικά είναι οι περιπτώσεις χρήσεις. Αυτά συνήθως είτε προέρχονται από το domain μοντέλο (οντότητες) ή είναι boundary/ GUI αντικείμενα.
    - Τα ρήματα είναι τα μηνύματα μεταξύ των αντικειμένων. Αυτά αντιπροσωπεύουν τις λειτουργίες του λογισμικού.
9. **Αναφέρουμε τις domain κλάσεις με το όνομά τους.** Θυμόμαστε από το προηγούμενο κεφάλαιο ότι το domain μοντέλο χρησιμεύει ως ένα γλωσσάρι του έργου που βοηθά να εξασφαλιστεί η συνεπής χρήση των όρων κατά την περιγραφή του χώρου προβλήματος. Όταν προσπαθούμε να οδηγήσουμε ένα αντικειμενοστραφή σχεδιασμό από τις περιπτώσεις χρήσης, είναι εξαιρετικά σημαντικό οι περιπτώσεις

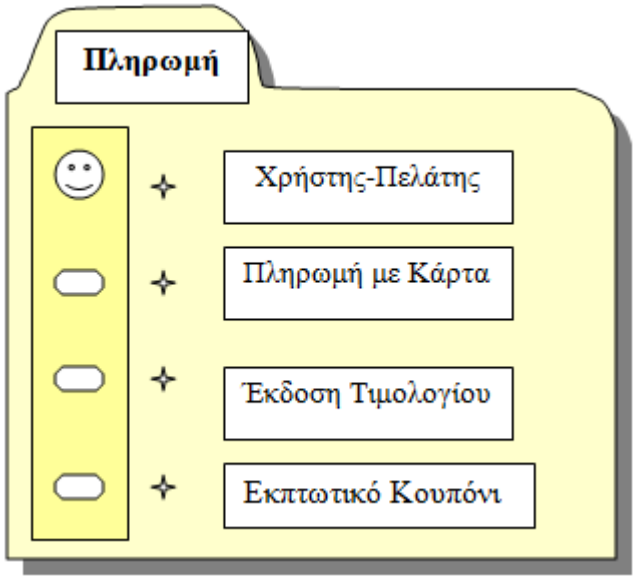
χρήσης να σχετίζονται με τα αντικείμενα. Πρέπει να σκεφτούμε το: πώς μπορούμε να οδηγήσουμε ένα μοντέλο αντικειμένου (object model) από τις περιπτώσεις χρήσης, εφόσον οι περιπτώσεις χρήσης δεν σχετίζονται με τα αντικείμενα; Η σύντομη απάντηση είναι δεν μπορούμε. Την ίδια στιγμή, δεν έχουμε πλήρη γνώση του τελικού μοντέλου αντικειμένων, όταν αρχίζουμε να γράφουμε περιπτώσεις χρήσης. Αυτό που γνωρίζουμε σε εκείνο το χρονικό σημείο είναι μια προκαταρκτική έκδοση του μοντέλου αντικειμένων που περιγράφει το πρόβλημα με ξεκάθαρο τρόπο-δηλαδή, το domain μοντέλο. Έτσι, πρέπει να συνδέσουμε τις περιπτώσεις χρήσης με τα domain αντικείμενα. Στην πράξη, αυτό σημαίνει αναφορά των domain αντικειμένων με το όνομά τους στο κείμενο των περιπτώσεων χρήσης.

10. **Αναφέρουμε τις Boundary κλάσεις με το όνομά τους.** Δεδομένου ότι είμαστε σε μια αποστολή όπου γράφουμε σαφείς απαιτήσεις συμπεριφοράς, και οι απαιτήσεις συμπεριφοράς σχεδόν πάντα περιλαμβάνουν το περιβάλλον εργασίας του χρήστη, είναι μια καλή ιδέα να μην γράψουμε αόριστες και διφορούμενες φράσεις όπως "Το σύστημα εμφανίζει μια σελίδα" στις περιπτώσεις χρήσης μας. Αντιθέτως, πρέπει να ονομάσουμε τις οθόνες μας ρητά-για παράδειγμα, "Το σύστημα εμφανίζει τη σελίδα Checkout."

5.3 Πακέτα Περιπτώσεων χρήσης

Πριν ξεκινήσουμε να γράφουμε τις περιπτώσεις χρήσης μας θα πρέπει να βρούμε ένα τρόπο να τις ομαδοποιήσουμε σε λογικές ομάδες. Ευτυχώς, η UML μας παρέχει ένα μηχανισμό πακέτων, το οποίο είναι ένας τρόπος για την ομαδοποίηση σχετιζόμενων στοιχείων (π.χ., κλάσεων, διαγραμμάτων, ή περιπτώσεων χρήσης). Τα πακέτα είναι βασικά ιεραρχημένα κοντέινερ που μπορεί να περιέχουν σχεδόν οποιαδήποτε κατασκευή UML, συμπεριλαμβανομένων και άλλων κοντέινερ. Τα UML πακέτα δεν είναι απολύτως ανόμοια από τα Java πακέτα, στο μέτρο που επιτρέπουν να διαιρέσουμε την εργασία μας σε μεγάλο βαθμό σε διάφορους τομείς.

Ένα τέτοιο πακέτο φαίνεται στο Σχήμα 5.4. Περισσότερα πακέτα περιπτώσεων χρήσης υπάρχουν στο Παράρτημα Γ'.



Σχήμα 5.4 Πακέτο ομάδας Πληρωμή

5.4 Σχέσεις (Relationships)

Τα διαγράμματα περιπτώσεων χρήσης χρησιμοποιούν τεσσάρων τύπου σχέσεις για να ενώσουν-συσχετίσουν τις περιπτώσεις χρήσης μεταξύ τους. Οι τέσσερις αυτές σχέσεις είναι οι εξής:

- **Include.** Σε μια μορφή αλληλεπίδρασης, μια δεδομένη περίπτωση χρήσης μπορεί να περιλαμβάνει μια άλλη. Η Include είναι μια κατευθυνόμενη σχέση μεταξύ δύο περιπτώσεων χρήσης, πράγμα που σημαίνει ότι η συμπεριφορά της περίπτωσης χρήσης που συμπεριλαμβάνεται, εισάγεται στη συμπεριφορά της περίπτωσης χρήσης που την περιλαμβάνει. Η περίπτωση χρήσης που συμπεριλαμβάνεται συχνά εξαρτάται από την έκβαση της περίπτωσης χρήσης που περιλαμβάνει. Αυτό είναι χρήσιμο για την εξαγωγή πραγματικά κοινών συμπεριφορών από πολλαπλές περιπτώσεις χρήσης σε μια ενιαία περιγραφή. Ο συμβολισμός είναι ένα διακεκομμένο βέλος από αυτήν που συμπεριλαμβάνει σε αυτήν που

περιλαμβάνεται, με την ετικέτα "« includes »". Δεν υπάρχουν παράμετροι ή τιμές επιστροφής.

- **Extend.** Σε μια άλλη μορφή αλληλεπίδρασης, μια συγκεκριμένη περίπτωση χρήσης (η επέκταση) μπορεί να επεκτείνει μία άλλη. Η σχέση δείχνει ότι η συμπεριφορά της περίπτωσης χρήση που επεκτείνεται μπορεί να εισαχθεί στην εκτεταμένη περίπτωση χρήσης υπό ορισμένες προϋποθέσεις. Ο συμβολισμός είναι ένα διακεκομμένο βέλος από την περίπτωση χρήσης που εκτείνεται προς την εκτεταμένη περίπτωση χρήσης, με την ετικέτα "« extends »". Σημειώσεις και περιορισμοί μπορεί να σχετίζονται με αυτή τη σχέση για την ανάδειξη των όρων υπό τους οποίους αυτή η συμπεριφορά θα εκτελεστεί. Οι σχεδιαστές μοντέλων χρησιμοποιούν την σχέση «επέκταση» για να δείξουν περιπτώσεις χρήσης που είναι «προαιρετικές» σε σχέση με την βασική περίπτωση χρήσης. Ανάλογα με την προσέγγιση των σχεδιαστικών μοντέλων το "προαιρετικό" μπορεί να σημαίνει "όχι δυνητικά εκτελέσιμο στην βασική περίπτωση χρήσης" ή μπορεί να σημαίνει "δεν απαιτείται για την επίτευξη του βασικού στόχου της περίπτωσης χρήσης".
- **Generalization (Γενίκευση).** Στην τρίτη μορφή της σχέσης μεταξύ των περιπτώσεων χρήσης, υπάρχει η σχέση γενίκευση / εξειδίκευση. Μια συγκεκριμένη περίπτωση χρήσης ενδέχεται να έχει κοινές συμπεριφορές, απαιτήσεις, περιορισμούς και υποθέσεις με μια πιο γενική περίπτωση χρήσης. Στην περίπτωση αυτή, τα περιγράφουμε μια φορά, και θα τα αντιμετωπίσουμε με τον ίδιο τρόπο, περιγράφοντας τις ενδεχόμενες διαφορές στη εξειδικευμένες περιπτώσεις. Ο συμβολισμός είναι μια σταθερή γραμμή που λήγει σε ένα κοίλο τρίγωνο που ξεκινά από την εξειδικευμένη στην πιο γενική περίπτωση χρήσης.
- **Associations (Συσχέτιση).** Οι συσχετίσεις μεταξύ των χαρακτήρων και των περιπτώσεων χρήσης αναφέρονται στα διαγράμματα περιπτώσεων χρήσης με συμπαγείς γραμμές. Η συσχέτιση υπάρχει κάθε φορά που ένας χαρακτήρας ασχολείται με μια αλληλεπίδραση

που περιγράφεται από την περίπτωση χρήσης. Οι συσχετίσεις μοντελοποιούνται ως γραμμές που συνδέουν τις περιπτώσεις χρήσης και τους χαρακτήρες μεταξύ τους, με προαιρετική αιχμή βέλους στο ένα άκρο της γραμμής. Το βέλος χρησιμοποιείται συχνά για να δείξει την κατεύθυνση της αρχικής επίκληση της σχέσης ή να αναφέρει τον κύριο χαρακτήρα στην περίπτωση χρήσης. Οι αιχμές βελών συνεπάγονται τον έλεγχο της ροής και δεν πρέπει να συγχέονται με ροές δεδομένων.

5.5 Περιπτώσεις χρήσης στην πράξη

Παρακάτω βλέπουμε ένα παράδειγμα περίπτωσης χρήσης. Είναι η περίπτωση χρήσης Επιλογή χαρακτηριστικών κράτησης δωματίου του πακέτου Κράτηση. (Περισσότερα διαγράμματα περιπτώσεων χρήσης μπορείτε να βρείτε στο cd που συνοδεύει το παρόν βιβλίο και βρίσκεται στη τελευταία σελίδα του βιβλίου).

Identifier	10
Author	natalia
Date	Aug 25, 2009 4:35:41 PM
Brief Description	Ο Χρήστης-Πελάτης επιλέγει τον προορισμό, τον αριθμό των δωματίων και ατόμων σε κάθε δωμάτιο και πατάει το κουμπί Αναζήτηση. Το σύστημα εμφανίζει την καρτέλα Διαθέσιμα ξενοδοχεία.
Preconditions	Ο Χρήστης-Πελάτης έχει επιλέξει την καρτέλα Ξενοδοχεία.

Flow of Events

	Actor Input	System Response
1	Ο Χρήστης-Πελάτης επιλέγει από το κεντρικό μενού την καρτέλα Ξενοδοχεία.	
2		Το σύστημα ανακτά τις προτεινόμενες πόλεις προς προορισμό από τον κατάλογο των Ξενοδοχείων.
3		Το σύστημα εμφανίζει την καρτέλα Ξενοδοχεία.
4	Ο Χρήστης-Πελάτης εισάγει τα κριτήρια αναζήτησης του επιθυμητού ξενοδοχείου-δωματίου (τον Προορισμό, τις Ημ.Άφιξης και Αναχώρησης, τον αριθμό Δωματίων και τους αριθμούς των ενηλίκων και παιδιών που θα διαμένουν	

		σε κάθε δωμάτιο.	
	5	Ο Χρήστης-Πελάτης πατάει το κουμπί Αναζήτηση.	
	6		Το σύστημα αναζητά τα διαθέσιμα Ξενοδοχεία-Δωμάτια στη Βάση Δεδομένων σύμφωνα με τις προτιμήσεις του Χρήστη-Πελάτη.
	7		Το σύστημα εμφανίζει την καρτέλα Διαθέσιμα Δωμάτια, ταξινομημένα κατά τιμή και αλφαβητικά.

Alternative Event (α)		**Actor Input**	**System Response**
	1	2(α)-Ο Χρήστης-Πελάτης εισάγει λάθος το όνομα του Προορισμού.	
	2		Το σύστημα ενημερώνει το Χρήστη-Πελάτη ότι η επιλογή που έκανε είναι λανθασμένη με την εμφάνιση του μηνύματος Ανύπαρκτος Προορισμός.

Alternative Event (β)		**Actor Input**	**System Response**
	1	7(β)-Δεν υπάρχουν διαθέσιμα Δωμάτια σύμφωνα με τις προτιμήσεις του Χρήστη-Πελάτη.	
	2		Το σύστημα εμφανίζει το μήνυμα "Δε βρέθηκαν Αποτελέσματα".

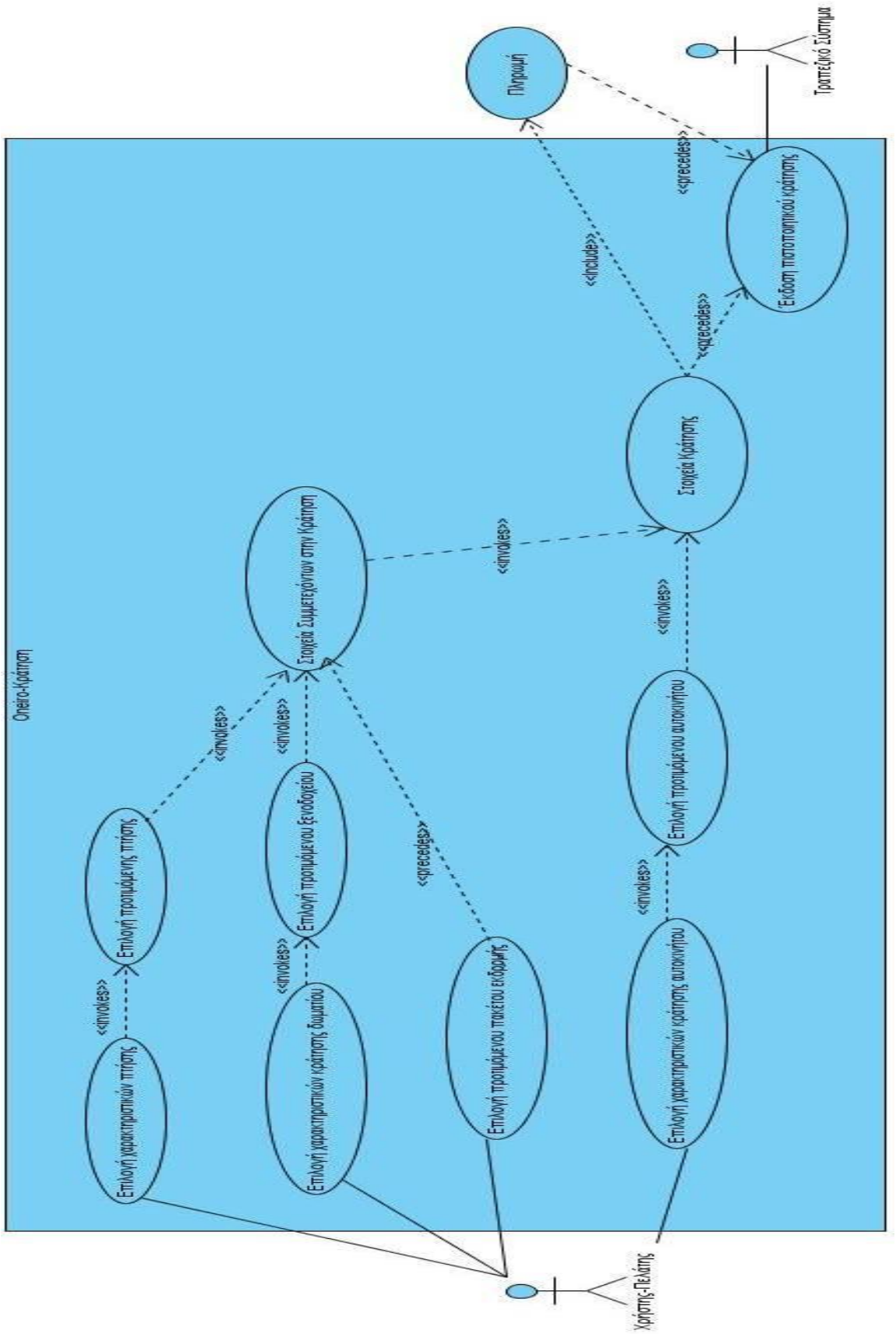
Επίσης δημιουργούμε μακέτες οθονών και πρωτότυπα ώστε να έχουμε μία πλήρη εικόνα του πως θα φαίνεται το σύστημα. Στο Σχήμα 5.5 δείχνεται η σελίδα Κράτηση Αεροπορικού Εισιτηρίου – Κριτήρια Χρήστη που είναι και η Αρχική Σελίδα του site.

Επίσης στο Σχήμα 5.6 παρουσιάζεται το διάγραμμα περιπτώσεων χρήσης για την εννοιολογική ενότητα Κράτηση.

Σχήμα 5.5 Μακέτα Οθόνης Κράτηση Αεροπορικού Εισιτηρίου - Κριτήρια Χρήστη

5.6 Περίληψη

Σε αυτό το κεφάλαιο, καλύψαμε λεπτομερώς πώς γράφουμε τις περιπτώσεις χρήσης που χρησιμοποιήθηκαν στον σχεδιασμό του λογισμικού μας. Στο Ορόσημο 1, θα έχουμε γράψει την πρώτη πρόχειρη έκδοση των περιπτώσεων χρήσης μας, έχοντας ρωτήσει εκτενώς τους πελάτες, τους τελικούς χρήστες και όλα τα ενδιαφερόμενα μέλη ώστε να εκμαιεύσουμε όσες περισσότερες πληροφορίες μπορούμε από αυτούς για το πώς το νέο σύστημα θα πρέπει να συμπεριφέρεται. Όμως ο ρόλος του πελάτη δεν τελειώνει εκεί. Στο επόμενο κεφάλαιο, θα καλύψει τις ανάγκες του σταδίου της επανεξέτασης, στο οποίο θα εξασφαλίζουμε ότι το domain model και οι περιπτώσεις χρήσης συνεργάζονται για την αντιμετώπιση των απαιτήσεων του πελάτη.



Σχήμα 5.6 Διάγραμμα Περιπτώσεων Χρήσης - Κράτηση

Κεφάλαιο 6: Επανεξέταση Απαιτήσεων (Requirements Review)

Η επανεξέταση απαιτήσεων διασφαλίζει ότι το σύστημα που περιγράφεται πραγματικά ταιριάζει με τις απαιτήσεις. Είναι μια συνεργατική επανεξέταση απαιτήσεων που περιλαμβάνει τον εκπρόσωπο του πελάτη (-ων), τους τελικούς χρήστες (δηλαδή τους ανθρώπους που πράγματι θα χρησιμοποιήσουν το σύστημα, ή που χρησιμοποιούν το τρέχον σύστημα που αντικαθίσταται), και των ανθρώπων του εμπορίου. Κατά βάση, όλοι οι συμμετέχοντες στο έργο που έχουν έννομο συμφέρον για την εξασφάλιση των απαιτήσεων τροφοδοτούν την άποψή τους για το σύστημα.

Στο κεφάλαιο αυτό, παρέχουμε μια επισκόπηση του σταδίου επανεξέτασης απαιτήσεων.

6.1 Οδηγίες Επανεξέτασης απαιτήσεων

Οι συμβουλές που δίνονται σε αυτό το κεφάλαιο συνοψίζονται στην παρακάτω λίστα.

1. **Βεβαιωνόμαστε ότι το domain model καλύπτει τα domain προβλήματα.** Στοχεύουμε σε ένα διάγραμμα domain model που έχει τις πιο σημαντικές αφαιρέσεις (abstractions) του domain προβλήματος. Όλα τα κουτιά για αυτό το διάγραμμα έχουν σαφείς ονόματα που οι χρήστες του συστήματος μπορούν να συσχετίσουν. Και το διάγραμμα αυτό πρέπει να γίνει γρήγορα. Περισσότερες λεπτομέρειες θα αποκαλυφθούν καθώς αναλύουμε τις περιπτώσεις χρήσεις.
2. **Δείχνουμε τις σχέσεις Γενίκευσης (Generalization) και Συνάθροισης (Aggregation).** Αυτές οι σχέσεις γενίκευση (είναι-ένα) και συνάθροιση (έχει-ένα) προχωρούν αρκετά προς την πραγματοποίηση του διαγράμματος και ειδικά σε αντίθεση με την ασάφεια.
3. **Περιγράφουμε τόσο τα Βασικά όσο και τα Εναλλακτικά Σενάρια σε ενεργητική φωνή.** Το γνωρίζουμε ήδη, αλλά οι περιπτώσεις χρήσης περιγράφουν πώς οι χρήστες χρησιμοποιούν το σύστημα, σε

χρόνο ενεστώτα ενεργητικής φωνής, και πρέπει να περιλαμβάνει τόσο τις συνήθεις (ηλιόλουστη ημέρα, sunny day) χρήσεις καθώς και τις λιγότερο τυπικές (βροχερή μέρα, rainy day) χρήσεις.

4. **Δεν αναμειγνύουμε τις λειτουργικές απαιτήσεις στο κείμενο των περιπτώσεων χρήσης.** Οι λειτουργικές απαιτήσεις είναι παθητική φωνή, απαιτήσεις τύπου "θα πρέπει" (π.χ., "Το σύστημα θα πρέπει κάνει αυτό»). Ένα κοινό λάθος είναι να συμπεριληφθούν αυτές στις περιγραφές περιπτώσεων χρήσης. Αντ' αυτού, οι λειτουργικές απαιτήσεις πρέπει να διατηρούνται χωριστά και να "κατανεμηθούν" στις περιπτώσεις χρήσης. Δηλαδή, οι περιπτώσεις χρήσης πληρούν τις λειτουργικές απαιτήσεις, αλλά και οι απαιτήσεις «θα» παθητικής φωνής δεν συνθέτουν τις περιπτώσεις χρήσης. Κρατάμε τις περιπτώσεις χρήσης που επικεντρώνονται στην χρήση του συστήματος (ενεργητική φωνή).
5. **Οργανώνουμε τις περιπτώσεις χρήσης σε πακέτα.** Μπορούμε να οργανώσουμε τις περιπτώσεις χρήσης σε λειτουργικά συναφείς τομείς (υποσυστήματα). Μία χρήσιμη στρατηγική είναι να προσδιοριστούν όλες τις περιπτώσεις χρήσης μέσα σε ένα λειτουργικό χώρο στο διάγραμμα περιπτώσεων χρήσης.
6. **Γράφουμε τις περιπτώσεις χρήσης στο πλαίσιο του Μοντέλου αντικειμένων.** Με μεγάλη διαφορά ο πιο αποτελεσματικός τρόπος για να στηρίξουμε τις περιπτώσεις χρήσης στο domain model είναι να φτιάξουμε πρώτα το domain model, και να γράψουμε τις περιπτώσεις χρήσης ώστε να αναφέρονται ρητά στα domain αντικείμενα με το όνομα τους. Μεγάλο μέρος της ασάφειας στις περιπτώσεις χρήσης απορρέει από το γεγονός ότι συχνά είναι γραμμένες με "όρους χρήστη" χωρίς να αναφέρονται ρητά και με ακρίβεια στα ιδιαίτερα "στοιχεία του domain προβλήματος» που επηρεάζονται από το ζητούμενο σενάριο. Έτσι το πρώτο βήμα για την αποσαφήνιση του κειμένου των περιπτώσεων χρήσης είναι να κάνουμε αυτό το κείμενο να αναφέρεται ρητά στα κατάλληλα domain αντικείμενα.

7. **Βάζουμε τις περιπτώσεις χρήσης στο πλαίσιο του περιβάλλοντος του χρήστη.** Για να βασίσουμε τις περιπτώσεις χρήση μας στο περιβάλλον εργασίας του χρήστη, ονομάζουμε τις οθόνες που θα συμμετάσχουν στην περίπτωση χρήσης, και να χρησιμοποιούμε αυτά τα ονόματα στο κείμενο των περιπτώσεων χρήσης. Οι περιπτώσεις χρήσης πραγματικά έχουν ανάγκη να συνδέονται όχι μόνο το μοντέλο αντικειμένου, αλλά και το πρότυπο GUI με τις αφηγηματικές περιγραφές συμπεριφοράς. Σε πρακτικό επίπεδο, αυτό συνήθως σημαίνει ότι θα πρέπει να ονομάσουμε τις οθόνες μας ρητά και να χρησιμοποιήσουμε αυτά τα ονόματα στο κείμενο της περίπτωσης χρήσης.
8. **Χρησιμοποιούμε Storyboards, μακέτες οθονών και GUI πρωτότυπα.** Βεβαιωνόμαστε ότι όλη η συμπεριφορά του χρήστη (π.χ., τα κουμπιά που μπορούν να κάνουν κλικ, τα μενού από τα οποία μπορούν να επιλέξουν) προορίζεται για τις περιπτώσεις χρήσης. Αυτές οι μακέτες δεν χρειάζεται να είναι απεικονίσεις υψηλής πιστότητας με κινούμενα κουμπιά ("το καλύτερο είναι εχθρός του καλού"). Πρέπει να είναι απλές, σαφής εικόνες που εστιάζουν ποια συμπεριφορά του συστήματος ενεργοποιείται από το χρήστη και πρέπει να είναι γρήγορες στη δημιουργία.
9. **Επανεξέταση των απαιτήσεων συμπεριφοράς με τους κατάλληλους ανθρώπους.** Επανεξέταση των περιπτώσεων χρήσης, domain model, καθώς και των πρωτοτύπων, μακετών/GUI οθονών με τους τελικούς χρήστες, τα ενδιαφερόμενα μέρη και τους ανθρώπους του εμπορίου, σε συνδυασμό με πιο τεχνικά μέλη του προσωπικού μας. Βεβαιωνόμαστε ότι οι απαιτήσεις γίνονται καλά κατανοητές και έχουν συμφωνηθεί από όλα τα μέρη πριν προχωρήσουμε προς την κατεύθυνση του σχεδιασμού. Ο στόχος της επανεξέτασης των απαιτήσεών μας είναι η επίτευξη της βασικής συμφωνίας μεταξύ όλων των μερών ότι οι περιπτώσεις χρήσης αποτυπώνουν τη συμπεριφορά των απαιτήσεων του συστήματος. Και προκειμένου να επιτευχθεί ο

στόχος αυτός, θα πρέπει να εξασφαλισθεί ότι η απαιτούμενη συμπεριφορά του συστήματος είναι σαφώς κατανοητή από όλους.

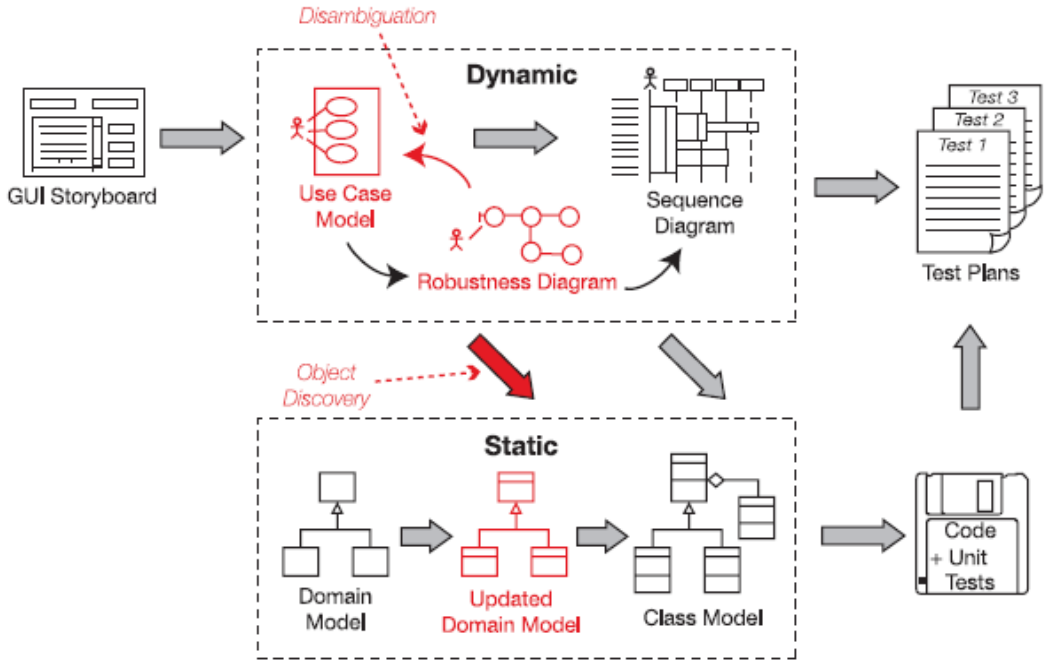
6.2 Επανεξέταση απαιτήσεων στην πράξη

Στο Ορόσημο 1 επαναλαμβάνουμε τις οδηγίες που μάθαμε σε αυτό το κεφάλαιο, αλλά καθώς και στο προηγούμενο ραφινάροντας το domain model μας, τα διαγράμματα περιπτώσεων χρήσης και τις ίδιες τις περιπτώσεις χρήσεις μέχρι αν είμαστε σίγουροι ότι ακολουθούνται όλοι οι κανόνες. Καθώς αλλάζουμε στοιχεία στις περιπτώσεις χρήσης μας δε ξεχνάμε να ανανεώνουμε και το domain model ώστε να ακολουθεί πιστά τις περιπτώσεις χρήσης, και το αντίστροφο. Επίσης τελειοποιούμε τα storyboards, τα πρωτότυπα και τις μακέτες οθονών.

6.3 Περίληψη

Σε αυτό το κεφάλαιο, καλύψαμε το ορόσημο της επανεξέταση των απαιτήσεων. Είναι ένα σημαντικό βήμα διότι εξασφαλίζει ότι οι απαιτήσεις είναι επαρκώς κατανοητές τόσο από την ομάδα ανάπτυξης όσο και από τον πελάτη / τους χρήστες / τους συμμετέχοντες στο έργο. Στο επόμενο κεφάλαιο, θα ξεκινήσουμε τον προκαταρκτικό σχεδιασμό, ένα βήμα το οποίο παρέχει εντατική ανάδραση στις περιπτώσεις χρήσης και στο domain μοντέλο και ουσιαστικά αποτελεί την γέφυρα ανάμεσα στην ανάλυση και το σχεδιασμό.

Κεφάλαιο 7: Ανάλυση Ευρωστίας



Σχήμα 7.1 Μέθοδος ICONIX σε λεπτομερειακή ροή εργασίας (Robustness Analysis)

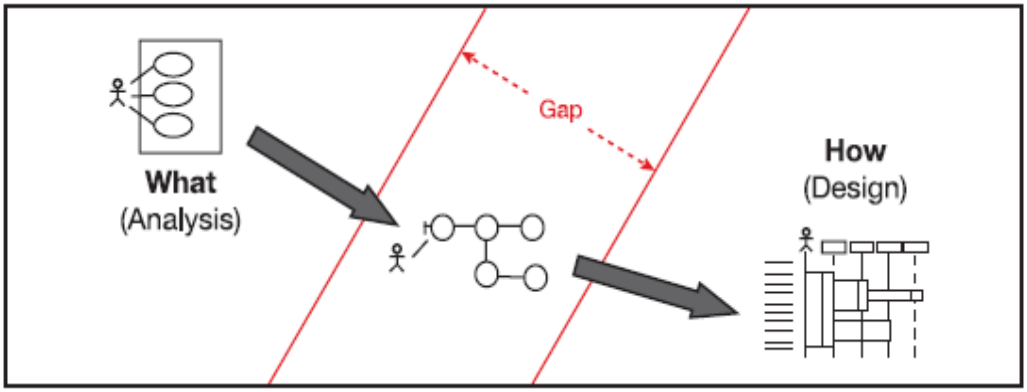
Για να πάρουμε από τις περιπτώσεις χρήσης το λεπτομερή σχεδιασμό (και στη συνέχεια τον κώδικα), θα πρέπει να συνδέσουμε τις περιπτώσεις χρήσης με αντικείμενα. Η τεχνική που περιγράφεται σε αυτό το κεφάλαιο, η ανάλυση ευρωστίας, μας βοηθά στο να γεφυρώσουμε το χάσμα από την ανάλυση προς το σχεδιασμό, κάνοντας ακριβώς αυτό. Με λίγα λόγια, είναι ένας τρόπος ανάλυσης του κειμένου των περιπτώσεων χρήσης και προσδιορισμού ενός πρώτου συνόλου αντικειμένων για κάθε περίπτωση χρήσης. Αυτά κατατάσσονται σε boundary objects, entity objects και controllers (που συχνά είναι περισσότερο λειτουργίες παρά αντικείμενα).

Ένα διάγραμμα ευρωστίας είναι μια εικόνα αντικειμένων μιας περίπτωσης χρήσης. Το διάγραμμα ευρωστίας και το κείμενο της περίπτωσης χρήσης πρέπει να ταιριάζουν απόλυτα, έτσι το διάγραμμα ευρωστία μας πιέζει να συνδέσουμε το κείμενο της περίπτωσης χρήσης με τα αντικείμενα. Αυτό μας επιτρέπει να παίρνουμε αντικειμενοστραφή σχέδια από τις περιπτώσεις χρήσης, και αυτό είναι πραγματικά η «μαγεία» της ανάλυσης ευρωστίας.

Σχεδιάζοντας ένα διάγραμμα ευρωστίας εξασφαλίζουμε ότι η περίπτωση χρήσης είναι γραμμένη στα πλαίσια του domain μοντέλου, δηλαδή όλοι οι όροι (ουσιαστικά και ονοματικές φράσεις) που βρίσκονται στο domain μοντέλο θα πρέπει να χρησιμοποιούνται επίσης άμεσα στο κείμενό των περιπτώσεων χρήσης μας.

7.1 Που τοποθετείται το Διάγραμμα Ευρωστίας στη διαδικασία

Κοιτάζοντας Σχήμα 7.2 , η ανάλυση ευρωστίας πραγματοποιείται στο σκοτεινό μέσο έδαφος ανάμεσα στην ανάλυση και το σχεδιασμό. Αν θεωρήσουμε ότι η ανάλυση (δηλαδή, οι περιπτώσεις χρήσης) είναι το «τι» και ο σχεδιασμός το «πώς», τότε η ανάλυση ευρωστίας είναι πραγματικά ένα προκαταρκτικό σχέδιο. Κατά τη διάρκεια αυτής της φάσης, θα αρχίσουμε να κάνουμε κάποιες προκαταρκτικές υποθέσεις σχετικά με το σχέδιό μας, και αρχίζουμε να σκεφτόμαστε για την τεχνική δομή και να σκεφτόμαστε μέσω των διαφόρων πιθανών στρατηγικών σχεδιασμού. Έτσι είναι μέρος της ανάλυση και μέρος του σχεδιασμού. Είναι επίσης μια σημαντική τεχνική για να αφαιρέσουμε την ασάφεια από το κείμενο που αφορά τη περίπτωση χρήσης μας.



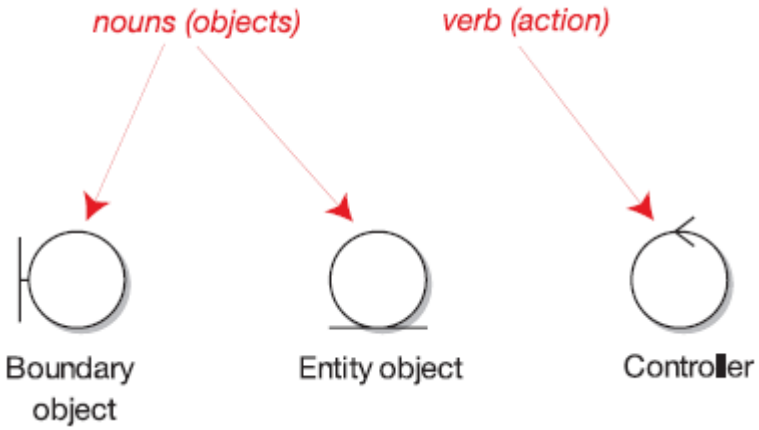
Σχήμα 7.2 Γεφύρωση του χάσματος μεταξύ του τι και του πως

7.2 Ανατομία ενός Διαγράμματος Ευρωστίας

Ένα διάγραμμα ευρωστία είναι κάπως ένα υβρίδιο μεταξύ ενός διαγράμματος κλάσης και ενός διαγράμματος δραστηριότητας. Είναι μια εικονογραφική αναπαράσταση της συμπεριφοράς που περιγράφεται από την περίπτωση χρήσης, παρουσιάζοντας τόσο τις κλάσεις που συμμετέχουν όσο και τη συμπεριφορά του λογισμικού, αν και αποφεύγετε σκοπίμως να απεικονιστεί ποιά κλάση είναι

υπεύθυνη για ποιά κομμάτια συμπεριφοράς. Κάθε κλάση εκπροσωπείται από ένα γραφικό εικονίδιο στερεότυπο (βλέπε Σχήμα 7.3). Ωστόσο, ένα διάγραμμα ευρωστίας μοιάζει περισσότερο σε ένα διάγραμμα δραστηριότητας (ή ένα διάγραμμα ροής), με την έννοια ότι το ένα αντικείμενο "μιλά" στο επόμενο αντικείμενο. Αυτή η ροή της δράσης αντιπροσωπεύεται από μια γραμμή μεταξύ των δύο αντικειμένων που μιλάνε το ένα στο άλλο.

Υπάρχει μια άμεση συσχέτιση ένα προς ένα μεταξύ της ροή της δράσης στο διάγραμμα ευρωστίας και τα βήματα που περιγράφονται στο κείμενο της περίπτωση χρήσης.



Σχήμα 7.3 Σύμβολα Ανάλυσης Ευρωστίας

Τα τρία στερεότυπα κλάσης που φαίνονται στο Σχήμα 7.3 είναι τα εξής:

- **Boundary objects.** Η "διασύνδεση" μεταξύ του συστήματος και του έξω κόσμου (δείτε πίσω στο Σχήμα 7.3). Τα boundary objects είναι συνήθως οθόνες ή ιστοσελίδες (δηλαδή, το στρώμα παρουσίασης με το οποίο αλληλεπιδρά ο χαρακτήρας).
- **Entity objects.** Κλάσεις από το μοντέλο domain.
- **Controllers.** Η «κόλλα» μεταξύ των boundary και τον entity objects.

Είναι χρήσιμο να σκεφτόμαστε τα boundary objects και entity objects ως ουσιαστικά, και τους controllers ως ρήματα. Ακολουθούμε τους παρακάτω κανόνες κατά το σχεδιασμό των διαγραμμάτων ευρωστία μας:

- Τα ουσιαστικά μπορούν να μιλούν σε ρήματα (και το αντίστροφο).
- Τα ουσιαστικά δε μπορούν να μιλούν με άλλα ουσιαστικά.

- Τα ρήματα μπορούν να μιλούν με άλλα ρήματα.

Οι κανόνες αυτοί βοηθούν στο να επιβάλουν ένα σχέδιο ουσιαστικό-ρήμα-ουσιαστικό στο κείμενο των περιπτώσεων χρήσης μας. Αν το κείμενό των περιπτώσεων χρήσης μας ακολουθεί αυτό το πρότυπο, τα διαγράμματα ευρωστίας είναι πανεύκολα στο να σχεδιαστούν. Η ανάλυση ευρωστίας παρέχει ένα λογικό έλεγχο των περιπτώσεων χρήσης.

7.3 Οδηγίες Ανάλυσης Ευρωστίας

Στην ενότητα αυτή, περιγράφεται η θεωρία πίσω από την ανάλυση ευρωστίας, διανθίζοντάς τη με παραδείγματα από το έργο του Διαδικτυακού Τουριστικού Γραφείου. Θα ξεκινήσουμε με τις πρώτες κατευθυντήριες γραμμές για την ανάλυση ευρωστίας μας.

1. **Θέτουμε το κείμενο της περίπτωσης χρήσης μας δίπλα από το διάγραμμα ευρωστίας που θα δημιουργήσουμε.** Με αυτόν τον τρόπο βοηθάμε πραγματικά στο να ενισχύσουμε το γεγονός ότι σχεδιάζουμε μία εικόνα αντικειμένων από τα γεγονότα που περιγράφονται στην περίπτωση χρήσης. Επιπλέον, θα δουλεύουμε την περίπτωση χρήσης μια πρόταση τη φορά καθώς θα σχεδιάζουμε το διάγραμμα, έτσι είναι βολικό να έχουμε το κείμενο από κοντά. Επειδή το διάγραμμα ευρωστίας είναι ουσιαστικά μια αναπαράσταση με εικόνες της περίπτωσης χρήσης, βοηθά να έχουμε το κείμενο σε ένα σημείωμα στο διάγραμμα: είναι δύο διαφορετικές όψεις του ίδιου πράγματος, έτσι θα πρέπει να είμαστε σε θέση να «περπατήσουμε» μέσα στο κείμενο και να το εντοπίσουμε στο διάγραμμα (και αντίστροφα).
2. **Παίρνουμε τις κλάσεις οντοτήτων (entity classes) από το domain model και προσθέτουμε όσες λείπουν.** Οι περισσότερες από τις οντότητες στο διάγραμμα ευρωστίας μας θα προέλθουν από το μοντέλο domain. Ωστόσο, από τη στιγμή που θέσαμε όριο χρόνου στην αρχική προσπάθεια μοντελοποίησης του domain model σε μερικές ώρες, είναι φυσικό να αναμένουμε ότι ίσως να λείπουν

ορισμένες domain κλάσεις. Όταν είμαστε στη διάρκεια της σύνταξης διαγραμμάτων ευρωστίας και συμβεί αυτό, φροντίζουμε να προσθέτουμε τις κλάσεις που λείπουν στο domain μοντέλο. Η μεθοδολογία ICONIX υποθέτει ότι το αρχικό domain μοντέλο μας θα είναι ελλιπής και αναμένει ότι λείπουν αντικείμενα τα οποία θα ανακαλύπτονται κατά την ανάλυση ευρωστίας. Στο βιβλίο αυτό, αναφερόμαστε σε αυτή τη διαδικασία ως ανακάλυψη αντικειμένων.

3. **Διορθώνουμε την περίπτωση χρήσης καθώς σχεδιάζουμε το Διάγραμμα Ευρωστίας.** Η πείρα έχει δείξει ότι η πρώτη προσπάθεια εγγραφής των περιπτώσεων χρήσης έχουν την τάση να παρουσιάζουν τα ακόλουθα χαρακτηριστικά: είναι συνήθως ασαφείς, διφορούμενες, ελλιπείς και εσφαλμένες. Η κατάργηση της ασάφειας από τις περιπτώσεις χρήσης είναι ένας από τους βασικούς σκοπούς αυτής της τεχνικής. Η «μαγεία» της τεχνικής αυτής είναι στην πραγματικότητα η σκληρή δουλειά: σχεδιάζοντας ένα διάγραμμα ευρωστίας μας αναγκάζει να δουλέψουμε την περίπτωση χρήσης παίρνοντας μία φράση τη φορά. Αυτή η απλή πράξη σχεδόν πάντα φέρνει στην επιφάνεια τα λάθη στο κείμενο της πρώτης προσπάθειας περίπτωσης χρήσης, έτσι είναι σημαντικό να ξαναγράψουμε την περίπτωση χρήσης, παράλληλα με το σχεδιασμό του διαγράμματος ευρωστίας.
4. **Κάντε ένα boundary object για κάθε οθόνη/σελίδα.** Σχεδιάζοντας ένα διάγραμμα ευρωστίας μπορούμε να επιβάλλουμε σαφείς ονοματολογία. Αν, για παράδειγμα, βρούμε ένα boundary object στο διάγραμμα ευρωστίας να έχει όνομα ιστοσελίδα, σταματάμε και προσπαθούμε να του δώσουμε ένα πραγματικό όνομα, όπως Σελίδα Στοιχεία Πελάτη.
5. **Οι Controllers είναι τυπικά λογικές λειτουργίες λογισμικού.** Είναι δυνατό να έχουμε control κλάσεις στο σχεδιασμό μας (π.χ., κλάσεις manager) και μπορούν να αντιπροσωπευτούν ως controllers στο διάγραμμα ευρωστίας. Ωστόσο, δεν είναι απόλυτο ότι κάθε controller σε ένα διάγραμμα ευρωστίας θα αποτελέσει μια πραγματική κλάση

ελέγχου. Σε πολλές περιπτώσεις, ο controller σε ένα διάγραμμα ευρωστίας απλώς χρησιμοποιείται ως σύμβολο κράτησης θέσης για μια λειτουργία του λογισμικού. Η κατάχρηση των κλάσεων controller (π.χ. ένας controller για κάθε περίπτωση χρήσης) σε ένα διάγραμμα ευρωστίας μπορεί να μας οδηγήσει πίσω στην λειτουργική αποσύνθεση, έτσι οι controller κλάσεις θα πρέπει να χρησιμοποιούνται με φειδώ. Η παρουσίαση ενός συνδυασμού αντικειμένων και των λειτουργιών τους είναι ένας από τους τρόπους με τον οποίο ένα διάγραμμα ευρωστίας διαφέρει ουσιαστικά από ένα διάγραμμα συνεργασίας.

6. **Δεν ανησυχούμε για την κατεύθυνση των βελών σε ένα Διάγραμμα Ευρωστίας.** Θυμόμαστε ότι το διάγραμμα ευρωστίας έχει δύο στόχους:
    - Να μας αναγκάσει να αποσαφηνίσουμε τα κείμενα των περιπτώσεων χρήσης.
    - Να μας βοηθήσει να βρούμε τα αντικείμενα που λείπουν από το domain model μας.

    Το ποια κατεύθυνση θα πάρουν τα βέλη δεν προσδίδει τίποτα στους παραπάνω στόχους.

7. **Δείχνουμε Επικαλούμενες (Invoked) περιπτώσεις χρήσης στο Διάγραμμα Ευρωστίας.** Μπορούμε να μεταφέρουμε μία περίπτωση χρήσης σε ένα διάγραμμα ευρωστίας αν γίνεται επίκληση από την περίπτωση χρήσης γονέα. Είναι ο απλούστερος τρόπος για να δείξουμε μια περίπτωση επικαλείται από μία άλλη σε ένα διάγραμμα ευρωστίας.
8. **Το Διάγραμμα Ευρωστίας αναπαριστά ένα προκαταρκτικό σχέδιο των περιπτώσεων χρήσης.** Εδώ είναι μερικές θεμελιώδεις αλήθειες σχετικά με την ανάπτυξη του συστήματος:
    - Είναι μια καλή ιδέα να αντιλαμβανόμαστε πλήρως τις απαιτήσεις πριν προχωρήσουμε στο σχεδιασμό.

- Είναι συχνά αδύνατο να κατανοήσουμε πλήρως τις απαιτήσεις χωρίς να κάνουμε κάποια διερευνητική σχεδίαση.

Αυτές οι δύο καταστάσεις μπορεί να φαίνονται αντιφατικές, αλλά η λύση είναι πολύ απλή: μπορούμε να κάνουμε μια προκαταρκτική μελέτη με σκοπό την επικύρωση της συμπεριφοράς των απαιτήσεων, πριν από τον πραγματικό σχεδιασμό, ο οποίος θα μας οδηγήσει στον κώδικα. Το διάγραμμα ευρωστίας αντιπροσωπεύει τα σχέδια μελέτης, ενώ ο πραγματικός σχεδιασμός εμφανίζεται στα διαγράμματα ακολουθίας.

9. **Τα αντικείμενα στο Διάγραμμα Ευρωστίας θα πάρουν μορφή στο λεπτομερή σχεδιασμό.** Κλάσεις boundary και entity στο διάγραμμα ευρωστίας θα μεταμορφωθούν γενικά σε στιγμιότυπα αντικειμένων στο διάγραμμα ακολουθίας, καθώς οι controllers θα γίνουν μηνύματα. Λαμβάνουμε υπόψη ότι τόσο τα boundary αντικείμενα όσο και τα entity αντικείμενα είναι ουσιαστικά, και ότι οι controllers είναι ρήματα (δηλαδή, μια δράση που εκτελείται σε ένα αντικείμενο). Ως εκ τούτου, είναι λογικό ότι οι controllers (ενεργειών) θα γίνουν μέθοδοι στις κλάσεις boundary και entity αντίστοιχα.

7.4 Πώς πραγματοποιούμε ανάλυση ευρωστίας;

Εκτελούμε ανάλυση ευρωστίας για μία περίπτωση χρήσης ενεργώντας μέσω του κειμένου περίπτωσης χρήσης, μία φράση κάθε φορά, και σχεδιάζοντας τον/τους χαρακτήρα(-ες), τα κατάλληλα boundary και entity αντικείμενα και τους controllers, καθώς και τις συνδέσεις μεταξύ των διαφόρων στοιχείων του διαγράμματος. Θα πρέπει να εγκαταστήσουμε το βασικό σενάριο και όλα τα εναλλακτικά σενάρια σε ένα διάγραμμα.

7.5 Διάγραμμα ευρωστίας στην πράξη

Παρακάτω, στο Σχήμα 7.4, παρουσιάζουμε το διάγραμμα ευρωστίας για την περίπτωση χρήσης Επιλογή χαρακτηριστικών κράτησης δωματίου. (Περισσότερα

διαγράμματα ευρωστίας για όλες τις περιπτώσεις χρήσης του συστήματός μας θα βρείτε στο συνοδευτικό cd που βρίσκεται στην τελευταία σελίδα του βιβλίου).

7.6 Ενημέρωση του στατικού domain model μας

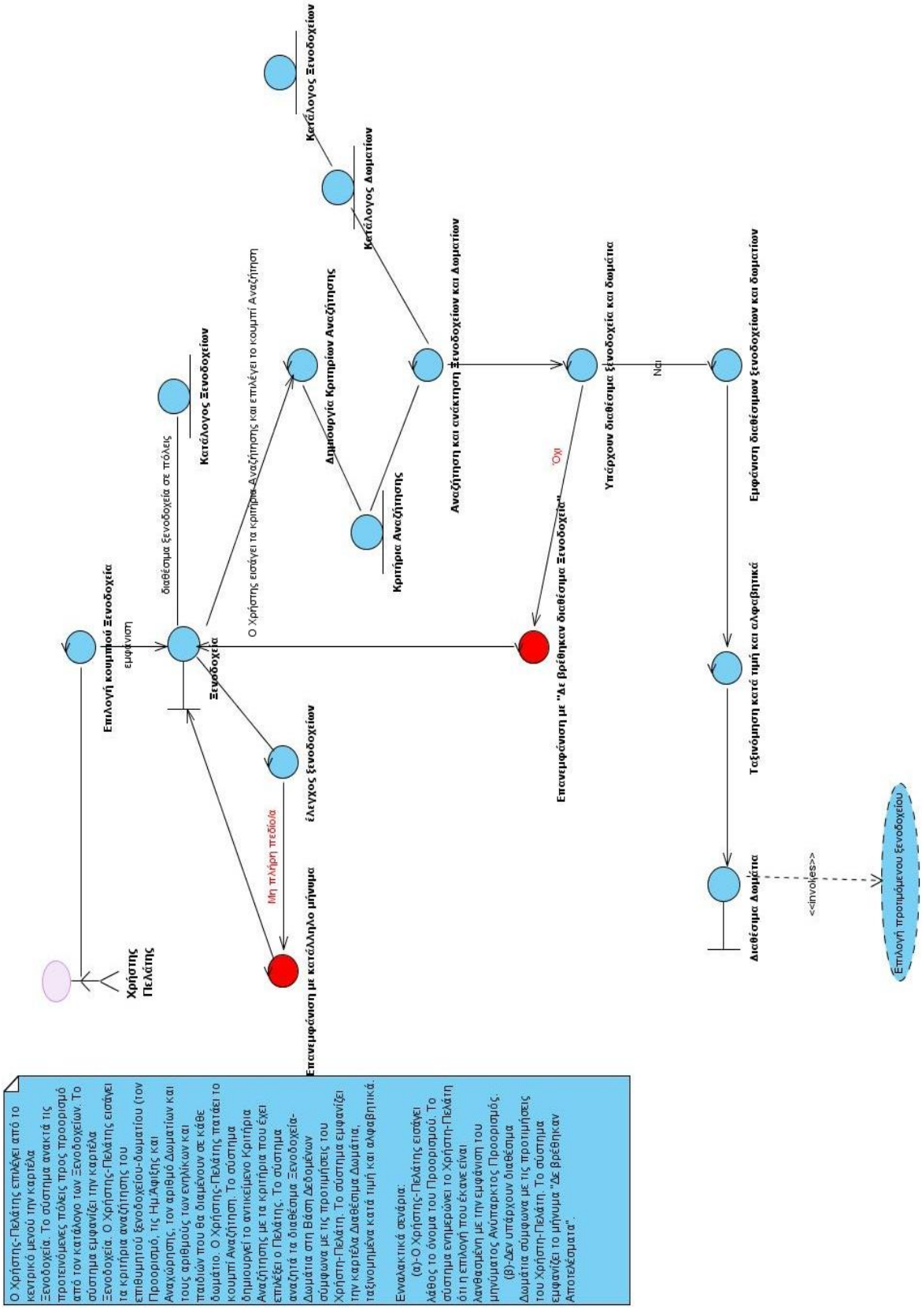
Κατά τη σύνταξη των διαγραμμάτων ευρωστίας, είναι μια καλή ιδέα να ενημερώνουμε σταδιακά το domain μοντέλο μας, καθώς προχωράμε. Θα ανακαλύψουμε σχεδόν αναπόφευκτα νέες κλάσεις domain. Θα προσδιορίζει επίσης τα χαρακτηριστικά που πρέπει να προστεθούν στις κλάσεις αυτές. Αυτό όλο πρέπει να γίνει στο domain model (γνωστό και ως ανάλυση σε επίπεδο στατικού μοντέλου), και είναι καλύτερο να το κάνουμε τώρα, που μόλις εντοπίσαμε αυτές τις αλλαγές ή τα νέα στοιχεία, πριν ξεχαστούν.

Τα νέα χαρακτηριστικά είναι δυνατόν να ανακαλυφθούν από το κείμενο των περιπτώσεων χρήσης, είτε από τα πρωτότυπα GUI, ή ακόμη και από τις λειτουργικές απαιτήσεις.

7.7 Περίληψη

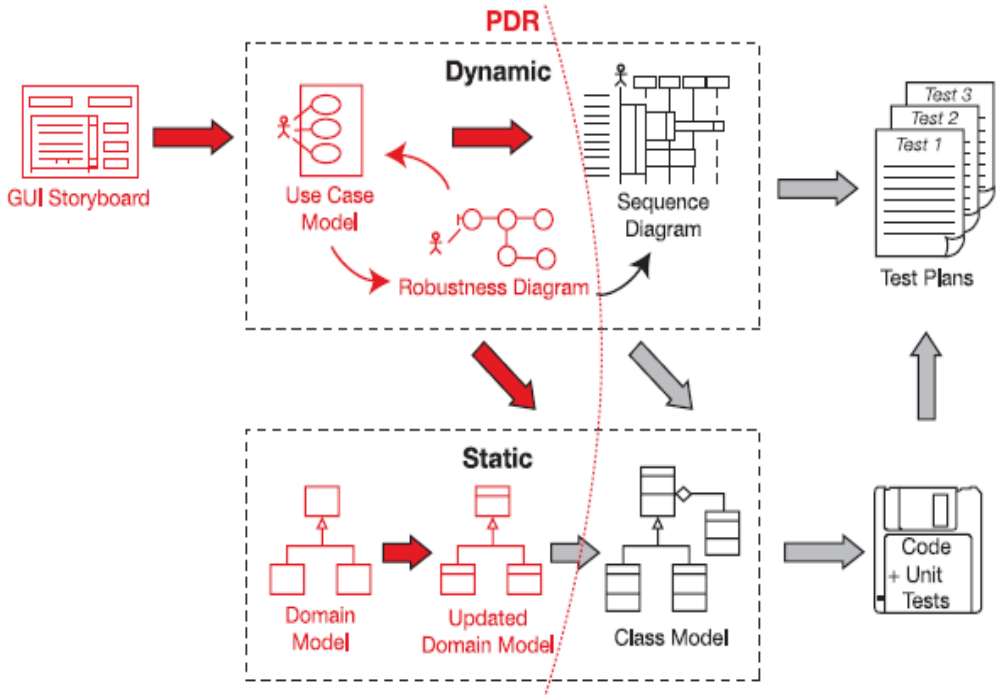
Στο κεφάλαιο αυτό, περάσαμε από την ανάλυση στο σχεδιασμό, χρησιμοποιώντας ένα από τα πιο χρήσιμα και ακόμη πιο καλά φυλαγμένα μυστικά της βιομηχανίας: την ανάλυση ευρωστίας.

Έτσι ερχόμαστε στο Ορόσημο 2: Προκαταρκτική Επισκόπηση Σχεδιασμού (Preliminary Design Review, PDR) που καλύπτεται στο επόμενο κεφάλαιο.



Σχήμα 7.4 Διάγραμμα Ευρωστίας Επιλογή χαρακτηριστικών κράτησης δωματίου

Κεφάλαιο 8: Προκαταρκτική Επισκόπηση Σχεδιασμού (Preliminary Design Review, PDR)



Σχήμα 8.1 Μεθοδολογία ICONIX σε λεπτομερειακή ροή εργασίας (PDR)

Η Προκαταρκτική Επισκόπηση Σχεδιασμού μας βοηθά να βεβαιωθούμε ότι τα διαγράμματα ευρωστίας, το domain μοντέλο, καθώς και το κείμενο που αφορά τις περιπτώσεις χρήσης ταιριάζουν όλα μεταξύ τους. Η επισκόπηση αυτή είναι η «πύλη» μεταξύ του προκαταρκτικού σχεδιασμού και των λεπτομερών σταδίων σχεδιασμού, για κάθε πακέτο περιπτώσεων χρήσης.

8.1 Οδηγίες Προκαταρκτικής Επισκόπησης Σχεδιασμού

Οι αρχές που δίνονται σε αυτό το κεφάλαιο συνοψίζονται στην παρακάτω λίστα.

1. **Χρησιμοποιούμε τον επισημαντή για να ελέγξουμε ότι το κείμενο της περίπτωσης χρήσης ταιριάζει με το διάγραμμα ευρωστίας.** Μερικές φορές σκεφτόμαστε ότι πρέπει να αναφέρουμε μόνο τη βασική πορεία δράσης σε ένα διάγραμμα ευρωστίας, ή ότι πρέπει να κάνουμε ένα ξεχωριστό διάγραμμα για κάθε εναλλακτική πορεία. Αλλά είναι καλύτερα να δούμε ολόκληρη την περίπτωση χρήσης (βασικό

σενάριο, αλλά και όλα τα εναλλακτικά σενάρια) σε ένα ενιαίο διάγραμμα ευρωστίας. Αν το διάγραμμα γίνεται πάρα πολύ μεγάλο (ή αν διαπιστώσουμε ότι τα εναλλακτικά έχουν τα δικά τους εναλλακτικά σενάρια), τότε είναι σίγουρη η κατάτμηση της περίπτωσης χρήσης. Αν επισημάνουμε μία φράση στο κείμενο των περιπτώσεων χρήσης και επισημάνουμε και τον αντικατοπτρισμό της φράσης στο διάγραμμα ευρωστίας, μέχρι να τελειώσει όλη η περίπτωση χρήσης (βασικό και εναλλακτικά σενάρια) τότε το διάγραμμά μας είναι σωστό και το ίδιο και η περίπτωση χρήσης μας. Αν όμως βρούμε ότι δεν ταιριάζουν τότε πρέπει να το διορθώσουμε.

2. **Βεβαιωνόμαστε ότι όλες οι οντότητες εμφανίζονται στο ανανεωμένο domain model.** Η ανακάλυψή αντικειμένων είναι ένας από τους πρωταρχικούς σκοπούς της ανάλυσης ευρωστίας, δεν έχει νόημα να ανακαλύψουμε νέα αντικείμενα στα σχετικά διαγράμματα ευρωστία μας και να μην τα προσθέσουμε στο μοντέλο κλάσης (class model) μας (το οποίο εξελίσσεται από το μοντέλο domain).
3. **Βεβαιωνόμαστε ότι μπορούμε να εντοπίσουμε τη ροή δεδομένων μεταξύ των κλάσεων και των οθονών.** Η περίπτωση χρήσης μας πολύ πιθανό να περιλαμβάνει χρήστες που καθορίζουν τις πληροφορίες χρησιμοποιώντας τις οθόνες του συστήματος. Τα δεδομένα αυτά πρέπει να βρουν το δρόμος προς τις κλάσεις οντοτήτων, οι οποίες κατέχουν τις αξίες που είναι εγγεγραμμένες ως ιδιότητες. Φυσικά, αυτό λειτουργεί επίσης και με τον άλλο τρόπο: οι τιμές από τις κλάσεις οντοτήτων θα εμφανίζονται στις οθόνες.
4. **Δεν πρέπει να ξεχάσουμε τα εναλλακτικά σενάρια και την συμπεριφορά τους.** Το έχουμε ξαναεπισημάνει αλλά είναι το πιο σημαντικό. Είναι η τελευταία μας ευκαιρία για να εντοπίσουμε τα εναλλακτικά σενάρια τα οποία είναι πολύ σημαντικά στην μετέπειτα πορεία της ανάπτυξης του συστήματός μας. Το να ξεχνάμε τα εναλλακτικά σενάρια είναι μία μεγάλη αστοχία στην ανάπτυξη λογισμικού. Για κάθε εναλλακτικό σενάριο, φυσικά, βεβαιωνόμαστε ότι η συμπεριφορά του συστήματος ως απάντηση στην κατάσταση που

προκαλεί είναι πλήρως εμπεριστατωμένη. Το να προσδιορίσουμε ότι μια εναλλακτική πορεία μπορεί να συμβεί είναι αναγκαίο, αλλά δεν επαρκεί για την ολοκλήρωση της περίπτωσης χρήσης μας. Εκτός από την καταχώριση των εναλλακτικών σεναρίων, είναι ζωτικής σημασίας να περιγράψουμε λεπτομερώς την ακριβή συμπεριφορά (του χρήστη και του συστήματος), του τρόπου με τον οποίο το εναλλακτικό σενάριο θα χειριστεί από το σύστημα.

5. **Βεβαιωνόμαστε ότι κάθε περίπτωση χρήσης καλύπτει και τις δύο πλευρές του διαλόγου χρήστης/σύστημα.** Ένα από τα πιο συνηθισμένα λάθη, είναι ότι γράφονται απλά μόνο τα βήματα που ακολουθεί ο χρήστης. Αυτή η λανθασμένη διαδικασία αγνοεί ένα θεμελιωδώς σημαντικό σημείο: στις περισσότερες περιπτώσεις, ο στόχος είναι να κατανοηθεί πλήρως και να καθοριστεί η συμπεριφορά του λογισμικού του συστήματος. Αν γράψουμε μόνο τις ενέργειες των χρηστών και αγνοήσουμε τη συμπεριφορά του συστήματος, δεν κάνουμε μεγάλη πρόοδο προς τον στόχο του καθορισμού της συμπεριφοράς του λογισμικού.
6. **Βεβαιωνόμαστε ότι δεν έχουμε παραβιάσει τους κανονισμούς σύνταξης της ανάλυσης ευρωστίας.** Καθ' όλη τη διάρκεια της επισκόπησης πρέπει να έχουμε στο νου μας τις εξής περιπτώσεις:
    - Οι χαρακτήρες συνδέονται μόνο με boundary objects.
    - Δεν υπάρχει επικοινωνία ουσιαστικό-ουσιαστικό, μεταξύ boundary/entity, boundary/boundary, ή entity/entity αντικειμένων χωρίς controllers μεταξύ τους. Οι controllers εκπροσωπούν την συμπεριφορά του συστήματος, έτσι θα ήταν πολύ κακό να τους αφήσουμε έξω.
7. **Βεβαιωνόμαστε ότι οι περιπτώσεις χρήσης είναι στο πλαίσιο τόσο του μοντέλου αντικειμένων όσο και του GUI.** Η μαγεία του επιπέδου αφαίρεσης που μόλις περιγράψαμε επιτυγχάνεται εύκολα με την τοποθέτηση της περίπτωση χρήσης στο πλαίσιο του μοντέλου αντικειμένου και του GUI. Στην πράξη, αυτό σημαίνει ότι έχουμε ονοματίσει τις οθόνες μας και τις domain κλάσεις μας. Η επίλυση της

ασάφειας των ονομάτων στη χρήση των domain αντικειμένων και των οθονών λύνει πολλά προβλήματα. Οι περιπτώσεις χρήσης σε αυτό το επίπεδο πρέπει επίσης να είναι στο πλαίσιο της (εξελισσόμενης) τεχνικής αρχιτεκτονικής του συστήματος, αλλά δεν πρέπει να περνούν στο λεπτομερή σχεδιασμό.

8. **Δεν πρέπει να παρασυρθούμε σε λεπτομερή σχεδιασμό.** Πρέπει να θυμόμαστε ότι το διάγραμμα ευρωστίας αντιπροσωπεύει μια εξιδανικευμένη, προκαταρκτική μελέτη, όχι τον «πραγματικό σχεδιασμό του λογισμικού". Στην πράξη, αυτό σημαίνει ότι οι αποφάσεις που σχετίζονται με την κατανομή της συμπεριφοράς μεταξύ των κλάσεων δεν πρέπει να γίνονται στα διαγράμματα ευρωστίας. Οι αποφάσεις αυτές καλύτερα αναβάλλονται μέχρι το σχεδιασμό των διαγραμμάτων ακολουθίας. Η ICONIX διαδικασία ακολουθεί την προσέγγιση δύο περασμάτων για να φτάσουμε στο λεπτομερή σχεδιασμό. Στο πρώτο πέρασμα, σκόπιμα αγνοεί "ποιος κάνει τι» και επικεντρώνεται στον εντοπισμό αντικειμένων, στην ονομασία οθονών και τη σαφή περιγραφή της συμπεριφοράς. Μόλις γίνει αυτό σωστά (και το έχουμε επαληθεύσει στη διάρκεια του PDR), είμαστε έτοιμοι να αναλάβουμε το πρόβλημα της εκχώρησης συμπεριφοράς (δηλαδή, πώς οι μέθοδοι κατανέμονται μεταξύ των κλάσεων) κατά τη διάρκεια του λεπτομερούς σχεδιασμού.

8.2 Προκαταρκτική Επισκόπηση Σχεδιασμού στην πράξη

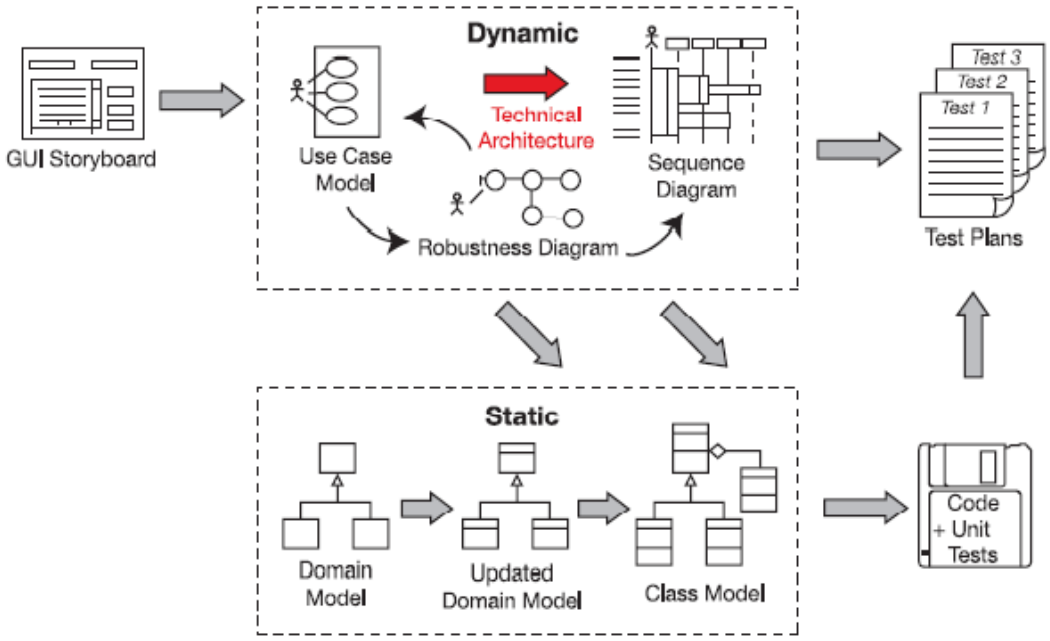
Έχοντας κατανοήσει τα βήματα των δύο τελευταίων κεφαλαίων περνάμε επαναληπτικά από όλες τις περιπτώσεις χρήσης τα διαγράμματα ευρωστίας και το domain μοντέλο και ελέγχουμε αν ακολουθούν τα βήματα αυτά και επιμελούμαστε τυχόν διορθώσεις. Είναι πολύ σημαντικό να είναι όλα σωστά πριν προχωρήσουμε στο σχεδιασμό γι’ αυτό το λόγο τονίστηκαν οι παραπάνω οδηγίες σε αυτό το κεφάλαιο. Όταν φτάσουμε στο σχεδιασμό θα είναι πλέον αργά να διορθώσουμε τα προβλήματα και ως αποτέλεσμα το σύστημά μας θα πάρει λάθος τροπή, το οποίο είναι τόσο επικίνδυνο όσο και ζημιογόνο οικονομικά.

8.3 Περίληψη

Στο κεφάλαιο αυτό, καλύψαμε το ορόσημο Προκαταρκτική Επισκόπηση Σχεδιασμού (Preliminary Design Review, PDR). Αυτό το βήμα περιλαμβάνει τη διασφάλιση ότι τα διαγράμματα ευρωστίας και το κείμενο των περιπτώσεων χρήσης ταιριάζουν μεταξύ τους και ότι και τα δύο είναι πλήρεις και εκπροσωπούν ορθά την επιθυμητή συμπεριφορά του συστήματος.

Μόλις το PDR είναι πλήρες, είμαστε έτοιμοι να προχωρήσουμε σε λεπτομερή σχεδιασμό, που καλύπτουμε στο κεφάλαιο 10. Η τεχνική αρχιτεκτονική (technical architecture, TA) είναι επίσης ένα σημαντικό βήμα. Η διατύπωση του TA αρχίζει κατά τη διάρκεια της ανάλυσης ευρωστίας, αλλά πραγματικά ξεκινά με υψηλή ταχύτητα λίγο πριν από το λεπτομερή σχεδιασμό. Έχουμε καλύψει το TA στο κεφάλαιο 9.

Κεφάλαιο 9: Τεχνική Αρχιτεκτονική (Technical Architecture, TA)



Σχήμα 9.1 Μεθοδολογία ICONIX σε λεπτομερειακή ροή εργασίας (technical architecture)

Ο σκοπός της τεχνικής αρχιτεκτονικής (ΤΑ) είναι να πάρουμε μια γενική ιδέα για το σύστημα που θα αναπτύξουμε. Θα είναι ένα web-based σύστημα; Ή ένα σύστημα πλούσιο σε πελάτες σε VB. NET ή Java Swing; Μήπως πρέπει να χρησιμοποιήσουμε ένα συγκεκριμένο πλαίσιο εφαρμογής;

Δεν υπάρχει τυποποιημένη σημειογραφία ή μορφότυπο για την τεκμηρίωση της ΤΑ. Το βάθος και η μορφή της τεχνικής αρχιτεκτονικής και των συμβάσεων για τη δημιουργία της ποικίλλουν σε μεγάλο βαθμό από εταιρία σε εταιρία. Στο κεφάλαιο αυτό, αναλύουμε τις τεχνολογίες, που αποφασίσαμε σε αυτό το στάδιο, να χρησιμοποιήσουμε στο σύστημά μας, το Διαδικτυακό Τουριστικό Γραφείο.

9.1 Τι είναι η τεχνική αρχιτεκτονική;

Η τεχνική αρχιτεκτονική (που αναφέρεται επίσης ως αρχιτεκτονική του συστήματος και αρχιτεκτονική λογισμικού) γενικά περιγράφει το σύστημα που σκοπεύουμε να χτίσουμε από την άποψη της δομής. Η αρχιτεκτονική είναι

φτιαγμένη για να ικανοποιήσει τις απαιτήσεις των επιχειρήσεων και το επίπεδο υπηρεσιών του συστήματος που πρόκειται να χτίσουμε. Η αρχιτεκτονική περιλαμβάνει (αλλά δεν περιορίζεται σε αυτά) την τοπολογία του συστήματος (οι κόμβοι server, φυσική τοποθεσία στο δίκτυο, την επιλογή του διακομιστή εφαρμογών, κ.λπ.).

Ένα καλό ΤΑ θα πρέπει να βασίζεται σε διεξοδική ανάλυση των εμπλεκόμενων "αριθμών", δηλαδή, τον αριθμό των ανθρώπων που θα χρησιμοποιούν το σύστημα ανά πάσα στιγμή, αν υπάρχουν ώρες αιχμής (και πότε είναι αυτές) , τον αριθμό των συναλλαγών ανά λεπτό, τα κριτήρια failover, και ούτω καθεξής. Οι αριθμοί αυτοί θα διαδραματίσουν έναν τεράστιο ρόλο στην απόφαση για το είδος του server εφαρμογών (ή web server) που θα πρέπει να χρησιμοποιηθεί, πόσες άδειες θα αγοραστούν και τεχνολογίες server και client θα πρέπει να χρησιμοποιηθούν.

9.2 Οδηγίες Τεχνικής Αρχιτεκτονικής

Οι αρχές που θα συζητηθούν σε αυτό το κεφάλαιο συνοψίζονται παρακάτω σε μερικές οδηγίες.

1. **Ξεχωρίζουμε τις λειτουργικές αρχιτεκτονικές, τις αρχιτεκτονικές δεδομένων και του συστήματος.** Οι αρχιτεκτονικές γενικά καλύπτουν τρεις περιοχές:
    - Το μοντέλο ανάπτυξης (διακομιστές δικτύου και εφαρμογής, και πώς ταιριάζουν μεταξύ τους, τοπολογία του συστήματος, προγράμματα περιήγησης στο Web που υποστηρίζονται κλπ.).
    - Το package/component model (διαχωρισμός των προβληματισμών σε διαφορετικά πακέτα).
    - Το μοντέλο δεδομένων.
2. **Βασίζουμε την αρχιτεκτονική στις απαιτήσεις.** Είναι δελεαστικό να βασίσουμε την αρχιτεκτονική στην τελευταία λέξη της τεχνολογίας, αντί να ακούμε τι προστάζουν οι απαιτήσεις και να λάβουμε μια αντικειμενική απόφαση που βασίζεται σε ότι χρειάζεται. Τα ζητήματα προϋπολογισμού είναι επίσης σημαντικά. Αν αποφασίσουμε για

παράδειγμα ότι, από τεχνικής άποψη, ο καλύτερος server για το έργο είναι ο «BankBreaker 8,0 Service Pack 12," είναι ικανός ο προϋπολογισμός να στηρίξει μία τέτοια αγορά; Υπάρχουν φθηνότερες (και πιο ισχυρές) εναλλακτικές λύσεις που ταιριάζουν με τις απαιτήσεις εξίσου καλά;

3. Εξετάζουμε τους παράγοντες από άποψη **επεκτασιμότητας**, **ασφάλειας** και **διαθεσιμότητας**.
4. Εξετάζουμε την **διεθνοποίηση** και την **τοπική προσαρμογή**.
5. **Θέτουμε «δύσκολα» ερωτήματα σε όλα τα άτομα που συμμετέχουν.** Ερωτήματα σχετικά με ζητήματα όπως η ασφάλεια, ο έλεγχος, η φορητότητα και η επεκτασιμότητα θα πρέπει να απαντηθούν τώρα, όχι έξι μήνες μετά το έργο.
6. Εξετάζουμε τη **δυνατότητα δοκιμών (testability).**
7. **Εξετάζουμε με ποια εξωτερικά συστήματα θα υπάρχει διασύνδεση.** Οργανώνουμε τις απαιτήσεις για οτιδήποτε αφορά τις συγχρονισμένες ή ασύγχρονες εξωτερικές αλληλεπιδράσεις του συστήματος. Για σύγχρονα συστήματα, σκεφτόμαστε εξωτερική διαθεσιμότητα του συστήματος. Είναι μια απαίτηση το σύστημά μας να μη μπορεί να λειτουργήσει χωρίς το άλλο σύστημα;

9.3 Τεχνική Αρχιτεκτονική στην πράξη

Σε αυτήν την ενότητα θα περιγράψουμε τις αρχιτεκτονικές που χρησιμοποιήθηκαν για την ανάπτυξη του συστήματος μας και είχαν οργανωθεί και αποφασιστεί σε αυτό το σημείο της ανάπτυξης του έργου μας, δηλαδή λίγο μετά την προκαταρκτική επισκόπηση σχεδιασμού και της ολοκλήρωσης των διαγραμμάτων ευρωστίας, του domain model και λίγο πριν το σχεδιασμό των διαγραμμάτων ακολουθίας.

9.3.1 Διακομιστής (Server)

Καθώς αποφασίσαμε ότι η εφαρμογή μας θα είναι διαδικτυακή έπρεπε να αποφασίσουμε και τον κατάλληλο web server που θα στήριζε την εφαρμογή αυτή και θα είναι υπεύθυνος για την αποστολή των στοιχείων μέσω του διαδικτύου.

Αποφασίσαμε λοιπόν ότι ο καταλληλότερος server για την εκπαιδευτική μας εφαρμογή είναι ο Apache Tomcat λόγω των παρακάτω πλεονεκτημάτων:

- **Προηγμένα χαρακτηριστικά.** Ο Apache Server έχει τα πιο προηγμένα χαρακτηριστικά στην αγορά των servers. Επίσης ο Apache είναι πάντα καινοτόμος και χρησιμοποιεί τα τελευταία πρωτόκολλα που χρησιμοποιούνται στο διαδίκτυο.
- **Ευελιξία.** Ο Apache Server μπορεί να προσαρμοστεί πολύ εύκολα, λόγω της αρθρωτής του δομής.
- **Ευκολία στο διαχειρισμό.** Η διαχείριση είναι ένα από τα κύρια στοιχεία κάθε είδους servers. Ο Apache Server έχει μια λίστα με αρχεία ρυθμίσεων που είναι καλά τεκμηριωμένη με όλες τις απαραίτητες πληροφορίες ώστε να μπορούμε να τα διαβάσουμε και να ενημερωθούμε για όλα τα χαρακτηριστικά και τις ρυθμίσεις του Apache Server. Επιπλέον, τα αρχεία ρυθμίσεων είναι σε μορφή ASCII και μπορούν να παραδοθούν χωρίς καμία επιπλοκή.
- **Ο Apache Server είναι ανοιχτή εφαρμογή.** Ένα από τα πλεονεκτήματα του Apache είναι ότι αποτελεί ένα επεκτάσιμο εργαλείο. Το API του Apache Server ανήκει στο Open Source Community (κοινότητα ανοιχτού κώδικα). Με άλλα λόγια, μπορούμε να προσθέσουμε ενότητες που δεν υπάρχουν στον Server Apache. Επίσης μπορούμε να γράψουμε το δικό μας κώδικα και να προσαρμόσουμε σε Apache Server τις λειτουργίες και τις βελτιώσεις που ταιριάζουν με τις ανάγκες μας. Η δημοτικότητα του Apache είναι το προϊόν της ενσωμάτωσης των χιλιάδων προγραμματιστών και προχωρημένων χρηστών ανά τον κόσμο, που προτείνουν σημαντικές αλλαγές και βελτιώσεις για τη δημιουργία ενός ισχυρού web server όπως Apache.

- **Ο Web Server του Apache είναι αποτελεσματικός.** Η αποτελεσματική χρήση του Apache είναι μια μεγάλη αρετή για έναν web server. Ο Apache Server έχει καταφέρει κάτι που δεν έχουν οι κάνει οι άλλοι διακομιστές web. Όλες οι προσπάθειες στο να έχουμε ένα Server Apache περισσότερο βελτιστοποιημένο έχουν πραγματικά πραγματοποιηθεί. Σήμερα μπορούμε να δούμε τον καρπό ενός πολύ σταθερού και ώριμου web server με μεγάλη απόδοση.
- **Φορητότητα και υποστήριξη.** Το μεγαλύτερο όφελος του Open Source Community είναι η υποστήριξη των λογισμικών τους πέντε αστέρων που δεν υπάρχει αλλού στην αγορά. Επίσης ο Apache προσφέρει μία θαυμάσια φορητότητα που μπορεί να εγκατασταθεί και να λειτουργήσει κάτω από πολλές πλατφόρμες με υψηλό επίπεδο φορητότητας. Ο Apache server χρησιμοποιείται επίσης σε κάθε hosting server.

9.3.2 Χρησιμοποίηση του Netbeans για την ανάπτυξη της εφαρμογής

Η ανάπτυξη μίας εφαρμογής διαδικτύου είναι αρκετά απαιτητική διαδικασία διότι απαιτεί την σύμπραξη πολλών διαφορετικών γλωσσών προγραμματισμού και ένα τρόπο συνδυασμού πολλών τεχνολογιών. Για παράδειγμα, μπορεί να χρειαστεί η χρησιμοποίηση της HTML και CSS για τη μορφοποίηση των σελίδων μας, JavaScript για μερικά κουμπιά rollover, Java servlets ή JSP για τη διαχείριση μίας φόρμας.

Με το Netbeans IDE δε χρειάζεται να γνωρίζουμε όλες τις γλώσσες ή το πώς συνδυάζονται οι διάφορες τεχνολογίες. Το Netbeans χειρίζεται όλες τις γλώσσες και συνδυάζει όλες τις τεχνολογίες άψογα για εμάς.

Έτσι επιλέξαμε μέσω του μενού του Netbeans τη δημιουργία ενός Web Application όπου περάσαμε και τον apache server ώστε να διαχειρίζεται τις συναλλαγές.

9.3.3 Χρησιμοποίηση του JQUERY

Το JQuery είναι μία βιβλιοθήκη JavaScript πολλαπλών διακομιστών που σχεδιάστηκε για την απλοποίηση της δέσμης ενεργειών σε HTML στην πλευρά του

πελάτη. Είναι ένα δωρεάν, ανοιχτού κώδικα λογισμικό που σχεδιάστηκε για την απλοποίηση της περιήγησης σε ένα κείμενο, τη δημιουργία animations, την επιλογή στοιχείων DOM, το χειρισμό γεγονότων και στην ανάπτυξη Ajax εφαρμογών. Εμείς τη χρησιμοποιήσαμε για το χειρισμό ενεργειών και τη δημιουργία animations.

9.3.4 Χρήση της PostgreSQL ως βάση δεδομένων

Η PostgreSQL είναι μία ισχυρή, ανοιχτού κώδικα, αντικειμενοστραφής σχεσιακή βάση δεδομένων. Έχει περισσότερα από 15 χρόνια ενεργής ανάπτυξης και μια αποδεδειγμένη αρχιτεκτονική η οποία έχει κερδίσει μια ισχυρή φήμη για την αξιοπιστία, την ακεραιότητα των δεδομένων, και την ορθότητα. Τρέχει σε όλα τα μεγάλα λειτουργικά συστήματα, συμπεριλαμβανομένου του Linux, UNIX (AIX, BSD, HP-UX, IRIX SGI, Mac OS X, Solaris, Tru64), και τα Windows. Είναι πλήρως συμβατό ACID, έχει την πλήρη στήριξη των ξένων κλειδιών (foreign keys), ενώσεων (joins), των όψεων (views), των σημείων ενεργοποίησης (triggers) και των αποθηκευμένων διαδικασιών (stored procedures) (σε πολλές γλώσσες). Περιλαμβάνει το μεγαλύτερο μέρος του SQL: 2008 τύπους δεδομένων, συμπεριλαμβανομένων των INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL και TIMESTAMP. Υποστηρίζει επίσης την αποθήκευση δυαδικών μεγάλων αντικειμένων, συμπεριλαμβανομένων εικόνων, ήχων ή βίντεο. Έχει μητρική διασυνδέσεις προγραμματισμού C / C + +, Java,. Net, Perl, Python, Ruby, Tcl, ODBC, μεταξύ άλλων, και εξαιρετική τεκμηρίωση.

Εμείς την επιλέξαμε για όλα τα παραπάνω αλλά και γιατί ταίριαξε απόλυτα στην εφαρμογή που υλοποιήσαμε, την έχουμε διδαχθεί στο τμήμα μας και είναι πολύ εύκολη στη χρήση και στην υλοποίηση. Σημαντικό ρόλο παρέχει η πληθώρα των τεκμηριώσεων και των tutorials που υπάρχουν στο διαδίκτυο και βοήθησαν πολύ στην υλοποίηση της βάσης μας.

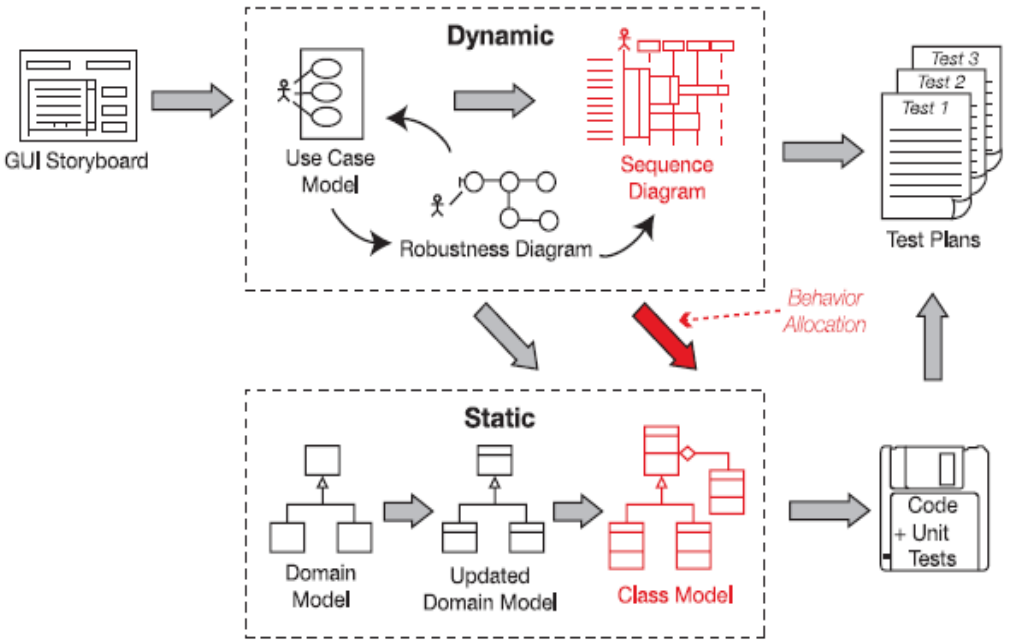
9.4 Περίληψη

Στο κεφάλαιο αυτό παρουσιάσαμε την τεχνική αρχιτεκτονική (ΤΑ) και πήραμε μία γεύση από τις τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη του προγράμματός μας. Αναφέρουμε ότι το ΤΑ διαφέρει από έργο σε έργο και δεν είναι

ο κεντρικός στόχος της ICONIX. Παρ' όλα αυτά είναι ένα καλό σημείο να επιλέξουμε τις τεχνολογίες και τις αρχιτεκτονικές που θα χρησιμοποιήσουμε για την ανάπτυξη της εφαρμογής μας. Μία ανάσα πριν το λεπτομερή σχεδιασμό.

Στο επόμενο κεφάλαιο αναπτύσσουμε το λεπτομερή σχεδιασμό και σχεδιάζουμε τα διαγράμματα ακολουθίας του Online Τουριστικού Γραφείου μας.

Κεφάλαιο 10: Διαγράμματα Ακολουθίας (Sequence Diagrams)



Σχήμα 10.1 Μεθοδολογία ICONIX λεπτομερειακή ροή εργασίας (Sequence Diagram)

Μόλις ολοκληρώσαμε την ανάλυση ευρωστίας, και έχουμε διεξάγει την προκαταρκτική επισκόπηση σχεδιασμού, είναι η ώρα να ξεκινήσει η αναλυτική προσπάθεια σχεδιασμού. Ως τώρα, το κείμενο των περιπτώσεων χρήσης μας πρέπει είναι πλήρες, ορθό, λεπτομερές και σαφές. Εν ολίγοις, οι περιπτώσεις χρήσης μας πρέπει να είναι σε μια κατάσταση, όπου να μπορούμε να δημιουργήσουμε ένα λεπτομερή σχεδιασμό από αυτές.

Όλα τα στάδια της διαδικασίας μέχρι στιγμής έχουν προετοιμάσει τις περιπτώσεις χρήσης για την αναλυτική δραστηριότητα του σχεδιασμού. Έχοντας ολοκληρώσει την ανάλυσή ευρωστίας και το PDR, θα πρέπει τώρα να έχουμε ανακαλύψει λίγο πολύ όλες τις domain κλάσεις που πρόκειται να χρειαστούμε.

10.1 Διαγράμματα Ακολουθίας και λεπτομερείς αντικειμενοστραφής σχεδιασμός (object-oriented design, OOD)

Αν θεωρήσουμε ότι ο προκαταρκτικός σχεδιασμός έχει να κάνει με την ανακάλυψη των κλάσεων (γνωστός και ως ανακάλυψη αντικειμένων), τότε ο λεπτομερής σχεδιασμός έχει να κάνει, αντιθέτως, σχετικά με την κατανομή συμπεριφοράς (γνωστός και ως κατανομή συμπεριφοράς), δηλαδή την κατανομή των λειτουργιών του λογισμικού που έχουν εντοπιστεί στο σύνολο των κλάσεων που ανακαλύφθηκαν κατά τη διάρκεια του προκαταρκτικού σχεδιασμού.

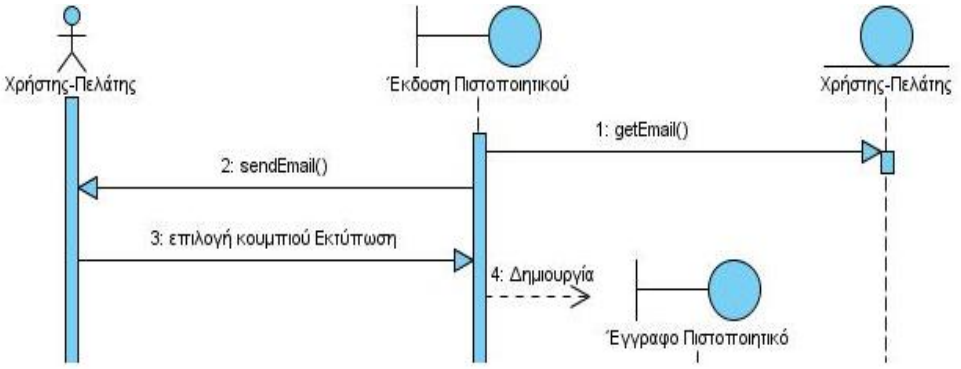
Όταν σχεδιάζουμε ένα διάγραμμα ακολουθίας, σαρώνουμε ακόμη μία φορά τον προκαταρκτικό σχεδιασμό, προσθέτοντας λεπτομέρειες.

Με τον προκαταρκτικό σχεδιασμό, κάναμε κάποιες ανεπίσημες πρώτες υποθέσεις για το πώς οι κλάσεις θα αλληλεπιδρούν μεταξύ τους. Τώρα ήρθε η ώρα να κάνουμε αυτές τις δηλώσεις ακριβείς, για να τις μετατρέψουμε σε λεπτομερή σχεδιασμό που λειτουργεί στο πλαίσιο της ΤΑ που έχουμε ορίσει.

Υπάρχει άμεση σχέση μεταξύ κάθε περίπτωση χρήσης, διαγράμματος ευρωστίας, καθώς και διαγραμμάτων ακολουθίας. Ακριβώς όπως σχεδιάσαμε ένα διάγραμμα ευρωστίας ανά περίπτωση χρήσης, θα σχεδιάσουμε επίσης ένα διάγραμμα ακολουθίας για κάθε περίπτωση χρήσης.

10.2 Σημειογραφία Διαγράμματος Ακολουθίας

Πριν αναλύσουμε τις βέλτιστες πρακτικές για την κατάρτιση διαγραμμάτων ακολουθίας από τις περιπτώσεις χρήσης, θα μας βοηθήσει να καταλάβουμε τα στοιχεία από τα οποία αποτελείται ένα διάγραμμα ακολουθίας (βλέπε Σχήμα 10.2)



Σχήμα 10.2 Σημειογραφία Διαγραμμάτων Ακολουθίας

Τα αντικείμενα στην κορυφή του διαγράμματος (Χρήστης-Πελάτης, Έκδοση Πιστοποιητικού, κ.λπ.) αλληλεπιδρούν μεταξύ τους με το πέρασμα μηνυμάτων εμπρός και πίσω. Οι κατακόρυφες διακεκομμένες γραμμές (ή γραμμές ζωής αντικειμένων) αντιπροσωπεύουν το χρόνο, οπότε η διαδικασία ξεκινάει με το πρώτο μήνυμα που βρίσκεται και πάνω πάνω (η σελίδα Έκδοση Πιστοποιητικού παίρνει το email (getEmail()) από το αντικείμενο Χρήστης-Πελάτης).

Ένας χαρακτήρας (ο Χρήστης-Πελάτης στην εικόνα 10.2) είναι ο χρήστης του οποίου την αλληλεπίδραση με το σύστημα έχουμε καθορίσει σε καθεμία από τις περιπτώσεις χρήσης. (Βλέπε «όρια του συστήματος» στο σχήμα 5.3.) Αναγνωρίζουμε τα εικονίδια των boundary objects και των entity objects από το διάγραμμα ευρωστίας του κεφαλαίου 7. (Στο Σχήμα 10.2, τα boundary objects είναι η Έκδοση Πιστοποιητικού και το Έγγραφο Πιστοποιητικό, ενώ το boundary object είναι ο Χρήστης-Πελάτης.)

Ωστόσο, παρατηρούμε ότι δεν υπάρχουν αντικείμενα controller στο διάγραμμα ακολουθίας (αν και θα μπορούσαν να υπάρχουν). Αυτό συμβαίνει γιατί όταν σχεδιάζουμε τα διαγράμματα ακολουθίας, οι controllers (τα ρήματα) μετατρέπονται σε μηνύματα στα boundary και entity αντικείμενα (τα ουσιαστικά).

Το επίκεντρο του ελέγχου, δηλαδή το ορθογώνιο κουτί που ακολουθεί κάθε στοιχείο των διαγραμμάτων ακολουθίας, αντιπροσωπεύει το χρόνο που μία συγκεκριμένη λειτουργία/μέθοδος έχει τον έλεγχο. Ξεκινάει με το βέλος που κατευθύνεται προς τη λειτουργία και τελειώνει όταν η λειτουργία επιστρέφει κάποια τιμή. Όπως θα δούμε παρακάτω είναι καλύτερα να απενεργοποιούμε στην αρχή το επίκεντρο ελέγχου γιατί αποσπά την προσοχή από το στόχο του σχεδιασμού.

10.3 Οδηγίες Διαγραμμάτων Ακολουθίας

Στην ενότητα αυτή, δείχνουμε πώς χρησιμοποιούμε τα διαγράμματα ακολουθίας ως μηχανισμό για την εξερεύνηση και τη συμπλήρωση του λεπτομερούς αντικειμενοστραφή σχεδιασμού για κάθε περίπτωση χρήσης. Και ξεκινάμε αναλύοντας μία λίστα με οδηγίες.

1. **Κατανόηση του γιατί σχεδιάζουμε διαγράμματα ακολουθίας.** Κατά την κατάρτιση των διαγραμμάτων ακολουθίας, εξερευνούμε σχολαστικά τα μέσα και έξω του λεπτομερούς σχεδιασμού για κάθε

περίπτωση χρήσης, σε μικροσκοπική λεπτομέρεια. Αυτό σημαίνει εξερεύνηση όχι μόνο του βασικού σεναρίου, αλλά και όλων των εναλλακτικών σε κάθε περίπτωση χρήσης. Είναι εκπληκτικό πόσα θέματα σχεδιασμού μπορεί να αλιευθούν σε αυτή τη φάση, εξοικονομώντας χρόνο από τον επανασχεδιασμό που θα προέκυπτε αργότερα. Τα διαγράμματα ακολουθίας έχουν πρωταρχικούς σκοπούς στη διαδικασία ICONIX:

- **Κατανομή συμπεριφοράς στις κλάσεις.** Εντοπίσαμε αυτές τις κλάσεις κατά τη διάρκεια της ανάλυσης ευρωστίας. Κατά τη διάρκεια της ακολουθίας διαγραμμάτων, οι controllers (επίσης ανακαλύπτονται κατά την ανάλυση ευρωστίας) μετατρέπονται σε λειτουργίες για τις κλάσεις. Ωστόσο, δεν χρειάζεται κατ' ανάγκην να καταλήγουν με σχέση 1:1 αντιστοιχία μεταξύ των controller και των λειτουργιών. Συχνά, ένας controller μετατρέπεται σε δύο ή περισσότερες λειτουργίες. Και όπως είπαμε και νωρίτερα, περιστασιακά ένας controller μπορεί επίσης να μετατραπεί σε μια controller κλάση.
- **Δείχνουμε λεπτομερειακά πως οι κλάσεις αλληλεπιδρούν μεταξύ τους σε όλη τη διάρκεια των περιπτώσεων χρήσης.** Κατά τη σχεδίαση των διαγραμμάτων ακολουθίας, θα πρέπει να διερευνούμε το πώς το σύστημα θα πραγματοποιήσει τη συμπεριφορά που περιγράφεται στις περιπτώσεις χρήσης μας. Αυτό επιτυγχάνεται καθώς σκεφτόμαστε και στη συνέχεια απεικονίζουμε τον τρόπο που τα αντικείμενα μας θα αλληλεπιδρούν μεταξύ τους κατά την εκτέλεση.
- **Οριστικοποιούμε την κατανομή των λειτουργιών μεταξύ των κλάσεων.** Αφού πραγματοποιηθεί η ανάλυση ευρωστίας, θα πρέπει ως τώρα να έχουν εντοπιστεί τουλάχιστον τα τρία τέταρτα των χαρακτηριστικών (attributes) (τα δεδομένα) σχετικά με τις κλάσεις μας, αλλά πολύ λίγες, ενδεχομένως, από τις λειτουργίες (operations) (συμπεριφορές). Αυτό συμβαίνει γιατί κατά τον προκαταρκτικό σχεδιασμό, τα στοιχεία αυτά δεν

υπήρχαν ώστε να κατανείμουμε τις λειτουργίες χωρίς να μαντέψουμε. Ωστόσο, τώρα που είμαστε στο στάδιο του λεπτομερούς σχεδιασμού, θα πρέπει να έχουμε τα πάντα σε θέση για να κατανείμουμε με ακρίβεια τη συμπεριφορά μεταξύ των κλάσεων μας.

2. **Φτιάχνουμε ένα διάγραμμα ακολουθίας για κάθε περίπτωση χρήσης.**
3. **Δείχνουμε πως η συμπεριφορά των περιπτώσεων χρήσης επιτυγχάνεται από τα αντικείμενα.** Οι controllers σε ένα διάγραμμα ευρωστίας γενικά παρουσιάζουν τις "λογικές" λειτουργίες του λογισμικού. Κάθε λογική λειτουργία μπορεί να επιτευχθεί με ένα ή περισσότερα μηνύματα μεταξύ των αντικειμένων που θα εμφανιστούν στο διάγραμμα ακολουθίας. Όπως σχεδιάζουμε τα μηνύματα, πραγματικά κατανέμουμε λειτουργίες σε κλάσεις, έτσι ώστε το διάγραμμα ακολουθίας να δείχνει την κατανομή της συμπεριφοράς μεταξύ των συνεργαζόμενων αντικειμένων.
4. **Παρουσιάζουμε το κείμενο των περιπτώσεων χρήσης στο διάγραμμα ακολουθίας.** Το κείμενο της περίπτωσης χρήσης που εμφανίζεται στο αριστερό περιθώριο του διαγράμματος ακολουθίας είναι πραγματικά μια σύμβαση μεταξύ των πελατών και των προγραμματιστών. Θα πρέπει να προσδιορίζει σαφώς τις απαιτήσεις συμπεριφοράς κατά το χρόνο εκτέλεσης που ικανοποιούν τις ανάγκες του πελάτη. Δεδομένου ότι έχουμε την παρούσα σύμβαση, είναι λογικό να χρησιμοποιήσουμε το διάγραμμα ακολουθίας για να αποτυπώσουμε γραφικά πώς ο σχεδιασμός ικανοποιεί τις απαιτήσεις συμπεριφοράς.
5. **Δεν πρέπει να ξοδεύουμε πολύ χρόνο για το επίκεντρο ελέγχου.**
6. **Αναθέτουμε λειτουργίες (operations) στις κλάσεις ενώ σχεδιάζουμε τα μηνύματα.** Είναι σημαντικό το διάγραμμα ακολουθίας να οδηγεί την κατανομή λειτουργιών στις κλάσεις.
7. **Επανεξετάζουμε το διάγραμμα κλάσεως (class diagram) καθώς εκχωρούμε λειτουργίες στις κλάσεις.** Εφόσον αναθέτουμε ενεργά

λειτουργίες σε κλάσεις, και δεδομένου ότι είναι πολύ εύκολο να κάνουμε λάθη ενώ σχεδιάζουμε τα διαγράμματα ακολουθίας, θα πρέπει συνεχώς ελέγχουμε ότι τα διαγράμματα ακολουθίας μας διασταυρώνονται με το διάγραμμα κλάσης μας για να είμαστε βέβαιοι ότι όταν αναθέτουμε μια λειτουργία σε μια κλάση, βγάζει νόημα.

8. **Καθαρίζουμε το στατικό μοντέλο πριν περάσουμε στην κρίσιμη επισκόπηση σχεδιασμού (critical design review, CDR).** Κοιτάμε προσεκτικά το στατικό μοντέλο, προσπαθώντας να τακτοποιήσουμε το σχεδιασμό, να επιλύσουμε θέματα σχεδιασμού του πραγματικού κόσμου, να εντοπίσουμε χρήσιμα σχεδιαστικά πρότυπα που μπορούν να συνυπολογίζονται για τη βελτίωση του σχεδιασμού, και ούτω καθεξής. Αυτό θα πρέπει τουλάχιστον να γίνει μέχρι το τελικό βήμα πριν προχωρήσουμε στο CDR. Αφού ολοκληρώσουμε τον προκαταρκτικό σχεδιασμό, το διάγραμμα κλάσεως θα πρέπει να έχει λιγότερες τρύπες από πριν την ανάλυση ευρωστίας. Οι πιθανότητες είναι ότι βρέθηκαν κάποιες πρόσθετες κλάσεις που έλειπαν από το domain model και πολλές από τις κλάσεις έχουν γεμίσει με λειτουργίες. Όμως ακόμα και αν σκεφτούμε ότι έχουμε κατανείμει τη συμπεριφορά του συστήματος στις κλάσεις, υπάρχει ακόμα πρόσθετη δουλειά μέχρι να φτάσουμε στον κώδικα.

10.4 Διαγράμματα Ακολουθίας στην πράξη

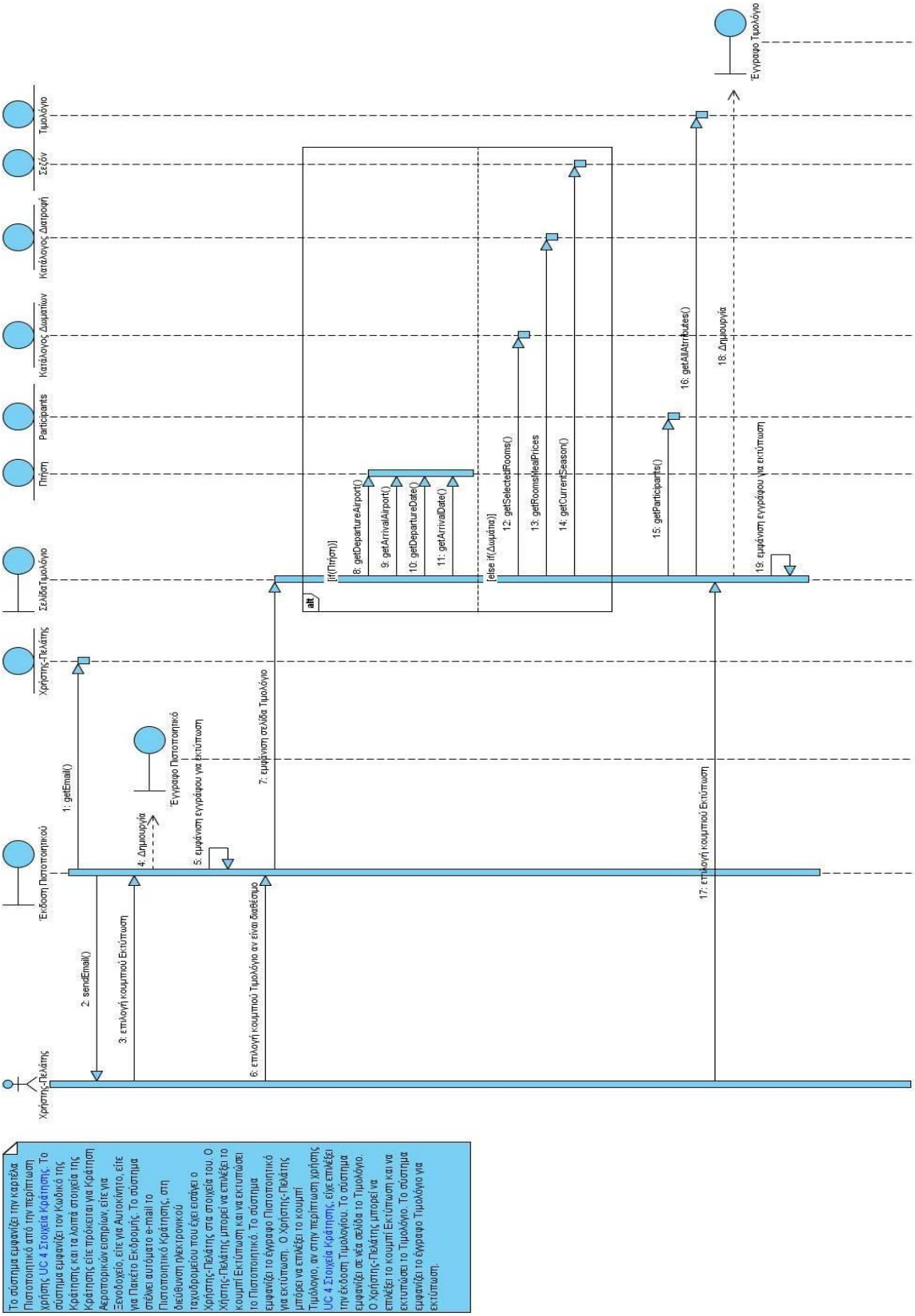
Αφού αναλύσαμε τις οδηγίες για το πώς θα σχεδιάσουμε ένα σωστό διάγραμμα ακολουθίας ήρθε η ώρα του σχεδιασμού. Στο Σχήμα 10.3 παρουσιάζουμε ολόκληρο το διάγραμμα ακολουθίας Έκδοσης Πιστοποιητικού Κράτησης, που απόσπασμά του χρησιμοποιήσαμε στο Σχήμα 10.2.

Επίσης παρουσιάζουμε μέρος του διαγράμματος κλάσης, στο Σχήμα 10.4. (Ολόκληρο το διάγραμμα κλάσης μπορεί να βρεθεί στο συνοδευτικό cd στο τέλος του βιβλίου.)

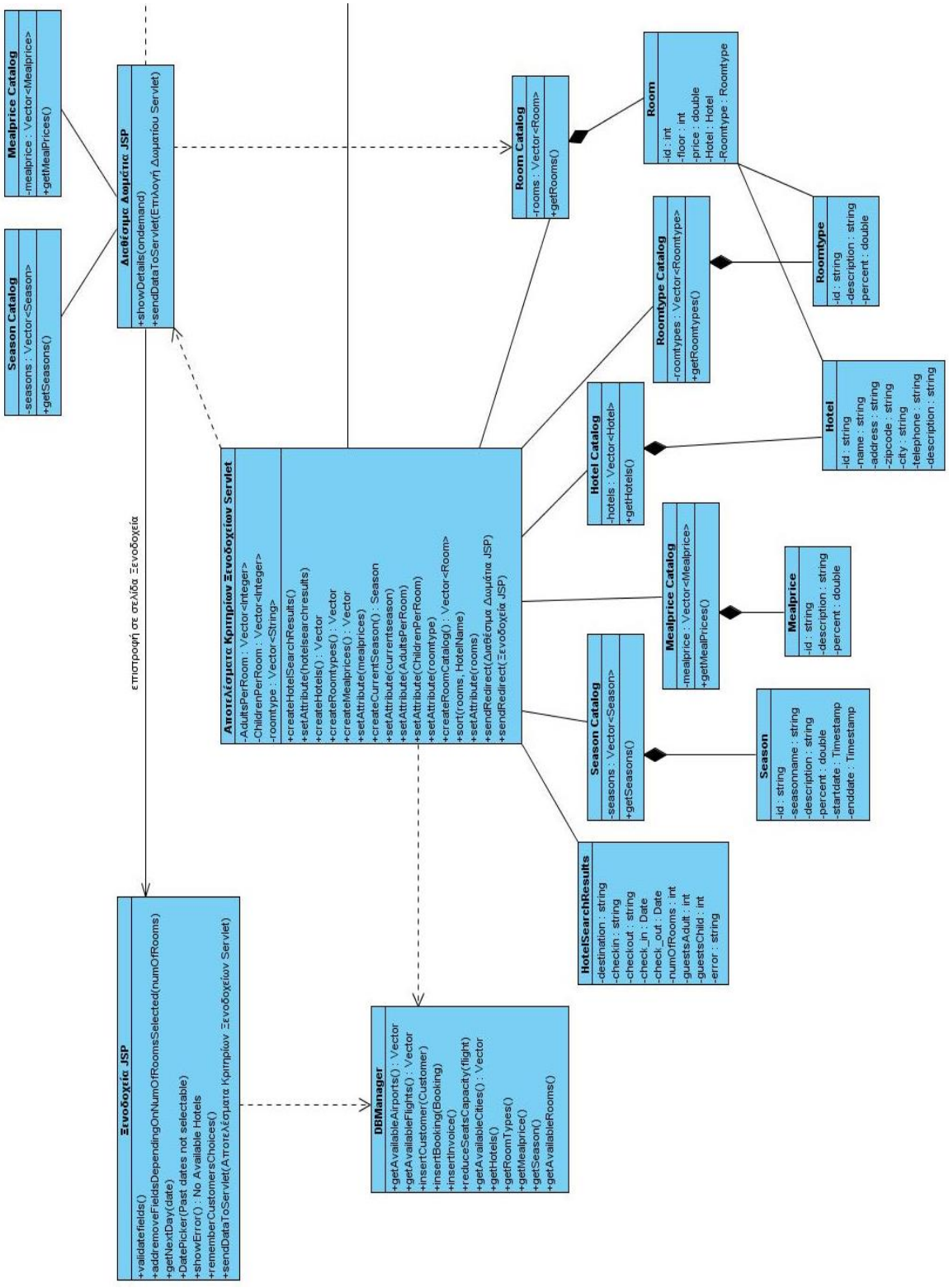
10.5 Περίληψη

Σε αυτό το κεφάλαιο καλύψαμε το λεπτομερή σχεδιασμό. Μόλις σχεδιάσουμε όλα τα διαγράμματα ακολουθίας για όλες τις περιπτώσεις χρήσης και ανανεώσουμε το στατικό μοντέλο μας, τότε μπορούμε να πούμε ότι τελειώσαμε με αυτό το στάδιο.

Τώρα είμαστε σχεδόν έτοιμοι να ξεκινήσουμε τη σύνταξη του κώδικα. Υπάρχει μία τελευταία στάση πριν την κωδικοποίηση, η Κρίσιμη Επισκόπηση Σχεδιασμού (Critical Design Review, CDR), που καλύπτουμε στο επόμενο Κεφάλαιο. Είναι ένα σημαντικό βήμα καθώς αποτελεί έναν έλεγχο πραγματικότητας για το σχεδιασμό.

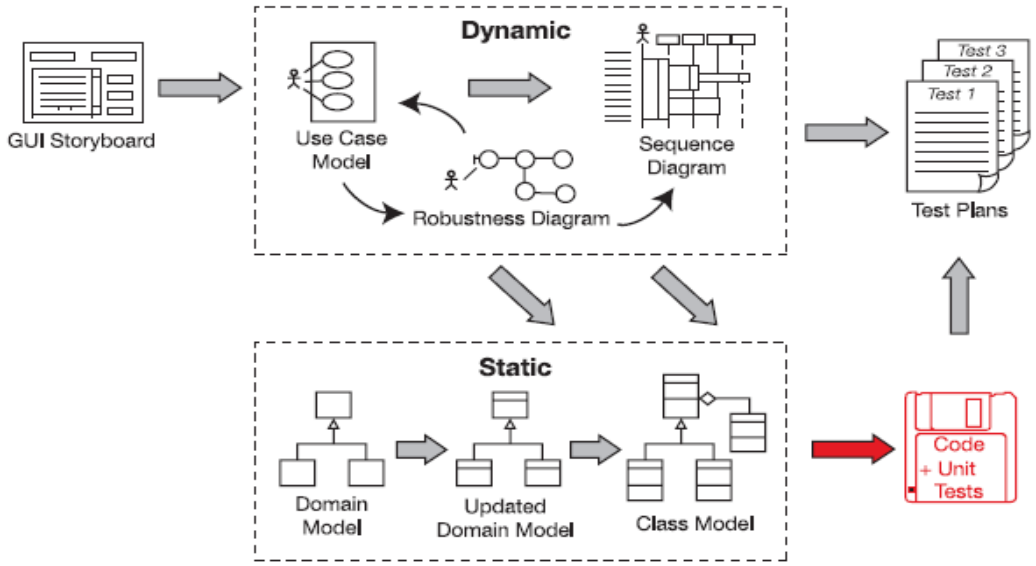


Σχήμα 10.3 Διάγραμμα Ακολουθίας Έκδοση Πιστοποιητικού Κράτησης



Σχήμα 10.4 Διάγραμμα Κλάσης (Class Diagram)

Κεφάλαιο 11: Κρίσιμη Επισκόπηση Σχεδιασμού (Critical Design Overview, CDR)



Σχήμα 11.1 Μεθοδολογία ICONIX σε λεπτομερειακή ροή εργασίας (CDR)

Το έργο μας θα πρέπει τώρα να είναι σε πολύ καλύτερη κατάσταση. Μέχρι τώρα, έχουμε χρησιμοποιήσει την ανάλυση ευρωστίας για να αποσαφηνίσουμε το κείμενο που αφορά τις περιπτώσεις χρήσης και να ανακαλύψουμε τις domain κλάσεις που λείπουν, έχουμε διεξάγει προκαταρκτική επισκόπηση σχεδιασμού (PDR) για να βεβαιωθούμε ότι οι περιπτώσεις χρήσης ταιριάζουν με αυτό που ο πελάτης θέλει πραγματικά, και έχουμε δημιουργήσει προσεκτικά ένα λεπτομερή σχεδιασμό για τις περιπτώσεις χρήσης που θα υλοποιήσουμε σε αυτή την έκδοση.

Έτσι, είμαστε σχεδόν έτοιμοι να ξεκινήσουμε την κωδικοποίηση. Υπάρχει μόνο ένα γρήγορο (αλλά ζωτικής σημασίας) ορόσημο: η Κρίσιμη Επισκόπηση Σχεδιασμού (Critical Design Review, CDR).

11.2 Οδηγίες για την Κρίσιμη Επισκόπηση Σχεδιασμού

Οι αρχές που διέπουν αυτό το κεφάλαιο μπορούν να συνοψιστούν στην ακόλουθη λίστα οδηγιών.

1. **Βεβαιωνόμαστε ότι το διάγραμμα ακολουθίας ταιριάζει με το αντίστοιχο κείμενο περίπτωσης χρήσης.** Θα πρέπει να είμαστε σε θέση να εντοπίσουμε από τις απαιτήσεις συμπεριφοράς (στο αριστερό περιθώριο) πώς οι απαιτήσεις αυτές θα εφαρμοστούν από τα μηνύματα που αποστέλλονται μεταξύ των αντικειμένων. Το διάγραμμα ακολουθίας θα πρέπει να παρέχει ένα οπτικό ίχνος των απαιτήσεων με μια ματιά. Δηλαδή, η σαφής απαιτήσεις συμπεριφοράς πρέπει να είναι ορατές στο αριστερό περιθώριο του διαγράμματος, και κατευθείαν προς τα δεξιά της κάθε πρότασης θα πρέπει να είναι τα αντικείμενα/μηνύματα που θα υλοποιήσουν τις απαιτήσεις αυτές. Έτσι θα πρέπει να είναι αρκετά απλό να ελέγχουμε τις απαιτήσεις, ενώ κοιτάζουμε το λεπτομερή σχεδιασμό. Η χρησιμοποίηση της επισήμανσης είναι ένα πολύ χρήσιμο εργαλείο όπως αναφέραμε και στο Ορόσημο 1.
2. **Βεβαιωνόμαστε ότι οι λειτουργίες έχουν ανατεθεί κατάλληλα στις κλάσεις.** Βεβαιωνόμαστε ότι κάθε μία από τις κλάσεις μας έχει μια συνεκτική, εστιασμένη σειρά δράσεων.
3. **Επανεξετάζουμε τα χαρακτηριστικά (attributes) και τις λειτουργίες (operations) των κλάσεων μας.** Θα πρέπει να επανεξετάσουμε τις κλάσεις στο διάγραμμα κλάσεων μας για να βεβαιωθούμε ότι έχουν όλες ένα κατάλληλο σύνολο χαρακτηριστικών και λειτουργιών. Ο ευκολότερος τρόπος να γίνει αυτό είναι η συνεχής αναπήδηση πέρα δώθε μεταξύ των διαγραμμάτων ακολουθίας και ενός αναλυτικού διαγράμματος κλάσεως που εμφανίζει όλα τα χαρακτηριστικά και τις λειτουργίες για τις κλάσεις.
4. **Βεβαιωνόμαστε ότι τα επιλεγμένα πρότυπα σχεδίασης βρίσκονται στα διαγράμματα ακολουθίας μας.** Αν και υπάρχουν ορισμένες εξαιρέσεις, η γενική κατευθυντήρια γραμμή εδώ είναι ότι το διάγραμμα ακολουθίας θα

πρέπει να δείχνει το "πραγματικό σχέδιο", όπως σκοπεύουμε να το υλοποιήσουμε με κώδικα.

5. **Επισημαίνουμε τις απαιτήσεις στις περιπτώσεις χρήσης και στις κλάσεις μας.** Επισημαίνουμε τις λειτουργικές (και μη λειτουργικές) απαιτήσεις στις περιπτώσεις χρήσης και τις κλάσεις μας για να βεβαιωθούμε ότι τις έχουμε καλύψει όλες. Ρυθμίζουμε το επίπεδο διατύπωσης σε ό, τι είναι κατάλληλο για το έργο και τον οργανισμό μας, αλλά αν θέλει κάποιος να εντοπίσει τις απαιτήσεις για το σχεδιασμό, αυτή είναι η καλύτερη στιγμή να το κάνει.
6. **Έλεγχος ορθότητας (επιστρεφόμενες τιμές, τύποι κτλ.).** Βεβαιωνόμαστε ότι όλα τα χαρακτηριστικά μας έχουν σωστά πληκτρολογηθεί, και ότι οι επιστρεφόμενες τιμές και οι λίστες παραμέτρων σχετικά με τις λειτουργίες/μεθόδους μας, είναι πλήρεις και ακριβείς.
7. **Δημιουργούμε τον κώδικα των επικεφαλίδων και τον ελέγχουμε προσεκτικά.**

11.3 Χρησιμοποίηση του Διαγράμματος Κλάσεως για τον εντοπισμό σφαλμάτων στα Διαγράμματα Ακολουθίας

Μια τεχνική που λειτουργεί πολύ καλά κατά τη διάρκεια του CDR είναι, όπως δείχνει ο τίτλος αυτής της ενότητας, η χρησιμοποίηση του διαγράμματος κλάσης για την ανεύρεση λαθών στα διαγράμματα ακολουθίας. Βασικά, αυτό συνεπάγεται την εστίαση σε διάφορες κλάσεις στο διάγραμμα κλάσης και την εύρεση μεθόδων σε λάθος κλάσεις, ή άλλες ανωμαλίες, και στη συνέχεια την εξεύρεση των ένοχων διαγραμμάτων ακολουθίας. Μπορούμε να βρούμε περισσότερα λάθη στα διαγράμματα ακολουθίας κοιτάζοντας το διάγραμμα κλάσης.

11.4 Περίληψη

Στο κεφάλαιο αυτό, εξετάσαμε την κρίσιμη επισκόπηση σχεδιασμού (CDR), ένα σημαντικό βήμα που πραγματοποιείται μεταξύ του λεπτομερούς σχεδιασμού

και της υλοποίησης. Όπως συζητήσαμε στην αρχή του κεφαλαίου αυτού, το CDR περιλαμβάνει τρεις βασικούς στόχους:

- Προσαρμογή του κειμένου περιπτώσεως χρήσης σε κάθε διάγραμμα ακολουθίας.
- Έλεγχος συνέχειας των μηνυμάτων.
- Επανέλεγχος για καλό σχεδιασμό.

Αν περάσαμε από τον λεπτομερή σχεδιασμό για κάθε περίπτωση χρήσης και τον επανεξετάσαμε για τα τρία αυτά κριτήρια, τότε το σχέδιό μας πρέπει πραγματικά να είναι έτοιμο για την κωδικοποίηση.

Στο επόμενο κεφάλαιο, υπεισερχόμαστε στην υλοποίηση. Αν έχουμε το λεπτομερή σχεδιασμό μας σωστό (όπως επαληθεύεται από το ορόσημο του CDR), τότε η υλοποίηση θα πρέπει να είναι μια σχετικά σύντομη και απλή διαδικασία.

Κεφάλαιο 12: Υλοποίηση (Πως μοιάζει το σύστημά μας)

Αφού ξοδέψαμε αρκετά κεφάλαια και χρόνο για να φτιάξουμε έναν πλήρη και άρτιο σχεδιασμό περνώντας από τις περιπτώσεις χρήσης, στην ανάλυση ευρωστίας και στο σχεδιασμό, ελέγχοντας κάθε στάδιο με το αντίστοιχο ορόσημό του, είμαστε σε θέση να ξεκινήσουμε την κωδικοποίηση.

Στο κεφάλαιο 1, περιγράψαμε με απλά παραδείγματα τι ενέργειες εκτελούν τα στοιχεία της εφαρμογής μας (JSP, servlets, κτλ.) και είδαμε και παραδείγματα από το πώς υλοποιούνται τα στοιχεία αυτά. Ο πλήρης κώδικας της εφαρμογής μας βρίσκεται στο συνοδευτικό cd του βιβλίου αυτού, που βρίσκεται στην τελευταία σελίδα.

Εδώ θα δείξουμε κάποιες εικόνες του συστήματος και συνοπτικά τις ενέργειες που εκτελούνται τόσο από τον χρήστη όσο και από το σύστημα.

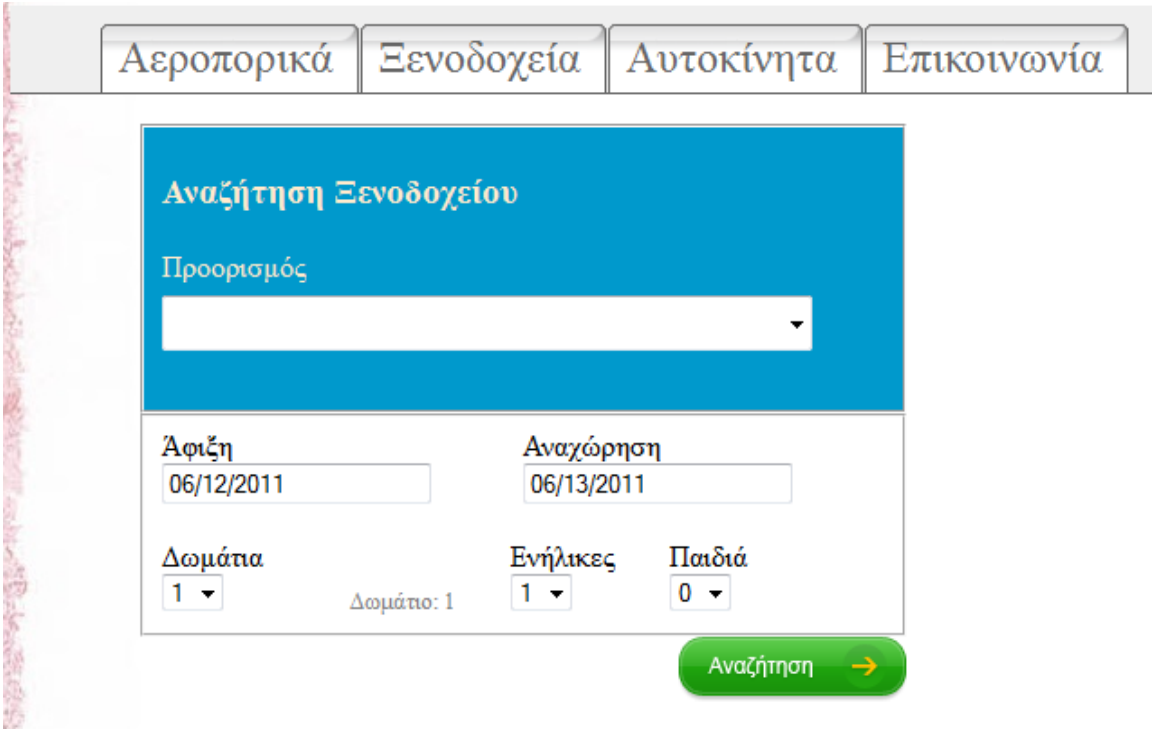
12.1 Κράτηση Δωματίου

Παρακάτω θα αναλύσουμε εν συντομία την κράτηση δωματίου σε ξενοδοχειακή μονάδα μέσω ενδεικτικών εικόνων από το σύστημα μας, «Online Τουριστικό Γραφείο».

Το Σχήμα 12.1 απεικονίζει την πρώτη σελίδα για την κράτηση δωματίου, σε αυτή τη σελίδα ο Χρήστης-Πελάτης μπορεί να επιλέξει από μία σειρά προορισμών, από το drop down μενού Προορισμός. Επίσης επιλέγει τις επιθυμητές ημερομηνίες check in και check out στα πεδία Άφιξη και Αναχώρηση και τέλος εισάγει σε πόσα δωμάτια επιθυμεί να κάνει κράτηση, καθώς και πόσοι ενήλικες και πόσα παιδιά θα βρίσκονται σε κάθε δωμάτιο. Τέλος επιλέγει το κουμπί Κράτηση.

Γίνεται έλεγχος των κριτηρίων και το σύστημα από την πλευρά του δημιουργεί το HotelSearchResultsBean με τα κριτήρια που εισήγαγε ο Χρήστης-Πελάτης για την κράτηση. Μεταβαίνουμε στο κατάλληλο servlet που χειρίζεται το request του Χρήστη-Πελάτη όπου δημιουργούνται τα απαραίτητα αντικείμενα και

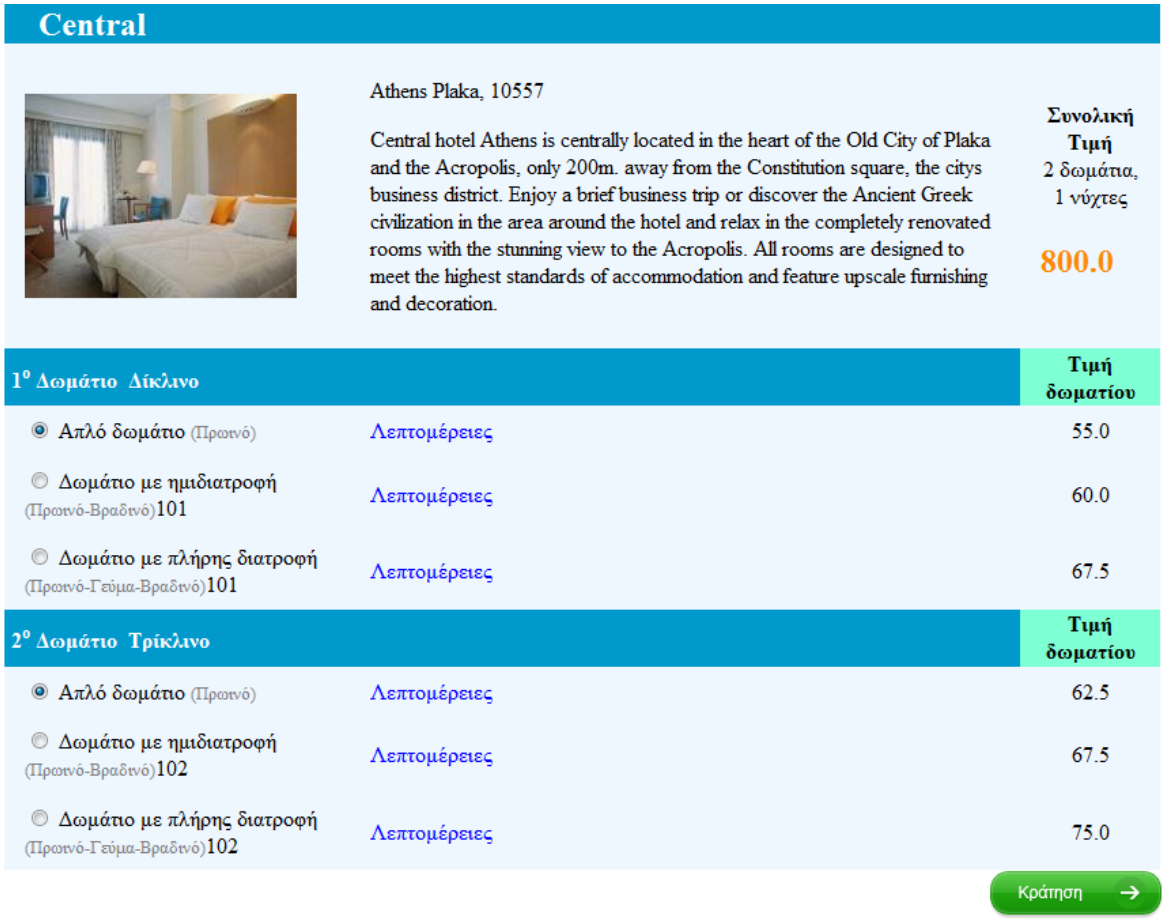
δημιουργείται η σύνδεση με τη βάση για τον εντοπισμό των διαθέσιμων δωματίων σύμφωνα με τα κριτήρια που εισήγαγε ο Χρήστης-Πελάτης. Επίσης αποθηκεύονται τα απαραίτητα στοιχεία στα session αντικείμενα ώστε να είναι διαθέσιμα ανά πάσα στιγμή κατά τη διάρκεια της συνεδρίας (session) του Χρήστη-Πελάτη. Επίσης περνάει τον έλεγχο στη σελίδα JSP που εμφανίζει τα αποτελέσματα της Αναζήτησης δείχνοντας τα διαθέσιμα δωμάτια σε ξενοδοχεία, σύμφωνα με τα κριτήρια του Χρήστη-Πελάτη.



Σχήμα 12.1 Επιλογή χαρακτηριστικών κράτησης δωματίου

Το σχήμα 12.2 παρουσιάζει τη σελίδα με τα διαθέσιμα δωμάτια, σύμφωνα με τις προτιμήσεις του Χρήστη-Πελάτη. Εμφανίζονται τα διαθέσιμα ξενοδοχεία που παρέχουν δωμάτια σύμφωνα με τα κριτήρια, ταξινομημένα κατά τιμή και όνομα. Σε κάθε ξενοδοχειακή μονάδα εμφανίζεται μία επιλογή δωματίου για κάθε δωμάτιο που έχει επιλέξει να κάνει κράτηση ο Χρήστης-Πελάτης. Για κάθε διαθέσιμο δωμάτιο εμφανίζονται τρεις επιλογές, όπου ο Χρήστης-Πελάτης επιλέγει το δωμάτιο που επιθυμεί ανάλογα με τη διατροφή που επιθυμεί (μόνο πρωινό, ημιδιατροφή ή πλήρης διατροφή) με ανάλογη διαφορά στην τιμή. Επίσης ο Χρήστης-Πελάτης μπορεί να επιλέξει το σύνδεσμο Λεπτομέρειες, κάτω από κάθε

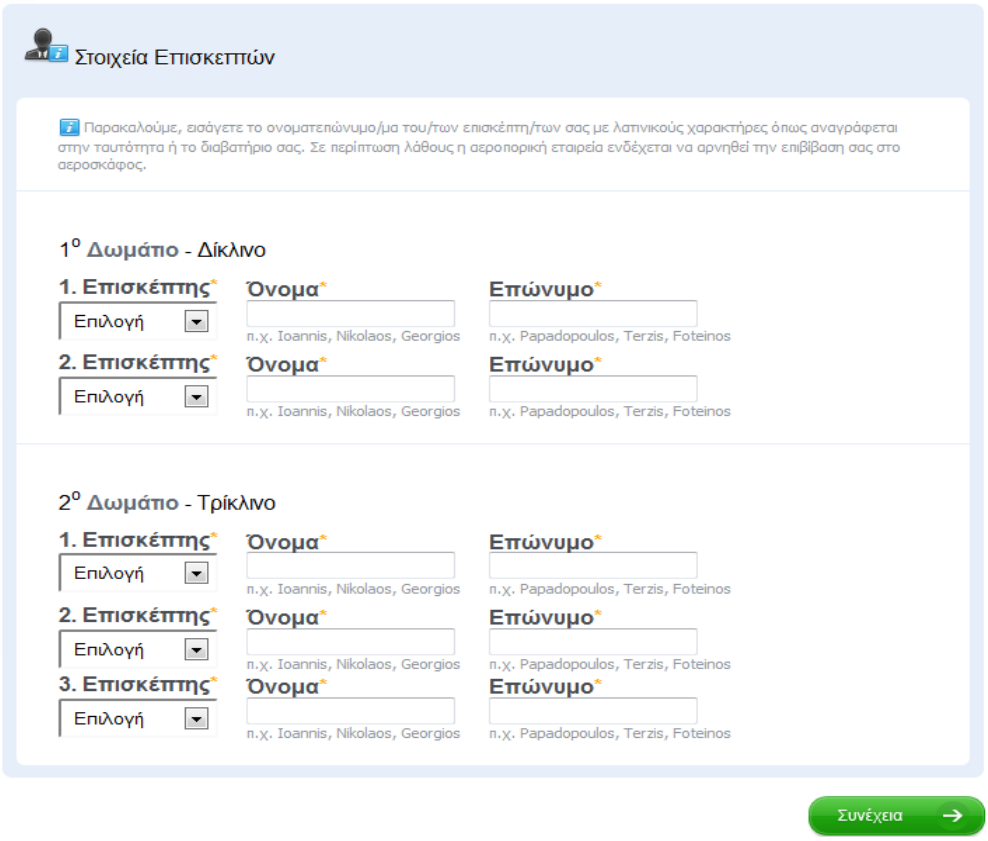
δωμάτιο και να ενημερωθεί για θέματα που αφορούν το δωμάτιο. Όταν ο Χρήστης-Πελάτης επιλέξει το/τα επιθυμητό/τα δωμάτιο/α τότε πατάει το κουμπί Κράτηση.



Σχήμα 12.2 Διαθέσιμα δωμάτια σε ξενοδοχειακή μονάδα

Το σύστημα μεταβαίνει στο servlet που χειρίζεται το request του Χρήστη-Πελάτη για τη δεύτερη σελίδα. Στο servlet αυτό αποθηκεύονται οι προτιμήσεις του Χρήστη-Πελάτη για το/τα επιθυμητό/τα δωμάτιο/α και αποθηκεύονται επίσης σημαντικές πληροφορίες στα αντικείμενα session της εφαρμογής. Ο έλεγχος περνά στη σελίδα JSP που εμφανίζει τα απαραίτητα πεδία προς συμπλήρωση με τα στοιχεία των συμμετεχόντων στην κράτηση.

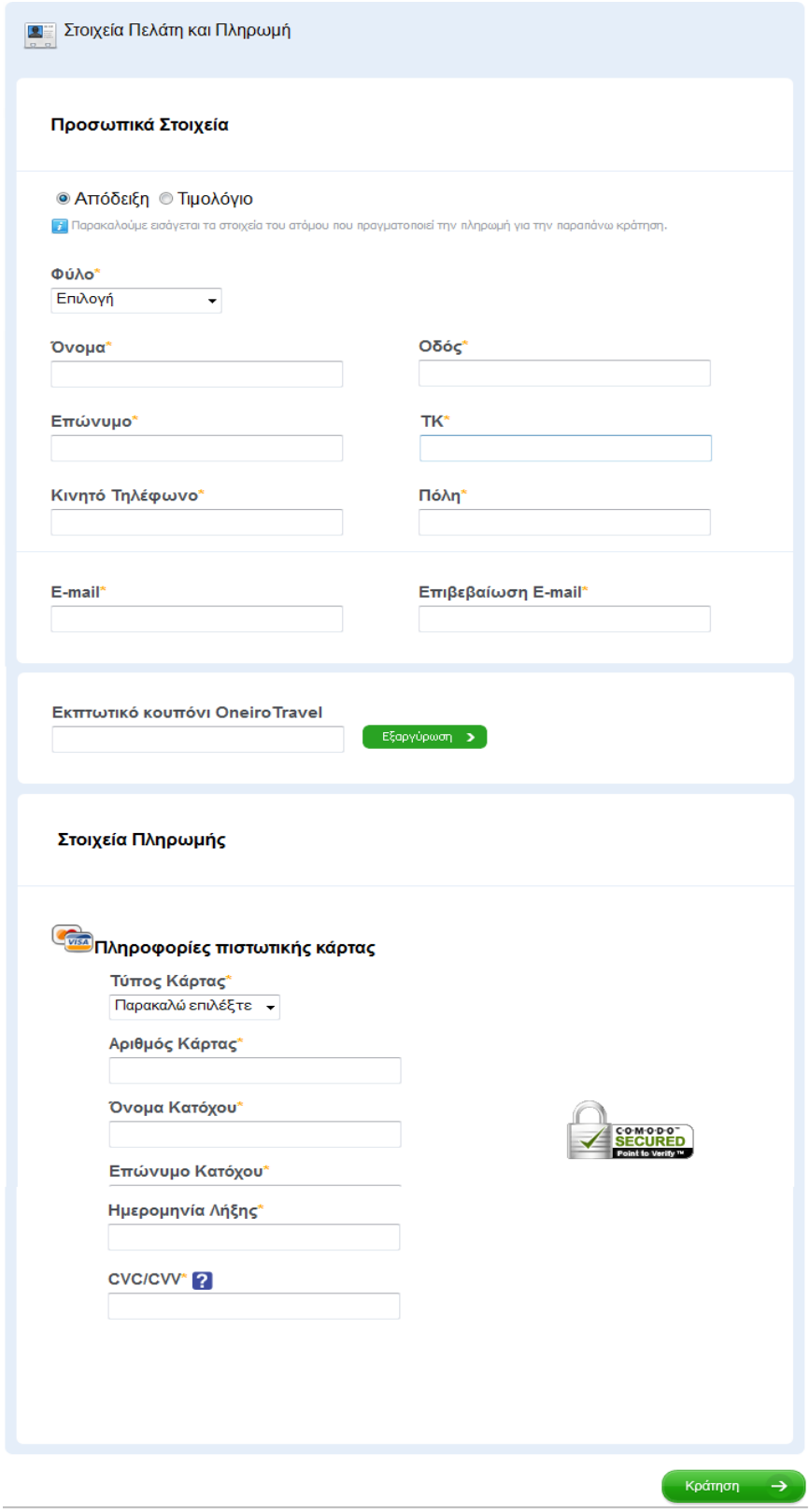
Το Σχήμα 12.3 δείχνει τα πεδία που ο χρήστης συμπληρώνει με τα στοιχεία των συμμετεχόντων στην κράτηση. Ο Χρήστης-Πελάτης εισάγει το ονοματεπώνυμο και το φύλο των συμμετεχόντων στην κράτηση σε κάθε δωμάτιο και στη συνέχεια πατάει το κουμπί Συνέχεια.



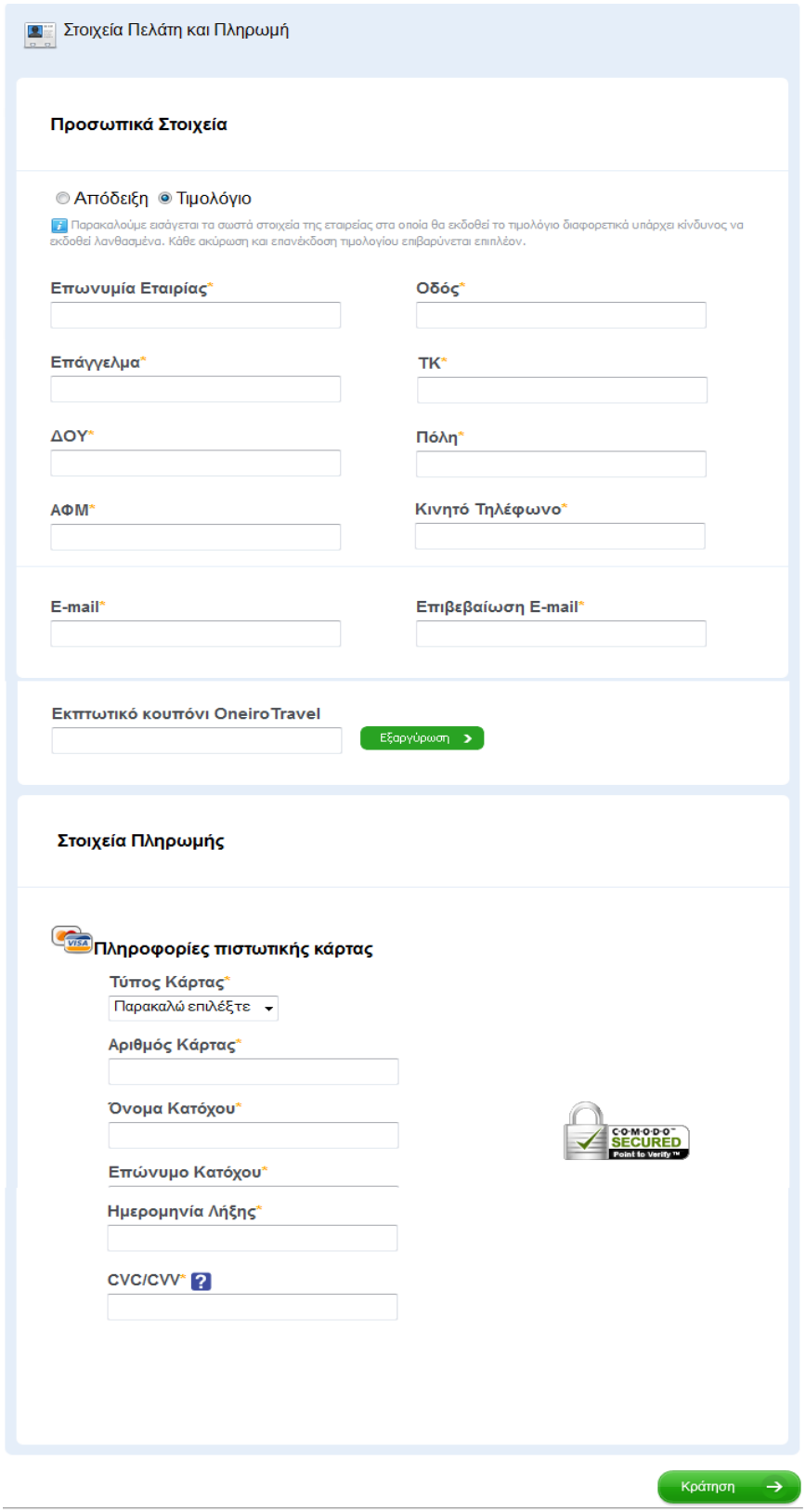
Σχήμα 12.3 Εισαγωγή στοιχείων συμμετεχόντων στην κράτηση

Το σύστημα ελέγχει τα στοιχεία που εισήγαγε ο Χρήστης-Πελάτης για την ορθότητά και την πληρότητά τους. Αν είναι λανθασμένα εμφανίζει κατάλληλη ένδειξη λάθους κάτω από κάθε πεδίο. Στη συνέχεια, και αν τα στοιχεία είναι σωστά, μεταβαίνει στο κατάλληλο servlet όπου αποθηκεύει τα δεδομένα που εισήγαγε ο Χρήστης-Πελάτης και περνάει τον έλεγχο στη JSP σελίδα που ασχολείται με την καταχώριση των στοιχείων του ατόμου που διεξάγει την κράτηση και τα στοιχεία της πληρωμής.

Τα Σχήματα 12.4, 12.5 παρουσιάζει τη σελίδα JSP όπου ο Χρήστης-Πελάτης επιλέγει αν θέλει απόδειξη ή τιμολόγιο. Αν επιλέξει απόδειξη, ο Χρήστης-Πελάτης εισάγει τα στοιχεία του ατόμου που εκτελεί την κράτηση και επίσης τα στοιχεία της Πληρωμής, όπως δείχνει το σχήμα 12.4. Αν όμως επιλέξει τιμολόγιο, ο Χρήστης-Πελάτης θα πρέπει να εισάγει τα στοιχεία της εταιρίας-επιχείρησης στην οποία θα εκδοθεί το τιμολόγιο και επίσης τα στοιχεία της Πληρωμής, όπως δείχνει το σχήμα 12.5. Στη συνέχεια πατάει το κουμπί Κράτηση.

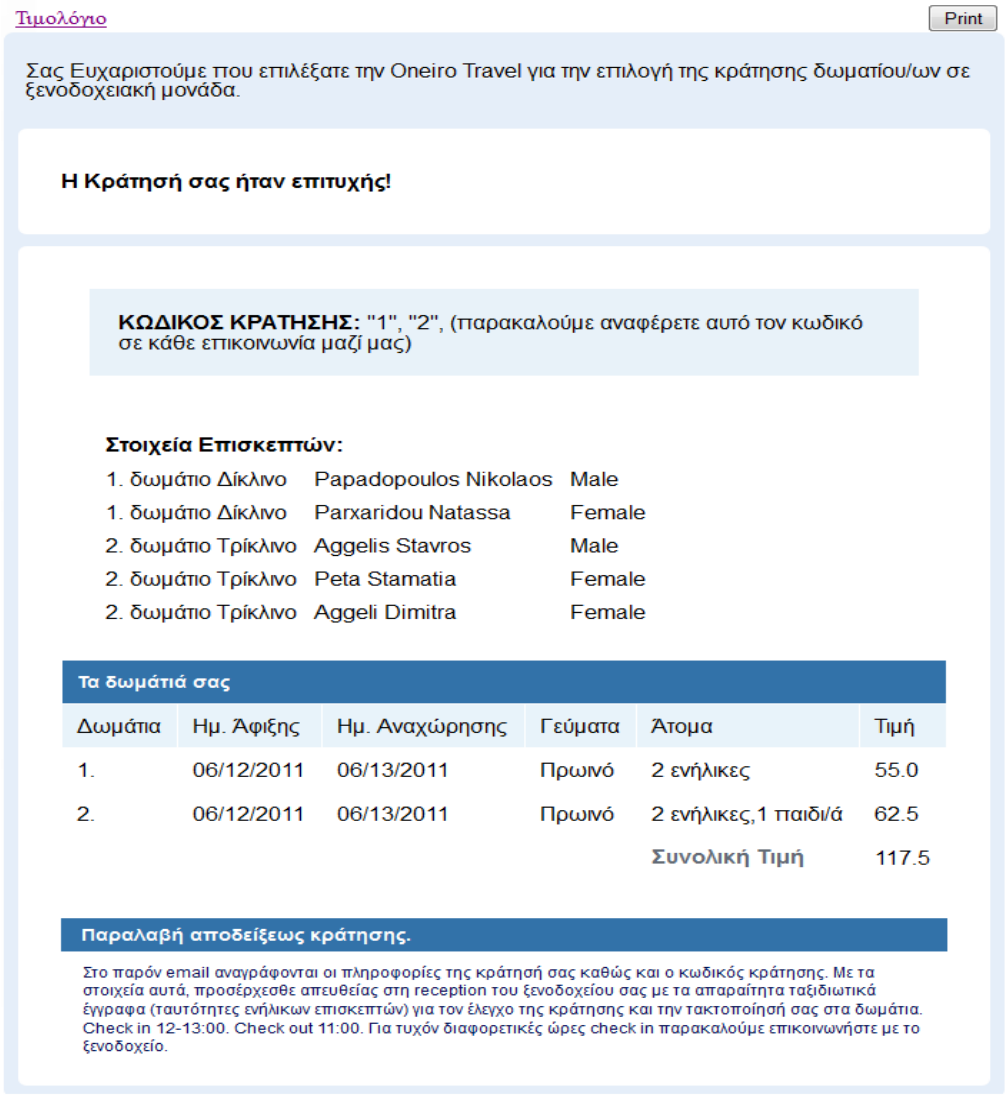


Σχήμα 12.4 Εισαγωγή στοιχείων Χρήστη-Πελάτη που εκτελεί την Κράτηση και στοιχείων Πληρωμής



Σχήμα 12.5 Εισαγωγή στοιχείων εταιρίας και στοιχείων Πληρωμής

Το σύστημα ελέγχει τα στοιχεί που εισήγαγε ο Χρήστης-Πελάτης και μεταβαίνει στο servlet που χειρίζεται το request του Χρήστη-Πελάτη. Δημιουργεί τα κατάλληλα αντικείμενα Κράτησης, δημιουργεί σύνδεση στη βάση όπου εισάγει τα κατάλληλα στοιχεία στους πίνακες, στέλνει το μήνυμα της κράτησης στο mail που έχει ορίσει ο Χρήστης-Πελάτης, με τη χρήση της βιβλιοθήκης JavaMail και μεταβιβάζει τον έλεγχο στη σελίδα JSP που εμφανίζει τα στοιχεία της κράτησης. Στο servlet επίσης θα έπρεπε να γίνεται και σύνδεση με ένα εξωτερικό σύστημα τραπέζης για τη διαχείριση της πληρωμής. Καθώς όμως είναι μία εκπαιδευτική εφαρμογή και όχι μία αληθινή το βήμα αυτό παραλύφθηκε.


Τιμολόγιο

Print

Σας Ευχαριστούμε που επιλέξατε την Oneiro Travel για την επιλογή της κράτησης δωματίου/ων σε ξενοδοχειακή μονάδα.

Η Κράτησή σας ήταν επιτυχής!

ΚΩΔΙΚΟΣ ΚΡΑΤΗΣΗΣ: "1", "2", (παρακαλούμε αναφέρετε αυτό τον κωδικό σε κάθε επικοινωνία μαζί μας)

Στοιχεία Επισκεπτών:

1. δωμάτιο Δίκλινο	Papadopoulos Nikolaos	Male
1. δωμάτιο Δίκλινο	Parxaridou Natassa	Female
2. δωμάτιο Τρίκλινο	Aggelis Stavros	Male
2. δωμάτιο Τρίκλινο	Peta Stamatia	Female
2. δωμάτιο Τρίκλινο	Aggeli Dimitra	Female

Τα δωμάτιά σας

Δωμάτια	Ημ. Άφιξης	Ημ. Αναχώρησης	Γεύματα	Άτομα	Τιμή
1.	06/12/2011	06/13/2011	Πρωινό	2 ενήλικες	55.0
2.	06/12/2011	06/13/2011	Πρωινό	2 ενήλικες,1 παιδί/ά	62.5
				Συνολική Τιμή	117.5

Παραλαβή αποδείξεως κράτησης.

Στο παρόν email αναγράφονται οι πληροφορίες της κράτησή σας καθώς και ο κωδικός κράτησης. Με τα στοιχεία αυτά, προσέρχεσθε απευθείας στη reception του ξενοδοχείου σας με τα απαραίτητα ταξιδιωτικά έγγραφα (ταυτότητες ενήλικων επισκεπτών) για τον έλεγχο της κράτησης και την τακτοποίησή σας στα δωμάτια. Check in 12-13:00. Check out 11:00. Για τυχόν διαφορετικές ώρες check in παρακαλούμε επικοινωνήστε με το ξενοδοχείο.

Σχήμα 12.6 Επιβεβαίωση Κράτησης

Στη σελίδα της επιβεβαίωσης κράτησης, που φαίνεται στο Σχήμα 12.6, παρουσιάζονται αναλυτικά τα στοιχεία της Κράτησης. Ο Χρήστης-Πελάτης μπορεί να επιλέξει να εκτυπώσει τη συγκεκριμένη σελίδα. Επίσης αν ο Χρήστης-Πελάτης έχει επιλέξει να εκδοθεί Τιμολόγιο μπορεί να επιλέξει το σύνδεσμο Τιμολόγιο όπου εμφανίζεται το Τιμολόγιο. Στη σελίδα Τιμολόγιο, φαίνεται στο Σχήμα 12.7, ο Χρήστης-Πελάτης μπορεί να επιλέξει την εκτύπωση της παρούσας σελίδας.

INVOICE

Oneiro Travel

Oneiro Travel Agency
Online Travel Agency
Smyrnis 25
Thessaloniki, 53646
Greece

Ημερομηνία : 2011/06/12
Αριθμός Τιμολογίου: 4

Λογαριασμός σε: C5
Savidis A.E
Smyrnis 20
Alexandroupoli, 597832
Greece

Περιγραφή
Κράτηση δωματίου/ων σε ξενοδοχείο. Κράτηση 2 δωαμτίων στην ξενοδοχειακή μονάδα Central, Plaka, στην πόλη Athens. Ημερομηνία Άφιξης 06/12/2011, Ημερομηνία Αναχώρησης 06/13/2011, (1 νύχτες), αριθμός ατόμων 4 ενήλικες, 1 παιδιά. Το 1 δωμάτιο είναι με πρωινό και θα φιλοξενηθούν οι κάτοθοι επισκέπτες: Papadopoulos Nikolaos, Parxaridou Natassa. Το 2 δωμάτιο είναι με πρωινό και θα φιλοξενηθούν οι κάτοθοι επισκέπτες: Aggelis Stavros, Peta Stamatia, Aggeli Dimitra.

Ποσότητα	Σύντομη Περιγραφή	Τιμή Μονάδας
1	Κράτηση μεπρωινό στην ξενοδοχειακή μονάδα Central στην πόλη Athens	55.0€
1	Κράτηση μεπρωινό στην ξενοδοχειακή μονάδα Central στην πόλη Athens	62.5€
	Συνολικό Ποσό Πληρωμής: 117.5€	

Αν έχετε οποιαδήποτε απορία σχετικά με το συγκεκριμένο τιμολόγιο επικοινωνήστε μαζί μας στο 8012556961 ή στείλτε μας email στη διεύθυνση oneiorotravel@agent.com

Σας ευχαριστούμε για τη συναλλαγή σας!

Σχήμα 12.7 Σελίδα Τιμολόγιο

Έτσι ολοκληρώνεται η κράτηση δωματίου/ων σε μία ξενοδοχειακή μονάδα από το υλοποιημένο σύστημά μας, «Online Τουριστικό Γραφείο». (Αναλυτικά η εφαρμογή βρίσκεται στο cd που συνοδεύει το βιβλίο και μέσα της κανείς θα βρει όλες τις λειτουργίες του συστήματος.).

12.2 Περίληψη

Στο κεφάλαιο αυτό παρακολουθήσαμε τον τρόπο με τον οποίο είναι υλοποιημένη η εφαρμογή μας, «Online Τουριστικό Γραφείο», μέσω ενός παραδείγματος κράτησης δωματίου. Απεικονίσαμε το layout του συστήματος και αναλύσαμε τις ενέργειες του Χρήστη-Πελάτη, καθώς και τις ενέργειες που κάνει από την πλευρά του το σύστημα.

Έτσι ολοκληρώθηκε το βιβλίο μας με τίτλο «Ανάπτυξη εκπαιδευτικής εφαρμογής “Τουριστικό γραφείο” με τη χρήση της μεθοδολογίας RUP». Εν κατακλείδι θα θέλαμε να πούμε ότι όλοι οι στόχοι πραγματοποιήθηκαν επιτυχώς και είμαστε έτοιμοι να ξεκινήσουμε επιπλέον έργα ανάπτυξης λογισμικού.

Συμπεράσματα και Προτάσεις

Ολοκληρώνοντας το έργο της ανάπτυξης λογισμικού για ένα διαδικτυακό τουριστικό γραφείο, έχουμε αποκτήσει ικανές γνώσεις ώστε να μπορέσουμε να βγάλουμε εις πέρας και άλλα παρόμοια έργα. Δε πρέπει όμως ποτέ να ξεχνάμε ότι κάθε έργο έχει τις δικές του ιδιομορφίες και τις δικές του ανάγκες. Επίσης ο τομέας της ανάπτυξης λογισμικού επεκτείνεται συνεχώς με νέες μεθοδολογίες και προτάσεις για γρηγορότερη και αρτιότερη ανάπτυξη. Τα παραπάνω δείχνουν ότι δε πρέπει να επαναπαυόμαστε στη γνώση που αποκτήθηκε για τη συγκεκριμένη εφαρμογή και θα πρέπει να αναζητούμε συνεχώς τρόπους για βελτίωση των προϊόντων που παράγουμε.

Στο συγκεκριμένο έργο, έγινε μία εξαιρετική δουλειά αποτύπωσης της λειτουργίας και της συμπεριφοράς ενός διαδικτυακού συστήματος. Καθώς όμως έγινε στους κόλπους της εκπόνησης μίας πτυχιακής εργασίας, ο χρόνος ήταν σαφώς περιορισμένος και προσπαθήσαμε να επικεντρωθούμε στα βασικά χαρακτηριστικά ενός συστήματος.

Έτσι θα μπορούσαμε ως περαιτέρω ολοκλήρωση του έργου να προσθέσουμε τη σύνδεση με το εξωτερικό σύστημα τραπέζης που λείπει, επιπλέον ελέγχους για «περίεργη» συμπεριφορά από τον Χρήστη-Πελάτη (πως θα πρέπει να αντιδρά το σύστημα σε εναλλακτικά σενάρια, όπου ο Χρήστης-Πελάτης δεν ακολουθήσει την πεπατημένη οδό) και εξόπλιση του συστήματος με διάφορους μηχανισμούς εναντίον επιθέσεων κακόβουλων προς τη βάση ή το ίδιο το σύστημα. Θα μπορούσαμε επίσης να εισάγουμε την ενοικίαση αυτοκινήτου που λείπει ή ακόμα και την ενοικίαση ακτοπλοϊκών εισιτηρίων και την ενοικίαση πακέτου εκδρομών. Επίσης η ανανέωση συνεχώς του συστήματος με τα δεδομένα της σύγχρονης εποχής και του σύγχρονου τρόπου ανάπτυξης λογισμικού, είναι μία ενέργεια που όχι μόνο επισημαίνεται αλλά επιβάλλεται, ώστε το σύστημα να είναι «ζωντανό» και λειτουργικό στη διάρκεια των χρόνων.

Βιβλιογραφία – Αναφορές

Διαδικτυακοί Τόποι

http://en.wikipedia.org/wiki/Main_Page

http://www.ibm.com

http://pdf.coreservlets.com

Βιβλία

Use Case Driven Object Modeling with UML, Theory and Practice, by Doug Rosenberg and Matt Stephens, 2007

Building J2EE Applications with the Rational Unified Process, by Peter Eeles, Kelli Houston, Wojtek Kozaczynski, 2002

Παράρτημα Α': Περιγραφή συστήματος Όνειρο

Το σύστημα Όνειρο είναι ένα λογισμικό σύστημα που υποστηρίζει μία εφαρμογή τουριστικών υπηρεσιών, παρέχοντας πλήρη κάλυψη όλων των απαιτήσεων ενός τουριστικού online γραφείου.

Το σύστημα περιλαμβάνει τις εξής υπηρεσίες: online κράτηση αεροπορικών εισιτηρίων, online κράτηση δωματίου σε ξενοδοχείο και online ενοικίαση αυτοκινήτου.

Επίσης το σύστημα παρέχει λειτουργία εγγραφής πελάτη, όπου ο χρήστης-πελάτης εισάγει τα στοιχεία του (όνομα, διεύθυνση, e-mail κτλ). Δεν απαιτείται εγγραφή του χρήστη-πελάτη στο σύστημα για να πραγματοποιηθεί επιτυχώς η επιθυμητή κράτηση και αυτό γιατί θέλουμε ο χρήστης-πελάτης να μπορεί να κάνει γρήγορα και εύκολα οποιαδήποτε ενέργεια επιθυμεί, συμβάλλοντας έτσι στις απαιτήσεις της πλέον πολυάσχολης πραγματικότητας. Η διαδικασία της εγγραφής είναι χρήσιμη γιατί ο χρήστης-πελάτης μπορεί να λαμβάνει κάποια προνόμια όπως εκπτωτικά κουπόνια ή ενημερωτικά e-mails για προσφορές και καινούριες παροχές. Οι λογαριασμοί των χρηστών κρατούνται στη βάση σε μία λίστα λογαριασμών.

Ο χρήστης-πελάτης έχει τη δυνατότητα να πλοηγείται μέσα στο interface της εφαρμογής και να επιλέγει όποια από τις παραπάνω υπηρεσίες επιθυμεί. Στην συνέχεια και με αναλυτικά βήματα ορίζει όλες τις επιλογές-παραμέτρους των κρατήσεων του.

Ο διαχειριστής της βάσης του συστήματος θα μπορεί να ενημερώνει τη βάση με νέα στοιχεία που θα αποφασίζει είτε η συγκεκριμένη εταιρία (π.χ. νέες συμβάσεις με ξενοδοχεία, εταιρίες ενοικίασης αυτοκινήτων, νέα πακέτα εκδρομών κτλ.) είτε οι εταιρίες με τις οποίες συνεργάζεται η εταιρία Όνειρο (π.χ. νέες πτήσεις, νέα μοντέλα αυτοκινήτων κτλ.). Ο διαχειριστής θα μπορεί επίσης να διαγράφει εγγραφές από τη βάση που δεν ανταποκρίνονται με τα δεδομένα εκείνης της

περιόδου (π.χ. ξενοδοχείο με το οποίο δεν συνεργάζεται πλέον η εταιρία) αλλά υπάρχει περιορισμός διαγραφής των εγγραφών για τις οποίες έχει γίνει κράτηση. Τέλος ο διαχειριστής της βάσης μπορεί να διενεργεί εργασίες πάνω στους λογαριασμούς των χρηστών-πελατών όπως εισαγωγή, διαγραφή και επεξεργασία των στοιχείων ενός πελάτη, όπου είναι απαραίτητο. Ο διαχειριστής της βάσης είναι κατ' επέκταση και διαχειριστής του όλου συστήματος.

1. Online κράτηση αεροπορικών εισιτηρίων

Ο χρήστης-πελάτης επιλέγει το αεροδρόμιο από το οποίο αποχωρεί και αεροδρόμιο επιθυμητού προορισμού. Επίσης επιλέγει ημερομηνία αναχώρησης καθώς και ημερομηνία επιστροφής αν έχει καθοριστεί πτήση με επιστροφή. Επιπλέον επιλογές γίνονται στο πόσα άτομα θα ταξιδέψουν και αν υπάρχουν παιδιά και βρέφη.

Ο χρήστης-πελάτης στέλνει τις επιλογές του μέσω του κουμπιού αναζήτησης και εμφανίζεται μία λίστα με τα διαθέσιμα δρομολόγια. Με ένα κλικ στις λεπτομέρειες θα μπορεί αν ελέγξει την περιγραφή της πτήσης καθώς και τα στοιχεία του αεροπλάνου. Ο χρήστης-πελάτης επιλέγει την ή τις επιθυμητές πτήσεις και κάνει κλικ στο κουμπί Κράτηση.

Στη συνέχεια ο χρήστης-πελάτης θα πρέπει να εισάγει τα στοιχεία του κάθε επιβάτη, όπως αναγράφονται στα επίσημα έγγραφα του καθενός, ώστε να μπορέσουν να εκδοθούν τα αντίστοιχα εισιτήρια της πτήσης.

Τέλος ο χρήστης-πελάτης επιβεβαιώνει την κράτηση και εισάγει τα απαραίτητα προσωπικά του στοιχεία ώστε να θεωρείται έγκυρη η κράτηση. Τα στοιχεία που πρέπει να συμπληρώσει είναι το ονοματεπώνυμο, με λατινικούς χαρακτήρες, κάθε επιβάτη που συμμετέχει στην κράτηση και η ηλικία, αν στην κράτηση συμμετέχουν και παιδιά. Επιπλέον το ονοματεπώνυμο και τα στοιχεία της διεύθυνσης του ατόμου που διεξάγει την κράτηση καθώς και ένας αριθμός τηλεφώνου και μία διεύθυνση e-mail. Τέλος τα στοιχεία της πιστωτικής κάρτας με την οποία θα γίνει πληρωμή όπως τύπος, ονοματεπώνυμο κατόχου, αριθμός κάρτας και CVC/CVV και ημερομηνία λήξης της κάρτας. Συλλέγονται ακόμα και

κάποια πρόσθετα στοιχεία (όπως αν ο πελάτης επιθυμεί τιμολόγιο, αν διαθέτει κάποιο εκπτωτικό κουπόνι κτλ), τα οποία όμως δεν είναι υποχρεωτικά για την διεξαγωγή μίας επιτυχής κράτησης.

Τέλος επιβεβαιώνονται τα στοιχεία από το σύστημα της τράπεζας, καταχωρείται η κράτηση στο σύστημά μας, αν είναι έγκυρη, και εμφανίζουμε το πιστοποιητικό της κράτησης και ταυτόχρονα στέλνεται και στον πελάτη με mail στη διεύθυνση του ηλεκτρονικού ταχυδρομείου που έχει ορίσει. Οι κρατήσεις καταγράφονται σε μία λίστα κρατήσεων στη βάση δεδομένων του συστήματος.

Επιπλέον περιορισμοί (Requirement constraints):

Ο χρήστης-πελάτης δεν μπορεί να εισάγει μη έγκυρα στοιχεία στις φόρμες. Πρέπει να γίνεται έλεγχος των πεδίων πριν τη καταχώρησή τους.

Τα παιδιά από 2 έως 11 ετών πληρώνουν το μισό αντίτιμο του εισιτηρίου, ενώ τα βρέφη έχουν μείωση τιμής κατά 80%.

Τα διαθέσιμα δρομολόγια εμφανίζονται ταξινομημένα ανά τιμή και ανά ώρα.

Τα υποχρεωτικά πεδία συμπλήρωσης στην online φόρμα προσωπικών στοιχείων του πελάτη πρέπει να είναι ευδιάκριτα.

2. Online κράτηση ξενοδοχείου

Ο χρήστης-πελάτης εισάγει τον προορισμό που θέλει να κάνει κράτηση δωματίου καθώς και την ημερομηνία άφιξης και αναχώρησης. Κατόπιν επιλέγεται ο αριθμός δωματίων που θέλει να κάνει κράτηση καθώς και πόσοι ενήλικες, αλλά και παιδιά, αν υπάρχουν, θα φιλοξενηθούν σε κάθε ένα δωμάτιο.

Το σύστημα θα δώσει αποτελέσματα σύμφωνα με τις προτιμήσεις του χρήστη και θα εμφανίσει τα αντίστοιχα ξενοδοχεία ταξινομημένα με βάση την τιμή, γράφοντας τον τύπο του/ων δωματίου/ων και την τιμή. Επιπλέον ο χρήστης μπορεί να επιλέξει αν θέλει παροχή μόνο πρωινού ή ημιδιατροφή(παροχή πρωινού και μεσημεριανού) ή πλήρης διατροφή (πρωινό, μεσημεριανό και

βραδινό) με διαφορά στη τιμή που θα αναγράφεται. Ο χρήστης-πελάτης επιλέγει το/τα επιθυμητό/τά δωμάτιο/α και κάνει κλικ στο κουμπί Κράτηση.

Στη συνέχεια ο χρήστης-πελάτης θα πρέπει να εισάγει τα στοιχεία του κάθε επισκέπτη σε κάθε δωμάτιο, όπως αναγράφονται στα επίσημα έγγραφα του καθενός, ώστε να μπορέσουν να εκδοθεί η ακριβής κράτηση του ξενοδοχείου.

Στη συνέχεια, εμφανίζεται η καρτέλα κράτηση στην οποία ο χρήστης-πελάτης πρέπει να συμπληρώσει κάποια προσωπικά του στοιχεία για να θεωρείται έγκυρη η κράτηση. Τα στοιχεία που πρέπει να συμπληρώσει είναι το ονοματεπώνυμο, με λατινικούς χαρακτήρες, του ατόμου που συμμετέχει στην κράτηση. Επιπλέον το ονοματεπώνυμο και τα στοιχεία της διεύθυνσης του ατόμου που διεξάγει την κράτηση καθώς και ένας αριθμός τηλεφώνου και μία διεύθυνση e-mail. Τέλος τα στοιχεία της πιστωτικής κάρτας με την οποία θα γίνει πληρωμή όπως τύπος, ονοματεπώνυμο κατόχου, αριθμός κάρτας και CVC/CVV και ημερομηνία λήξης της κάρτας. Συλλέγονται ακόμα και κάποια πρόσθετα στοιχεία (όπως αν ο πελάτης επιθυμεί τιμολόγιο και επίσης ενημερώνεται για την πολιτική της ακύρωσης της κράτησης), τα οποία όμως δεν είναι υποχρεωτικά για την διεξαγωγή μίας επιτυχής κράτησης.

Τέλος επιβεβαιώνονται τα στοιχεία από το σύστημα της τράπεζας, καταχωρείται η κράτηση στο σύστημά μας, αν είναι έγκυρη, και εμφανίζουμε το πιστοποιητικό της κράτησης και ταυτόχρονα στέλνεται και στον πελάτη με mail στη διεύθυνση του ηλεκτρονικού ταχυδρομείου που έχει ορίσει. Οι κρατήσεις καταγράφονται σε μία λίστα κρατήσεων στη βάση δεδομένων του συστήματος.

Επιπλέον περιορισμοί (Requirement constraints):

Ο αριθμός των δωματίων που μπορεί να εισάγει ο κάθε χρήστης για κάθε κράτηση είναι το πολύ 4. Επίσης ο αριθμός των ατόμων που μπορεί να φιλοξενηθούν σε ένα δωμάτιο είναι το πολύ 4.

Τα υποχρεωτικά πεδία συμπλήρωσης στην online φόρμα προσωπικών στοιχείων του πελάτη πρέπει να είναι ευδιάκριτα.

Ο χρήστης-πελάτης δεν μπορεί να εισάγει μη έγκυρα στοιχεία στις φόρμες. Πρέπει να γίνεται έλεγχος των πεδίων πριν τη καταχώρησή τους.

3. Online ενοικίαση αυτοκινήτου

Ο χρήστης-πελάτης εισάγει την πόλη-τοποθεσία στην οποία θέλει να κάνει την ενοικίαση του αυτοκινήτου. Επίσης επιλέγει και την ημερομηνία παραλαβής και επιστροφής του αυτοκινήτου καθώς και τις αντίστοιχες ώρες.

Στη συνέχεια το σύστημα εμφανίζει μία λίστα αυτοκινήτων που είναι διαθέσιμα, μαζί με όλες τις απαραίτητες πληροφορίες για κάθε αυτοκίνητο, αλλά και πληροφορίες για την ασφάλεια που παρέχεται μαζί με την ενοικίαση. Τα αποτελέσματα εμφανίζονται ταξινομημένα ως προς την τιμή. Ο χρήστης-πελάτης επιλέγει το επιθυμητό όχημα και κάνει κλικ στο κουμπί Κράτηση.

Το σύστημα εμφανίζει μία σελίδα με τα στοιχεία της ενοικίασης, όπου ο χρήστης πρέπει να επιλέξει κάποια επιπλέον χαρακτηριστικά για τον τόπο από τον οποίο θέλει να παραλάβει το όχημα και αν θέλει να υπάρχει κάποιος έξτρα εξοπλισμός, όπως παιδικό κάθισμα.

Προχωρώντας με την κράτηση, ο χρήστης-πελάτης πρέπει να εισάγει τα προσωπικά του στοιχεία ώστε να είναι έγκυρη η ενοικίαση. Τα στοιχεία που πρέπει να συμπληρώσει είναι το ονοματεπώνυμο, διεύθυνση, ένας αριθμός τηλεφώνου και μία διεύθυνση e-mail. Τέλος τα στοιχεία της πιστωτικής κάρτας με την οποία θα γίνει πληρωμή όπως τύπος, ονοματεπώνυμο κατόχου, αριθμός κάρτας και CVC/CVV και ημερομηνία λήξης της κάρτας. Συλλέγονται ακόμα και κάποια πρόσθετα στοιχεία (όπως αν ο πελάτης επιθυμεί τιμολόγιο, αν διαθέτει κάποιο εκπτωτικό κουπόνι ή αν ο χρήστης επιθυμεί να κλείσει ασφάλεια ακύρωσης κτλ), τα οποία όμως δεν είναι υποχρεωτικά για την διεξαγωγή μίας επιτυχής κράτησης.

Τέλος επιβεβαιώνονται τα στοιχεία από το σύστημα της τράπεζας, καταχωρείται η κράτηση στο σύστημά μας, αν είναι έγκυρη, και εμφανίζουμε το πιστοποιητικό της κράτησης και ταυτόχρονα στέλνεται και στον πελάτη με mail στη

διεύθυνση του ηλεκτρονικού ταχυδρομείου που έχει ορίσει. Οι κρατήσεις καταγράφονται σε μία λίστα κρατήσεων στη βάση δεδομένων του συστήματος.

Παράρτημα Β': Λίστα Domain Objects από την αρχική περιγραφή συστήματος

ΟΥΣΙΑΣΤΙΚΟ	ΠΛΗΘΟΣ	ΚΑΤΗΓΟΡΙΑ
online κράτηση	46	*Έννοια*
ακύρωση κράτησης	2	*Έννοια*
πακέτα εκδρομών	2	*Έννοια*
εγγραφή καταχώρισης	7	*Έννοια*
διαγραφή καταχώρισης	2	*Έννοια*
επεξεργασία καταχώρισης	1	*Έννοια*
login	4	*Έννοια*
Χρήστης-Πελάτης	33	*Ρόλος*
Λογαριασμός Χρήστη	1	*Έννοια*
ονοματεπώνυμο	6	*Χαρακτηριστικό της έννοιας << Λογαριασμός Χρήστη >>*
διεύθυνση	5	*Χαρακτηριστικό της έννοιας << Λογαριασμός Χρήστη >>*
τηλέφωνο επικοινωνίας	4	*Χαρακτηριστικό της έννοιας << Λογαριασμός Χρήστη >>*
e-mail	9	*Χαρακτηριστικό της έννοιας << Λογαριασμός Χρήστη >>*
username	1	*Χαρακτηριστικό της έννοιας << Λογαριασμός Χρήστη >>*

password	1	*Χαρακτηριστικό της έννοιας << Λογαριασμός Χρήστη >>*
Εκπτωτικό Κουπόνι	3	*Έννοια*
αεροδρόμιο (αναχώρησης-προορισμού)	3 (1-2)	*Χαρακτηριστικό της έννοιας <<online κράτηση>>*
ημερομηνία (αναχώρησης-επιστροφής)	8 (4+4)	*Χαρακτηριστικό της έννοιας <<online κράτηση>>*
ώρα (αναχώρησης-επιστροφής)	5 (3+2)	*Χαρακτηριστικό της έννοιας <<online κράτηση>>*
ενήλικες	2	*Χαρακτηριστικό της έννοιας <<online κράτηση>>*
παιδιά	5	*Χαρακτηριστικό της έννοιας <<online κράτηση>>*
βρέφη	2	*Χαρακτηριστικό της έννοιας <<online κράτηση>>*
τιμή	8	*Χαρακτηριστικό της έννοιας <<online κράτηση>>*
προορισμός διαμονής	1	*Χαρακτηριστικό της έννοιας <<online κράτηση>>*
σύνολο δωματίων	1	*Χαρακτηριστικό της έννοιας <<online κράτηση>>*
ενήλικες	1	*Χαρακτηριστικό της έννοιας <<online κράτηση>>*
Ξενοδοχείο	1	*Έννοια*
Δωμάτιο	3	*Έννοια*
τύπος δωματίου	1	*Χαρακτηριστικό της έννοιας <<Δωμάτιο>>*
διατροφή	1	*Χαρακτηριστικό της έννοιας <<Δωμάτιο>>*

πρωινό	1	*Τιμή του χαρακτηριστικού <<διατροφή>>*
ημιδιατροφή	1	*Τιμή του χαρακτηριστικού <<δωμάτιο>>*
πλήρης διατροφή	1	*Τιμή του χαρακτηριστικού <<δωμάτιο>>*
προορισμός ενοικίασης αυτοκινήτου	2	*Χαρακτηριστικό της έννοιας <<online κράτηση>>*
Αυτοκίνητο	2	*Έννοια*
ασφάλεια	1	*Χαρακτηριστικό της έννοιας <<Αυτοκίνητο>>*
τύπος αυτοκινήτου	1	*Χαρακτηριστικό της έννοιας <<Αυτοκίνητο>>*
εξτρά εξοπλισμός	1	*Χαρακτηριστικό της έννοιας <<Αυτοκίνητο>>*
παιδικό κάθισμα	1	*Τιμή του χαρακτηριστικού <<εξτρά εξοπλισμός>>*
ονοματεπώνυμο	1	*Χαρακτηριστικό της έννοιας <<online κράτηση>>*
ηλικία	1	*Χαρακτηριστικό της έννοιας <<online κράτηση>>*
Πληρωμή	4	*Έννοια*
τιμολόγιο	3	*Χαρακτηριστικό της έννοιας <<Πληρωμή>>*
Πιστωτική κάρτα	4	*Έννοια*
τύπος	4	*Χαρακτηριστικό της έννοιας <<Πιστωτική κάρτα>>*
ονοματεπώνυμο κατόχου	4	*Χαρακτηριστικό της έννοιας <<Πιστωτική κάρτα>>*

αριθμός κάρτας	4	*Χαρακτηριστικό της έννοιας <<Πιστωτική κάρτα>>*
CVC/CVV	4	*Χαρακτηριστικό της έννοιας <<Πιστωτική κάρτα>>*
ημερομηνία λήξης	4	*Χαρακτηριστικό της έννοιας <<Πιστωτική κάρτα>>*
Βάση Δεδομένων	6	*-?-*
Διαχειριστής Βάσης Δεδομένων	3	*Ρόλος*
Λίστα Λογαριασμών Χρηστών	2	*-?-*
Εξωτερικά συνεργαζόμενες εταιρίες	1	*-?-*
Λίστα Χρηστών	2	*-?-*
Λίστα Αεροδρομίων	1	*-?-*
Λίστα Κρατήσεων	3	*-?-*
Λίστα Ξενοδοχείων	1	*-?-*
Λίστα Αυτοκινήτων	1	*-?-*
Λίστα Πακέτου Εκδρομών	2	*-?-*

Οδηγός χρήσης λογισμικού

Το παρών βιβλίο συνοδεύεται και με ένα cd όπου υπάρχουν τα εξής στοιχεία: η πτυχιακή εργασία σε αρχείο word, ένας φάκελος Oneiro που περιέχει τις απαραίτητες κλάσεις και αρχεία του προγράμματός μας (αποτελεί δηλαδή την υλοποίηση του έργου μας) και ένα αρχείο όπου βρίσκονται οι πίνακες της βάσης μας.

Για να μπορέσετε να τρέξετε την υλοποίηση του online ταξιδιωτικού γραφείου Oneiro θα πρέπει να εκτελέσετε κάποιες ενέργειες στο σύστημά σας. Από το φάκελο Oneiro θα πρέπει να αντιγράψετε τον φάκελο WEB-INF στον τοπικό σας διακομιστή (κατά προτίμηση Tomcat 6.0.20).

Στη συνέχεια το αρχείο database.backup θα πρέπει να γίνει import στην τοπική σας βάση (π.χ PostgreSQL ή άλλη). Για να μπορέσετε να συνδεθείτε στη βάση θα πρέπει να ακολουθήσετε μία από τις παρακάτω ενέργειες:

1. Να μεταβείτε στο αρχείο DBConnection.java της υλοποίησης, που το βρίσκεται κάτω από το Oneiro\src\java\ oneirotravel\dbconnection και αλλάζεται τις μεταβλητές String που είναι δηλωμένες σύμφωνα με τα στοιχεία της τοπική σας βάση ή
2. Δημιουργείται στη βάση σας ένα χρήστη με τα στοιχεία που δείχνουν οι παραπάνω μεταβλητές.

Τέλος θα πρέπει να ορίσετε τα mappings του διακομιστή σας ώστε το νέο application να φαίνεται στο /Oneiro της τοπική σας σελίδας.

Τα διαγράμματα της μεθοδολογίας ICONIX θα τα βρείτε στον φάκελο vpworkspace όπου μπορείτε να τον διαχειριστείτε έχοντας την εφαρμογή Visual Paradigm. Μέσα σε αυτό το workspace βρίσκονται όλες οι περιπτώσεις χρήσης, το domain model, το class model, και τα διαγράμματα ευρωστίας και ακολουθίας για κάθε περίπτωση χρήσης. Επίσης περιλαμβάνει τα mock-ups screens της εφαρμογής μας.