Software Engineering Project

Ομαδική δραστηριότητα (aka project) μαθήματος «Τεχνολογία Λογισμικού»

Στο παρόν εγχειρίδιο περιγράφονται όλες οι οδηγίες για το project του μαθήματος «*Τεχνολογία Λογισμικού*» και όλες οι κανονιστικές διατάξεις που διέπουν την παράδοση του project. Θεωρήστε το παρόν ως "project manual" και συμβουλευτείτε το τακτικά και ιδιαίτερα πριν από κάθε παράδοση τμηματικής εργασίας. Σε περίπτωση αλλαγών ή βελτιώσεων (π.χ. προσθήκη παραδειγμάτων) η έκδοση που υπάρχει στο eClass θα ανανεώνεται, άρα θα πρέπει να έχετε κάθε φορά την τρέχουσα έκδοση.

Έκδοση 03.00

Πάτρα, Ιανουάριος 2025

Contents

1. Σκοπός

Σκοπός του project είναι να γνωρίσετε στην πράξη[1] τις βασικές φάσεις ανάπτυξης λογισμικού και να συνεργαστείτε σε ομάδες. Τις ομάδες θα έχετε την ευκαιρία και τη δυνατότητα να τις δημιουργήσετε ή να τις επιλέξετε εσείς, και να τις οργανώσετε όπως θέλετε. Η εμπειρία εργασίας και συνεργασίας σε ομάδες είναι σημαντική δεξιότητα που δεν πρέπει να αμελήσετε να καλλιεργήσετε γιατί, όπως θα διαπιστώσετε ως απόφοιτοι, είναι σημαντικό προσόν ενός μηχανικού.

Στο project αυτό θα προτείνετε και θα υλοποιήσετε ένα έργο ανάπτυξης λογισμικού σε μικρότερη, όμως, κλίμακα. Θα ξεκινήσετε από τη φάση της ανάλυσης των απαιτήσεων, θα συνεχίσετε με αρχική και λεπτομερή σχεδίαση, και θα ολοκληρώσετε με υλοποίηση μέρους του έργου (και ανάρτηση του κώδικα στο Git του έργου) και προτάσεις για έλεγχο (φάση ελέγχου τεχνολογίας λογισμικού) της υλοποίησης.

Η επιλογή ενός καλού έργου είναι σημαντικό στοιχείο για την επιτυχία του project, άρα σκεφτείτε πολύ τι έργο θα προτείνετε.

Τι είναι καλό έργο; Ιδανικά κάτι που θα σκεφτόσαστε μήπως αξίζει παράλληλα με τις σπουδές σας να το υλοποιήσετε πραγματικά! Μια ιδέα δηλαδή που να έχει πραγματική αξία και που θα μπορέσετε να την υλοποιήσετε και να έχει κέρδος ή αξία για το κοινωνικό σύνολο. Επίσης, ένα καλό έργο δεν μπορεί να είναι κάτι τόσο απλό που κάποιος το φτιάχνει μόνος του, ή που η ομάδα σας θα μπορούσε να το υλοποιήσει στα πλαίσια του project. Για παράδειγμα ένα εργαλείο που σβήνει τα metadata από αρχεία pdf είναι μερικές γραμμές κώδικα και θα το έγραφε κάποιος σε 10’ χωρίς να φτιάξει ομάδα, ένα σύστημα διαχείρισης αρχείων pdf που θα παρέχει λειτουργίες όπως split a file, merge files, convert, add a page, delete metadata, κ.λπ. αρχίζει να γίνεται ενδιαφέρον ως έργο (μόλις κάψαμε μια ιδέα), αλλά ακόμα δεν έχει σημαντική πολυπλοκότητα ή σημαντικές λειτουργίες για να αποτελέσει κάτι πολύπλοκο ως έργο για να ασχοληθεί μια ομάδα (ένας καλός προγραμματιστής θα το έφτιαχνε σε πολύ λιγότερο από τις 60 ώρες που θα βάλει στο project). Αν όμως αρχίσουμε να προσθέτουμε και λειτουργίες διαμοίρασης αρχείων, υποστήριξη αρχείων σε cloud, κ.λπ. αρχίζει να γίνεται ένα έργο ικανοποιητικής πολυπλοκότητας. Αν βάλουμε και διαφορετικά ήδη πελατών (online πελάτης που το χρησιμοποιεί free για περιορισμένο μέγεθος αρχείων, online πελάτης που πληρώνει για μια χρήση, online συνδρομητής που του παρέχουμε ένα πλήρες πακέτο λειτουργιών και cloud space, εταιρία που έχει ένα αριθμό χρηστών και συνεργατικές λειτουργίες) τότε αρχίζει να γίνεται ένα ενδιαφέρον έργο (που το κάψαμε ως θέμα). Σκεφτείτε λοιπόν μια εξαιρετική ιδέα που αν είναι τόσο καλή, θα αξίζει να αποτελέσει τεχνοβλαστό και να έχετε επαγγελματικό όφελος και όχι κάτι που είναι C&P από κάτι που έχετε χρησιμοποιήσει ήδη και σας αρέσει! Σημαντική, αν και όχι τόσο όσο το ίδιο το έργο είναι και η **σύντομη ονομασία του έργου σας** (3 έως 12 χαρακτήρες). Με αυτή την ονομασία θα αναφέρεστε και εσείς και εμείς στο έργο σας για το επόμενο εξάμηνο (ελπίζουμε) ή για τα επόμενα 3 χρόνια! Φροντίστε να είναι κάτι χαρακτηριστικό για το έργο σας.

Το project είναι λοιπόν μια εξομοίωση της δουλειάς που θα έκανε η ομάδα σας αν εσείς που αποτελείτε την ομάδα είχατε αναλάβει στην πραγματικότητα το έργο που προτείνετε. Η εξομοίωση όμως θα πρέπει

[1] Η διαφορά της θεωρίας από την πράξη έχει συζητηθεί εκτενώς σε μαθήματα προηγουμένων ετών, οπότε δεν χρειάζεται να επεκταθούμε!

να φτάνει ως ένα μέρος της ανάπτυξης, αν και προφανώς λόγο περιορισμού στο χρόνο περιμένουμε ότι πολλές λειτουργίες δεν θα έχουν αναπτυχθεί.

Η ύλη που αφορά το project θα διδαχθεί στις παραδόσεις και τα φροντιστήρια του μαθήματος, η παρακολούθηση των οποίων είναι σημαντικότατη για την επιτυχή ολοκλήρωση του project.

Το project αυτό είναι (υπό ΚΣ) το τελευταίο μεγάλο project που θα υλοποιήσετε κατά τη διάρκεια των σπουδών στο Τμήμα. Προσπαθήστε να είναι μια δουλειά για την οποία θα είστε υπερήφανοι και κάτι που θα θέλετε να δείξετε (π.χ. το Git του έργου σας) ως απόφοιτοι.

Disclaimer: Στο ανά χείρας εγχειρίδιο χρησιμοποιούμε την έκφραση «ο φοιτητής» με ουδέτερη έννοια (δηλαδή εννοούμε προφανώς και «η φοιτήτρια») για να μην γράφουμε συνεχώς «η φοιτήτρια ή ο φοιτητής». Δεν αποτελεί κάποιο είδος διάκρισης, απλά είναι ένας τρόπος να είμαστε πιο σύντομοι.

2. Ορολογία

Προς αποφυγή παρεξηγήσεων, επειδή το project αφορά ένα έργο λογισμικού θα χρησιμοποιούμε σε όλο το κείμενο τους παρακάτω όρους:

- **Project:** Αφορά το project *per se*, δηλαδή όταν μιλάμε, για παράδειγμα, για την ανάθεση προσωπικού στις δραστηριότητες του project, μιλάμε για το ποιο μέλος της ομάδας σας θα αναλάβει τι. Παράδειγμα: «*Η σχεδίαση του Use Case 1 έγινε από τον Κώστα … και η σχεδίαση του Use Case 2 έγινε από την Μαρία*
- **Έργο:** Αφορά το έργο λογισμικού που προτείνετε ως ομάδα και για το οποίο θα γίνει ανάλυση, σχεδίαση και μερική υλοποίηση. Παράδειγμα «*Το έργο μας είναι η δημιουργία ενός νέου Progress για το Πανεπιστήμιο Πατρών και για αυτό έχουν ετοιμαστεί 8 Use Cases*».
- **Ομάδα Project:** Είναι τα 5 μέλη της ομάδας σας. Αναλυτικά για την ομάδα έργου ακολουθούν στο σχετικό τμήμα «Ομάδα έργου».
- **Παραδοτέο:** Αφορά ένα πακέτο τεχνικών κειμένων που παραδίδονται μαζί σε μια καθορισμένη ημερομηνία.
- **Τεχνικό κείμενο:** Κατά κανόνα δεν είναι όλα κείμενα (τα περισσότερα δεν είναι κείμενο), αλλά για λόγους απλότητας θα ονομάζουμε τεχνικό κείμενο κάθε στοιχείο του έργου με το οποίο θα ασχοληθείτε. Έτσι ακόμα και ο τελικός κώδικας θα είναι ένα τεχνικό κείμενο (αν και απλά θα παραπέμπει στο Git της ομάδας σας). Κάθε τεχνικό κείμενο έχει ένα κωδικό που θα σας έχει δοθεί και θα πρέπει να τον χρησιμοποιείτε.
- **Παράδειγμα Έργου:** Για τις ανάγκες καλύτερης περιγραφής θα χρησιμοποιούμε ως παράδειγμα έργου ένα υποθετικό «*Νέο Progress για το Πανεπιστήμιο Πατρών*». Προφανώς, και αυτό δεν μπορεί να είναι η δική σας πρόταση, ούτε παρόμοιες προτάσεις, δηλαδή μην στείλετε ως πρόταση έργου κάτι σαν «*Ηλεκτρονική Γραμματεία για Πανεπιστήμια*» ή κάτι συναφές. Για την ανάλυση και σχεδίαση δίνουμε παραδείγματα από ένα μικρό έργο (ατομική εργασία προηγούμενων ετών για αυτό και είναι τόσο απλό) που αφορούσε ένα παιχνίδι τύπου «Μονόπολη».
- **Σύντομη Ονομασία Έργου:** Είναι μια σύντομη ονομασία του έργου σας από 3 έως 12 χαρακτήρες το πολύ. Για παράδειγμα, «Progress», «Μονόπολη». Σκεφτείτε το σαν brand name και χρησιμοποιήστε το σε κάθε αναφορά στο έργο σας.

3. Επικοινωνία και παράδοση του project

Όλη η επικοινωνία του μαθήματος Τεχνολογία Λογισμικού και κατά συνέπεια και όλη η επικοινωνία για το project, και η παράδοση του project θα γίνεται **αποκλειστικά μέσω του eClass του μαθήματος**. Οι παραδόσεις των τμημάτων του project θα γίνονται μέσω eClass (θα εμφανίζονται ως Εργασίες του μαθήματος) και απορίες (μόνο ότι δεν συζητήθηκε στο φροντιστήριο) μπορούν να ερωτηθούν ως μήνυμα στο eClass. Απορίες με e-mail και γενικά κάθε επικοινωνία μέσω e-mail θα αγνοείται. Το ίδιο ισχύει (αυστηρά) και για παραδόσεις του project που γίνονται μέσω e-mail.

Επειδή κάθε χρόνο, με την ολοκλήρωση των παραδόσεων και λίγο πριν τις εξετάσεις, δεχόμαστε μια σειρά από e-mail της μορφής «*δεν έκανα το project, αλλά μήπως θα μπορούσα να…*», ή «*διάβασα τους κανόνες, αλλά εγώ… ισχύουν και για εμένα;*», διαβάστε προσεκτικά τον κανονισμό του project και θεωρήστε ότι **ισχύει για όλους χωρίς εξαιρέσεις**. Το ίδιο και για το «*απορίες με e-mail και γενικά κάθε επικοινωνία μέσω e-mail θα αγνοείται*».

Το project έχει καταληκτικές ημερομηνίες παράδοσης και μια μικρή ανοχή σε καθυστερήσεις. Δεν θα γίνει δεκτό, όμως, κανένα project μετά την τελική καταληκτική ημερομηνία (ανακοινώνεται στο eClass με την έναρξη του μαθήματος), άρα φροντίστε να μην καθυστερήσετε.

4. Κανονισμός Project

Ακολουθούν διαδικαστικά που αφορούν το project τα οποία μελετήστε τα προσεκτικά προς αποφυγή παρεξηγήσεων.

Το project ως προϋπόθεση συμμετοχής στις εξετάσεις

Το project είναι υποχρεωτικό και **η παράδοσή του αποτελεί προϋπόθεση συμμετοχής στις εξετάσεις**. Φοιτητής που δεν έχει ολοκληρώσει το project δεν δικαιούται να συμμετέχει στις εξετάσεις και αν το κάνει δεν θα πάρει βαθμό, αφού το γραπτό του δεν θα διορθωθεί. Αναλυτικά για τη βαθμολογία ενός project ακολουθούν στο τμήμα «Βαθμολογία Project».

Διάρκεια ισχύος του βαθμού ενός φοιτητή στο project

Ο **βαθμός ενός φοιτητή μπορεί να κρατηθεί για 4 συμμέτοχες στην γραπτή εξέταση του μαθήματος**. Εάν όμως ο φοιτητής δεν καταφέρει να επιτύχει στις εξετάσεις αυτές τις 4 φορές, τότε ο βαθμός του project διαγράφεται και πρέπει να το εκπονήσει εκ νέου.

Τμηματική παράδοση (παραδοτέα) του project

Η παράδοση του project γίνεται τμηματικά, όπως θα γινόταν τμηματικά η παράδοση των παραδοτέων ενός έργου λογισμικού. Οι τμηματικές παραδόσεις θα έχουν καθορισμένες ημερομηνίες και θα γίνονται με τη μορφή εργασιών στο eClass. Αναλυτικά για τα παραδοτέα ακολουθούν στο τμήμα «Παραδοτέα Project».

Η παράδοση κάθε παραδοτέου μπορεί να γίνει είτε: α) στις ημερομηνίες που έχουν καθοριστεί, είτε β) με καθυστέρηση. Η καθυστέρηση δεν μπορεί να υπερβαίνει τις 15 ημέρες συνολικά για όλα τα παραδοτέα και υπάρχει ρήτρα καθυστέρησης (δηλαδή η καθυστέρηση σε κάποιο παραδοτέο έχει συνέπειες στη βαθμολογία του project). Αναλυτικά ακολουθούν στο τμήμα «Βαθμολογία Project».

Η παράδοση κάθε παραδοτέου θα γίνεται αυστηρά **μόνο από ένα μέλος της ομάδας σας**. Σε περίπτωση διπλής παράδοσης (δηλαδή που δύο ή περισσότερα μέλη της ομάδας σας έχουν παραδώσει κάτι διαφορετικό για το ίδιο παραδοτέο), θα αγνοούνται όλα. Ο φοιτητής που παραδίδει μπορεί να αλλάξει στο επόμενο παραδοτέο, **αν και είναι ισχυρά προτεινόμενο να είναι ο ίδιος πάντα που θα εκπροσωπεί την ομάδα**, αλλά πάντα θα είναι ένας.

Καταληκτική ημερομηνία παράδοσης του project

Η τελική καταληκτική ημερομηνία για την παράδοση του project είναι η ημερομηνία λήξης του τελευταίου παραδοτέου. Μετά από αυτή την ημερομηνία, όσες ομάδες δεν παρέδωσαν project θα πρέπει να το επαναλάβουν την επόμενη χρονιά. **Δεν υπάρχει καμία δυνατότητα παράτασης για κανένα λόγο.**

Περιπτώσεις αντιγραφής και λογοκλοπής

Project που είναι προϊόντα αντιγραφής ή λογοκλοπής θα μηδενίζονται και ανάλογα με τη σοβαρότητα της περίπτωσης το θέμα θα παραπέμπεται στη Επιτροπή Δεοντολογίας του Τμήματος. Για αποφυγή λογοκλοπής διαβάστε το τμήμα «λογοκλοπή».

5. Ομάδα Project

Σύμφωνα με το περίγραμμα μαθήματος, η εκτιμώμενη προσπάθεια ανά φοιτητή που έχει μελετήσει την αντίστοιχη ύλη[2] και έχει παρακολουθήσει τις παραδόσεις και τα φροντιστήρια υπολογίζεται (με βάση τα ECTS του μαθήματος) ως **60 ώρες**. Οι ομάδες project θα πρέπει να αποτελούνται από **5 φοιτητές** και η προσπάθεια που αναμένεται για την επιτυχή εκπόνηση ενός project είναι περίπου **300 ώρες εργασίας**. Δεν θα γίνουν δεκτά project με ομάδες 3, 4 φοιτητών ή 6 και περισσοτέρων φοιτητών με καμία δικαιολογία!!!

Δημιουργία ομάδας

Μπορείτε να χρησιμοποιήσετε το forum στο eClass, ή οποιοδήποτε άλλο τρόπο θέλετε, για να δημιουργήσετε τις ομάδες σας. Από τη δική μας εμπειρία σας ενημερώνουμε ότι μεικτές ομάδες σχετικά με τις δεξιότητες των μελών κάθε ομάδας λειτουργούν πολύ καλύτερα! Για παράδειγμα, μια ομάδα με 5 πολύ καλούς προγραμματιστές, μπορεί να μην είναι τόσο αποδοτική όσο μια ομάδα με πιο μοιρασμένες δεξιότητες. Δημιουργήστε τις ομάδες σας πολύ προσεκτικά. *Αν δεν μπορείτε να σχηματίσετε ομάδα με 5 άτομα, η μόνη εναλλακτική είναι να δεχτείτε την ομάδα που θα οριστεί για εσάς από τους διδάσκοντες.*

Αλλαγές στα μέλη μιας ομάδας

Αλλαγές στη σύνθεση μιας ομάδας μπορούν να γίνουν σε όλη τη διάρκεια του project. Αλλαγές μπορούν να γίνουν με αποχώρηση ή προσθήκη κάποιου μέλους ή αντικατάσταση κάποιου. Σε περίπτωση όμως μιας αλλαγής θα πρέπει να υπάρχει αναλυτική τεκμηρίωση στο επόμενο παραδοτέο του ρόλου του νέου μέλους και αντίστοιχη (αναμόρφωση) όλων των σχετικών παραδοτέων. Είναι προφανές ότι προσθήκες της τελευταίας στιγμής ενδέχεται να έχουν συνέπειες στην αξιολόγηση του project, αν δεν μπορούν να τεκμηριώσουν τη συμμετοχή του νέου μέλους της ομάδας και την ισότιμη κατανομή της προσπάθειας.

Ομάδες που αποφάσισαν να μην συνεχίσουν νωρίς στην αρχή του ακαδημαϊκού έτους, αλλά κάποια μέλη των ομάδων θέλουν να ενταχθούν σε άλλες ομάδες και να συνεισφέρουν μπορούν να το κάνουν με αυτή τη διαδικασία, είτε συζητώντας απευθείας με ομάδες, είτε χρησιμοποιώντας το forum.

Ρόλοι στην ομάδα

Τα μέλη της ομάδας μπορούν να έχουν διάφορους ρόλους που μπορούν να είναι είτε καθορισμένοι, είτε εκ περιτροπής. Η συμβουλή μας είναι να εξασκηθείτε σε αλλαγή ρόλων εκ περιτροπής σε κάθε παραδοτέο, αλλά η επιλογή είναι δική σας. Για παράδειγμα, σε ένα τυπικό μοντέλο διαχείρισης, ρόλοι που θα μπορούσαν να υπάρχουν στην ομάδα είναι ο “project manager” δηλαδή αυτός που έχει την ευθύνη ανάθεσης των εργασιών και επίβλεψης της πορείας των εργασιών που αναλαμβάνουν τα μέλη της ομάδας και ο “quality manager” δηλαδή αυτός που έχει την ευθύνη για τον καθορισμό των προδιαγραφών ποιότητας των παραδοτέων σας και τον έλεγχο της ποιότητας. Προφανώς, αυτοί οι ρόλοι μπορεί να είναι και εκ περιτροπής, είτε ανά παραδοτέο, είτε και σε διαφορετικά τμήματα του

[2] Δείτε στο περίγραμμα μαθήματος την αναμενόμενη προσπάθεια μελέτης. Η εκτίμηση προσπάθειας για το project αφορά φοιτητές που έχουν ήδη μελετήσει. Αν κατά τη διάρκεια των συναντήσεων της ομάδας σας για το project χρειάζεται να μάθετε το αντικείμενο οι χρόνοι θα είναι κατά πολύ μεγαλύτεροι!

παραδοτέου (π.χ. ο Κώστας ελέγχει το τμήμα του παραδοτέου που έγραψε η Μαρία, η Μαρία το τμήμα που έγραψε ο Νίκος, κ.λπ.).

Σε κάθε τεχνικό κείμενο μπορείτε να εμπλακείτε όλοι (έχει νόημα για μεγάλα τεχνικά κείμενα να μοιράσετε κομμάτια τους, π.χ. να αναλάβει καθένας από ένα use case), αλλά καλύτερα να μοιράζετε και να οργανώνετε τη δουλειά σας κατάλληλα. Θυμίζουμε, ότι μπορείτε να δουλέψετε όλοι σε όλα (δεν το προτείνουμε) ή να μοιράσετε τη δουλειά, αλλά να εμπλέξετε τουλάχιστον δύο μέλη της ομάδας σας σε κάθε τεχνικό κείμενο (π.χ. ο ένας να το συντάσσει και κάποιος να το ελέγχει). Άρα δεν είναι ανάγκη όλα τα μέλη να αναφέρονται και ως συμμετέχοντες σε κάθε τεχνικό κείμενο, μόνο όσοι πραγματικά εργάστηκαν και τι έκαναν.

Για την ανάλυση-σχεδίαση-υλοποίηση κώδικα-έλεγχος **είναι καλό να μοιράσετε τις εργασίες ώστε κάθε μέλος της ομάδας σας να αναλάβει τουλάχιστον δύο use case**, να σχεδιάσει ίδιο αριθμό robustness diagram, ίδιο αριθμό sequence diagram, να γράψει κώδικα για τις μεθόδους που ασχολήθηκε και να γράψει και τα αντίστοιχα test case. Έτσι **ο καθένας από εσάς θα έχει την εμπειρία ενός μικρού έργου** και θα πρέπει να συνεργαστείτε για τα τμήματα που ολοκληρώνουν το τελικό σύστημα/έργο σας από τα μικρότερα module. Προφανώς, μπορείτε να δουλέψετε και με άλλη μέθοδο, αλλά αν για παράδειγμα επιλέξετε κάποιος να κάνει μόνο use case, κάποιος μόνο robustness diagram και κάποιος μόνο sequence diagram δεν είναι καθόλου καλή πρακτική! Πρώτα από όλα κάποιος θα έχει δυσανάλογο βάρος σε μια παράδοση, ενώ οι άλλοι δεν θα έχουν σχεδόν τίποτε να κάνουν και μετά θα μάθετε ο καθένας ένα μόνο μέρος της διαδικασίας.

Συνεργασία με άλλες ομάδες

Μπορείτε να συνεργαστείτε με άλλες ομάδες για να ανταλλάξετε εμπειρίες για τη μέθοδο εργασίας και για τον τρόπο που προσεγγίζετε το έργο σας. Το «*Δώστε μας το παραδοτέο σας και θα το αλλάξουμε τόσο που δεν θα μας καταλάβουν*» δεν είναι (και δεν θα είναι κάτι που θα λειτουργήσει για δύο διαφορετικά έργα)! Σε αυτές τις περιπτώσεις μεγαλύτερη ευθύνη έχει η ομάδα που δίνει το έργο της παρά η ομάδα που το αποδέχεται, αλλά δυστυχώς και οι δύο θα τιμωρηθούν.

6. Παραδοτέα Project

Το project της Τεχνολογίας Λογισμικού είναι κάτι δυναμικό και θα εξελίσσεται όσο θα εξελίσσεται η ίδια η επιστήμη. Άρα ανά χρόνο μπορεί να υπάρχουν αλλαγές στα παραδοτέα και στα «τεχνικά κείμενα» που αποτελούν κάθε παραδοτέο. Αυτό σημαίνει ότι ομάδες που εργάστηκαν για το project το χρόνο n, θα πρέπει να προβούν σε κάποιες αλλαγές αν θέλουν να το υποβάλουν το χρόνο n+1. Στο **παρόν εγχειρίδιο περιγράφεται πάντα η τρέχουσα έκδοση των παραδοτέων του project**. Η αναλυτική περιγραφή κάθε τεχνικού κειμένου υπάρχει στο τμήμα «Περιγραφές Τεχνικών Κειμένων».

Υποχρεωτικά και προαιρετικά παραδοτέα

Το παραδοτέα του project θα υποβληθούν τμηματικά σε 4 παραδόσεις. Από αυτές, η πρώτη και η τελευταία είναι υποχρεωτικές, ενώ η δεύτερη και η τρίτη είναι προαιρετικές. Σας προτρέπουμε να ετοιμάσετε και να υποβάλετε τα παραδοτέα σας και στις 2 προαιρετικές παραδόσεις, γιατί θα σας βοηθήσει να έχετε μια ομαλή ροή στη δουλεία σας και θα έχετε τη δυνατότητα να εντοπίσετε και να διορθώσετε τυχόν αδυναμίες ή αστοχίες, αρκετά νωρίς στην πορεία του project.

Ονομασία των τεχνικών κειμένων που αποτελούν κάθε παραδοτέο

Κάθε παραδοτέο αποτελείται από διάφορα τεχνικά κείμενα. Ένα «τεχνικό κείμενο» μπορεί να μην είναι κείμενο (συνήθως είναι διαγράμματα ή και κώδικας), αλλά για λόγους απλότητας θα τα ονομάζουμε όλα τεχνικά κείμενα. Κάθε τεχνικό κείμενο έχει ένα κωδικό (δίνεται παρακάτω με μπλε χρώμα και μια έκδοση). Μπορείτε να τροποποιείτε ή να βελτιώνετε την έκδοση κάθε τεχνικού κειμένου με κάθε παράδοση και πρέπει να αναφέρετε ότι είναι νέα έκδοση.

Οι τμηματικές παραδόσεις θα αφορούν τα παρακάτω. Προσοχή, αναλυτική περιγραφή όλων των παρακάτω μπορείτε να βρείτε στο τμήμα «Περιγραφές Τεχνικών Κειμένων».

- **Παραδοτέο 1 (Υποχρεωτικό):** Ένα αρχείο .pdf που θα περιλαμβάνει τα παρακάτω.
  - Project-description-v0.1
  - Use-case-v0.1
  - Domain-model-v0.1

  Προαιρετικά: Project-code-v0.1, αν έχετε ξεκινήσει την ανάπτυξη μπορείτε να δώσετε μια πρώτη έκδοση του κώδικά σας στο Git σας. Προσοχή: όχι κώδικα για τα use cases, αλλά για στοιχεία του UI.

- **Παραδοτέο 2 (Προαιρετικό):** Ένα αρχείο .pdf που θα περιλαμβάνει τα παρακάτω.
  - Robustness-diagram-v0.1
  - Use-case-v0.2 (αν υπάρχει νέα έκδοση, κανονικά θα πρέπει να υπάρχει)
  - Domain-model-v0.2 (αν υπάρχει νέα έκδοση, κανονικά θα πρέπει να υπάρχει)

  Προαιρετικά: Project-code-v0.x, αν έχετε ήδη (πολύ πιθανό) ξεκινήσει κώδικα δώστε τη νέα έκδοση (v0.x) όπως προκύπτει από το use-case diagram. Αν όχι τότε δώστε την πρώτη έκδοση (v0.1) του κώδικα στο Git σας.

- **Παραδοτέο 3 (Προαιρετικό):** Ένα αρχείο .pdf που θα περιλαμβάνει τα παρακάτω.

- Sequence-diagram-v0.1
- Robustness-diagram-v0.2 (αν υπάρχει νέα έκδοση)
- Use-case-v0.3 (αν υπάρχει νέα έκδοση)
- Domain-model-v0.3 (αν υπάρχει νέα έκδοση, κανονικά θα πρέπει να υπάρχει)
- Project-code-v0.x, αν έχετε ήδη (πολύ πιθανό) ξεκινήσει κώδικα δώστε τη νέα έκδοση (v0.x) όπως προκύπτει από το use-case diagram. Αν όχι τότε δώστε την πρώτη έκδοση (v0.1) του κώδικα στο Git σας.
- Test-cases-v0.1

- **Παραδοτέο 4-Τελικό (Υποχρεωτικό):** Ένα αρχείο .pdf που θα περιλαμβάνει τα παρακάτω.

  Υποχρεωτικά:
  - Όλες τις τελικές εκδόσεις για Sequence-diagram, Robustness-diagram, Use-case, και Domain-model, είτε έχει αλλάξει στην τελική έκδοση, είτε αν δεν έχει αλλάξει την τελευταία έκδοση, όπως περιγράφεται αναλυτικά στο τμήμα «Περιγραφές Τεχνικών Κειμένων».
  - Κυρίως σε αυτή τη φάση θα πρέπει να ασχοληθείτε με την τελική έκδοση του κώδικα Project-code-v1.0, η οποία θα πρέπει να περιλαμβάνει ικανοποιητική λειτουργικότητα τουλάχιστον για τα use case του έργου σας τα οποία αναλύσατε.
  - Class-diagram-v1.0
  - Test-cases-v1.0
  - Project-description-v1.0

Μορφοποίηση των παραδοτέων και stylistics

Όλα τα παραδοτέα θα παραδοθούν σε ένα συνολικό αρχείο pdf με ονόματα όπως ορίζονται παραπάνω συγκεντρωμένα σε ένα συμπιεσμένο αρχείο.

Προσπαθήστε τα κείμενά σας να είναι άρτια κείμενα τόσο τεχνικά όσο και σε επίπεδο μορφοποίησης. Προσπαθήστε το λεκτικό μέρος να είναι με σωστά ελληνικά, τα σχήματα σχεδιασμένα όμορφα και με σωστές αναφορές σε αυτά ώστε να είναι ξεκάθαρο σε ποιο αναφέρεστε. Χρησιμοποιήστε το σύστημα αναφοράς του κειμενογράφου που χρησιμοποιείτε (π.χ. εικόνα 1) και όχι εκφράσεις όπως στην παρακάτω εικόνα. Μπορείτε να συντάξετε τις αναφορές σε ό,τι θέλετε (LaTeX, Word, Open Office, άλλο), να σχεδιάσετε τα σχήματα με όποιο εργαλείο θέλετε και γράψετε κώδικα με όποιο IDE θέλετε. Απλά αναφέρετε τα εργαλεία που χρησιμοποιήσατε για κάθε παραδοτέο. Χρήση εργαλείων που αυτοματοποιούν τη δική σας δουλειά και σας βοηθούν να μεταβείτε από φάση σε φάση (π.χ. Visual Paradigm) θα εκτιμηθεί ιδιαίτερα. Υπάρχουν free εκδόσεις ή εκδόσεις που μπορείτε να χρησιμοποιήσετε για περιορισμένο χρόνο (π.χ. το Visual Paradigm έχει 30 ημέρες free trial ανά χρήστη).

Σε κάθε τεχνικό κείμενο που παραδίνετε θα πρέπει να υπάρχουν:

- Μια αρχική σελίδα με τον τίτλο του τεχνικού κειμένου, τον κωδικό του και την έκδοση (π.χ. Use-case-v0.2), και η σύντομη ονομασία του έργου σας υποχρεωτικά. Προαιρετικά μπορεί να υπάρχει και ένα λογότυπο της ομάδας σας ή του έργου σας (μην φορτώνετε την αρχική σελίδα με πολλή πληροφορία).
- Τα ονόματα και οι ΑΜ των μελών της ομάδας σας (στην αμέσως επόμενη σελίδα).

- Οι ρόλοι των μελών της ομάδας σας για αυτό το κείμενο (π.χ. editor, contributor, peer reviewer) αν υπάρχουν. Επίσης, όπως είπαμε πριν, τα μέλη της ομάδας που συμμετείχαν σε αυτό το τεχνικό κείμενο. Θυμίζουμε, ότι μπορείτε να δουλέψετε όλοι σε όλα (δεν το προτείνουμε) ή να μοιράσετε τη δουλειά, αλλά να εμπλέξετε τουλάχιστον δύο μέλη της ομάδας σας σε κάθε παραδοτέο. Άρα δεν είναι ανάγκη όλα τα μέλη να αναφέρονται και ως συμμετέχοντες στο κείμενο, μόνο όσοι πραγματικά εργάστηκαν και τι έκαναν.
- Ανάλογα με την έκδοση του συγκεκριμένου κειμένου (που θα πρέπει να είναι ξεκάθαρη στην αρχική σελίδα), σε περίπτωση που είναι αναθεωρημένη έκδοση σε κάτι που έχει παραδοθεί σε προηγούμενη έκδοση θα πρέπει να υπάρχει μια σύντομη περιγραφή τι άλλαξε και γιατί (καλύτερα αμέσως μετά την αρχική σελίδα). Για παράδειγμα, αν χρειάζεται να υποβάλετε το Use-case-v0.2 θα πρέπει να αναφέρετε τι άλλαξε από το Use-case-v0.1 και γιατί χρειάστηκε νέα έκδοση. Αν η έκδοση είναι τελική (π.χ. Use-case-v1.0), θα πρέπει να αναφέρετε αν έχει αλλάξει κάτι από την τελευταία έκδοση ή αν η τελική είναι ίδια με την τελευταία έκδοση. (Μια δήλωση στην αρχική σελίδα «ή έκδοση v1.0 είναι ίδια με την έκδοση v0.3 αρκεί».
- Εργαλεία που χρησιμοποιήθηκαν (π.χ. τα σχήματα 1-5 έγιναν με το εργαλείο …, τα σχήματα 6-8 με το εργαλείο …, ο κώδικας που παρουσιάζεται αναπτύχθηκε με το IDE …, κ.λπ.). **Μην παραδίδετε ποτέ μεμονωμένα αρχεία**, αλλά τα πάντα να έχουν ενσωματωθεί στο pdf σας!
- Αναφορές σε πηγές που χρησιμοποιήσατε όπως στο παράδειγμα του παρόντος (στο τέλος του κειμένου).

Αλλαγές από έκδοση v0.n σε έκδοση v0.n+1 και η έκδοση v1.0

Κάθε τεχνικό κείμενο μπορεί να παραδοθεί πολλές φορές. Θεωρητικά σε κάθε παραδοτέο μπορείτε να δίνετε και μια νέα έκδοση ενός. Η πρώτη έκδοση θα ονομαστεί v0.1, η δεύτερη v0.2 ενώ η τελική έκδοση θα ονομαστεί v1.0. Κάθε έκδοση θα πρέπει να είναι αυτόνομη, δηλαδή να μην υπάρχουν παραπομπές σε προηγούμενη έκδοση. Θεωρήστε ότι **κάποιος θα διαβάσει μόνο την τελευταία έκδοση και εκεί πρέπει να υπάρχει ό,τι χρειάζεται**. Παρόλα αυτά σε κάθε νέα έκδοση θα πρέπει να περιγράφεται ο λόγος που έγινε η αλλαγή και τι ακριβώς άλλαξε. Επίσης, οι αλλαγές μέσα στο κείμενο θα πρέπει να είναι ευδιάκριτες, χρησιμοποιώντας για αυτές π.χ. άλλο χρώμα γραμματοσειράς ή χρώμα επισήμανσης

Παράδειγμα κακής αναφοράς σε προηγούμενη έκδοση: Στο sequence diagram της εικόνας 4 στην έκδοση v0.2 θα πρέπει να αλλάξει μόνο το όνομα “inventory” σε “items_list” για να συνάδει με τον κώδικα. Δεν δίνουμε νέο σχήμα στην τρέχουσα έκδοση v0.3.

Παράδειγμα καλής αναφοράς σε προηγούμενη έκδοση: Το sequence diagram της εικόνας 4 που ακολουθεί έχει αλλάξει από την προηγούμενη έκδοση. Η αλλαγή αφορά μόνο το όνομα “inventory” που άλλαξε “items_list” για να συνάδει με τον κώδικα που υποβάλαμε.

Προφανώς, οι αλλαγές που θα κάνετε δεν θα αφορούν κάτι τόσο απλό και θα πρέπει να περιγράφουνε αναλυτικά τι έγινε.

Ως έκδοση **v1.0 ονομάστε όλα τα τελικά τεχνικά κείμενα που θα παραδώσετε με την τελική παράδοση του project σας**. Αν το τελευταίο τεχνικό κείμενο για κάτι ήταν η έκδοση v0.2 για παράδειγμα και σε αυτή δεν έγινε καμία αλλαγή, στο v1.0 απλά να αναφέρετε «*είναι η έκδοση v0.2 χωρίς καμία αλλαγή*».

Παράδοση των παραδοτέων μέσω eClass

Φροντίστε τα παραδοτέα σας να έχουν μικρό μέγεθος. Δεν έχει νόημα να επισυνάπτετε εικόνες ιδιαίτερα υψηλής ανάλυσης για παράδειγμα που να ανεβάζουν το μέγεθος του pdf. Στην ιδιαίτερα απίθανη, όμως, περίπτωση που θα υπάρξει πρόβλημα με το eClass (ξεπεράσετε το όριο που δέχεται), τότε στο παραδοτέο σας θα πρέπει να υπάρχουν όλα τα ζητούμενα, αλλά μπορεί κάποιο υλικό να παραπέμπει (μέσα από τα κείμενα) σε κάποιο ανοικτό[3] cloud σύστημα αναγράφοντας ξεκάθαρα μέσα στο κείμενο τους λόγους που το κάνατε (για παράδειγμα: «*Επειδή στο έργο μας τα γραφικά έχουν ιδιαίτερη σημασία, αναπτύξαμε 180 εικόνες υψηλής ανάλυσης*[4] *με όλες τις αρχικές οθόνες του χρήστη. Στο κείμενο έχουμε 2-3 παραδείγματα, αλλά όλες τις εικόνες μπορείτε να τις βρείτε εδώ*»).

[3] Δηλαδή να μην απαιτεί από εμάς να ανοίξουμε λογαριασμό για να κατεβάσουμε αυτό που μας στείλατε.

[4] Έλεος! Μην κάνετε κάτι τέτοιο!

7. Βαθμολογία Project

Το project βαθμολογείται συνολικά με την τελική του παράδοση. Τα επιμέρους παραδοτέα επιτελούν τους εξής στόχους: α) να έχετε μια ομαλή οργάνωση της εργασίας που απαιτείται κατανεμημένη ομοιόμορφα χρονικά, β) να έχετε κάποιας μορφής αναπληροφόρηση από τους διδάσκοντες ώστε να εντοπίσετε και να διορθώσετε προβλήματα πολύ νωρίς και να μην προκύψουν προβλήματα με την τελική παράδοση, και γ) να υπάρχει καλύτερη εποπτεία των έργων που θα προταθούν και να αποφευχθούν κακές πρακτικές παρόμοιων Έργων.

Ημερομηνίες παραδόσεων και ρήτρα καθυστέρησης

Για κάθε παραδοτέο υπάρχει μια ημερομηνία παράδοσης που ανακοινώνεται στο eClass. Σε περίπτωση καθυστέρησης θα υπάρχει ρήτρα που θα υπολογίζεται ως 2% για κάθε ημέρα καθυστέρησης αθροιστικά. Η ημέρα καθυστέρησης ορίζεται ως n+1sec[5] από την ώρα παράδοσης n και για κάθε 24 ώρες προστίθεται μια ημέρα καθυστέρησης. Μια ομάδα δεν μπορεί να ξεπεράσει συνολικά τις 15 ημέρες καθυστέρηση. Σε αυτή την περίπτωση το project μηδενίζεται αυτόματα. Η καθυστέρηση στο τελευταίο παραδοτέο δεν μπορεί να ξεπερνά τις 2 ημέρες (24 ώρες) για πρακτικούς λόγους.

Για παράδειγμα, αν μια ομάδα καθυστερήσει την παράδοση ενός παραδοτέου κατά 5 ημέρες θα έχει μείωση 10% στον τελικό βαθμό του project, αν μια άλλη ομάδα καθυστερήσει κατά 3 ημέρες ένα παραδοτέο και κατά 7 ημέρες ένα άλλο θα έχει μείωση 20% στον τελικό βαθμό του project, κοκ. Ο κανόνας αυτός δεν ισχύει για το τελικό παραδοτέο (Παραδοτέο 4ο) που θα πρέπει να παραδοθεί αυστηρά στην ημερομηνία που ορίζεται στο eClass και μέχρι 24 ώρες μετά τη λήξη αυτής. Η συνολική καθυστέρηση δεν μπορεί να υπερβαίνει τις 15 ημέρες. Σε περίπτωση που μια ομάδα τις υπερβεί αυτομάτως το project χαρακτηρίζεται ως «fail» και δεν αξιολογείται.

Παραδείγματα:

- Μια ομάδα καθυστέρησε μόνο στο 2ο παραδοτέο και αντί να το παραδώσει στις 22:00 που ήταν η καταληκτική ώρα της προθεσμίας, το έδωσε 2 λεπτά μετά. Δεν καθυστέρησε σε τίποτε άλλο. Τότε ο βαθμός του project θα είναι ο βαθμός που θα λάμβανε κανονικά μείον 2% του βαθμού αυτού.
- Μια άλλη ομάδα καθυστέρησε κατά 15 ώρες το 1ο παραδοτέο, κατά 2 ημέρες και 10 λεπτά το 2ο παραδοτέο και κατά 3 ημέρες και 20 ώρες το 3ο παραδοτέο. Δεν καθυστέρησε σε τίποτε άλλο. Η ρήτρα υπολογίζεται ως 1 ημέρα για το 1ο παραδοτέο, 3 ημέρες για το 2ο παραδοτέο και 4 ημέρες για το 4ο παραδοτέο, άρα ο βαθμός του project θα είναι ο βαθμός που θα λάμβανε κανονικά μείον 16% του βαθμού αυτού.
- Μια άλλη ομάδα δεν καθυστέρησε τίποτε, αλλά δεν πρόλαβε να υποβάλει το τελικό παραδοτέο του project ούτε στις 24 ώρες που ήταν η μέγιστη παράδοση. Έστειλε όμως mail αμέσως λίγο μετά την προθεσμία επειδή το eClass είχε κλείσει τη δυνατότητα παράδοσης. Η ομάδα αυτή μηδενίζεται, αλλά μπορεί να κρατήσει τη δουλειά της για το επόμενο έτος.

[5] Το ρολόι συστήματος του eClass είναι αυτό που καθορίζει την ώρα παράδοσης και ό,τι εμφανίζει είναι τελικό. Αν το ρολόι εμφανίζει «καθυστερημένη παράδοση», ανεξαρτήτως για πόσα λεπτά ή δευτερόλεπτα, η ρήτρα ενεργοποιείται αυτόματα.

- Μια άλλη ομάδα, καθυστέρησε 10 ημέρες το 1ο παραδοτέο και άλλες 10 το 2ο παραδοτέο, αλλά συνεχίζει κανονικά. Μπορεί να συνεχίσει, αλλά το project είναι ήδη fail. Η δουλειά που κάνει πάντως θα είναι χρήσιμη για τον επόμενο χρόνο.

Ατομικοί βαθμοί φοιτητών

Ατομικοί βαθμοί φοιτητών υπολογίζονται μόνο για όσους συμμετείχαν σε project. Κάθε project με ομάδα *n* ατόμων θα λάβει ένα βαθμό *B,* **που θα είναι ο τελικός βαθμός μετά τις ρήτρες**. Για κάθε project η ομάδα που το υλοποίησε θα δηλώσει μια κατανομή της προσπάθειας E_i (i=1…n) για κάθε ένα από τους n συμμετέχοντες. Το E_i παίρνει τιμές από 0.12 έως 0.25, για τους περιορισμούς δείτε παρακάτω. **Ο βαθμός κάθε φοιτητή υπολογίζεται ως $B*n*E_i$**.

Ισχύουν τα παρακάτω:

- Αν για κάποιους φοιτητές προκύπτουν ατομικοί βαθμοί >10 τότε μετατρέπονται σε βαθμό project=10 (δεν υπάρχει δηλαδή βαθμός project μεγαλύτερος του 10 σε καμία περίπτωση).
- Ακόμα και αν κάποιος φοιτητής ενός επιτυχημένου project έχει ατομικό βαθμό < 5 μπορεί να συμμετάσχει στις εξετάσεις του Ιουνίου και του Σεπτεμβρίου κανονικά και αν επιτύχει στο μάθημα (δηλαδή έχει συνολικό τελικό βαθμό >=5) τότε θεωρείται ότι ολοκλήρωσε το μάθημα κανονικά[6].

Κατανομή προσπάθειας

Για κάθε project *n* φοιτητών, η ομάδα που το υλοποίησε θα δηλώσει μια κατανομή της προσπάθειας E_n για κάθε έναν από τους n συμμετέχοντες, την οποία θα προσθέσει στο αρχείο Project-Description-1.0 στο τελευταίο και υποχρεωτικό παραδοτέο (Παραδοτέο 4ο). Για την κατανομή ισχύουν οι εξής περιπτώσεις: α) η ομάδα συμφωνεί **ομόφωνα** ότι η προσπάθεια όλων των φοιτητών ήταν ισοδύναμη άρα $E_i = 1/n$ (σε αυτή την περίπτωση απλά αναφέρει ότι «*η προσπάθεια όλων των μελών της ομάδας ήταν ισοδύναμη*» και τίποτε άλλο), β) η ομάδα συμφωνεί σε μια κατανομή που διαφέρει από το $E_i = 1/n$, *(με sum(E_i)=1)* και το προτείνει αναφέροντας ότι η απόφαση είναι **ομόφωνη**, γ) η ομάδα δεν μπορεί να συμφωνήσει ομόφωνα σε μια κατανομή της προσπάθειας και αναφέρονται όλες οι απόψεις (θεωρητικά μπορεί να υπάρχουν και n απόψεις) με αντίστοιχη τεκμηρίωση κάθε άποψης. Σε αυτή την περίπτωση οι διδάσκοντες εξετάζουν τις προτάσεις, συγκρίνουν την εργασία όπως προκύπτει στα παραδοτέα και αποφασίζουν. Σε κάθε πρόταση κατανομής, κάθε E_i θα πρέπει να ορίζεται **με ακρίβεια 2 δεκαδικών ψηφίων**.

Τελικός βαθμός μαθήματος

Ο τελικός βαθμός μαθήματος, υπολογίζεται ως **50% ο βαθμός του project και 50% ο βαθμός των εξετάσεων** και για να καταχωρηθεί βαθμός στην ηλεκτρονική γραμματεία πρέπει **ο βαθμός εξετάσεων να είναι τουλάχιστον 5**. Σε αντίθετη περίπτωση **καταχωρείται ο βαθμός της γραπτής εξέτασης** και ισχύουν όσα έχουν αναφερθεί παραπάνω (δείτε τον ατομικό βαθμό project και τη διάρκεια ισχύος του project).

[6] Για παράδειγμα αν κάποιος έχει συμμετάσχει σε project με τελικό βαθμό 6.5, αλλά ο ατομικός του βαθμός είναι 4.0 μπορεί να συμμετάσχει στις εξετάσεις. Αν βαθμολογηθεί στις εξετάσεις με 6.0 ή περισσότερο θα περάσει κανονικά το μάθημα.

8. Λογοκλοπή

Οτιδήποτε αναφέρετε στο project θα πρέπει να έχει αναφορά στις πηγές σας με τον ορθό τρόπο. Αν κάτι δεν αναφέρεται, θεωρείται αυτόματα ότι είναι προϊόν δική σας πνευματικής εργασίας. Αν όμως εντοπιστεί ότι κάτι τέτοιο δεν ισχύει, τότε το project μηδενίζεται αυτόματα, ασχέτως από την έκταση του φαινομένου (δηλαδή αν αφορά μόνο ένα σχήμα, μια εικόνα, ή ένα ολόκληρο παραδοτέο).

Επειδή το Project τρέχει για πολλοστή χρονιά είναι πολύ εύκολο να πάρετε ένα επιτυχημένο project, να το αντιγράψετε και να ελπίζετε ότι δεν θα το καταλάβουμε. **Μην το κάνετε**, ας μην βάζουμε δουλειά στην επιτροπή δεοντολογίας! Σε περίπτωση τέτοιας αντιγραφής θα υπάρχουν συνέπειες πέρα από της αυτονόητης (μηδενισμός του Project).

Στην περίπτωση που κάποιο τμήμα του έργου έχει βασιστεί σε κάτι που αναφέρετε, θα πρέπει να διευκρινίζεται λεπτομερώς και με τρόπο που δεν επιδέχεται αμφισβήτηση τι ακριβώς διαφοροποιεί τη δική σας δουλειά. Ακολουθούν παραδείγματα:

Παράδειγμα ορθής αναφοράς εικόνας που έχει αλιευθεί στο διαδίκτυο

Η Εικόνα 1 της αρχικής οθόνης του κινητού έχει δανειστεί αυτούσια από την ιστοσελίδα με τίτλο «User Interface Design Inspiration- 54 UI Design Examples»[7] με στόχο να δείξουμε ένα παράδειγμα για το πώς θα μπορούσε να είναι η τελική εφαρμογή. Όλες οι επόμενες οθόνες έχουν σχεδιαστεί με το εργαλείο Microsoft Visio 2019 και είναι πρωτότυπες δικές μας προτάσεις. Στην επόμενη έκδοση του συστήματος θα έχουμε εκδόσεις όλων των οθονών σχεδιασμένες από εμάς συμπεριλαμβανομένης και αυτής της εικόνας 1.

Εικόνα 1: Παράδειγμα αρχικής οθόνης

[7] https://www.designyourway.net/drb/user-interface-design-inspiration-40-ui-design-examples/

Παράδειγμα ορθής αναφοράς υλικού από πηγή που αναφέρεται στη βιβλιογραφία

Σε όποια παραδοτέα κρίνετε αναγκαίο μπορεί να υπάρχει βιβλιογραφία. Υλικό που έχει χρησιμοποιηθεί από πηγές που αναφέρετε στη βιβλιογραφία θα πρέπει επίσης να επεξηγείται λεπτομερώς. Για παράδειγμα:

Στο domain model διάγραμμα που ακολουθεί η οντότητα «παραγγελία» έχει βασιστεί σε μεγάλο βαθμό στο παράδειγμα της σελίδας 34 του βιβλίου (Rosenberg & Scott, 2001) που υπάρχει στη βιβλιογραφία. Επειδή η συγκεκριμένη οντότητα εκεί περιγράφεται πολύ αναλυτικά και με τρόπο που καλύπτει απόλυτα τη δική μας εφαρμογή, θεωρήσαμε υπερβολή να δώσουμε κάτι νέο. Παρόλα αυτά, όλα τα υπόλοιπα στοιχεία του domain model διαγράμματος είναι προϊόν δικής μας εργασίας.

Παράδειγμα ορθής αναφοράς υλικού από άλλη πηγή

Η πρόταση έργου που υποβάλαμε βασίστηκε σε παλαιότερο project στο οποίο μετείχαν τρία από τα πέντε μέλη της ομάδας μας και συγκεκριμένα οι … Επειδή το ζητούμενο του project έχει αλλάξει σημαντικά, το μόνο που κρατήσαμε από το παλαιότερο project ήταν η αρχική ιδέα με τις διαφοροποιήσεις που περιγράφονται ακολούθως… Όλα τα υπόλοιπα (use cases, robustness diagram, domain model) που θα βρείτε στο συγκεκριμένο παραδοτέο έχουν υλοποιηθεί αποκλειστικά από εμάς για τις ανάγκες του έργου που περιγράφουμε.

Παράδειγμα ορθής αναφοράς υλικού από κώδικα

Το μέρος του κώδικα που αφορά τη διεπαφή με κινητές συσκευές δεν έχει αναπτυχθεί μόνο από εμάς, αλλά είναι τμήμα ενός open source project που μπορεί να βρεθεί στο Git του συγκεκριμένου project[8]. Επειδή ήταν αναγκαίο για την καλύτερη παρουσίαση της δουλειάς μας το ενσωματώσαμε στον δικό μας κώδικα, αποδίδοντας όλα τα πνευματικά δικαιώματα στους δημιουργούς, μέλος της ομάδας ανάπτυξης ήταν και το μέλος της ομάδας μας ο … Όλα τα υπόλοιπα τμήματα κώδικα που θα βρείτε στο δικό μας Git είναι αποκλειστικά προϊόν δικής μας εργασίας.

[8] Σελίδα του έργου για το οποίο μιλάτε.

9. Υποστήριξη του Project

Παραδείγματα σχετικά με το project της Τεχνολογίας Λογισμικού που θα σας βοηθήσουν στην ανάπτυξη του έργου σας θα δίνονται σε κάθε φροντιστήριο του μαθήματος. Εκεί μπορούν να συζητηθούν και απορίες σχετικά με το project. Επίσης, μπορεί να γίνει αξιολόγηση μέσω του eClass σε κάποια παραδοτέα.

Προσοχή: Τα φροντιστήρια έχουν ως σκοπό την εμβάθυνση στην ύλη και την επίλυση αποριών για ύλη που έχει διδαχθεί στις παραδόσεις του μαθήματος. Δεν μπορούν να υποκαθιστούν τις παραδόσεις του μαθήματος και θα τρέχουν στο ρυθμό όσων έχουν παρακολουθήσει και μελετήσει τις παραδόσεις. Άρα για την επιτυχή παρακολούθηση των φροντιστηρίων (συνήθως φροντιστήριο θα γίνεται για ύλη που διδάχθηκε την προηγούμενη βδομάδα) θα πρέπει να ισχύουν δύο προϋποθέσεις: α) να έχετε παρακολουθήσει τη σχετική παράδοση και β) να έχετε μελετήσει και να έχετε προετοιμάσει απορίες.

Απορίες σχετικά με το project μπορούν επίσης να επιλύονται στις ώρες γραφείου των διδασκόντων και πιθανότατα και σε απορίες πολύ περιορισμένης κλίμακας και στο forum του μαθήματος.

10. Περιγραφές Τεχνικών Κειμένων

Όπως αναφέραμε, ονομάζουμε ως «τεχνικό κείμενο» για λόγους απλότητας κάτι που δεν είναι κατά κανόνα κείμενο (τα περισσότερα δεν είναι κείμενο). Άρα, για λόγους απλότητας θα ονομάζουμε τεχνικό κείμενο κάθε στοιχείο του έργου με το οποίο θα ασχοληθείτε και ακολούθως εξηγούμε τι είναι το καθένα από αυτά. Έτσι ακόμα και ο τελικός κώδικας θα είναι ένα τεχνικό κείμενο (αν και απλά θα παραπέμπει στο Git της ομάδας σας). Κάθε τεχνικό κείμενο έχει ένα κωδικό (είναι με μπλε χρώμα) που θα πρέπει να τον χρησιμοποιείτε και μια έκδοση (την οποία και είναι σημαντικό να αναφέρετε και να τονίζετε τι έχει αλλάξει από την προηγούμενη έκδοση).

Project-description (αφορά την περιγραφή του **έργου και στο τελικό τη συνεισφορά κάθε μέλους της ομάδας**)

Στην πρώτη έκδοση (v0.1), δώστε την αρχική περιγραφή έργου σε φυσική γλώσσα (ελληνικά) όπως θα την έδινε ένας πελάτης, χωρίς τεχνικούς όρους και χωρίς να υπεισέρχεστε σε τεχνικές ή σχεδιαστικές λεπτομέρειες.

Για το **παράδειγμα έργου του νέου Progress** σωστή περιγραφή μιας λειτουργίας είναι: «*Ο φοιτητής θα πρέπει να μπορεί να δηλώνει μαθήματα, να βλέπει τι μαθήματα έχει δηλώσει και να ενημερώνεται με μήνυμα ηλεκτρονικού ταχυδρομείου αν πλησιάζει η καταληκτική προθεσμία και δεν έχει δηλώσει μαθήματα*». Αντίθετα περιγραφές του τύπου: «*Ο φοιτητής θα βλέπει μια λίστα μαθημάτων οργανωμένα ανά έτος και θα μπορεί με δεξί κλικ να επιλέγει…*» μπαίνουν σε σχεδιαστική λεπτομέρεια που δεν έχει νόημα σε αυτή τη φάση (list, right click είναι design level details και όχι προδιαγραφές), ενώ περιγραφές του τύπου: «*Αν ο φοιτητής δεν έχει δηλώσει μαθήματα, το σύστημα θα στέλνει ένα μήνυμα 5 ημέρες πριν την καταληκτική ημερομηνία, ακολούθως 2 ημέρες πριν την καταληκτική ημερομηνία και μετά 6 ώρες πριν την καταληκτική ημερομηνία αναφέροντας επείγον στο subject*» είναι λεπτομέρειες που δεν έχει νόημα η συζήτησή τους σε αυτή τη φάση (θα είχε νόημα όμως η συζήτηση αυτών των λεπτομερειών σε επίπεδο use cases).

Η επιλογή ενός καλού έργου είναι σημαντικό στοιχείο για την επιτυχία του project,
άρα σκεφτείτε πολύ τι έργο θα προτείνετε.

Τι είναι καλό έργο; Ιδανικά κάτι που θα σκεφτόσαστε μήπως αξίζει παράλληλα με τις σπουδές σας να το υλοποιήσετε πραγματικά! Μια ιδέα δηλαδή που να έχει πραγματική αξία και που θα μπορέσετε να την υλοποιήσετε και να έχει κέρδος ή αξία για το κοινωνικό σύνολο. Επίσης, ένα καλό έργο δεν μπορεί να είναι κάτι τόσο απλό που κάποιος το φτιάχνει μόνος του, ή που η ομάδα σας θα μπορούσε να το υλοποιήσει στα πλαίσια του project. Για παράδειγμα ένα εργαλείο που σβήνει τα metadata από αρχεία pdf είναι μερικές γραμμές κώδικα και θα το έγραφε κάποιος σε 10’ χωρίς να φτιάξει ομάδα, ένα σύστημα διαχείρισης αρχείων pdf που θα παρέχει λειτουργίες όπως split a file, merge files, convert, add a page, delete metadata, κ.λπ. αρχίζει να γίνεται ενδιαφέρον ως έργο (μόλις κάψαμε μια ιδέα), αλλά ακόμα δεν έχει σημαντική πολυπλοκότητα ή σημαντικές λειτουργίες για να αποτελέσει κάτι πολύπλοκο ως έργο για να ασχοληθεί μια ομάδα (ένας καλός προγραμματιστής θα το έφτιαχνε σε πολύ λιγότερο από τις 60 ώρες που θα βάλει στο project). Αν όμως αρχίσουμε να προσθέτουμε και λειτουργίες διαμοίρασης αρχείων, υποστήριξη αρχείων σε cloud, κ.λπ. αρχίζει να γίνεται ένα έργο ικανοποιητικής πολυπλοκότητας. Αν βάλουμε και διαφορετικά ήδη πελατών (online πελάτης που το χρησιμοποιεί free για περιορισμένο

μέγεθος αρχείων, online πελάτης που πληρώνει για μια χρήση, online συνδρομητής που του παρέχουμε ένα πλήρες πακέτο λειτουργιών και cloud space, εταιρία που έχει ένα αριθμό χρηστών και συνεργατικές λειτουργίες) τότε αρχίζει να γίνεται ένα ενδιαφέρον έργο (που το κάψαμε ως θέμα). Σκεφτείτε λοιπόν μια εξαιρετική ιδέα που αν είναι τόσο καλή, θα αξίζει να αποτελέσει τεχνοβλαστό και να έχετε επαγγελματικό όφελος και όχι κάτι που είναι C&P από κάτι που έχετε χρησιμοποιήσει ήδη και σας αρέσει!

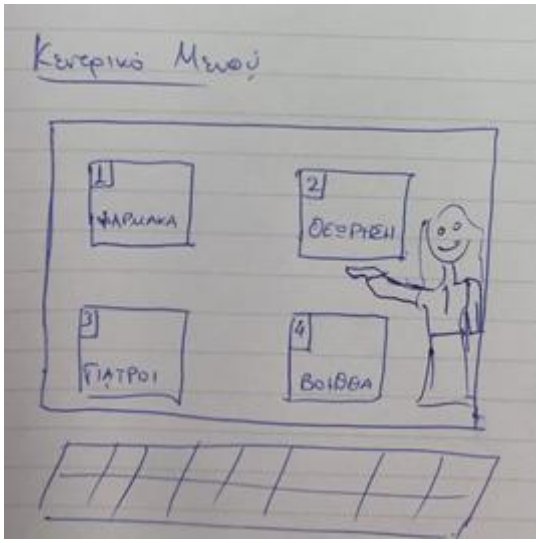
Σημαντική, αν και όχι τόσο όσο το ίδιο το έργο είναι και η **σύντομη ονομασία του έργου σας** (θυμίζουμε 3 έως 12 χαρακτήρες). Με αυτή την ονομασία θα αναφέρεστε και εσείς και εμείς στο έργο σας για το επόμενο εξάμηνο (ελπίζουμε) ή για τα επόμενα 3 χρόνια! Φροντίστε να είναι κάτι χαρακτηριστικό για το έργο σας.

Αν η ιδέα και η περιγραφή των λειτουργιών διαφοροποιηθεί σημαντικά από την έκδοση v0.1, θα πρέπει να υποβληθεί νέα έκδοση. **Μην υποβάλετε νέα έκδοση όταν περιγράφετε πιο αναλυτικά κάτι** (αυτό θα γίνει για παράδειγμα στα use cases), αλλά μόνο αν αποκλίνετε σημαντικά από αυτό που περιγράψατε.

Στην πρώτη έκδοση (v0.1), θα πρέπει να υπάρχουν ενδεικτικές mock-up screens όπως θα τις παρουσιάζατε στον πελάτη για τις βασικές λειτουργίες του έργου σας. Για το **παράδειγμα έργου του νέου Progress**, σίγουρα θα πρέπει να υπάρχει μια οθόνη (ή περισσότερες όπου χρειάζεται) με τις βασικές λειτουργίες ενός φοιτητή, ενός καθηγητή και της γραμματείας. Μπορούν να σχεδιαστούν σε οποιοδήποτε σχεδιαστικό εργαλείο (π.χ. Word draw) ή ακόμα και σε χαρτί.

Προαιρετικά στην επόμενη έκδοση (v0.2), μπορείτε να δώσετε υλοποιημένες οθόνες είτε με χρήση κάποιου rapid prototyping tool, είτε με κώδικα στο IDE που χρησιμοποιείτε.

Στην τελική έκδοση (v1.0), όποιες οθόνες έχουν υλοποιηθεί θα πρέπει να ενσωματωθούν, δείχνοντας το αρχικό παράδειγμα (mock-up) και την τελική οθόνη που υλοποιήθηκε. Μπορεί στην τελική έκδοση να υπάρχει και το πλήρες ιστορικό των οθονών. Στην Εικόνα 2 φαίνεται μια αρχική mock-up οθόνη που έχει σχεδιαστεί από συμφοιτητές σας, αλλά στα πλαίσια ενός workshop 2 ωρών (για αυτό και είναι τόσο απλή) με στυλό και χαρτί. Στην Εικόνα 3 φαίνεται η ίδια οθόνη όπως σχεδιάστηκε πάλι μέσα στο workshop, ενώ στην Εικόνα 4 είναι η τελική οθόνη, όπως αναπτύχθηκε τελικά.



Εικόνα 2: Παράδειγμα mock-up screen

Εικόνα 3: Αρχική σχεδίαση

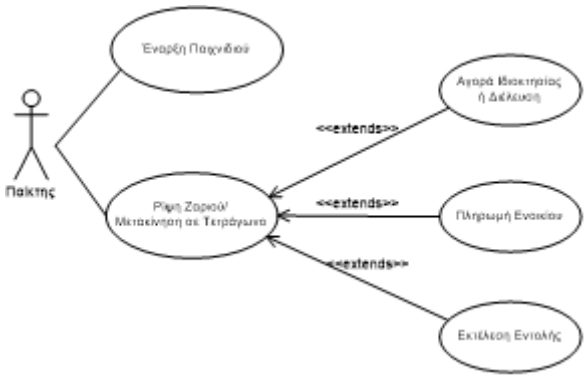
Εικόνα 4: Τελική οθόνη συστήματος

Προσοχή: Τα παραπάνω παραδείγματα έχουν υλοποιηθεί σε workshop μέσα στις 2 ώρες που διαρκούσε το workshop, άρα δεν είναι συγκρίσιμα με τη δουλειά που αναμένεται από μια ομάδα φοιτητών που σε λίγο θα γίνουν μηχανικοί!

ΣΗΜΑΝΤΙΚΟ: Θα δηλώσει μια κατανομή της προσπάθειας E_n για κάθε έναν από τους n συμμετέχοντες, η οποία θα προστεθεί στο αρχείο Project-Description-1.0, δηλαδή στο τελευταίο και υποχρεωτικό παραδοτέο.

Use-case

Στην πρώτη έκδοση (v0.1), περιγράψτε τους πιθανούς χειριστές του έργου σας και δώστε το συνολικό μοντέλο περιπτώσεων χρήσης του έργου σας.



Εικόνα 5: Use case model από ένα απλό έργο

Το παράδειγμα στην Εικόνα 5 είναι από μια ατομική εργασία στο μάθημα και είναι ιδιαίτερα απλό. Το δικό σας use case model λογικά θα έχει ένα σημαντικό αριθμό από use case (τουλάχιστον δύο σύνθετα use-case για κάθε μέλος της ομάδας). Για παράδειγμα στην ηλεκτρονική γραμματεία, μόνο ο καθηγητής έχει use cases όπως: αλλαγή προσωπικών στοιχείων, αλλαγή password, αίτημα σύνδεσης με μάθημα, αίτημα διαγραφής από μάθημα, αίτημα ξεκλειδώματος μαθήματος, υποβολή βαθμών για μάθημα, ανάρτηση βαθμών για μάθημα, έκδοση βαθμολογίου, κ.λπ.

Αν δεν προκύπτει σημαντικός αριθμός από use cases και αν στο domain model δεν προκύπτουν αρκετές κλάσεις, τότε μάλλον υλοποιείτε κάτι ιδιαίτερα απλό και πιθανότατα θα πρέπει να σκεφτείτε να προσθέσετε λειτουργίες στο έργο σας, έτσι ώστε να μην υλοποιείτε κάτι τόσο απλό. **Μην διστάσετε να αλλάξετε σε αυτή τη φάση το έργο σας**, αν χρειάζεται, ώστε να επεκταθεί σε κάτι που θα έχει ικανοποιητική εργασία ομάδας.

Ακολούθως στην πρώτη έκδοση (v0.1), θα πρέπει να αποφασίσετε πόσα από τα use cases του έργου σας θα αναλύσετε-σχεδιάσετε-υλοποιήσετε. Είναι προφανές ότι δεν μπορεί να γίνει για όλα, άρα επιλέξτε μερικά χαρακτηριστικά use cases του έργου σας και μοιράστε τα. Ιδανικά θα πρέπει κάθε μέλος της ομάδας σας να έχει δύο τουλάχιστον use cases εκτός αν κάτι είναι ιδιαίτερα πολύπλοκο και αναλάβει μόνο ένα, ή αν κάτι είναι ιδιαίτερα απλό και κάποιος σχεδιάσει περισσότερα από δύο.

Στην επιλογή των use cases που θα αναλύσετε να αποφύγετε use cases ιδιαίτερα γενικά (για παράδειγμα login χρήστη στην εφαρμογή) που δεν έχουν νόημα ειδικά για την εφαρμογή σας και να παρουσιάσετε use cases που έχουν ιδιαίτερη σημασία για το έργο σας και μόνο αυτό. **Η κακή επιλογή use cases ή η επιλογή τετριμμένων use cases είναι λόγος μη αποδοχής του project σας**. Στην καλύτερη για εσάς περίπτωση θα σας γυρίσουμε πίσω και θα σας ζητήσουμε να σχεδιάσετε άλλα use cases και στη χειρότερη δεν θα γίνει αποδεκτό στην τελική αξιολόγηση.

Για κάθε use case περιγράψτε αναλυτικά τη βασική και την εναλλακτική ροή. Ακολουθεί παράδειγμα (κείμενο με μπλε χρώμα) από το παιχνίδι στην Εικόνα 5, όπου περιγράφεται το use case “Αγορά Ιδιοκτησίας ή Διέλευση»:

Βασική Ροή «Πληρωμή ενοικίου»

1. Το σύστημα ελέγχει αν ο ιδιοκτήτης του συγκεκριμένου τετραγώνου κατέχει όλα τα τετράγωνα ιδίου χρώματος και διαπιστώνει ότι αυτό δεν ισχύει
2. Το σύστημα ανακτά την τιμή του ενοικίου για το συγκεκριμένο τετράγωνο και την εμφανίζει στην οθόνη "Πληρωμή Ενοικίου" με τον τίτλο του τετραγώνου, την τιμή ενοικίου και πλήκτρο επιβεβαίωσης
3. Ο παίκτης επιβεβαιώνει την πληρωμή
4. Το σύστημα αφαιρεί το ποσό που αντιστοιχεί στο ενοίκιο από το χρηματικό υπόλοιπο του παίκτη που διέρχεται από το τετράγωνο και προσθέτει το αντίστοιχο ποσό στο χρηματικό υπόλοιπο του παίκτη που είναι ο ιδιοκτήτης του τετραγώνου
5. Ο παίκτης κλείνει το μήνυμα "Πληρωμή Ενοικίου"
6. Το σύστημα ελέγχει το χρηματικό υπόλοιπο του παίκτη και διαπιστώνει ότι είναι μεγαλύτερο του μηδενός
7. Το σύστημα εμφανίζει την οθόνη "Ταμπλό"

Εναλλακτική Ροή 1

1.α.1 Το σύστημα διαπιστώνει ότι ο ιδιοκτήτης του τετραγώνου κατέχει όλα τα τετράγωνα ιδίου χρώματος

1.α.2 Το σύστημα ανακτά την τιμή του ενοικίου για το συγκεκριμένο τετράγωνο και εμφανίζει το διπλάσιό της στην οθόνη "Πληρωμή Ενοικίου"

1.α.3 Ο παίκτης επιβεβαιώνει την πληρωμή

1.α.4 Το σύστημα αφαιρεί το ποσό που αντιστοιχεί στο διπλάσιο του ενοικίου από το χρηματικό υπόλοιπο του παίκτη που διέρχεται από το τετράγωνο και προσθέτει το αντίστοιχο ποσό στο χρηματικό υπόλοιπο του παίκτη που είναι ο ιδιοκτήτης του τετραγΠΑΑΣ

1.α.5 Η περίπτωση χρήσης συνεχίζεται από το βήμα 5 της βασικής ροής

Εναλλακτική Ροή 2

6.α.1 Το σύστημα διαπιστώνει ότι το χρηματικό υπόλοιπο του παίκτη είναι ίσο ή μικρότερο του μηδενός και εμφανίζει μήνυμα "Αποχώρηση από το Παιχνίδι"

6.α.2 Ο παίκτης κλείνει το μήνυμα

6.α.3 Το σύστημα διαγράφει τον παίκτη που αποχώρησε και εμφανίζει την οθόνη "Ταμπλό" αφαιρώντας το πιόνι με το χρώμα που αντιστοιχεί στον παίκτη που αποχώρησε

Στη δεύτερη έκδοση (v0.2) των use case που θα υποβληθεί μετά τη σχεδίαση των robustness diagram θα έχουν προκύψει (από την ανάλυση ευρωστίας) αλλαγές στην περιγραφή των use case. Όσο καλά και να έχετε περιγράψει τα use case η ανάλυση ευρωστίας θα σας βοηθήσει να βελτιώσετε κάποια πράγματα στις ροές σας και να δώσετε εναλλακτικές περιγραφές. Δώστε τη δεύτερη έκδοση των use case περιγράφοντας τι ακριβώς αλλαγές έχετε κάνει μετά την ανάλυση ευρωστίας. Ένας καλός και σύντομος τρόπος να το κάνετε, είναι να έχετε το κείμενο που άλλαξε με άλλο χρώμα στο κείμενο της έκδοσης v0.2, εξηγώντας (σύντομα) και γιατί έγινε η αλλαγή. **Πιθανότατα θα πρέπει να σχεδιάσετε εκ νέου το use case diagram**.

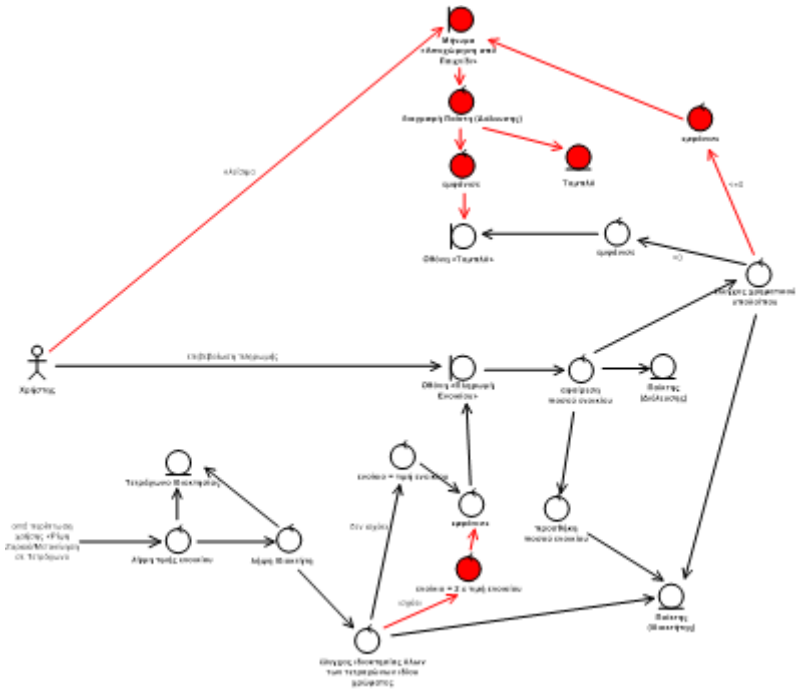
Για παράδειγμα, μια τέτοια αλλαγή στη βασική ροή στο παράδειγμα παραπάνω μετά την ανάλυση ευρωστίας θα μπορούσε να είναι: «*Στο βήμα 1 της βασικής ροής, για να υπάρχει η δυνατότητα εύρεσης του ιδιοκτήτη του τετραγώνου στο οποίο βρίσκεται το πιόνι του παίκτη που διέρχεται, διευκρινίζεται ότι πρέπει η πληροφορία αυτή (ο ιδιοκτήτης) να αντληθεί από το τετράγωνο*».

Είναι πολύ πιθανό (αλλά όχι σίγουρο, αν έχετε κάνει πολύ καλή ανάλυση ευρωστίας) **να προκύψουν αλλαγές στα use cases και μετά τα διαγράμματα ακολουθίας**. Σε αυτή την περίπτωση θα πρέπει να επανέλθετε με νέα έκδοση (v0.3) περιγράφοντας τις αλλαγές με παρόμοιο τρόπο (άλλο χρώμα και περιγραφή).

Robustness-diagram

Στην πρώτη έκδοση (v0.1), σχεδιάστε τα διαγράμματα ευρωστίας (robustness diagram) για όλα τα use case που έχετε παραδώσει. Ιδανικά το ίδιο μέλος της ομάδας σας που σχεδίασε κάθε use case θα πρέπει να συνεχίσει με το ανάλογο robustness diagram (και αργότερα με το sequence diagram και τον κώδικα). **Συντονιστείτε ως ομάδα**, ώστε να έχετε ένα κοινό τρόπο σχεδίασης και να μην είναι το κείμενο αλλοπρόσαλλο (π.χ. κάποιος να σημειώνει τις εναλλακτικές ροές με μπλε χρώμα, άλλος με πράσινο και άλλος με κόκκινο, ή ακόμα χειρότερα ο ένας να σχεδιάζει με word draw, ο άλλος με Visio και ο άλλος με Visual Paradigm). Μια καλή τακτική είναι ο καθένας σας να ελέγξει τα διαγράμματα κάποιου άλλου.

Στην Εικόνα 6 είναι ένα απλό (λόγω του απλού use case) παράδειγμα robustness diagram για το use case που δείξαμε στην Εικόνα 5. Προσέξτε ότι οι εναλλακτικές ροές εμφανίζονται με άλλο χρώμα.



Εικόνα 6: Παράδειγμα robustness diagram

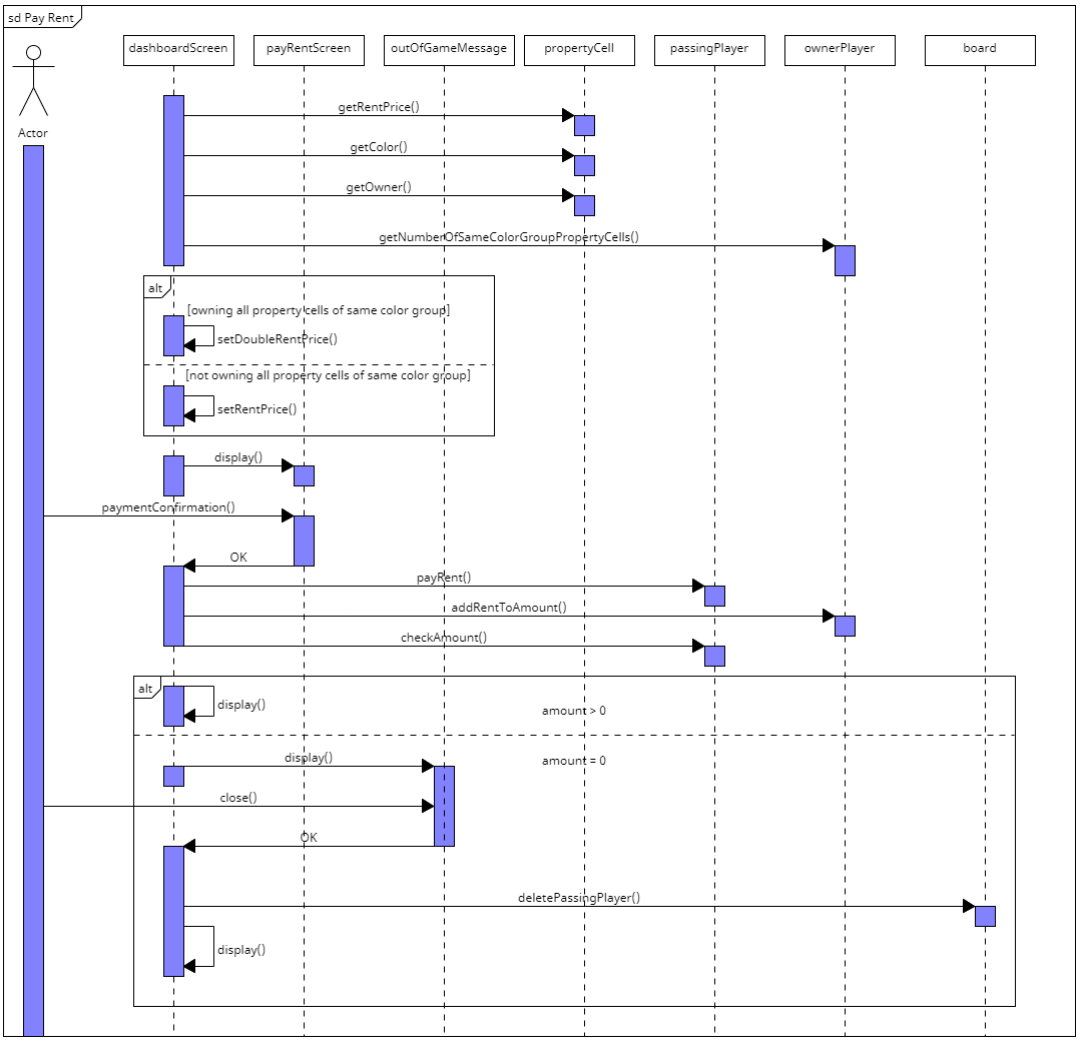
Μην ξεχνάτε το σκοπό για τον οποίο σχεδιάζουμε τα διαγράμματα ευρωστίας: α) για να ανακαλύψουμε προβλήματα και λάθη στα use cases και να τα βελτιώσουμε και β) για να εντοπίσουμε στοιχεία για το

domain model του έργου μας. Άρα θα πρέπει όλη αυτή η γνώση να περάσει στις νέες εκδόσεις των use cases και του domain model. Αν δεν το κάνετε, τότε η όλη διαδικασία πάει χαμένη!

Αν κατά τη σχεδίαση των διαγραμμάτων ακολουθίας προκύψουν αλλαγές στα robustness diagram τότε θα πρέπει οπωσδήποτε να υποβληθεί η δεύτερη έκδοση (v0.2). Το να έχετε διαγράμματα ακολουθίας που δεν έχουν καμία σχέση με τα διαγράμματα ευρωστίας είναι σοβαρό λάθος. Λογικά δεν θα χρειαστούν επόμενες εκδόσεις, αλλά αν χρειαστεί τις υποβάλλετε.

Sequence-diagrams

Στην πρώτη έκδοση (v0.1), των διαγραμμάτων ακολουθίας, αξιοποιήστε τη γνώση από τα διαγράμματα ευρωστίας και σχεδιάστε τα αντίστοιχα sequence diagram. Ιδανικά το ίδιο μέλος της ομάδας σας που σχεδίασε κάθε use case θα πρέπει να συνεχίσει με το ανάλογο robustness diagram (και αργότερα με το sequence diagram και τον κώδικα). **Συντονιστείτε ως ομάδα**, ώστε να έχετε ένα κοινό τρόπο σχεδίασης και να μην είναι το κείμενο αλλοπρόσαλλο (π.χ. είναι πιθανό να αναφέρεστε στις ίδιες κλάσεις σε δύο διαφορετικά διαγράμματα άρα θα πρέπει να αναφέρεστε με τον ίδιο τρόπο, ή ακόμα χειρότερα ο ένας να σχεδιάζει με word draw, ο άλλος με Visio και ο άλλος με Visual Paradigm). Μια καλή τακτική είναι ο καθένας σας να ελέγξει τα διαγράμματα κάποιου άλλου.



Εικόνα 7: Παράδειγμα sequence diagram

Στην Εικόνα 7 είναι ένα παράδειγμα sequence diagram για το ίδιο use case και robustness diagram που χρησιμοποιήσαμε ως παράδειγμα πιο πριν, σχεδιασμένο με το UMLet. Μην ξεχνάτε το σκοπό για τον οποίο σχεδιάζουμε τα διαγράμματα ευρωστίας: α) για να γνωρίζουμε τα μηνύματα που ανταλλάσσουν οι κλάσεις και να γράψουμε τις κατάλληλες μεθόδους στον κώδικα, άρα αυτή η γνώση θα πρέπει να μεταφερθεί στον κώδικα (αν συνδέσετε κάποιο εργαλείο όπως το Visual Paradigm με κάποιο IDE αυτό μπορεί να γίνει και αυτόματα σε επίπεδο ονομάτων και κλήσεων) και β) για να εντοπίσουμε στοιχεία για το domain model του έργου μας. Άρα θα πρέπει όλη αυτή η γνώση να περάσει στη νέα έκδοση του domain model και κυρίως στον κώδικα. Αν δεν το κάνετε, τότε η όλη διαδικασία πάει χαμένη!

Λογικά δεν θα χρειαστούν επόμενες εκδόσεις διαγραμμάτων ακολουθίας, αλλά αν χρειαστεί τις υποβάλλετε. Αν κατά την υλοποίηση, γράφοντας κώδικα κάτι δεν μπορεί να υλοποιηθεί όπως έχει σχεδιαστεί θα πρέπει οι αλλαγές που θα γίνουν στον κώδικα να απεικονιστούν και στα διαγράμματα ακολουθίας. Το να έχετε διαγράμματα ακολουθίας που δεν έχουν καμία σχέση με τον κώδικά σας είναι σοβαρό λάθος. Αν όντως προκύψουν αλλαγές τότε θα πρέπει οπωσδήποτε να υποβληθεί η δεύτερη έκδοση (v0.2).

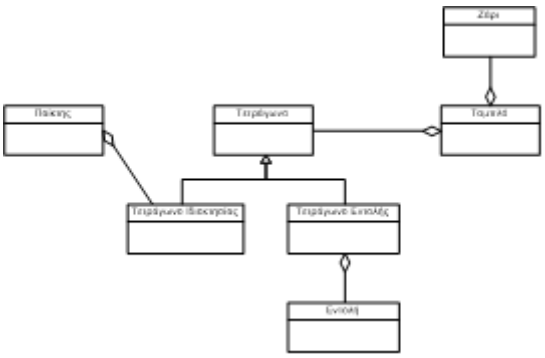
Domain-model

Στην πρώτη έκδοση (v0.1), θα πρέπει να αναφέρετε τις υποψήφιες κλάσεις του πεδίου (domain) του προβλήματος. Μπορείτε να βασιστείτε στη λεκτική περιγραφή του προβλήματος και να καταγράψετε όλα τα ουσιαστικά (εκτός από αυτά που αναφέρονται στο ίδιο το έργο/εφαρμογή/πρόβλημα) ως υποψήφιες κλάσεις (συνήθως στην ονομαστική του ενικού). Ακολούθως θα συζητήσετε τις κλάσεις και θα καταλήξετε σε κάποιες πιθανές (ενώ θα αποκλείσετε κάποιες άλλες).

Για παράδειγμα, στο παράδειγμα έργου της ηλεκτρονικής γραμματείας, παραδείγματα από πιθανές κλάσεις θα μπορούσαν να είναι:

- Φοιτητής
- Έτος εγγραφής φοιτητή
- Αριθμός μητρώου φοιτητή
- Καθηγητής
- Υπάλληλος γραμματείας
- Τμήμα
- Μάθημα
- Βαθμός
- Μονάδες ECTS

Οι κλάσεις αυτές είναι "υποψήφιες" υπό την έννοια ότι μπορεί να αποτελέσουν τελικά και κλάσεις της σχεδίασης του συστήματος, αλλά προφανώς μπορεί να μην διατηρηθούν ως κλάσεις μέχρι το πέρας της αντικειμενοστρεφούς ανάλυσης και σχεδίασης. Επίσης, κάποιες από τις κλάσεις αυτές πιθανόν να αποτελέσουν ιδιότητες (attributes) άλλων κλάσεων. Είναι αυτονόητο ότι υπάρχουν πολλές εναλλακτικές λύσεις στο στάδιο αυτό και σε κάθε λύση είναι δυνατόν να επιλεγεί ένα διαφορετικό σύνολο κλάσεων. Όμως είναι σημαντικό να γίνει σωστά αυτή η διαδικασία **για να έχετε ως ομάδα μια κοινή ορολογία** που θα την τηρείτε σε όλες τις φάσεις της ανάλυσης και σχεδίασης. Για παράδειγμα δεν μπορεί κάποιο μέλος της ομάδας σας να λέει σε ένα use case «Αριθμός μητρώου φοιτητή» και κάποιος άλλος «ΑΜ» και αργότερα στον κώδικα ο ένας να γράφει “arithmos_mitroou” και ο άλλος “am_foititi”.



Εικόνα 8: Παράδειγμα domain model από την έκδοση v0.2

Στην πρώτη έκδοση (v0.1), θα πρέπει να δώσετε ένα αρχικό σχήμα του domain model (είναι το πρώιμο class diagram, με τις βασικές κλάσεις και τις συνδέσεις ανάμεσά τους, αλλά χωρίς ακόμα να υπάρχουν

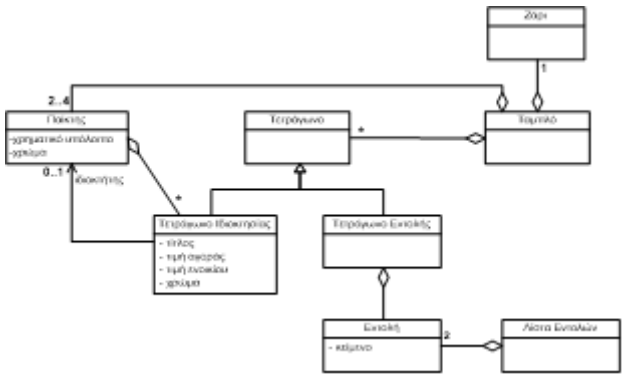
attributes και methods. Ένα απλό παράδειγμα ενός τέτοιου σχήματος από ένα παιχνίδι (ήταν από ατομική εργασία για το ακαδημαϊκό έτος 2017-2018) παρουσιάζεται στην Εικόνα 8.

Η πρώτη έκδοση (v0.1) θα πρέπει να περιλαμβάνει και μια σύντομη περιγραφή των κλάσεων, δηλαδή για το παραπάνω παράδειγμα θα αρκούσε κάτι σαν:

- Παίκτης: Οντότητα που περιλαμβάνει τις ιδιότητες κάθε πραγματικού παίκτη που συμμετέχει στο παιχνίδι, όπως το αναγνωριστικό του και το χρηματικό του υπόλοιπο.
- Ταμπλό: Μια (κυκλική) λίστα από τα τετράγωνα που αντιστοιχεί στην επιφάνεια όπου παίζεται το παιχνίδι. Περιλαμβάνει τετράγωνα.
- Τετράγωνο: Η γενική οντότητα που αναφέρεται στα τετράγωνα που περιλαμβάνει το ταμπλό. Χαρακτηρίζεται από τη θέση του στο ταμπλό.
- Τετράγωνο Ιδιοκτησίας: Ειδικότερη περίπτωση τετραγώνου που μπορεί να αποτελέσει ιδιοκτησία κάποιου παίκτη, αν αυτός το αγοράσει. Στον ιδιοκτήτη ενός τέτοιου τετραγώνου καταβάλλεται ενοίκιο κατά τη διέλευση.
- Τετράγωνο Εντολής: Ειδικότερη περίπτωση τετραγώνου. Δεν μπορεί να αποτελέσει ιδιοκτησία κάποιου παίκτη. Κατά τη διέλευση από αυτό λαμβάνεται κάθε φορά (με τυχαίο τρόπο) μια εντολή από μια λίστα εντολών που πρέπει να εκτελέσει ο παίκτης.
- Εντολή: Οντότητα που περιλαμβάνει την περιγραφή μιας εντολής που πρέπει να εκτελέσει ο χρήστης και τυχόν οδηγίες.
- Ζάρι: Οντότητα που αντιστοιχεί στο πραγματικό ζάρι. Επιστρέφει έναν τυχαίο αριθμό από το 1 μέχρι το 6 όταν ζητηθεί.

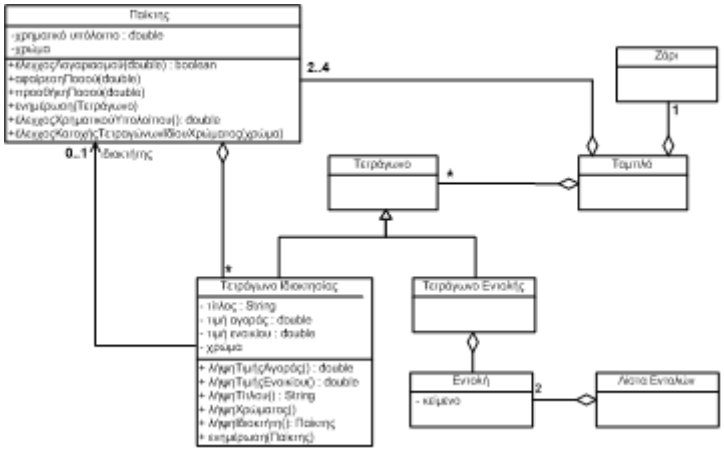
Σε αυτό το σημείο έχετε και μια εκτίμηση της πολυπλοκότητας του συστήματος. Αν προκύπτει κάτι τόσο απλό όσο το παραπάνω, μάλλον είναι ατομική εργασία και όχι έργο ανάλογο του project. Σε παρόμοιες περιπτώσεις πιθανότατα θα πρέπει να σκεφτείτε να προσθέσετε λειτουργίες στο έργο σας, έτσι ώστε να μην υλοποιείτε κάτι τόσο απλό. **Μην διστάσετε να αλλάξετε σε αυτή τη φάση το έργο σας**, αν χρειάζεται, ώστε να επεκταθεί σε κάτι που θα έχει ικανοποιητική εργασία ομάδας.

Στις επόμενες εκδόσεις του domain model αξιοποιείτε την εμπειρία από την ανάλυση ευρωστίας και τη σχεδίαση των robustness diagrams, και την ανάλυση των διαγραμμάτων ακολουθίας, και υποβάλετε τη δεύτερη έκδοση (v0.2).



Εικόνα 9: Εμπλουτισμένο domain model μετά την ανάλυση ευρωστίας

Η δεύτερη έκδοση του domain model (v0.2) θα πρέπει να περιλαμβάνει όλες τις αλλαγές που έχουν προκύψει στις κλάσεις, να έχει (αν χρειάζεται) βελτιωμένη περιγραφή των κλάσεων και να έχει συμπεριλάβει στο στατικό μοντέλο attributes που έχουν προκύψει. Ένα παράδειγμα υπάρχει στην Εικόνα 9, από μια μικρή ατομική εργασία, άρα είναι ένα ιδιαίτερα απλό μοντέλο. Τέλος, μετά την ανάλυση των διαγραμμάτων ακολουθίας θα πρέπει να προσθέσετε και τις μεθόδους που έχουν προκύψει, αλλά και όσα νέα attributes έχουν προκύψει. Ένα τέτοιο (ιδιαίτερα απλό) παράδειγμα παρουσιάζεται στην Εικόνα 10.



Εικόνα 10: Παράδειγμα domain model μετά τα διαγράμματα ακολουθίας

Class-diagram

Στην πρώτη έκδοση (v1.0), παραδώστε το τελικό class diagram, που θα βασιστεί σε μεγάλο βαθμό στην τελευταία έκδοση του domain model, μετά τις τελικές λεπτομέρειες που προέκυψαν κατά την ανάπτυξη του κώδικα. Πρακτικά είναι το domain model με μερικές προσθήκες που αφορούν λεπτομέρειες υλοποίησης.

Προσοχή τα ονόματα των κλάσεων, μεθόδων και attributes θα πρέπει να είναι ίδια με αυτά που δημιουργήσατε στον κώδικα, άρα καλύτερα ονόματα με λατινικούς χαρακτήρες και όχι ελληνικά όπως στο παράδειγμα.

Project-code (αφορά τον κώδικα του **έργου**)

Μπορείτε να ξεκινήσετε να γράφετε κώδικα από το παραδοτέο 1, αλλά μην γράφετε κώδικα για τα use cases γιατί αυτό ακυρώνει το σκοπό όλης της διαδικασίας! Μπορείτε όμως να ξεκινήσετε να γράφετε κώδικα για τμήματα του user interface.

Αρχίστε να γράφετε κώδικα όταν έχετε τις αντίστοιχες πληροφορίες στο domain model. Δηλαδή όταν έχετε τις κλάσεις και τα attributes μπορείτε να τις δημιουργήσετε, όταν έχετε τις μεθόδους μπορείτε να τις ονομάσετε, κ.λπ. Το μεγάλο μέρος του κώδικα θα γραφεί όταν θα έχετε σχεδιάσει τα sequence diagrams. Εκεί αναμένουμε να υπάρχουν τουλάχιστον όλα τα σχετικά που να υλοποιούν όλα όσα έχετε σχεδιάσει στα sequence diagrams και στο class diagram του έργου σας. Το βάθος των λειτουργιών που θα υλοποιήσετε εξαρτάται από εσάς και από το χρόνο που θα διαθέσετε στην ανάπτυξη, αλλά προσπαθήστε ό,τι αναλύσατε και σχεδιάσατε στα πλαίσια του έργου σας να υλοποιηθεί, αφήνοντας

εκτός λειτουργίες για τις οποίες επιλέξατε να μην δώσετε use cases. Εάν υπάρχει αναμενόμενη λειτουργικότητα από κλάσεις που δεν έχουν υλοποιηθεί, μπορεί να γίνει με ένα προκαθορισμένο μήνυμα.

Για παράδειγμα, αν στο παιχνίδι που δείξαμε ως παράδειγμα στο domain model δεν έχει φτιαχτεί η class Dice (αν και είναι κάτι αστείο όπως φαίνεται στο παρακάτω παράδειγμα σε Java), μπορεί να φτιάξετε μια κλάση που να επιστρέφει πάντα 6 ώστε να μπορέσετε να εξετάσετε τη λειτουργία των κλάσεων που έχετε σχεδιάσει.

```
public class Dice {

	public int getRandomNumber() {

		int result;
		result = (int) (Math.random() * 6);
		result += 1;

		return (result);
	}
}
```

Χρόνο για να γράψετε κώδικα έχετε πριν το Παραδοτέο 3, στο οποίο ουσιαστικά θα υποβάλετε τις τελευταίες τελικές εκδόσεις των τεχνικών κειμένων που έχετε ήδη ολοκληρώσει (με μικρές αλλαγές όπου και αν χρειάζεται) και κυρίως θα πρέπει να ολοκληρώσετε τον κώδικα του έργου σας. Ιδανικά, όμως, μπορείτε να έχετε πολλά στοιχεία του κώδικα από πριν, δουλεύοντας κατά τη διάρκεια των προηγούμενων παραδοτέων.

Στο τεχνικό κείμενο project-code δεν θα υπάρχει κώδικας. Θα υπάρχει απλά **παραπομπή στο Git σας** και μια σύντομη περιγραφή τι θα βρούμε εκεί (αν και αυτή μπορεί επίσης να υπάρχει στο Git σας). Απλά, στις παραδόσεις πριν την τελική παράδοση, επειδή μπορεί μέχρι να δούμε εμείς τον κώδικα, εσείς να έχετε κάνει αλλαγές, ενημερώστε μας για ποια έκδοση αναφέρεστε.

Στο Git του έργου σας θα πρέπει να υπάρχει όχι μόνο ο κώδικας, αλλά και η πρόοδος στην ανάπτυξη (διαφορετικές εκδόσεις) και ο τρόπος συνεργασίας (ποιος ανέβασε τι). Επειδή **υπάρχει αναλυτικό ιστορικό** μπορούμε άνετα να δούμε πότε μια ομάδα συνεργάστηκε και πότε ένα μέλος της ομάδας ανέβασε τα πάντα βιαστικά και στο παρά πέντε!

Test-cases (αφορά test cases για τον κώδικα του **έργου**)

Με βάση αυτά που διδαχθήκατε στο μάθημα, επιλέξτε μια ή περισσότερες μεθόδους ελέγχου και δώστε τα δεδομένα ελέγχου με τα οποία ελέγξατε τον κώδικά σας. Ιδανικά μπορούν να παραδοθούν δύο εκδόσεις με test cases, η πρώτη έκδοση (v0.1) με την προτελευταία έκδοση του κώδικα στο παραδοτέο 3 και η τελική έκδοση (v1.0) μαζί με το τελικό παραδοτέο 4. Στην τελική έκδοση μπορείτε να προσθέσετε νέα δεδομένα ελέγχου για κώδικα που προσθέσατε και αλλάξατε, αλλά και αποτελέσματα ελέγχων στον κώδικά σας.

11. Συνηθισμένα λάθη

- Δεν έχετε ένα καλό project brand name (π.χ. έχετε κάτι σαν Ε.Τ.Λ.Γ.Ν.Π) και ακόμα χειρότερα δεν το χρησιμοποιείτε σε όλα τα παραδοτέα σας. Εμείς στο έργο σας θα αναφερόμαστε με αυτό το όνομα.
- Παρακολουθείτε το μάθημα «δια αντιπροσώπου». Παρουσιάστηκε το φαινόμενο ομάδες να στέλνουν στο μάθημα ένα μέλος της ομάδας το οποίο με τη σειρά του τους μετέφερε τι ειπώθηκε. Τελικά προφανώς είχαμε «σπασμένο τηλέφωνο» και αρκετές παρεξηγήσεις!
- Φτιάχνετε τον κώδικα αγνοώντας την ανάλυση και σχεδίαση που έχει προηγηθεί! Επειδή πολλές ομάδες καταλήγουν (κακώς) να κάνουν κάτι εύκολο, τελικά κάποιος γράφει τον κώδικα χωρίς να δώσει και μεγάλη σημασία στις φάσεις που έχουν προηγηθεί.
- Το Git του έργου σας έχει μηδενική συνεργατικότητα και δραστηριότητα. Στο Git θα βάλετε (ο καθένας ατομικά) τον κώδικα που συνεισφέρατε, θα καταχωρήσετε τις αλλαγές, θα δημιουργήσετε branches αν χρειαστεί και θα αλλάξετε ότι χρειαστεί. Αν αυτό δεν φαίνεται και απλά μπει κάποιος την τελευταία μέρα και ανεβάσει όλο τον κώδικα τότε η συνεισφορά σας δεν μπορεί να κριθεί. Το Git σας θα πρέπει να είναι “ζωντανό” σε όλη τη διάρκεια του έργου σας!

ΚΑΛΗ ΣΑΣ ΕΠΙΤΥΧΙΑ ΣΤΟ PROJECT

Αναφορές

Rosenberg, D., & Scott, K. (2001). *Applying use case driven object modeling with UML: an anotated e-commerce example*: Addison-Wesley Professional.

Stellman, A., & Greene, J. (2014). *Learning agile: Understanding scrum, XP, lean, and kanban*: " O'Reilly Media, Inc.".