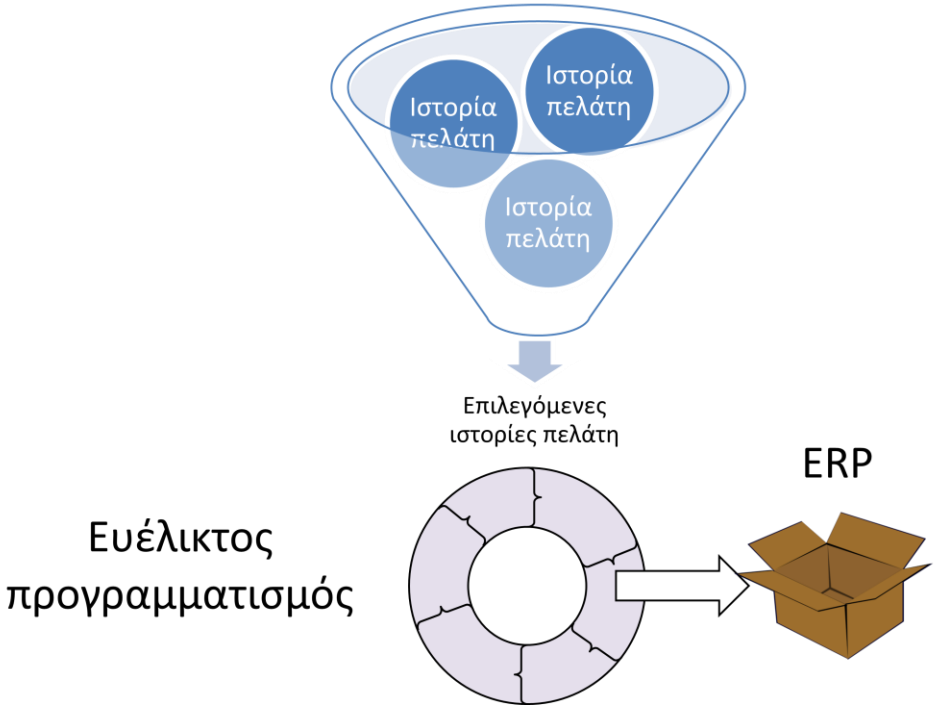
ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πτυχιακή εργασία

«ΕΥΕΛΙΚΤΟ ERP. ΥΛΟΠΟΙΗΣΗ ΕΝΟΣ ΜΙΚΡΟΥ ΣΥΣΤΗΜΑΤΟΣ ERP.»



Των φοιτητών

Λιάρα Ευαγγέλου, Παντελάκη Αριστείδη
Αρ. Μητρώου: 042485, 042465

Επιβλέπων καθηγητής

Σφέτσος Παναγιώτης

Θεσσαλονίκη 2011

Περιεχόμενα

1 Εισαγωγή

Το φαινόμενο της αποτυχίας μεγάλων έργων, που αφορούν την υλοποίηση νέων προγραμμάτων, είναι αρκετά συχνό. Το γεγονός αυτό αποτρέπει αρκετούς οργανισμούς να επενδύσουν στις νέες τεχνολογίες. Επίσης, το κόστος καθώς και ο χρόνος υλοποίησης των προγραμμάτων αυτών, λειτουργούν ως απαγορευτικοί παράγοντες, με αποτέλεσμα αρκετές εταιρίες να μη μετατρέπονται σε ανταγωνιστικές. Για τους παραπάνω λόγους, η έννοια του ευέλικτου προγραμματισμού γίνεται όλο και πιο διαδεδομένη και όλο και περισσότερες εταιρίες λογισμικού τη χρησιμοποιούν. Όπως θα δούμε σε επόμενα κεφάλαια, ο ευέλικτος προγραμματισμός εξασφαλίζει την παράδοση κώδικα, ακόμη κι όταν το επιθυμητό λογισμικό είναι σε φάση υλοποίησης.

Ένα υπολογιστικό σύστημα που πρέπει να υπάρχει σε κάθε εταιρία, είναι το ERP(Enterprise Resource planning), σκοπός του οποίου είναι η αυτοματοποίηση αρκετών επιχειρησιακών λειτουργιών. Για τη δημιουργία τέτοιων προγραμμάτων απαιτούνται αρκετά κεφάλαια και μεγάλος χρόνος υλοποίησης, με αποτέλεσμα ο πελάτης, λόγω αλλαγής συνθηκών-ευκαιριών, να επιθυμεί την τροποποίηση αρκετών πτυχών του προγράμματος κατά τη δημιουργία του. Ο ευέλικτος προγραμματισμός συνεισφέρει σε αυτή την κατεύθυνση, δηλαδή στην τροποποίηση των επιθυμιών του πελάτη, με αποτέλεσμα την αύξηση της εμπιστοσύνης του πελάτη.

Στη συγκεκριμένη πτυχιακή εργασία, θα αναλύσουμε δύο μεθοδολογίες του ευέλικτου προγραμματισμού, ενώ παράλληλα θα χρησιμοποιηθεί μία από αυτές για την υλοποίηση ενός μικρού ERP. Επιπροσθέτως, θα ασχοληθούμε με τον τρόπο υλοποίησης ενός Agile ERP και θα σχολιάσουμε τα αποτελέσματα του πειράματος, αποδεικνύοντας με αυτό τον τρόπο κατά πόσο επωφελούμαστε από τη χρήση του ευέλικτου προγραμματισμού.

2 Ευέλικτος προγραμματισμός

Πολλές γνωστές εταιρίες παραγωγής κώδικα, όχι μόνο υιοθέτησαν τον ευέλικτο προγραμματισμό, αλλά τον υποστηρίζουν και τον προτείνουν ανεπιφύλακτα. Μερικές από αυτές είναι: Microsoft, Symantec, Google και η IBM. Τι είναι όμως ο ευέλικτος προγραμματισμός και γιατί το χρησιμοποιούν; Ο ευέλικτος προγραμματισμός αναφέρεται σε ομάδα προγραμματιστικών μεθοδολογιών, οι οποίες έχουν κοινές αρχές. Πριν αναλύσουμε τις μεθοδολογίες αυτές και τις τεχνικές τους, θα κάνουμε μία ιστορική αναδρομή για να κατανοήσουμε την ανάγκη της δημιουργίας τους.

Στον προγραμματισμό πάντα υπήρχε η ανάγκη για δημιουργία μεθόδων οι οποίες όχι μόνο θα προέβλεπαν το κόστος, αλλά και το χρόνο τον οποίο θα χρειαζόταν το κάθε έργο για την ολοκλήρωσή του. Κάτι τέτοιο δε συνέβη για δεκαετίες, ούτε ακόμα όταν έγινε ευρέως αποδεκτός ο αντικειμενοστραφής προγραμματισμός. Τα πράγματα έγιναν ακόμα χειρότερα με το πέρας του χρόνου, αφού τα προγράμματα έπρεπε να είναι πιο πολύπλοκα, έτσι ώστε να είναι πιο ανταγωνιστικά. Σύμφωνα με στατιστικές μελέτες, το 31.1% των έργων δεν ολοκληρώνονταν, ενώ το 52.7% είχε έξοδα πολύ μεγαλύτερα από τα έξοδα που είχαν εκτιμηθεί τα οποία άγγιζαν το 189% της αρχικής εκτίμησης. Όλα αυτά, είχαν ως συνεπεία είτε την πλήρη οικονομική καταστροφή, είτε τη μείωση της αξιοπιστίας των εταιριών εγγραφής κώδικα. Όμως, πλήγμα δεχόταν και ο πελάτης ο οποίος με τη χρησιμοποίηση ενός νέου προγράμματος στόχευε στο να καταστεί πιο ανταγωνιστικός, αλλά αυτό δε δημιουργούταν εγκαίρως. Ακόμα και οι uml μεθοδολογίες, αν και προσδιορίζουν με ακρίβεια τον τρόπο με τον οποίο θα λειτούργει το εκάστοτε πρόγραμμα, δεν είναι ικανές να λύσουν το άλυτο φαινομενικά παραπάνω πρόβλημα.

Το 1986, [Brooks] είχε εκτιμηθεί ότι καμιά μεθοδολογία δε θα καταφέρει να εκτιμήσει με σχετική ακρίβεια το κόστος αλλά και το χρόνο υλοποίησης των

έργων μέχρι το 1996, όπως κι έγινε. Η αρχή όμως, είχε ξεκινήσει το 1974 από τον Edmonds ο οποίος παρουσίασε τη δική του μεθοδολογία για παραγωγή κώδικα. Το 1986 έκανε την εμφάνιση του το scrum, ενώ το 1996 παρουσιάστηκε το λεγόμενο extreme programming(xp). Οι προηγούμενες δύο μεθοδολογίες δεν είχαν την εποχή εκείνη μεγάλη απήχηση, λόγω των ανατρεπτικών τους ιδεών για την οργάνωση και την παραγωγή που αφορά την εγγραφή κώδικα, αλλά ενσωματώθηκαν αργότερα στις μεθοδολογίες του ευέλικτου προγραμματισμού οπού και άλλαξε δραστικά την αποδοχή τους. Ουσιαστικά, η δημιουργία του ευέλικτου προγραμματισμού ξεκίνησε από τα μέσα της δεκαετίας του ‘90, όταν οι αντιδράσεις για τις «βαριές» μεθόδους ολοένα και αυξάνονταν. Αυτές οι μέθοδοι, όπως το μοντέλο του καταρράκτη, θεωρούσαν ότι οι αναλυτές έχουν κατανοήσει τις ανάγκες του πελάτη καθώς κι ότι ο πελάτης δε θα αλλάξει τις προδιαγραφές του προγράμματος πριν αυτό ολοκληρωθεί, κάτι που δε συνέβαινε ποτέ. Το αποτέλεσμα, όπως είναι εμφανές από τα προηγούμενα ποσοστά εγκατάλειψης έργων και μεγάλων εξόδων, ήταν η δημιουργία του agile manifesto το Φεβρουάριο του 2001, το οποίο υπογράφθηκε από τα δεκαεφτά άτομα, που το δημιούργησαν. Σύμφωνα με αυτό:

«Σας παρουσιάζουμε τους καλύτερους τρόπους για τη δημιουργία προγραμμάτων μέσω χρησιμοποίησής τους ή βοηθώντας άλλους να τις χρησιμοποιήσουν. Μέσα από αυτήν την εργασία καταλήξαμε στις παρακάτω αξίες:

- Άτομα και αλληλεπιδράσεις έναντι διεργασιών και εργαλείων
- Λειτουργικό πρόγραμμα έναντι περιεκτικών εγγράφων
- Συνεργασία πελάτη έναντι διακανονισμού συμβολαίου
- Απάντηση στην αλλαγή έναντι ακολουθίας του σχεδίου

Αυτά είναι, αν και υπάρχει αξία στα αντικείμενα που βρίσκονται δεξιά, εμείς δίνουμε περισσότερη αξία σε αυτά που βρίσκονται αριστερά.

Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas.»

Με τη διακήρυξη του agile manifesto, δημιουργείται και ο ευέλικτος προγραμματισμός, ο οποίος βασίζεται στις παραπάνω αρχές και βάση αυτών έχουμε τα κοινά χαρακτηριστικά που τα διακρίνουμε σε κάθε ευέλικτη προγραμματιστική μεθοδολογία. Στις παραπάνω αξίες ‘κρύβονται’ αυτά τα κοινά χαρακτηριστικά. Ειδικότερα:

- Ευχαρίστηση του πελάτη μέσω συνεχούς και γρήγορης παρουσίασης-παράδοσης χρήσιμου κώδικα.
- Αυτοοργανούμενες ομάδες.
- Συνεχής τροποποίηση του κώδικα, ώστε να επιτευχθεί η απαιτούμενη ποιότητα.
- Η μικρή χρονικά παράδοση χρήσιμου κώδικα (συνήθως εβδομαδιαία).
- Με παράδοση χρήσιμου κώδικα μπορεί να παρουσιασθεί η πρόοδος της κατασκευής του προγράμματος.
- Απλή οργάνωση.
- Συνεχής και σε καθημερινή βάση επικοινωνία του πελάτη με την ομάδα δημιουργίας του προγράμματος.
- Η προσωπική επικοινωνία(πρόσωπο με πρόσωπο) των εμπλεκομένων ατόμων.
- Η συνεχής αλλαγή των προδιαγραφών είναι ευπρόσδεκτη σε όλη τη διαδικασία παραγωγής του έργου.
- Η εύρεση ευκαιριών και η ενσωμάτωσή τους.
- Εύκολη ενσωμάτωση των αλλαγών.

- Η δημιουργία του έργου από παρακινούμενα άτομα τα οποία χρίζουν εμπιστοσύνης.

Από τα παραπάνω μπορούμε να εξαγάγουμε τις καινοτομίες, τις οποίες χρησιμοποιούν οι ευέλικτες προγραμματιστικές μεθοδολογίες, καθιστώντας τες πρωτοποριακές και αποτελεσματικές. Αυτή τη στιγμή θα τονίσουμε τις πιο βασικές μιας και θα αναλυθούν όλες σε άλλο κεφάλαιο.

Μια από τις καινοτομίες είναι η παρουσία του πελάτη σε όλη τη διαδικασία της παραγωγής του προγράμματος. Η παρουσία αυτή είναι κάθε άλλο παρά παθητική, αφού ο πελάτης συμμετέχει στη δημιουργία του προγράμματος αναλαμβάνοντας πολλούς και διαφορετικούς ρόλους. Οι χρήστες του συστήματος συμμετέχουν στις καθημερινές συναντήσεις και ορίζουν την αξία των πτυχών του προγράμματος, καθορίζοντας με τον τρόπο αυτό τη σειρά, με την οποία θα υλοποιηθούν οι διάφορες λειτουργίες. Ακόμη, όχι μόνο απαντούν στις ερωτήσεις των προγραμματιστών σχετικά με τις ορολογίες που χρησιμοποιούν κατά την εργασία τους, αλλά εκφράζουν τη γνώμη τους για το παραγόμενο αποτέλεσμα, καθορίζοντας έτσι την ανάγκη για τροποποίησή του η όχι. Άλλος σημαντικός ρόλος είναι του ελεγκτή(tester), τον οποίο αναλαμβάνουν συχνά, δίνοντας στο πρόγραμμα τα κατάλληλα δεδομένα και περιμένοντας το κατάλληλο αποτέλεσμα. Όπως γίνεται αντιληπτό, ο ευέλικτος προγραμματισμός προϋποθέτει την παρουσία του πελάτη-χρηστών του νέου συστήματος, αλλιώς κινδυνεύει να χάσει μέρος της ευελιξίας του.

Μια δεύτερη καινοτομία είναι η ταχύτητα με την οποία παραδίδεται ο κώδικας στον πελάτη. Συνήθως, αναλόγως την ευέλικτη μεθοδολογία που θα διαλέξουμε, κάθε μήνα έχουμε την παράδοση λειτουργικού κώδικα στον πελάτη. Ο κώδικας αυτός είναι άμεσο αποτέλεσμα και των επιλογών του πελάτη, αφού αυτός καθορίζει όχι μόνο το τι θεωρεί σημαντικό για υλοποίηση, αλλά και τη λειτουργικότητα του. Με τον τρόπο αυτό, ο πελάτης αποκτά μια διαφορετική σχέση με τους δημιουργούς του κώδικα από ό,τι με τις παλαιότερες μεθοδολογίες, για τρεις κυρίως λόγους. Ο πρώτος έχει να κάνει με την άμεση προβολή του παραγόμενου αποτελέσματος. Στον δεύτερο, έχουμε τη μέτρηση της προόδου, που γίνεται εύκολα αντιληπτή από το τι έχει

υλοποιηθεί μέχρι κάποια χρονική στιγμή. Ο τρίτος, όπως προείπαμε, αφορά την επέμβαση-συμμετοχή του πελάτη ή εκπροσώπων του, κατά την παραγωγική διαδικασία. Καλό είναι να τονίσουμε, ότι αν και γίνεται παράδοση κώδικα κάθε εβδομάδα, δεν είναι σωστό να θεωρούμε ότι το έργο θα τελειώσει πιο γρήγορα από ό,τι θα τελείωνε εάν δεν υιοθετούσαμε κάποια ευέλικτη μεθοδολογία.

Η τελευταία καινοτομία, που θα αναφερθεί στο κεφάλαιο αυτό, αφορά την τροποποίηση των αναγκών του πελάτη που δεν αποτελούν τροχοπέδη κατά τη διαδικασία της δημιουργίας του προγράμματος. Ο πελάτης συχνά δε γνωρίζει, για αρκετό χρονικό διάστημα, τις πραγματικές του ανάγκες ή παρουσιάζονται καινούριες, με αποτέλεσμα να επιδιώκει είτε την αλλαγή είτε την εισαγωγή νέων λειτουργιών στο παραγόμενο πρόγραμμα. Σε παλιότερες μεθοδολογίες δε θα μπορούσε να επιτύχει όλες αυτές τι αλλαγές. Αντιθέτως, στον ευέλικτο προγραμματισμό, όπως αναφέρει και το όνομά του άλλωστε, το προηγούμενο πρόβλημα δεν υφίσταται, λόγω του τρόπου με τον οποίο γράφεται ο κώδικας.

Όπως σε κάθε μεθοδολογία, έτσι και στον ευέλικτο προγραμματισμό, έχουμε διαφωνίες σχετικά με το κατά πόσο είναι αποτελεσματικός. Η επιχειρηματολογία της αρνητικής αυτής κριτικής εστιάζεται στα παρακάτω:

- Λειτουργεί μόνο όταν τα στελέχη είναι έμπειρα.
- Υπολείπεται σε δομή και σε απαραίτητα έγγραφα (documentation).
- Χρειάζεται χρόνο και αλλαγή νοοτροπίας για να χρησιμοποιηθεί σωστά.
- Η μη λειτουργική ποιότητα είναι δύσκολο να τοποθετηθεί σαν ιστορία χρήστη(user story).
- Αυξάνει τον κίνδυνο του μη σωστού ελέγχου των αλλαγών λόγω έλλειψης επαρκούς καθορισμού των απαιτήσεων.
- Προωθεί το μη επαρκή σχεδιασμό.

- Είναι αδύνατο να υπάρξει ρεαλιστική εκτίμηση του χρόνου, κατά τον οποίο το έργο θα έχει ολοκληρωθεί, αφού στην αρχή του έργου δε γνωρίζει κανένας τις απαιτήσεις του έργου.

- Μπορεί να είναι αναποτελεσματική τεχνική, λόγω του ότι ο κώδικας μπορεί να τροποποιείται-ξαναεγγράφεται συνεχώς, καθώς αλλάζουν οι απαιτήσεις του προγράμματος.

Είναι αλήθεια, ότι ο ευέλικτος προγραμματισμός για να είναι επιτυχής, χρειάζεται αρκετή προσπάθεια και μεγάλη εμπειρία, τουλάχιστον από μερικά άτομα που την εφαρμόζουν. Ωστόσο, ο σκοπός της εργασίας αυτής είναι να επαληθεύσει ή να διαψεύσει τις παραπάνω κριτικές, κάνοντας γνωστές και εφαρμόζοντας τις τεχνικές του ευέλικτου προγραμματισμού.

3 Ακραίος Προγραμματισμός(eXtreme Programming)

Ο ευέλικτος προγραμματισμός, τείνει να αντικαταστήσει κάθε παλαιότερη μεθοδολογία ανάπτυξης λογισμικού, λόγω των πλεονεκτημάτων της, με αποτέλεσμα να καθίσταται όλο και πιο αποδεκτός. Αν και δεν έχουν γίνει έρευνες για την ευέλικτη προγραμματιστική μεθοδολογία, η οποία έχει χρησιμοποιηθεί περισσότερο, διαφαίνεται ότι ο ακραίος προγραμματισμός(eXtreme Programming) κατακτά όλο και περισσότερες ομάδες ανάπτυξης λογισμικού. Αυτό οφείλεται στις τεχνικές που ενσωματώνει, οι οποίες διαφοροποιούνται και μάλιστα σε μεγάλο βαθμό, από κάθε άλλη προγραμματιστική μεθοδολογία, κάτι που την καθιστά ξεχωριστή. Για το λόγο αυτό ονομάστηκε ακραία. Καλό είναι να τονίσουμε ότι οι περισσότερες από τις τεχνικές της εφαρμόζονται από πολλές ευέλικτες μεθοδολογίες, με διαφορετικές όμως ονομασίες, όπως για παράδειγμα η επανάληψη(iteration) στο scrum αναφέρεται ως σύντομος δρόμος ταχύτητας(sprint).

Όπως έχουμε αναφέρει, όλες οι προγραμματιστικές μεθοδολογίες, ευέλικτες και μη, αναφέρουν τον όρο «κύκλος ζωής» του προγράμματος. Στον κύκλο ζωής έχουμε την ανάλυση των απαιτήσεων, το σχεδιασμό, την υλοποίηση, τον έλεγχο και τέλος την παράδοση και τη συντήρηση. Για όλες τις μη ευέλικτες μεθοδολογίες, τα παραπάνω στάδια θεωρούνται διακριτά, σε αντίθεση με τις ευέλικτες οι οποίες έχουν την ικανότητα να ενεργούν ταυτόχρονα σε όλα τα στάδια του κύκλου ζωής, διότι δίνουν έμφαση στην καθημερινή και συνεχή επικοινωνία. Ο ακραίος προγραμματισμός δεν αποτελεί εξαίρεση και για το λόγο αυτό υπόσχεται μειωμένες προϋποθέσεις σε σχεδιασμό και έλεγχο, όσο παράδοξο κι αν ακούγεται. Η υπόσχεση επιτυγχάνεται λόγω των τεχνικών της, οι οποίες σε καθημερινή βάση όχι μόνο ελέγχουν τον παραγόμενο κώδικα, αλλά εξαναγκάζουν το συνεχή σχεδιασμό. Αξιοσημείωτο είναι το ότι με τη χρήση όλων των σταδίων του κύκλου ζωής, η εβδομαδιαία παράδοση κώδικα στον πελάτη είναι εφικτή, και σαν αποτέλεσμα έχουμε όχι μόνο την ικανοποίηση του πελάτη, αλλά και τη γρήγορη ανατροφοδότηση, που αφορά το παραγόμενο αποτέλεσμα. Έτσι επιτυγχάνεται η τροποποίηση και η σωστή κατασκευή των λειτουργιών που επιθυμούν οι τελικοί χρήστες του προγράμματος. Αυτό όμως, είναι μόνο η αρχή. Ίσως το σημαντικότερο πλεονέκτημα να βρίσκεται στην ευκολία που διέπει τον ακραίο προγραμματισμό στο να αποδέχεται τις αλλαγές που απαιτεί ο πελάτης, σε οποιαδήποτε χρονική στιγμή.

Αν και δεν υπάρχουν κανόνες για τη χρήση αυτής της προγραμματιστικής μεθοδολογίας, η εμπειρία είναι το καταλληλότερο εφόδιο για τη συνεχή βελτίωση των διαδικασιών του ακραίου προγραμματισμού ώστε να επιτύχουμε τους στόχους μας. Τι γίνεται όμως, όταν δεν έχουμε την απαραίτητη εμπειρία; Η απάντηση βρίσκεται στο συγκεκριμένο κεφάλαιο, αλλά η αρχή γίνεται με την προσαρμογή των σταδίων του κύκλου ζωής στις τεχνικές των ευέλικτων προγραμματιστικών μεθόδων. Έτσι, με βάση το agile manifesto έχουμε:

- Το σχεδιασμό(planning).
- Τη σκέψη(thinking).

- Την Ανάπτυξη-Υλοποίηση(Developing).
- Τη Συνεργασία(Collaborating).
- Την Κυκλοφορία(releasing).

Στα στάδια αυτά καταχωρούνται όλες οι τεχνικές του ακραίου προγραμματισμού τις οποίες θα αναλύσουμε αμέσως.

3.1 Προαπαιτήσεις του ακραίου προγραμματισμού

Ο ακραίος προγραμματισμός χρησιμοποιεί εκλεπτυσμένες τεχνικές οι οποίες μας εξασφαλίζουν την ευελιξία. Όμως, οι τεχνικές αυτές για να χρησιμοποιηθούν σωστά, παρουσιάζουν μερικές απαιτήσεις χωρίς τις οποίες δε θα οδηγηθούμε στο επιθυμητό αποτέλεσμα.

Οι έξι προϋποθέσεις είναι οι εξής:

- Υποστήριξη Προϊσταμένων.
- Αποδοχή ευέλικτου προγραμματισμού από την ομάδα.
- Κοντινή τοποθέτηση των μελών της ομάδας.
- Παρουσία των πελατών-χρηστών κατά την ανάπτυξη του λογισμικού.
- Δημιουργία ομάδων με τον προτεινόμενο αριθμό ατόμων.
- Χρήση όλων των τεχνικών του ακραίου προγραμματισμού.

Εκτός όμως από τις προαπαιτήσεις, θα τονίσουμε εν συντομία μερικές καταστάσεις, που αν τις επιδιώξουμε θα αντιμετωπισθούν αυτομάτως πολλά προβλήματα. Μερικές από αυτές είναι:

- Η χρήση του ακραίου προγραμματισμού σε καινούριο έργο.
- Η πρόσληψη έμπειρων στελεχών και ειδικότερα των διαχειριστών και των προπονητών.
- Η πρόσληψη προγραμματιστών με σχεδιαστικές ικανότητες.
- Η πρόσληψη ατόμων με πλήρη απασχόληση.
- Η δημιουργία θετικού κλίματος στην ομάδα.
- Η αποκλειστική χρήση των ατόμων μόνο για ένα έργο.

3.2 Η επανάληψη του ακραίου προγραμματισμού

Πριν αναλύσουμε τις τεχνικές του ακραίου προγραμματισμού, είναι προτιμότερο να καταλάβουμε, εν συντομία, τη σειρά με την οποία θα χρησιμοποιήσουμε τις τεχνικές του. Όπως έχουμε αναφέρει, όλα τα στάδια του κύκλου ζωής του λογισμικού εφαρμόζονται ταυτόχρονα και μάλιστα σε κάθε επανάληψη. Στο συγκεκριμένο σημείο, θα εστιάσουμε στην κατανόηση του πώς λειτουργεί ο ακραίος προγραμματισμός. Καταρχάς, επανάληψη(iteration) είναι το χρονικό διάστημα, μιας εβδομάδας συνήθως, κατά το οποίο θα πρέπει να παραδοθούν στον πελάτη οι λειτουργίες του προγράμματος, που έχουν καθοριστεί κατά την αρχή του χρονικού διαστήματος. Με άλλα λόγια, στις επαναλήψεις παρατηρούνται όλα τα στάδια του κύκλου ζωής και στο τέλος αυτής παραδίδεται ο κώδικας, που έχει καθοριστεί στο στάδιο του σχεδιασμού.

Ας υποθέσουμε λοιπόν, ότι μας έχει ανατεθεί η ανάπτυξη ενός νέου έργου κι ότι έχουμε προσλάβει το κατάλληλο προσωπικό. Ο πελάτης, μας

προσφέρει τον απαραίτητο αριθμό χρηστών που θα χρειασθούμε και γενικά τηρούμε όλες τις προϋποθέσεις, για την εφαρμογή του ακραίου προγραμματισμού. Πρωτίστως, θα πρέπει να χωρίσουμε το προσωπικό μας σε ομάδες καθηκόντων και ύστερα να δημιουργήσουμε ομάδες, αναλόγως πάντα με τον αριθμό του προσωπικού.

Έτσι, θα πρέπει να γίνει ο πρώτος σχεδιασμός επανάληψης, κατά τον οποίο αφιερώνεται περισσότερος χρόνος από ό,τι στους επόμενους σχεδιασμούς, λόγω του ότι θα πρέπει οι πελάτες να αναγνωρίσουν τις λειτουργίες που επιθυμούν από το προς κατασκευή πρόγραμμα. Αξίζει να τονίσουμε, ότι δεν είναι απαραίτητο να γνωρίζουμε όλες τις λειτουργίες που επιθυμούμε να υλοποιηθούν, αφού ο ευέλικτος προγραμματισμός υποστηρίζει την αυξητική ανάπτυξη. Συνεπώς, οι χρήστες σε συνεργασία με το διαχειριστή του έργου αναγνωρίζουν τις επιθυμητές λειτουργίες και τις βαθμολογούν, με βάση τη σημαντικότητα, δηλαδή τη σειρά με την οποία θα υλοποιηθούν. Ύστερα, ενημερώνουν το διαχειριστή προϊόντος, ο οποίος τις χωρίζει σε ιστορίες πελάτη και σε συνεργασία με τους προγραμματιστές εκτιμούν το χρόνο που αυτές θα χρειασθούν μέχρι να είναι έτοιμες. Στο σημείο αυτό, ο διαχειριστής προϊόντος με βάση τη σπουδαιότητα των λειτουργιών, πρέπει να αποφασίσει για το ποιες λειτουργίες θα υλοποιηθούν στην επανάληψη. Όπως είναι φυσικό, δεν επιλέγει τυχαία τον αριθμό των λειτουργιών, αλλά με βάση έναν αριθμό προόδου του έργου τη λεγόμενη ταχύτητα ανάπτυξης(velocity). Η ταχύτητα ανάπτυξης είναι ένας αριθμός, με βάση τον οποίο ενημερώνεται η ομάδα για το φόρτο εργασίας, που μπορεί να αναλάβει σε κάθε επανάληψη. Έτσι, κατά την πρώτη επανάληψη δεν υφίσταται αυτός ο αριθμός οπότε τον εκτιμούμε χωρίς όμως να επιβαρύνεται η πρόοδος του έργου. Επίσης, η επιλογή του αριθμού των λειτουργιών προς υλοποίηση στη συγκεκριμένη επανάληψη, γίνεται και με τη σύμφωνη γνώμη των προγραμματιστών, που είναι τα καταλληλότερα άτομα για την εκτίμηση του χρόνου δημιουργίας της κάθε λειτουργίας. Αφού έχουν αποφασισθεί όλες οι λειτουργίες κι έχουν κατατμηθεί σε ιστορίες πελάτη, η σκυτάλη περνά στους προπονητές της κάθε ομάδας που επιλέγουν τις ιστορίες τις οποίες θα κατασκευάσουν.

Στο σημείο αυτό, ο σχεδιασμός της επανάληψης σταματά και ουσιαστικά ξεκινά η επανάληψη, οπότε το κάθε άτομο αναλαμβάνει τα καθήκοντα που του έχουν ανατεθεί. Μερικοί από τους χρήστες με τους ελεγκτές, αναλαμβάνουν την κατασκευή και τον πλήρη έλεγχο του κώδικα, οι προγραμματιστές οργανώνονται με βάση τον προγραμματισμό σε ζεύγη και οι ειδικοί τομέα με τους χρήστες προετοιμάζονται ώστε να είναι αποτελεσματικοί στο να απαντούν σε κάθε ερώτηση που αφορά τον τομέα. Ο διαχειριστής του προϊόντος, καθημερινά ελέγχει την πορεία του έργου με βάση τις εκτιμήσεις των προγραμματιστών, ώστε να αναγνωρίσει την ταχύτητα ανάπτυξης (velocity) και να κάνει διορθωτικές κινήσεις όποτε είναι αναγκαίο.

Σε καθημερινή βάση, η κάθε ομάδα πραγματοποιεί συνάντηση, τη λεγόμενη όρθια συνάντηση, ώστε όχι μόνο να αναγνωρισθεί η πορεία του έργου αλλά και να γίνουν γνωστά τα όποια προβλήματα παρουσιάστηκαν. Ακόμη, συνήθως ανά ώρα πραγματοποιείται η ολική ένωση του κώδικα ώστε να ελεγχθεί η λειτουργία του.

Στο τέλος κάθε επανάληψης, παραδίδεται ο ελεγμένος και έτοιμος κώδικας στον πελάτη, του οποίου η αξιολόγηση λαμβάνεται σοβαρά υπόψη. Αν και αυτή η διαδικασία εφαρμόζεται στον ακραίο προγραμματισμό για την ανάπτυξη του έργου, αξίζει να σημειωθεί ότι δεν έχουμε αναφέρει αρκετές τεχνικές, όπως οι κανόνες εγγραφής κώδικα. Για τη σωστή εκμετάλλευση των δυνατοτήτων του ακραίου προγραμματισμού θα χρειασθεί αρκετός καιρός, αλλά το αποτέλεσμα θα ξεπεράσει κάθε προσδοκία.

3.3 Η Σκέψη(Thinking)

Η σκέψη αφορά το στάδιο, κατά το οποίο αναγνωρίζουμε το τι κάνουμε, το γιατί το κάνουμε και τέλος το εάν είναι ευεργετική η χρήση του. Άρα, κατά το στάδιο αυτό, ελέγχουμε τις τεχνικές που χρησιμοποιούμε και προσπαθούμε να τις τροποποιήσουμε, έτσι ώστε να τις εκμεταλλευτούμε κατά το μέγιστο δυνατό. Σύμφωνα με τα παραπάνω, ο ακραίος προγραμματισμός

απαιτεί συχνή προσοχή σε ότι αφορά τις τεχνικές του, ώστε να τις βελτιώσουμε. Στη σκέψη παρατηρούμε ότι ο ακραίος προγραμματισμός απαιτεί τις πέντε παρακάτω τεχνικές.

- Ο προγραμματισμός σε ζεύγη, όπου το ένα εκ των δύο ατόμων αναλαμβάνει την οργάνωση του κώδικα, ώστε να ενσωματωθεί ομαλά με το υπόλοιπο πρόγραμμα.
- Η παρακινητική εργασία, η οποία ασχολείται με το να ωθεί τα άτομα στο να δώσουν τον καλύτερό τους εαυτό.
- Ο πληροφοριακός χώρος εργασίας, παρέχει σε όλα τα άτομα την εύκολη πρόσβαση σε πληροφορίες. Η εύκολη πρόσβαση αφορά κυρίως την οπτική.
- Η ανάλυση ρίζας προβλήματος, με την οποία βρίσκουμε τα αίτια του προβλήματος και προσπαθούμε να τα εξαλείψουμε.
- Η αναθεώρηση, που αναλύει τις διαδικασίες ανάπτυξης λογισμικού και τις αναβαθμίζει.

Στο σημείο αυτό θα πρέπει να τονίσουμε ότι οι δύο τελευταίες τεχνικές απαιτούν δεκαπέντε λεπτά καθημερινώς, ώστε να τις εκμεταλλευτούμε σωστά.

3.3.1 Ο Προγραμματισμός σε Ζεύγη (Pair programming)

Ο προγραμματισμός σε ζεύγη είναι μία από τις καινοτόμες τεχνικές του ακραίου προγραμματισμού, και χάρη στην αποτελεσματικότητά της εφαρμόζεται συχνά κι από ομάδες, που δεν αναπτύσσουν λογισμικό με βάση τον ακραίο προγραμματισμό. Αντίθετα, υπάρχουν αρκετοί που τη θεωρούν σαν αναποτελεσματική τεχνική, όχι μόνο εξ αιτίας του κόστους της, αλλά και επειδή δημιουργεί το συναίσθημα δυσαρέσκειας στο προγραμματιστή, όταν υπάρχει κοντά του άτομο που ελέγχει τον κώδικα.

Οι δημιουργοί του ακραίου προγραμματισμού υποστηρίζουν ότι η συνεργασία δύο ατόμων κατά την εγγραφή του κώδικα αποσκοπεί στο να αναπτύξει προγράμματα με λιγότερες ελαττωματικές καταστάσεις, σε μικρότερο χρόνο και με την απαιτούμενη ποιότητα. Όταν δημιουργούμε τα ζευγάρια, το ένα άτομο για κάποιο χρονικό διάστημα αναλαμβάνει τη συγγραφή του κώδικα. Το άτομο αυτό ονομάζεται οδηγός(driver). Το δεύτερο άτομο είναι ο λεγόμενος πλοηγός(navigator), που ο ρόλος του αφορά την οργάνωση του κώδικα, ώστε να ενσωματωθεί χωρίς προβλήματα στο παραγόμενο πρόγραμμα. Εκτός όμως από την εργασία αυτή, όχι μόνο ελέγχει τον εγγραφόμενο κώδικα ενημερώνοντας με διακριτικό τρόπο τον οδηγό για τα σφάλματά του, αλλά και οργανώνει τη σειρά με την οποία θα υλοποιηθούν οι ζητούμενες λειτουργίες του κώδικα. Με τον τρόπο αυτό, ο οδηγός απασχολείται μόνο με τη σωστή συγγραφή του κώδικα επιδιώκοντας την απαραίτητη ποιότητα, οπότε η ανάπτυξη του λογισμικού επιταχύνεται αισθητά. Μπορεί να είναι αρκετά ενοχλητικό το να υπάρχει άτομο που παρακολουθεί τη συγγραφή του κώδικα, αλλά το όφελος είναι η παραγωγικότητα, δηλαδή η προσπάθεια για σωστό και ποιοτικό κώδικα που πηγάζει από την παρουσία του δεύτερου ατόμου.

Συχνά παρατηρούμε τη δημιουργία ζευγαριών που αποτελείται από ένα άπειρο άτομο και από ένα έμπειρο. Αν και είναι θεμιτή η κίνηση αυτή αφού το άπειρο άτομο αποκτά απαραίτητες γνώσεις, είναι σαφές ότι το να παρέχουμε τη δυνατότητα σε έναν νέο προγραμματιστή να ειδικευτεί σε κάποια βιβλιοθήκη, επιφέρει όχι μόνο αύξηση παραγωγικότητας, αλλά καθιστά τον προγραμματιστή αυτόν απαραίτητο. Με τον τρόπο αυτό, του δημιουργείται αίσθημα ευφορίας, που τον παρακινεί στο να συνεχίσει τη σκληρή εργασία.

Στον αντίποδα, υπάρχουν αρκετοί που υποστηρίζουν ότι τα οφέλη της συγκεκριμένης τεχνικής δεν υφίστανται και μάλιστα τη θεωρούν αντιπαραγωγική. Τα επιχειρήματά τους είναι τα εξής:

- Χρειάζεται διπλάσιους προγραμματιστές, άρα και μεγαλύτερο κόστος.

- Είναι δύσκολη η συγκέντρωση των προγραμματιστών όταν υπάρχει και δεύτερο άτομο που ελέγχει τον κώδικα.
- Μειώνεται η παραγωγικότητα στα ζευγάρια όπου υπάρχει μεγάλη διαφορά εμπειρίας.
- Οι προγραμματιστές δύσκολα διαπραγματεύονται τον τρόπο γραφής του κώδικα.
- Οι συχνές παρατηρήσεις σφαλμάτων μπορούν να διαταράξουν το φιλικό κλίμα.

Λόγω των εμποδίων που αναφέρθηκαν, παρουσιάζονται προτάσεις για ομαλότερη χρήση του προγραμματισμού σε ζεύγη:

- Συχνή εναλλαγή ρόλων.
- Συχνή επικοινωνία και συνεργασία.
- Αποφυγή δημιουργίας του ιδίου ζεύγους πάνω από μία φορά την ίδια μέρα.
- Δημιουργία φιλικού κλίματος.
- Αλλαγή ζευγαριών όταν είναι φανερή η ανάγκη για νέα προοπτική.
- Δημιουργία νέων ζευγαριών όταν υπάρχει ανάγκη για επίλυση προβλήματος από εξειδικευμένα άτομα.
- Η χρήση άνετου εξοπλισμού.
- Η δυνατότητα δημιουργίας ζευγαριών χωρίς την επέμβαση-επιβολή άλλων στελεχών.

3.3.2 Η Παρακινητική Εργασία (Energized work)

Η τεχνική αυτή, αφορά τους τρόπους με τους οποίους τα άτομα που έχουν αναλάβει την ανάπτυξη του έργου, όχι μόνο να δίνουν τον καλύτερό τους εαυτό για την κατάκτηση του στόχου, αλλά να είναι και πρόθυμοι να φέρουν εις πέρας αυτή τη δύσκολη εργασία. Την τεχνική αυτή την εφαρμόζουν συνήθως οι προπονητές, με τρείς βασικούς τρόπους, εκ των οποίων ο τελευταίος απορρέει από την οργάνωση και από τις τεχνικές του ακραίου προγραμματισμού.

- Η απαγόρευση των υπερωριών, ακόμη κι όταν το έργο δεν αναπτύσσεται με τον επιθυμητό ρυθμό.
- Η χρήση διαλλειμάτων, ειδικά όταν εμφανίζονται προβλήματα που φαινομενικά απαιτούν μη ευδιάκριτες λύσεις.
- Η δημιουργία φιλικού κλίματος και συναισθημάτων που προκύπτουν από την επίτευξη των στόχων.

3.3.3 Ο Πληροφοριακός Χώρος Εργασίας (Informative workspace)

Αρκετές είναι οι περιπτώσεις, στις οποίες παρατηρείται η έλλειψη των απαραίτητων πληροφοριών κατά την ανάπτυξη του λογισμικού, με αποτέλεσμα την καθυστέρηση της συγγραφής του κώδικα. Η αναζήτηση των πληροφοριών αυτών όχι μόνο απαιτεί συνήθως μεγάλο χρονικό διάστημα, αλλά και επηρεάζει τη συγκέντρωση του ατόμου, που επιζητά τις συγκεκριμένες πληροφορίες. Επιπροσθέτως, η παραπάνω αναζήτηση μπορεί να καταλήξει σε απασχόληση συναδέλφου για την παροχή των πληροφοριών με ακόμα μεγαλύτερες και δυσμενέστερες συνέπειες. Έτσι, ο ακραίος προγραμματισμός εισάγει την τεχνική του πληροφοριακού χώρου εργασίας, δηλαδή την εύκολη πρόσβαση σε απαραίτητες πληροφορίες. Η τεχνική αυτή επικεντρώνεται στη σωστή διαμόρφωση του χώρου και στη χρήση των κατάλληλων αντικειμένων για εύκολη οπτική πρόσβαση στις επιζητούμενες πληροφορίες.

Οι τρόποι ώστε να έχουμε εύκολη πρόσβαση στην πληροφορία που μας ενδιαφέρει είναι οι εξής:

- Η αγορά και η χρήση αρκετών πινάκων.
- Η σωστή χωροταξική τοποθέτηση των ομάδων και των πινάκων, ώστε να μεγιστοποιηθεί η οπτική επαφή με τους πίνακες.
- Η χρήση διαγραμμάτων για εύκολη και γρήγορη απόκτηση πληροφοριών.
- Η χρήση καρτελών για τις ιστορίες πελάτη και μάλιστα με διαφορετικά χρώματα, εάν είναι εφικτό.

3.3.4 Η Ανάλυση Ρίζας Προβλήματος(Root-Cause analysis)

Όπως αναφέρει και το όνομά της, η τεχνική αυτή αφορά την ανεύρεση του αιτίου, στο οποίο οφείλεται η δημιουργία μη ορθών λειτουργιών-καταστάσεων του προγράμματος (bugs). Είναι απολύτως φυσιολογικό, οι προγραμματιστές ασχέτως εμπειρίας, να κάνουν προγραμματιστικά λάθη. Σύμφωνα με την παρατήρηση του Norm Kerth:

«Οποιοσδήποτε θα κάνει το καλύτερο δυνατό με βάση τις ικανότητες, τις γνώσεις και τους πόρους που του παρέχονται»

Εάν αναλογισθούμε την παραπάνω πρόταση, τότε μπορούμε να αναγνωρίσουμε την ευθύνη των τεχνικών που χρησιμοποιούνται κατά την ανάπτυξη του λογισμικού. Για το λόγο αυτό, ο ευέλικτος προγραμματισμός προτείνει την ανάλυση ρίζας προβλήματος, που χρησιμοποιείται από τους προπονητές και το διαχειριστή του έργου, για την ανεύρεση και τροποποίηση των τεχνικών, στις οποίες οφείλεται η δημιουργία δυσλειτουργιών.

Ένας από τους προτεινόμενους τρόπους για την ανεύρεση των τεχνικών, στις οποίες οφείλονται οι προβληματικές καταστάσεις του προγράμματος, είναι η ερώτηση με χρήση του ‘γιατί’. Με τον τρόπο αυτό, αναρωτιόμαστε γιατί συμβαίνουν κάποιες καταστάσεις και μάλιστα με τη συνέχιση της χρήσης τέτοιων ερωτήσεων, μπορούμε να αναγνωρίσουμε την ρίζα του προβλήματος σε μικρό χρονικό διάστημα. Έτσι, αναγνωρίζουμε όχι μόνο την τεχνική που πρέπει να τροποποιηθεί, αλλά και την πτυχή της τεχνικής αυτής λαμβάνοντας παράλληλα τα κατάλληλα μέτρα, ώστε να εξαλειφθεί, κατά το δυνατόν, το πρόβλημα που προέκυψε.

3.3.5 Η Αναθεώρηση(Retrospectives)

Συχνά η ανάγκη τροποποίησης των εργασιακών συνηθειών για την επίτευξη ευνοϊκότερων αποτελεσμάτων, καθίσταται αναγκαία. Για το λόγο αυτό, ο ευέλικτος προγραμματισμός ενσωματώνει την τεχνική της αναθεώρησης. Σύμφωνα με την τεχνική αυτή, μετά από αρκετές συναντήσεις και διαδικασίες που αφορούν την εύρεση ιδεών και λύσεων για την τροποποίηση πτυχών των τεχνικών, επιτυγχάνουμε την ευνοϊκότερη δυνατή χρήση του ακραίου προγραμματισμού. Η πιο συχνή αναθεώρηση αφορά την επανάληψη και ονομάζεται αναθεώρηση επανάληψης. Έτσι λοιπόν, αφιερώνεται μία ώρα στην προγραμματιστική ομάδα, για την επιλογή ενός κοινού στόχου που ονομάζεται στόχος αναθεώρησης(retrospective objective).

3.4 Η Συνεργασία(collaboration)

Η συνεργασία είναι το στάδιο του κύκλου ζωής κατά το οποίο δίνεται έμφαση στην επικοινωνία, η οποία αποτελεί την ραχοκοκαλιά του ακραίου προγραμματισμού. Για να στηρίξουμε την παρατήρηση αυτή, απλά αναλογιστείτε ότι όσο περισσότερες πληροφορίες μας παρέχονται, τόσο

επιταχύνεται η ανάπτυξη του λογισμικού, αλλά και οι προγραμματιστές καθίστανται πιο αποτελεσματικοί. Για το λόγο αυτό, οι περισσότερες τεχνικές του ακραίου προγραμματισμού βασίζονται και παροτρύνουν την επικοινωνία, είτε είναι μεταξύ ομάδων είτε μεταξύ δύο ατόμων. Επειδή η επικοινωνία που μας παρέχει τις κατάλληλες πληροφορίες συχνά καθίσταται δύσκολη, υπάρχουν αρκετές τεχνικές που την ενισχύουν. Ο ακραίος προγραμματισμός ενσωματώνει τις παρακάτω επτά τεχνικές, ώστε να καταστεί η επικοινωνία αποτελεσματική.

- Η εμπιστοσύνη.
- Η στενή συνεργασία.
- Η ‘κοινή’ γλώσσα.
- Οι όρθιες συναντήσεις.
- Οι κανόνες κώδικα.
- Τα δείγματα επανάληψης.
- Οι αναφορές.

3.4.1 Η Εμπιστοσύνη(Trust)

Η δημιουργία του συναισθήματος της ομαδικότητας αποτελεί βασικό αντικείμενο της τεχνικής της εμπιστοσύνης. Με την απόκτηση της ομαδικότητας, παρατηρείται το φαινόμενο κατά το οποίο δίδεται προτεραιότητα στη συνεργασία, δηλαδή τα άτομα αναλαμβάνουν καθήκοντα για θέματα που πρέπει να επιλυθούν, αντιμετωπίζοντάς τα σαν προσωπική ευθύνη, και σε συνεργασία με άλλα πρόσωπα επιδιώκουν την ολοκλήρωσή του. Η παραπάνω πρόταση μας αποκαλύπτει το φαινόμενο της στενής συνεργασίας, που επιφέρει το επιθυμητό αποτέλεσμα, δηλαδή τη γρήγορη ανάπτυξη του λογισμικού χωρίς επιπτώσεις στην ποιότητά του.

Ένα αρκετά συχνό φαινόμενο είναι της αντιπαλότητας μεταξύ προγραμματιστών και πελατών, με αποτέλεσμα να δρουν σαν δύο ξεχωριστές ομάδες, με όσες καταστρεπτικές συνέπειες συνεπάγεται αυτό. Την αντιμετώπιση της κατάστασης αυτής αναλαμβάνει η τεχνική της εμπιστοσύνης, κατά την οποία οι προγραμματιστές και οι πελάτες θα πρέπει να εργάζονται στενά, ώστε να αναγνωρίσουν οι δύο αυτές ομάδες καθηκόντων τις ευαισθησίες της άλλης.

3.4.2 Η Στενή Συνεργασία(Sitting together)

Σε παλαιότερες προγραμματιστικές μεθόδους, η μείωση της επικοινωνίας θεωρούταν σα λύση για την αντιμετώπιση των καθυστερήσεων και των παρανοήσεων, που μπορούσε να δημιουργήσει η παρατεταμένη επικοινωνία. Για το λόγο αυτό, επιδιωκόταν η μείωση των ερωτήσεων με τη συγγραφή του κειμένου των απαιτήσεων του πελάτη από τους αναλυτές. Αν και είναι αρκετά καλή η προσέγγιση του προβλήματος, στην πράξη παρουσιάζονται αρκετά ψεγάδια και το καταλαβαίνουμε εάν αναλογισθούμε τον όγκο εργασίας των αναλυτών, που θα πρέπει να προνοήσουν απαντώντας σε κάθε πιθανή ερώτηση. Ο ακραίος προγραμματισμός, σε αντίθεση με τις παλαιότερες προγραμματιστικές μεθοδολογίες, προσεγγίζει το πρόβλημα αυτό με τελείως διαφορετικό τρόπο.

- Οι ομάδες που αλληλεπιδρούν συχνά θα πρέπει να εργάζονται σε κοντινή απόσταση.
- Η οργάνωση του χώρου να επιτρέπει τη δημιουργία του ιδιωτικού κλίματος, χωρίς όμως να εμποδίζεται η επικοινωνία και η οπτική πρόσβαση στους πίνακες που θα τοποθετηθούν.

Τα πλεονεκτήματα της προσέγγισης αυτής είναι:

- Η άμεση ανταπόκριση σε ερωτήσεις που προκύπτουν κατά την ανάπτυξη του λογισμικού.

- Η οπτική επίδειξη του προβλήματος.
- Μείωση χρόνου ολοκλήρωσης.

Τις παρατηρήσεις μας αυτές τις βασίζουμε στο πείραμα της Teasley. Σύμφωνα με αυτό, η στενή συνεργασία όχι μόνο διπλασίασε την παραγωγικότητα, αλλά μείωσε επίσης το χρόνο ολοκλήρωσης του έργου κατά το ένα τρίτο. Τα εκπληκτικά αυτά αποτελέσματα επιτυγχάνονται μέσα από τη συχνή επικοινωνία. Η Teasley μας αναφέρει ενδεικτικά:

«Παλαιότερες μελέτες μας, έδειξαν ότι λιγότερο από το 30% του χρόνου ενός προγραμματιστή αφιερώνεται σε παραδοσιακές εργασίες προγραμματισμού, καθώς και λιγότερο από 20% σε συγγραφή του κώδικα. Ο υπόλοιπος χρόνος αφιερώνεται σε συναντήσεις, επίλυση προβλημάτων, έλεγχος προϊόντων κτλ.»

Αξίζει να σημειωθεί, ότι οι συχνές ερωτήσεις καθυστερούν την ανάπτυξη του έργου, λόγω των συνεχών διακοπών, με αποτέλεσμα πολλές προγραμματιστικές μεθοδολογίες να αποθαρρύνουν τη συχνή αυτή επικοινωνία. Όμως, ο ακραίος προγραμματισμός αντιμετωπίζει τις καταστάσεις αυτές με την τεχνική του προγραμματισμού σε ζεύγη, αφού ο οδηγός συνεχίζει απερίσπαστος την εργασία του, ενώ ο πλοηγός επικοινωνεί με το άτομο που χρειάζεται βοήθεια.

3.4.3 Κοινή Γλώσσα(ubiquitous language)

Όπως οι πελάτες έχουν τους δικούς τους όρους για τους διάφορους τομείς, έτσι και τα πρόσωπα που έχουν αναλάβει τη δημιουργία του προγράμματος επικοινωνούν με όρους από την επιστήμη της πληροφορικής ώστε να σχεδιασθεί και να υλοποιηθεί σωστά το έργο. Αυτό λοιπόν, μπορεί να προκαλέσει προβλήματα στην επικοινωνία μεταξύ των πελατών και των προγραμματιστών με καταστροφικές συνέπειες. Έτσι, πρέπει να επιτευχθεί η δημιουργία μιας κοινής γλώσσας.

Η γλώσσα αυτή συνδυάζει τον κώδικα του προγράμματος με όρους του τομέα, ώστε η συγγραφή του κώδικα παρουσιάζεται σαν αρκετά δύσκολη διαδικασία, ενώ παράλληλα η μετατροπή του κώδικα σε ορολογίες κατανοητές από τους πελάτες να καθίσταται αρκετά εύκολη. Η καινούρια αυτή γλώσσα, πρέπει να τροποποιείται και να εξελίσσεται, ενσωματώνοντας νέες λέξεις και ορολογίες πάντα όμως με τη σύμφωνη γνώμη των ειδικών στον τομέα.

3.4.4 Όρθιες-καθημερινές Συναντήσεις(stand-up meetings)

Σε κάθε προγραμματιστική μεθοδολογία αναγνωρίζεται η ανάγκη για τη γνώση και ενημέρωση της προόδου του έργου. Έτσι, ο ακραίος προγραμματισμός χρησιμοποιεί τις όρθιες συναντήσεις, οι οποίες είναι καθημερινές και μάλιστα διαφορετικές για κάθε ομάδα. Στον τίτλο του κεφαλαίου προσθέσαμε και τη λέξη καθημερινές για να τονίσουμε το χρονικό διάστημα κατά το οποίο χρησιμοποιείται η τεχνική αυτή.

Ο ευέλικτος προγραμματισμός έχει οργανώσει τον τρόπο με τον οποίο θα διεξάγονται οι συναντήσεις αυτές, με αποτέλεσμα τη μείωση του χρόνου ενημέρωσης της ομάδας και μάλιστα σε τέτοιο βαθμό, ώστε τις περισσότερες φορές τα δεκαπέντε λεπτά να είναι αρκετά. Στη μείωση συνεισφέρει και το ότι κάθε ομάδα διεξάγει τη δική της συνάντηση σε διαφορετική χρονική στιγμή από τις άλλες.

Σε κάθε καθημερινή συνάντηση, το κάθε άτομο της ομάδας απαντά σε τρία ερωτήματα:

- Με τι ασχολήθηκα χθες;
- Με τι θα ασχοληθώ σήμερα;
- Τι προβλήματα συνάντησα που δυσχεραίνουν την εργασία μου;

Η κάθε ομάδα οργανώνει ξεχωριστή συνάντηση στην οποία παρευρίσκονται και οι δύο διαχειριστές, ώστε να αποκτήσουν συνολική εικόνα

για την κατάσταση του έργου. Για το λόγο αυτό ποτέ δυο συναντήσεις δε διεξάγονται ταυτόχρονα.

3.4.5 Οι Κανόνες Κώδικα(coding standards)

Είναι γνωστό ότι ο κάθε προγραμματιστής συγγράφει τον κώδικα με ξεχωριστή μορφοποίηση. Η κατάσταση αυτή επιφέρει αρκετά προβλήματα, όταν η συγγραφή του κώδικα δεν αναλαμβάνεται από ένα άτομο. Οπότε, ο ευέλικτος προγραμματισμός προτείνει τη δημιουργία κανόνων σε ότι αφορά τη μορφοποίηση του κώδικα.

Η δημιουργία των κανόνων αυτών ξεκινά από την πρώτη επανάληψη, ώστε να επιτευχθεί η κατάλληλη μορφοποίηση και οργάνωση κατά τη συγγραφή του κώδικα. Έτσι, διοργανώνεται συνάντηση των προγραμματιστών για τη διευθέτηση των ζητημάτων αυτών, κατά την οποία θα υπάρξουν αρκετές διαφωνίες για τη μορφοποίηση που θα υιοθετηθεί. Κατά τη συνάντηση αυτή θα πρέπει να αναφερθούν τα παρακάτω θέματα:

- Εργαλεία ανάπτυξης λογισμικού.
- Τεχνικές ανάπτυξης λογισμικού.
- Μορφοποίηση των αρχείων.
- Χειρισμός σφαλμάτων.
- Σχεδιασμός.
- Καταγραφές.
- Κανόνες κατασκευής

Ο βασικός όμως στόχος τους, είναι η κατασκευή των ελάχιστων μορφοποιήσεων που θα γίνουν αποδεκτές από όλους του προγραμματιστές, ώστε να εφαρμοσθούν. Με τον τρόπο αυτό, επιτυγχάνουμε τη συνοχή,

παραμερίζοντας εν μέρει την τελειότητα που θα απολαμβάναμε εάν μπορούσαμε να επιβάλλουμε μία πλήρη κοινά αποδεκτή μορφοποίηση. Στο σημείο αυτό αξίζει να τονίσουμε, ότι για τη συνάντηση αυτή ο προτεινόμενος χρόνος που θα πρέπει να αφιερώσουμε περιορίζεται σε μία ώρα.

3.4.6 Τα Δείγματα Επανάληψης(Iteration demo)

Μία από τις καινοτομίες του ευέλικτου προγραμματισμού είναι η ανά τακτά χρονικά διαστήματα παράδοση χρήσιμου κώδικα στον πελάτη. Η καινοτομία αυτή προσφέρει αρκετά πλεονεκτήματα και για το λόγο αυτό ενσωματώθηκε στον ευέλικτο προγραμματισμό, σαν τεχνική των δειγμάτων επανάληψης. Από το όνομα της τεχνικής αυτής, εύκολα αναγνωρίζουμε ότι η παράδοση του έτοιμου και χρήσιμου κώδικα επιτυγχάνεται μετά από κάθε επανάληψη.

Για να επιτύχουμε όλα αυτά τα καταπληκτικά αποτελέσματα, απαιτείται πειθαρχία και αυστηρή οργάνωση. Οι προγραμματιστές πρέπει να επιδιώκουν την ανάπτυξη κώδικα υψηλής ποιότητας, οι πελάτες να γνωρίζουν τις λειτουργίες που θα πρέπει να παραχθούν και οι ελεγκτές συνεχώς να ελέγχουν τον κώδικα. Όπως γίνεται κατανοητό, ο ευέλικτος προγραμματισμός χρησιμοποιεί τεχνικές που καθιστούν την ανάπτυξη του λογισμικού αρκετά απαιτητική διαδικασία, μειώνοντας έτσι την επικινδυνότητα εγκατάλειψης του έργου. Η τεχνική αυτή θεωρείται αρκετά σημαντική, αφού:

- Ενημερώνει τον πελάτη για την πορεία του έργου.
- Ο πελάτης παραλαμβάνει λειτουργικό κώδικα που μπορεί να χρησιμοποιήσει.
- Ενημερώνεται η ομάδα για τις προθέσεις του πελάτη.
- Καθορίζεται η πορεία-τροχιά του έργου.

- Βελτιώνει τις σχέσεις πελάτη με την ομάδα ανάπτυξης λογισμικού.
- Βελτιώνει την καταξίωση της ομάδας ανάπτυξης λογισμικού.

3.4.7 Οι Αναφορές(Reports)

Κατά τη χρήση του ακραίου προγραμματισμού, παρατηρούμε ότι η ροή πληροφοριών μεταξύ των μελών που συμμετάσχουν στην ανάπτυξη του λογισμικού είναι συνεχής. Οι καθημερινές συναντήσεις, καθώς και ο προγραμματισμός σε ζεύγη, επιφέρουν μία αρκετά καλή γνώση για την κατάσταση του έργου ώστε να μη χρειάζεται η δημιουργία αναφορών. Όμως, εκτός από την ενημερωμένη ομάδα ανάπτυξης λογισμικού, ο πελάτης επενδύει αρκετά στη σωστή λειτουργία του παραγόμενου λογισμικού και σαν αποτέλεσμα η γνώση για την πορεία της ανάπτυξης καθίσταται σημαντική. Για λόγους οργάνωσης της γνώσης αυτής, εμφανίζονται δύο κύριες κατηγορίες αναφορών.

Η πρώτη κατηγορία αφορά τις αναφορές προόδου(progress reports), δηλαδή την ενημέρωση των πελατών και των ανωτέρων στελεχών για την επανάληψη και για τις λειτουργίες που έχουν ενσωματωθεί στο αναπτυσσόμενο λογισμικό. Η δεύτερη μεγάλη κατηγορία αποτελείται από τις αναφορές διαχείρισης(management reports), που στοχεύουν στην ενημέρωση των προϊσταμένων για την γενική κατεύθυνση του έργου. Με τον τρόπο αυτό, αποφασίζονται οι στόχοι που πρέπει να επιτευχθούν, ώστε να στεφθεί με επιτυχία η δημιουργία του προγράμματος. Οι συγκεκριμένες αναφορές δεν είναι εύκολες στο να δημιουργηθούν, αλλά αντίθετα απαιτούν εξειδικευμένη ανάλυση στο σύνολο των τεχνικών του ακραίου προγραμματισμού.

3.5 Ο Σχεδιασμός(Planning)

Σε όλες τις προγραμματιστικές μεθοδολογίες αναφέρεται συχνά η ανάγκη του σχεδιασμού. Μάλιστα, πολλές μεθοδολογίες δίνουν τόση έμφαση, που χρησιμοποιούν εξειδικευμένες τεχνικές για το σκοπό αυτό, ώστε να επιτύχουν στην ανάπτυξη του λογισμικού. Όσο περισσότερος χρόνος αφιερώνεται για το σχεδιασμό, τόσο αυξάνονται οι πιθανότητες όχι μόνο για την ολοκλήρωση του έργου, αλλά και για την ορθή ανάπτυξή του. Επιπροσθέτως, παρατηρείται ότι όσο μεγαλύτερο το πρόγραμμα για την ανάπτυξή του, τόσο καθίσταται αναγκαιότερη η χρήση του σχεδιασμού και για το λόγο αυτό ο ακραίος προγραμματισμός ενσωματώνει αρκετές τεχνικές που συνεισφέρουν προς την κατεύθυνση αυτή. Συγκεκριμένα, ο ακραίος προγραμματισμός, λόγω του ότι είναι αρκετά δύσκολο το να προβλέψει κανείς όλες τις λειτουργίες του προγράμματος και τις καταστάσεις που μπορεί να προκύψουν, τροποποιεί τη συμπεριφορά αυτή θεωρώντας ότι είναι προτιμότερο να περιμένουμε τις κατάλληλες ευκαιρίες, τις οποίες θα εκμεταλλευτούμε. Αν και η συγκεκριμένη προσέγγιση φαίνεται παρακινδυνευμένη, ο ακραίος προγραμματισμός με την ενσωμάτωση των παρακάτω επτά τεχνικών μπορεί να επιτύχει το επιθυμητό αποτέλεσμα. Οι τεχνικές αυτές είναι:

- Το σχέδιο κυκλοφορίας, το οποίο μας παρουσιάζει το τι πρέπει να κάνουμε για να φτάσουμε την επιτυχία.
- Το παιχνίδι του σχεδιασμού, που συνδυάζει τις δεξιότητες των ατόμων της ομάδας.
- Η διαχείριση των κινδύνων.
- Ο σχεδιασμός επανάληψης.
- Η χαλάρωση, που επιτρέπει στην ομάδα να παραδίδουν ελεγμένο κώδικα σε κάθε επανάληψη.
- Οι ιστορίες πελάτη.

- Οι εκτιμήσεις, που επιτρέπουν στην ομάδα να υπολογίσουν το χρόνο για την ανάπτυξη του λογισμικού.

3.5.1 Το Σχέδιο Κυκλοφορίας(Release Planning)

Σε κάθε επανάληψη υλοποιούνται συγκεκριμένες λειτουργίες που θα ενσωματωθούν στο επιθυμητό λογισμικό. Οι λειτουργίες αυτές καθορίζονται στην αρχή κάθε επανάληψης, μετά από συμφωνία του διαχειριστή προϊόντος με τους χρήστες. Όμως, η συνεχής παράδοση έτοιμου κώδικα δε βασίζεται μόνο στην επιλογή των λειτουργιών προς υλοποίηση, αλλά προϋποθέτει εξειδικευμένες τεχνικές και σωστή οργάνωση. Για το λόγο αυτό, ο ευέλικτος προγραμματισμός χρησιμοποιεί την τεχνική του σχεδίου κυκλοφορίας, η οποία είναι καθοριστική για τη σωστή πορεία του έργου.

Γενικά υπάρχουν δύο είδη σχεδίων κυκλοφορίας. Το πρώτο είναι το καθοδηγούμενο από το εύρος(scope boxed), ενώ το δεύτερο καθοδηγούμενο από το χρόνο(time boxed). Στο πρώτο σχέδιο αναλύουμε με ακρίβεια τις λειτουργίες που θα υλοποιηθούν χωρίς όμως να γνωρίζουμε το χρόνο κατά τον οποίο θα είναι έτοιμες. Αντίθετα, στο σχέδιο καθοδηγούμενο από το χρόνο, με βάση τα περιθώρια χρόνου που έχουμε, καθορίζουμε τον αριθμό των λειτουργιών που θα υλοποιήσουμε. Επειδή είναι κοινά αποδεκτό ότι αυτού του είδους τα σχέδια επιφέρουν ευνοϊκότερα αποτελέσματα, λόγω της χρονικά σταθερής τους παράδοσης κώδικα, θα αναλύσουμε μόνο το συγκεκριμένο σχέδιο.

Για να χρησιμοποιήσουμε σωστά το σχέδιο καθοδηγούμενο από το χρόνο, θα πρέπει να αποφασίσουμε τις ημερομηνίες που θα παραδώσουμε χρήσιμο κώδικα. Η παράδοση κώδικα κάθε μήνα θεωρείται ικανοποιητική, σε αντίθεση με την παράδοση κάθε τρίμηνο. Αφού έχουμε αποφασίσει για το χρόνο, θα πρέπει να αναλογισθούμε τις λειτουργίες που θα ενσωματωθούν στο χρονικό αυτό διάστημα. Έτσι, ο διαχειριστής προϊόντος σε συνεργασία με τους χρήστες, κατηγοριοποιεί τις λειτουργίες με βάση τη σημαντικότητά τους

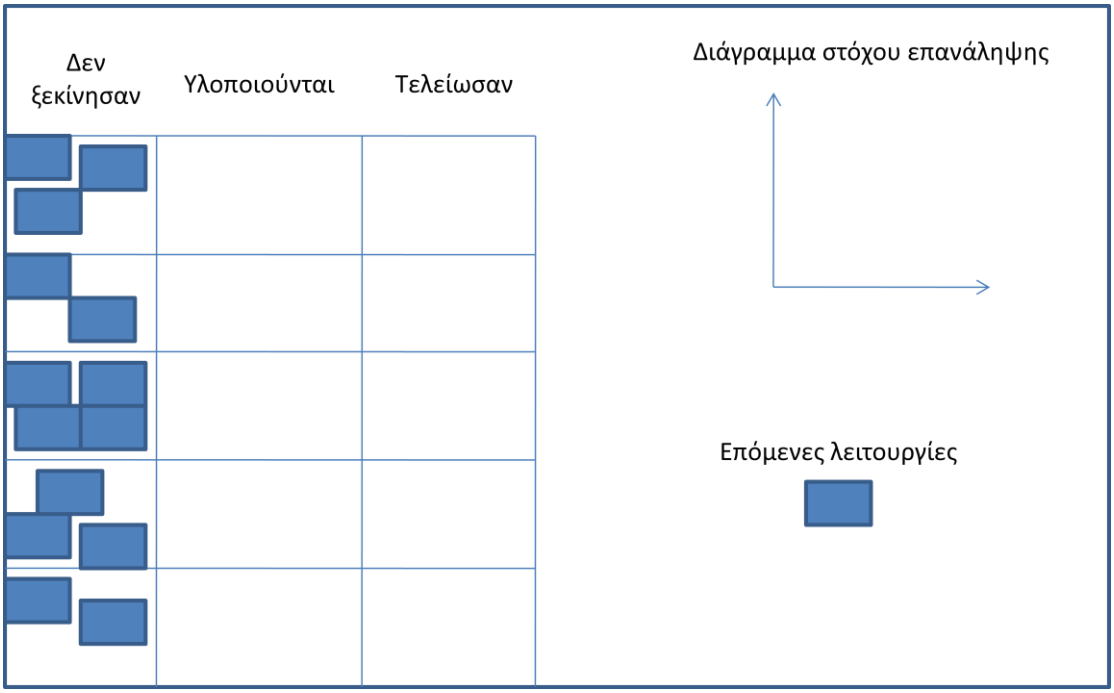
και τις χωρίζει σε ιστορίες πελάτη, αναφέροντάς τες στους προγραμματιστές, ώστε να εκτιμηθεί ο χρόνος που απαιτεί η κάθε μία από αυτές για την πλήρη δημιουργία της. Όταν εκτιμηθούν όλες, ο διαχειριστής προϊόντος, με βάση τον αριθμό της ταχύτητας ανάπτυξης, επιλέγει τις σημαντικότερες λειτουργίες που μπορούν να υλοποιηθούν σε κάθε επανάληψη και δημιουργεί κάρτες, οι οποίες αντιπροσωπεύουν τις ιστορίες πελάτη τοποθετώντας τες στον πίνακα επανάληψης. Το αποτέλεσμα λοιπόν της προηγούμενης διαδικασίας είναι το σχέδιο κυκλοφορίας, ενώ η ίδια η διαδικασία είναι η τεχνική του σχεδιασμού της επανάληψης.

Στο σημείο αυτό, καλό είναι να αναφέρουμε τον υπολογισμό της ταχύτητας ανάπτυξης που καθορίζει ουσιαστικά το ρυθμό ανάπτυξης του έργου. Το πρώτο βήμα αφορά την εύρεση των ημερών της επανάληψης, που εργάζεται ο κάθε προγραμματιστής και ύστερα η άθροιση τους. Το επόμενο βήμα, είναι η εύρεση του ποσοστού συγκέντρωσης, που μπορούμε να το βρούμε μόνο εάν έχουμε υπολογίσει την προηγούμενη ταχύτητα. Εάν έχουμε την προηγούμενη ταχύτητα, τότε τη διαιρούμε με το συνολικό αριθμό ημερών εργασίας και έτσι βρίσκουμε το ποσοστό συγκέντρωσης. Το τελικό στάδιο για την ανεύρεση της αναμενόμενης ταχύτητας, υπολογίζεται από τον πολλαπλασιασμό του ποσοστού συγκέντρωσης με το συνολικό αριθμό ημερών εργασίας. Άρα, έχουμε:

Προηγούμενη Ταχύτητα / Σύνολο ημερών εργασίας = Ποσοστό συγκέντρωσης

Σύνολο ημερών εργασίας * Ποσοστό συγκέντρωσης = Εκτιμώμενη Ταχύτητα.

Εφόσον ο διαχειριστής προϊόντος γνωρίζει την ταχύτητα, με βάση τις εκτιμήσεις των προγραμματιστών για το χρόνο ολοκλήρωσης της ανάπτυξης της κάθε λειτουργίας, επιλέγει σύμφωνα με τη σημαντικότητα της κάθε λειτουργίας αυτές, που μπορούν να υλοποιηθούν σε μία επανάληψη. Ύστερα, τις χωρίζει σε ιστορίες πελάτη και τις αναρτά στον πίνακα επανάληψης όπως φαίνεται στην παρακάτω εικόνα.

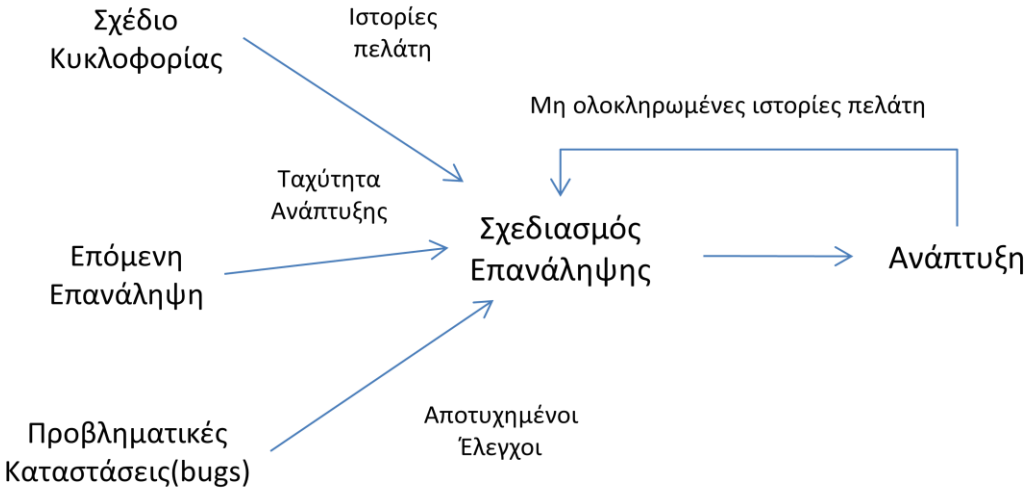


Για τον επιτυχή βραχυπρόθεσμο σχεδιασμό, θα πρέπει να χρησιμοποιηθούν οι κατάλληλοι ορίζοντες σχεδιασμού. Μερικοί από αυτούς είναι:

- Ορισμός του οράματος για ολόκληρο το έργο.
- Ορισμός των δύο επόμενων παραδόσεων κώδικα.
- Ορισμός των απαραίτητων λειτουργιών που θα υλοποιηθούν κατά την επανάληψη.
- Δημιουργία ιστοριών πελάτη για τη συγκεκριμένη επανάληψη.
- Εκτίμηση και κατηγοριοποίηση των ιστοριών πελάτη.
- Εύρεση των απαιτήσεων των πελατών και δημιουργία ελέγχων πελάτη.

Αξίζει να σημειωθεί ότι ο κοντινός χρονικά σχεδιασμός είναι ακριβέστερος και ασφαλέστερος από το μακροπρόθεσμο, για τη ανάπτυξη

του λογισμικού. Επιπροσθέτως, καθίσταται όχι μόνο αρκετά δύσκολη η ανάλυση όλων των επαναλήψεων, αλλά και αδύνατη λόγω μη γνώσης της πορείας του έργου.



3.5.2 Το Παιχνίδι του Σχεδιασμού(Planning game)

Για τη δημιουργία του σχεδίου της κυκλοφορίας χρησιμοποιείται η τεχνική, που ονομάζεται παιχνίδι του σχεδιασμού. Η τεχνική αυτή επιδιώκει τη μεγιστοποίηση των χρήσιμων πληροφοριών, για τη σωστή εκμετάλλευση του σχεδίου κυκλοφορίας.

Το ζητούμενο κατά την ανάπτυξη λογισμικού, είναι η μεγιστοποίηση της αξίας του με το μικρότερο δυνατό κόστος. Οι χρήστες του συστήματος γνωρίζουν το πώς θα αυξήσουν την αξία του προγράμματος, ενώ αντίθετα οι προγραμματιστές, με τη δημιουργία ποιοτικού κώδικα μειώνουν το κόστος της ανάπτυξης του λογισμικού. Έτσι, το παιχνίδι του σχεδιασμού αναλαμβάνει να συμπεριλάβει τις απόψεις και ιδέες των δύο αυτών ομάδων καθηκόντων κατά το σχεδιασμό, ώστε να επιτευχθούν οι στόχοι την ομάδας. Για το λόγο αυτό, οι δύο ομάδες καθηκόντων συναντώνται και χρησιμοποιούν την τεχνική που αναλύουμε:

- Οποιοσδήποτε μπορεί να δημιουργήσει ιστορίες πελάτη.
- Οι προγραμματιστές εκτιμούν το χρόνο υλοποίησης των ιστοριών πελάτη.
- Οι χρήστες τοποθετούν, με βάση τη σημαντικότητα, την κάθε ιστορία πελάτη.
- Τα βήματα αυτά επαναλαμβάνονται μέχρι να έχουν εκτιμηθεί και τοποθετηθεί όλες οι ιστορίες πελάτη.

Το αποτέλεσμα της τεχνικής αυτής είναι η δημιουργία ενός ενιαίου σχεδίου, κατά το οποίο οι ιστορίες όχι μόνο είναι τοποθετημένες με βάση τη σημαντικότητα τους, αλλά έχει εκτιμηθεί και ο χρόνος για την υλοποίησή τους.

3.5.3 Η Διαχείριση των Κινδύνων (Risk management)

Σε παλαιότερες προγραμματιστικές μεθοδολογίες παρατηρούνταν αρκετές δυσκολίες σε ότι αφορά τον υπολογισμό παράδοσης του τελικού προγράμματος. Αντιθέτως, στον ευέλικτο προγραμματισμό γίνεται χρήση της ταχύτητας σα μέτρο σύγκρισης της προόδου του έργου. Όμως, ακόμα και με τον τρόπο αυτό, δεν μπορούμε να επιβεβαιώσουμε τις εκτιμήσεις μας βασιζόμενοι μόνο στην ταχύτητα. Υπάρχουν ανεπιθύμητες περιπτώσεις, όπου είτε κάποιο άτομο αρρώστησε είτε κάποιος υπολογιστής χάλασε κτλ. Για το λόγο αυτό, ο ακραίος προγραμματισμός ενσωματώνει την τεχνική της διαχείρισης του κινδύνου, που επιτρέπει όχι μόνο την εκτίμηση της ολοκλήρωσης του έργου, αλλά και την επίτευξη της εκτίμησης αυτής.

Σε κάθε έργο παρατηρούνται καταστάσεις που επιβραδύνουν την ανάπτυξη του έργου, όπως η εισαγωγή νέων λειτουργιών. Οι καταστάσεις αυτές, δρουν σαν πολλαπλασιαστής στην εκτίμηση της ολοκλήρωσης του έργου μας, με αποτέλεσμα ο ακραίος προγραμματισμός να απαιτεί τη δημιουργία πινάκων για την εύρεση της πρόσθετης καθυστέρησης. Παρακάτω

αναρτούμε τον πίνακα που προτείνουν ο DeMarco με τον Lister, για την εύρεση της καθυστέρησης από κοινούς κινδύνους.

Ποσοστό πιθανότητας	Αυστηρή	Επικίνδυνη	Περιγραφή
10%	X1	X1	Σχεδόν απίθανη
50%	X1,4	X2	50-50 πιθανότητες
90%	X1,8	X4	Σχεδόν σίγουρη

Στην πρώτη στήλη παρατηρούμε τα ποσοστά επιτυχίας που αφορούν την πρόβλεψη που έχουμε κάνει για την ολοκλήρωση του έργου. Έτσι, η πρώτη γραμμή, μας αναφέρει ότι έχουμε μόνο 10% επιτυχία να ολοκληρώσουμε το έργο στο χρόνο που προβλέψαμε. Στη δεύτερη και τρίτη στήλη παρατηρούμε τους πολλαπλασιαστές που αυξάνουν την ολοκλήρωση του έργου. Αλλά ποια από τις δύο στήλες μας αντιπροσωπεύει; Αυτό εξαρτάται από την ομάδα της ανάπτυξης του λογισμικού. Εάν έχουμε σταθερή ταχύτητα και αντιμετωπίζουμε με ευκολία τα όποια προβλήματα παρουσιάζονται, τότε η δεύτερη στήλη θα χρησιμοποιηθεί. Σε αντίθετη περίπτωση, η τρίτη. Γνωρίζοντας την ταχύτητα, τους πολλαπλασιαστές επικινδυνότητας και το επίπεδο επικινδυνότητας, μπορούμε να υπολογίσουμε την ολοκλήρωση της ανάπτυξης του λογισμικού, με βάση τον επόμενο κανόνα.

Υπολειπόμενοι πόντοι =((υπολειπόμενες επαναλήψεις – επίπεδο επικινδυνότητας) * ταχύτητα)/ πολλαπλασιαστής επικινδυνότητας.

Για παράδειγμα, έστω ότι απομένουν δέκα επαναλήψεις σε μια σχετικά έμπειρη ομάδα.

Υπολειπόμενοι πόντοι = (10 - 2) *14 = 112

10% πιθανότητα : 112 / 1 = 122 πόντοι

50 % πιθανότητα : 112 / 1,4 = 80 πόντοι

90% πιθανότητα : 112 / 1,8 = 62 πόντοι

Άρα, έχουμε 10% πιθανότητα να ολοκληρώσουμε και τους 112 πόντους, 50% να ολοκληρώσουμε 80 πόντους και 90% να ολοκληρώσουμε 62 πόντους στο προβλεπόμενο χρόνο.

3.5.4 Η χαλάρωση(Slack)

Η τεχνική της χαλάρωσης σκοπεύει στην αφιέρωση αρκετού χρόνου, ώστε να αντιμετωπιστούν προβλήματα που προκύπτουν κατά την ανάπτυξη του λογισμικού. Με τον τρόπο αυτό, επιτυγχάνουμε όχι μόνο ομαλότερη ταχύτητα ανάπτυξης, αλλά και εκπλήρωση των στόχων που έχουμε θέσει σε κάθε επανάληψη.

Ένας καλός τρόπος για αξιοποίηση του ελεύθερου αυτού χρόνου είναι η έρευνα. Σύμφωνα με τον τρόπο αυτό, τα άτομα που συμμετέχουν στη διαδικασία ανάπτυξης του λογισμικού, επιλέγουν την έρευνα με την οποία θα ασχοληθούν. Με τον τρόπο αυτό αποκτούν συγκεκριμένες δεξιότητες που ίσως επιταχύνουν την ανάπτυξη του λογισμικού. Επιπροσθέτως, επειδή οι περισσότεροι προγραμματιστές διακρίνονται για την επιθυμία τους να καταστούν σημαντικοί, η απόκτηση νέων δεξιοτήτων θα επισπευσθεί με όποια οφέλη κι αν αυτό συνεπάγεται. Είναι σημαντικό να αναφέρουμε, ότι ο χρόνος της χαλάρωσης δε θα πρέπει να είναι περισσότερος αλλά ούτε και λιγότερος από όσο χρειαζόμαστε, αφού και στις δύο περιπτώσεις το αντίκτυπο στην ταχύτητα ανάπτυξης είναι αρκετά μεγάλο. Για το λόγο αυτό, ο διαχειριστής του προϊόντος θα πρέπει να αναθεωρεί συχνά το χρόνο, που αφιερώνεται στην τεχνική αυτή, με βάση όχι μόνο τα προβλήματα που καλούνται να αντιμετωπίσουν, αλλά και τη σταθερότητα ή μη που παρουσιάζει η ταχύτητα ανάπτυξης.

3.5.5 Οι Ιστορίες Πελάτη(Customer Stories)

Συχνά, στις περισσότερες τεχνικές του ακραίου προγραμματισμού, αναφερόμαστε στις ιστορίες πελάτη, οι οποίες είναι λειτουργίες που επιθυμούν οι πελάτες να εμπεριέχονται στο πρόγραμμα. Οι χρήστες λοιπόν, δημιουργούν τις ιστορίες πελάτη σε συνεργασία με το διαχειριστή προϊόντος, ώστε όχι μόνο να γνωστοποιηθούν οι προσδοκίες, αλλά και να αποδοθεί αξία στο αναπτυσσόμενο λογισμικό. Οι ιστορίες πελάτη θα πρέπει να ενσωματώνουν:

- Την αξία του πελάτη, δηλαδή τις λειτουργίες που επιθυμεί για υλοποίηση.
- Κριτήρια ελέγχου σωστής υλοποίησης.
- Πρόσθετες πληροφορίες που χρησιμοποιούνται κατά το σχεδιασμό.

Κατά το σχεδιασμό κυκλοφορίας, οι λειτουργίες χωρίζονται σε ιστορίες πελάτη και αναγράφονται σε κάρτες. Επιπροσθέτως, θα πρέπει να αποφασισθεί ποιες λειτουργίες θα υλοποιηθούν κατά την επόμενη επανάληψη. Έτσι, στις κάρτες αυτές αναγράφεται η σημαντικότητά τους, δηλαδή ένας αριθμός που αναφέρει την αξία τους, που καθορίζεται από τους χρήστες. Επίσης, στις κάρτες αυτές αναγράφεται και ο εκτιμώμενος χρόνος υλοποίησης που έχει υπολογισθεί από τους προγραμματιστές.

Αν και δεν υπάρχει κάποιος κανόνας για το τι πρέπει να αναγράφεται στις κάρτες, παρουσιάζουμε τις πιο κοινές πληροφορίες:

- Ένας μοναδικός αριθμός(id), ώστε να ξεχωρίζουμε την ιστορία πελάτη από τις υπόλοιπες.
- Ένα όνομα, το οποίο θα χρησιμοποιούμε όταν αναφερόμαστε σε συγκεκριμένη ιστορία.

- Μία σύντομη περιγραφή, από την οποία οι προγραμματιστές θα αναγνωρίζουν τις πτυχές που πρέπει να υλοποιηθούν.
- Ένας αριθμός, που αφορά τη σημαντικότητα σύμφωνα πάντα με τον πελάτη.
- Οι εκτίμηση για τις μέρες που χρειάζεται, ώστε να υλοποιηθεί πλήρως.

Στο σημείο αυτό, αξίζει να τονιστεί, ότι όσο μεγαλύτερος είναι ο αριθμός της σημαντικότητας, τόσο αναγκαία θεωρείται από τους πελάτες η συγκεκριμένη ιστορία και σαν αποτέλεσμα να υλοποιηθεί συντομότερα από άλλες. Επειδή συχνά παρατηρείται το φαινόμενο κατά το οποίο οι χρήστες προσθέτουν νέες ιστορίες, καλό είναι ο αριθμός σημαντικότητας να μην είναι συνεχής, αλλά να υπάρχουν αριθμοί που να μην έχουν εισαχθεί σε κάποια ιστορία.

3.5.6 Η Εκτίμηση(Estimating)

Για τη σωστή επιλογή του αριθμού των ιστοριών πελάτη, που θα υλοποιηθούν κατά την επανάληψη, οι προγραμματιστές θα πρέπει να εκτιμήσουν το χρόνο που θα χρειασθεί η κάθε μία για την πλήρη υλοποίησή της. Επίσης, η ανάγκη για τη σωστή μέτρηση του αριθμού ταχύτητας ανάπτυξης αναγνωρίστηκε εγκαίρως, με αποτέλεσμα τη δημιουργία της τεχνικής της εκτίμησης. Η τεχνική αυτή, αφορά την όσο το δυνατόν πλησιέστερη προσέγγιση του πραγματικού χρόνου ολοκλήρωσης, που επιτυγχάνεται από τη χρήση εκλεπτυσμένων μεθοδολογιών.

Η εκτίμηση του χρόνου που χρειάζεται μία ιστορία στο να υλοποιηθεί, θεωρείται από τα πιο δύσκολα καθήκοντα που αναλαμβάνουν οι προγραμματιστές. Συχνά, οι εκτιμήσεις τους δεν αντιπροσωπεύουν το πραγματικό χρόνο υλοποίησης της ιστορίας πελάτη, λόγω του μη ορθού υπολογισμού των προβληματικών καταστάσεων που προκύπτουν. Για το

λόγο αυτό, ο ακραίος προγραμματισμός προτείνει μια διαφορετική προσέγγιση του θέματος αυτού, δηλαδή την εκτίμηση των προγραμματιστών με βάση την ιδανικότερη κατάσταση. Αυτό σημαίνει ότι οι προγραμματιστές πρέπει να αναλογισθούν τις μέρες που θα χρειασθούν για την ολοκλήρωση της ιστορίας πελάτη όταν δεν υπάρχουν διακοπές στην εργασία τους. Βέβαια, το αποτέλεσμα δεν αντιπροσωπεύει την πραγματική κατάσταση, αφού είναι απαλλαγμένο από τις διακοπές και τα προβλήματα που ίσως προκύψουν. Ωστόσο, ο συνδυασμός του με την ταχύτητα ανάπτυξης επιφέρει μια αποδεκτή λύση στο συγκεκριμένο πρόβλημα.

Η ταχύτητα ανάπτυξης σε έμπειρες ομάδες τείνει να είναι σταθερή. Ακόμη κι όταν η ομάδα μας δεν έχει εξοικειωθεί με καμία ευέλικτη προγραμματιστική μεθοδολογία, τότε η ταχύτητα ανάπτυξης μας επιδεικνύει την απόδοση της ομάδας μας. Συγκεκριμένα, κατά την περίπτωση όπου δεν έχουμε αναπτύξει όλες τις προσδοκώμενες ιστορίες πελάτη, η προσδοκώμενη ανάπτυξη μειώνεται, παρουσιάζοντάς μας όχι μόνο την πραγματική, αλλά υπολογίζοντας και την ακριβέστερη ανάπτυξη στην επόμενη επανάληψη. Ομοίως, όταν έχουν ολοκληρωθεί όλες οι προσδοκώμενες ιστορίες πελάτη και υπάρχει αρκετός χρόνος μέχρι την παράδοση των νέων λειτουργιών, τότε επιλέγεται ακόμα μία ιστορία πελάτη, οπότε αυξάνεται και η ταχύτητα ανάπτυξης. Το αποτέλεσμα και των δύο περιπτώσεων είναι η ακριβέστερη εύρεση του φόρτου εργασίας, που μπορεί να αναλάβει η ομάδα σε κάθε επανάληψη και με τον τρόπο αυτό επιτυγχάνεται η εκτίμηση για την ολοκλήρωση του έργου. Ακόμη, όταν μια ιστορία δεν έχει ολοκληρωθεί πλήρως, τότε οι πόντοι της δεν μετριούνται κατά την εύρεση της προηγούμενης ταχύτητας. Αν αναλογιστούμε ότι οι εκτιμήσεις θα πρέπει να γίνονται με βάση τον ιδανικότερο χρόνο ολοκλήρωσης της κάθε ιστορίας, τότε δεν κινδυνεύουμε να αποδώσουμε περισσότερο χρόνο ολοκλήρωσης από ό,τι θα χρειασθεί στην πραγματικότητα, με αποτέλεσμα την ορθότερη μέτρηση της ταχύτητας. Για παράδειγμα, αποφεύγεται η περίπτωση κατά την οποία έχουμε αποδώσει πέντε πόντους σε μία ιστορία, αλλά ολοκληρώθηκε σε τρείς μέρες κι από ένα μόνο προγραμματιστή.

3.6 Ανάπτυξη-Υλοποίηση(Developing)

Όλες οι προγραμματιστικές μεθοδολογίες, ευέλικτες και μη, επικεντρώνονται στη βελτίωση των τεχνικών που αφορούν την εγγραφή του κώδικα, ώστε να αναπτυχθεί το ζητούμενο λογισμικό στο μικρότερο χρόνο, με το μικρότερο κόστος. Τον τρόπο που συγγράφουμε κώδικα καλείται να δώσει το στάδιο της υλοποίησης, κατά το οποίο επιτυγχάνεται η επιθυμητή ποιότητα, που όχι μόνο μειώνει το κόστος αλλά και το χρόνο ανάπτυξης του λογισμικού. Όμως, η υλοποίηση, εκτός από τη συγγραφή του κώδικα, θα πρέπει να εξασφαλίσει και τη σωστή λειτουργία των δημιουργημένων λειτουργιών. Άρα, κατά το στάδιο αυτό παρατηρούμε και τη χρήση τεχνικών ελέγχου του κώδικα. Ο ακραίος προγραμματισμός χρησιμοποιεί οκτώ τεχνικές, για την επίτευξη της ποιότητας. Αυτές είναι:

- Αυξητική ανάπτυξη.
- Έλεγχοι χρήστη.
- Ανάπτυξη οδηγούμενη από ελέγχους.
- Αναδόμηση του κώδικα.
- Απλός σχεδιασμός.
- Αυξητικός σχεδιασμός.
- Λύσεις χαλάρωσης.
- Βελτίωση απόδοσης.

3.6.1 Η Αυξητική Ανάπτυξη(Incremental Requirements)

Σύμφωνα με την τεχνική της αυξητικής ανάπτυξης, η ανάπτυξη της κάθε ιστορίας πελάτη επιτυγχάνεται με τη συγγραφή μικρών πτυχών του

κώδικα, που αφού έχουν ελεγχθεί, σταδιακά εμπλουτίζονται με νέες λειτουργίες. Με τον τρόπο αυτό, όχι μόνο μειώνουμε δραστικά τις προβληματικές καταστάσεις, αλλά και συγγράφεται κώδικας υψηλής ποιότητας. Οι πτυχές της κάθε λειτουργίας αφορούν τις απαιτήσεις του πελάτη, άρα καθίσταται αναγκαία η γνωστοποίησή τους στους προγραμματιστές. Για το λόγο αυτό, κατά τη διαδικασία ανάπτυξης κάποιας απαίτησης του πελάτη, δίδεται η δυνατότητα στους χρήστες να αποφασίσουν για τις επόμενες, ώστε να εξοικονομηθεί πολύτιμος χρόνος. Βέβαια, αρκετές λειτουργίες αναγράφονται στην κάρτα της εκάστοτε ιστορίας, αλλά δεν είναι λίγες οι φορές όπου οι χρήστες τροποποιούν τις απαιτήσεις τους.

3.6.2 Οι Έλεγχοι Πελατών(Customer Tests)

Στην αρχή κάθε επανάληψης, ο διαχειριστής προϊόντος ελέγχει τις ιστορίες πελάτη, που θα υλοποιηθούν για λειτουργίες, οι οποίες είτε δε θα γίνουν κατανοητές από τους προγραμματιστές, είτε θα οδηγηθούν σε εσφαλμένες καταστάσεις. Εφόσον υπάρχουν τέτοιες ιστορίες, τότε για τις ομάδες που ανέλαβαν την υλοποίηση της κάθε μίας από αυτές, οργανώνεται συνάντηση με τους χρήστες για την αποσαφήνιση όλων των πτυχών. Κατά τις συναντήσεις αυτές, όχι μόνο απαντώνται ερωτήσεις, αλλά δίνονται και παραδείγματα, για να επιτευχθεί σε μεγάλο ποσοστό η κατανόηση των ιστοριών πελάτη από τους προγραμματιστές. Τα παραδείγματα αυτά, είναι οι έλεγχοι πελατών, που έχουν σκοπό όχι μόνο την κατανόηση των πτυχών, που αφορούν τον τομέα, από τους προγραμματιστές, αλλά και τη χρήση τους κατά τον έλεγχο του λογισμικού.

3.6.3 Ανάπτυξη Οδηγούμενη από Ελέγχους(Test-driven Development TDD)

Ένα από τα κυριότερα προβλήματα, που παρουσιάζονταν στις μη ευέλικτες προγραμματιστικές μεθόδους, ήταν ο έλεγχος κατά το τελευταίο στάδιο της ανάπτυξης. Επειδή η συγγραφή του κώδικα απαιτεί αρκετό κόπο και δεξιότητες, παρουσιάζονται αρκετά σφάλματα, είτε στη σύνταξη του κώδικα είτε στη λογική του υπόσταση. Αυτό είχε ως αποτέλεσμα τη χρονική καθυστέρηση, αλλά και το αυξημένο κόστος για την επιδιόρθωση των σφαλμάτων που αρκετές φορές ήταν απαγορευτικό. Αντιθέτως, στις ευέλικτες προγραμματιστικές μεθοδολογίες, λόγω της ταυτόχρονης χρήσης όλων των σταδίων ζωής του λογισμικού, τα σφάλματα αναγνωρίζονται εγκαίρως, και μάλιστα πριν παραδοθούν οι νεοδημιουργηθείσες λειτουργίες. Όμως, ο ακραίος προγραμματισμός δεν αρκέστηκε μόνο στην προηγούμενη παρατήρηση, αλλά καινοτομεί με τη χρήση της τεχνικής που ονομάζεται ανάπτυξη οδηγούμενη από ελέγχους(TDD), η οποία μας παρουσιάζει τα σφάλματα ακριβώς όταν εμφανίζονται.

Η τεχνική αυτή, είναι ουσιαστικά ένας κύκλος ελέγχου, κωδικοποίησης και αναδόμησης του κώδικα. Όταν δημιουργείται μια νέα λειτουργία, τότε παρουσιάζεται συνεχής χρήση τέτοιων ελέγχων, μέχρι την πλήρη υλοποίηση της λειτουργίας. Σύμφωνα με τους Janzen και Saiedian, που μελέτησαν τη συγκεκριμένη τεχνική, παρουσιάσθηκε αισθητή μείωση δυσλειτουργιών, που εμφανίζονται μετά την παράδοση του κώδικα. Εκτός όμως από το πλεονέκτημα αυτό, παρουσιάζεται ταυτόχρονα, βελτίωση στην ποιότητα του κώδικα και μερική προστασία απέναντι σε προβλήματα, που θα εμφανισθούν στο μέλλον. Στον αντίποδα, θα χρειασθεί αρκετός χρόνος και κόπος μέχρι να καταστεί η χρήση της τεχνικής αυτής αποτελεσματική, αλλά αξίζει τον κόπο.

Η ανάπτυξη οδηγούμενη από ελέγχους, αφορά τη λειτουργική συνέπεια του κώδικα, δηλαδή το να ελέγχει εάν λειτουργεί με βάση τις προσδοκίες του προγραμματιστή. Για το λόγο αυτό, οι προγραμματιστές εγγράφουν τους ελέγχους αυτούς πριν τη σύνταξη του κώδικα. Αν και ακούγεται παράδοξο, το πλεονέκτημα που απολαμβάνουμε, είναι η προσέγγιση στο τι θέλουμε να υλοποιηθεί και όχι στο πώς θα υλοποιηθεί. Με τον τρόπο αυτό, όχι μόνο δεν παραλείπεται καμία πτυχή της ιστορίας πελάτη, αλλά καθίσταται ευκολότερη και η συγγραφή του κώδικα, αφού γίνεται

ευκολότερα γνωστό το τι πρέπει να υλοποιηθεί για να ολοκληρωθεί ο έλεγχος με επιτυχία.

Για τη χρήση της τεχνικής αυτής, πρέπει να ακολουθηθούν τέσσερα μικρά βήματα. Όμως, ας φανταστούμε πρώτα την τεχνική αυτή σαν ένα κύκλο εργασιών, που συμβαίνει αρκετές φορές για την κάθε ιστορία ξεχωριστά και όταν τελειώσει ο παραχθείς κώδικας, θα είναι όσο το δυνατόν ποιοτικότερος και λειτουργικός. Καλό είναι να τονίσουμε ότι η ενασχόληση με κάθε κύκλο δε θα ξεπερνά τα πέντε λεπτά.

- Ανεύρεση των πτυχών, που περιλαμβάνει η ιστορία πελάτη και η δημιουργία μικρών ελέγχων, που τις ενσωματώνουν.
- Έλεγχος ύπαρξης πτυχών.
- Αναδόμηση κώδικα, μέχρι να επιτευχθεί η απαραίτητη ποιότητα.
- Επανάληψη βημάτων, μέχρι την ενσωμάτωση όλων των πτυχών.

Όπως έχουμε αναφέρει, οι έλεγχοι αυτοί θα πρέπει να χρησιμοποιούνται δύο φορές σε κάθε λεπτό. Εάν δεν ισχύει κάτι τέτοιο, τότε το πιο πιθανό είναι οι αλλαγές που κάνουμε να μην είναι αρκετά μικρές, όπως προστάζει η τεχνική αυτή, με αποτέλεσμα να μην επωφελούμαστε πλήρως από αυτή.

3.6.4 Η Αναδόμηση του Κώδικα(Refactoring)

Κατά την ανάπτυξη του λογισμικού, παρατηρείται το φαινόμενο μείωσης της ποιότητας του κώδικα, δηλαδή εξαλείφεται η οργάνωσή-δομή του, με αποτέλεσμα να απαιτείται αρκετή προσπάθεια είτε για την ανάγνωσή του είτε για την τροποποίησή του. Καθώς συνεχίζεται η ανάπτυξη του λογισμικού, το συγκεκριμένο φαινόμενο παρατηρείται ακόμα εντονότερα και έχει ως αποτέλεσμα το αυξημένο κόστος συντήρησης. Ένα μέτρο για τον περιορισμό της κατάστασης αυτής, ήταν η δημιουργία του αντικειμενοστραφή

προγραμματισμού. Ωστόσο, για την περαιτέρω αντιμετώπιση του προαναφερθέντος φαινομένου, που ονομάζεται σπαγγέτι, ο ευέλικτος προγραμματισμός δημιούργησε την τεχνική της αναδόμησης του κώδικα. Η τεχνική αυτή επιδιώκει τη διατήρηση της λειτουργικότητας, ενώ τροποποιείται η δομή του κώδικα, ώστε να καταστεί ευκολότερη η ανάγνωση και η τροποποίησή του. Η τεχνική της αναδόμησης του κώδικα, τροποποιεί τη δομή του κώδικα, όχι με το να την αναδημιουργήσει, αλλά με το να αναγνωρίσει τις πτυχές του κώδικα που χρίζουν τροποποίησης-βελτίωσης.

Για την αναδόμηση του κώδικα πρέπει να γίνουν τα παρακάτω:

- Χρήση πίνακα σε συνδυασμό διαγραμμάτων UML για την ανάλυση του κώδικα.
- Μικρές τροποποιήσεις και ο συνεχής έλεγχος για τη διατήρηση της λειτουργικότητας.

Θα πρέπει να υπενθυμίσουμε, ότι η συγκεκριμένη τεχνική δεν πρέπει να παραλείπεται σε καμιά περίπτωση, ειδικότερα όταν το προς ανάπτυξη πρόγραμμα είναι αρκετά χρονοβόρο για τη δημιουργία του. Με τη χρήση της αναδόμησης του κώδικα επιτυγχάνουμε τη ζητούμενη ποιότητα, που έχουμε αναφέρει αρκετές φορές στο παρελθόν.

3.6.5 Ο Απλός Σχεδιασμός(simple design)

Ο ευέλικτος προγραμματισμός, συχνά αναφέρει την ανάγκη για συγγραφή απλού κώδικα και μάλιστα με την προηγούμενη τεχνική προσπαθεί να επιτύχει το σκοπό αυτό. Επιπροσθέτως, οι προγραμματιστές αρκετές φορές επιδιώκουν την εύρεση του απλούστερου σχεδιασμού. Με τον απλό σχεδιασμό επιτυγχάνουμε:

- Εύκολη τροποποίηση της δομής του κώδικα.
- Μείωση κόστους συντήρησης.

- Αποτροπή υλοποίησης λειτουργιών, που δεν έχουν καθοριστεί για ανάπτυξη.

Όπως και στην αναδόμηση του κώδικα, έτσι και στη συγκεκριμένη τεχνική θα τονισθεί η ανάγκη για την εξάλειψη διπλότυπου κώδικα, ο οποίος όχι μόνο επιφέρει κακή δομή του κώδικα, αλλά και δυσκολία στην τροποποίησή του. Την προηγούμενη παρατήρηση την επεσήμανε ο Martin Fowler, όπου μας ανέφερε ενδεικτικά:

«Ένα από τα πράγματα που προσπάθησα να κάνω είναι να βρω είτε απλούστερους κανόνες είτε κανόνες για την αναγνώριση του καλού ή του κακού σχεδιασμού. Πιστεύω ότι ένας από τους σημαντικότερους κανόνες είναι η αποφυγή διπλοτύπων.»

3.6.6 Αυξητικός Σχεδιασμός(incremental Design)

Μία από τις καινοτομίες του ευέλικτου προγραμματισμού, είναι η ενσωμάτωση νέων λειτουργιών-επιθυμιών του πελάτη, κατά την ανάπτυξη του λογισμικού. Έτσι, δημιουργήθηκε η τεχνική του αυξητικού σχεδιασμού, η οποία επιτρέπει τη σταδιακή δημιουργία τεχνικής υποδομής σε κάθε επανάληψη.

Ο αυξητικός σχεδιασμός εφαρμόζεται κατά τη χρήση της ανάπτυξης, η οποία καθοδηγείται από ελέγχους και στην οποία προστίθενται κάθε φορά λίγες γραμμές κώδικα που ελέγχονται εξονυχιστικά. Γίνεται εύκολα κατανοητό, ότι ο αυξητικός σχεδιασμός προωθεί τη σταδιακή ανάπτυξη κώδικα, ώστε να επιτευχθεί ο έλεγχός του αλλά και ο απλός σχεδιασμός. Άρα, τα στάδια που ακολουθούν οι προγραμματιστές είναι τα παρακάτω:

- Δημιουργία της απλούστερης δομής, που είναι και λειτουργική.
- Σταδιακή ανάπτυξη του κώδικα, επεκτείνοντας τις λειτουργίες.
- Συνεχής βελτίωση της δομής(αναδόμηση του κώδικα).

3.6.7 Βελτίωση απόδοσης (Performance Optimization)

Κατά τον έλεγχο του αναπτυσσόμενου προγράμματος, οι ελεγκτές σε συνεργασία με τους χρήστες, εκτός της αρμοδιότητας τους για την εύρεση σφαλμάτων, συντάσσουν πορίσματα σχετικά με την απόδοση του συστήματος. Με τον όρο απόδοση εννοούμε την ταχύτητα, με την οποία εκτελούνται οι διάφορες λειτουργίες του συστήματος. Όπως γίνεται κατανοητό, ακόμη και οι μικρές καθυστερήσεις μερικών δευτερολέπτων μπορούν να επιφέρουν όχι μόνο αρνητικά σχόλια από τον πελάτη, αλλά και αρνητικές επιχειρηματικές επιπτώσεις, όσον αφορά το σκοπό δημιουργίας των λειτουργιών. Έτσι, η τεχνική της βελτίωσης απόδοσης με τη χρήση ελέγχων, επιδιώκει την ανεύρεση και διόρθωση του κώδικα, που ευθύνεται στις καθυστερήσεις. Όμως αξίζει να τονισθεί, ότι το παραπάνω εγχείρημα δεν είναι πάντα επιτεύξιμο, εάν αναλογιστούμε ότι υπάρχουν αρκετοί παράγοντες καθυστέρησης που δεν αντιμετωπίζονται, όπως είναι των βάσεων δεδομένων.

Οι έλεγχοι της απόδοσης, θα πρέπει να επιφέρουν τους στόχους του πελάτη, ώστε να καταστούν ικανοποιητικοί. Οι στόχοι αυτοί μπορεί να αφορούν:

- Την ρυθμοαπόδοση(throughput), που μετρά τις λειτουργίες, οι οποίες μπορούν να εκτελεστούν σε συγκεκριμένο χρονικό διάστημα.
- Την καθυστέρηση(latency) μέχρι την εκτέλεση της λειτουργίας, που είναι αποδεκτή από τους χρήστες.
- Την ανταπόκριση(responsiveness), που είναι ο αποδεκτός χρόνος μέχρι την ανατροφοδότηση των χρηστών, κατά την ολοκλήρωση μιας λειτουργίας.

Για το σωστό έλεγχο της επιθυμητής επίδοσης, οι πελάτες θα πρέπει να αναφέρουν όλους τους επιθυμητούς προαναφερόμενους χρόνους. Με τον

τρόπο αυτό, οι προγραμματιστές αναγνωρίζουν το στόχο της απόδοσης και επιδιώκουν την επίτευξή του.

3.7 Η Κυκλοφορία(Releasing)

Το τελευταίο στάδιο του κύκλου ζωής, στο οποίο θα αναφερθούμε είναι το στάδιο της κυκλοφορίας. Κατά τη διάρκεια της διαδικασίας ανάπτυξης κώδικα, η ομάδα σε εβδομαδιαία βάση παραδίδει λειτουργικό και έτοιμο για χρήση κώδικα-λειτουργίες, που έχουν καθοριστεί κατά το στάδιο του σχεδίου κυκλοφορίας. Για να γίνει εφικτό το συγκεκριμένο εγχείρημα, η ομάδα ανάπτυξης θα πρέπει να επικεντρωθεί σε ορισμένα σημαντικά θέματα, όπως η εύρεση τεχνικών-μεθοδολογιών για τη σωστή και γρήγορη συγγραφή κώδικα με κύριο μέλημα, την όσο το δυνατόν ταχύτερη ανάπτυξη κώδικα, με σαφώς ποιοτικά αποτελέσματα. Μία από τις τεχνικές αυτές είναι η δεκάλεπτη ανάπτυξη, στην οποία βασίζεται το συγκεκριμένο στάδιο του κύκλου ζωής.

Έχει ιδιαίτερη σημασία να κατανοήσουμε τη χρησιμότητα της ανάπτυξης αυτής, όπως και τα πλεονεκτήματα που προσφέρει, έναντι των μεθόδων που είχαν χρησιμοποιηθεί κατά το παρελθόν. Σε παλαιότερες προγραμματιστικές μεθόδους τα στάδια του κύκλου ζωής ήταν διακριτά. Αυτό είχε ως αποτέλεσμα την έλλειψη της δυνατότητας παράλληλης εκτέλεσης των σταδίων ανάπτυξης του λογισμικού, γεγονός που επέφερε όχι μόνο τη συχνή εγκατάλειψη έργων, αλλά και αυξημένο κόστος για τη συντήρησή του. Όμως, τα προβλήματα δε σταματούν εδώ. Ο πελάτης συχνά βρισκόταν σε αντιπαράθεση με την ομάδα ανάπτυξης λογισμικού, εφόσον δεν του δινόταν η δυνατότητα παρέμβασης και τροποποίησης των αρχικών απαιτήσεών του. Γι' αυτό το λόγο, ο ευέλικτος προγραμματισμός ενσωματώνει την τεχνική της δεκάλεπτης ανάπτυξης.

Για να επιτευχθεί η δεκάλεπτη ανάπτυξη, ο ευέλικτος προγραμματισμός ενσωματώνει στο στάδιο της κυκλοφορίας, τις έξι παρακάτω τεχνικές:

- Έτοιμο από κάθε άποψη (“Done done”)
- Χωρίς Σφάλματα (no bugs)
- Έλεγχος Έκδοσης (version control)
- Δεκάλεπτη Ανάπτυξη (A ten minute build)
- Συχνή Ενοποίηση (Continuous integration)
- Κοινή Δικαιοδοσία Κώδικα(Collective code ownership)

3.7.1 Έτοιμο από κάθε άποψη(Done done)

Μπορούμε πολύ εύκολα να κατανοήσουμε την έννοια της τεχνικής αυτής, αν αναλογιστούμε το εξής: “Δε θα ήταν ευχής έργο, αν μετά την ολοκλήρωση μίας επιμέρους λειτουργίας του υπό ανάπτυξη συστήματος, δε χρειαζόταν ξανά να αναφερθούμε σε αυτή, για τυχόν τροποποιήσεις ή παραλείψεις που εντοπίστηκαν;”. Εάν ίσχυε η παραπάνω πρόταση, τότε θα είχε ως αποτέλεσμα την ελαχιστοποίηση του συνολικού χρόνου διεκπεραίωσης του έργου, καθώς και λιγότερα έξοδα που ίσως προέκυπταν από τυχόν λάθη ή παραλείψεις. Επιπροσθέτως, η ανεύρεση του χρόνου ολοκλήρωσης του έργου θα εμφανίζονταν ευκολότερη από ποτέ, ενώ παράλληλα εμφανίζεται δραστική μείωση του μη παραγωγικού χρόνου, δηλαδή του χρόνου που χρησιμοποιείται για οποιαδήποτε άλλη διαδικασία, εκτός της ανάπτυξης νέων λειτουργιών. Ένα παράδειγμα είναι η επιδιόρθωση σφαλμάτων, που αξίζει να τονισθεί ότι πρέπει να αντιμετωπισθεί άμεσα.

Αυτή είναι η έννοια της συγκεκριμένης τεχνικής, δηλαδή η κάθε λειτουργία η οποία ολοκληρώνεται, θα πρέπει να είναι πλήρης, να εξυπηρετεί τις ανάγκες που είχαν ορισθεί, να είναι πλήρως λειτουργική και σαφώς ποιοτική γεγονός που συνεπάγεται με την ικανοποίηση του πελάτη. Για το λόγο αυτό, ο ακραίος προγραμματισμός χρησιμοποιεί τη συγκεκριμένη τεχνική, η οποία προτρέπει τους προγραμματιστές να έχουν ολοκληρώσει

κάθε ενέργεια που αφορά μια λειτουργία πριν αναπτύξουν μια άλλη. Αυτό, σημαίνει ότι μόνο όταν καθίσταται δυνατή η άμεση παρουσίαση της λειτουργίας στον πελάτη, χωρίς καμία τροποποίηση για την επίδειξη των δυνατοτήτων της, μπορεί η ομάδα ανάπτυξης του λογισμικού να ενασχοληθεί με την υλοποίηση άλλης επιθυμητής λειτουργίας.

Η σωστή χρήση της τεχνικής αυτής, είναι ίσως μια από τις πιο πολύπλοκες διαδικασίες των ευέλικτων προγραμματιστικών μεθοδολογιών, καθώς η παρουσία της υφίσταται καθ' όλη τη διάρκεια της ανάπτυξης λογισμικού, ενώ ταυτόχρονα απαιτεί τη σωστή χρήση όλων των υπολοίπων τεχνικών. Επίσης, λόγω του ότι είναι μια από τις καινοτομίες των ευέλικτων προγραμματιστικών μεθοδολογιών, η ομάδα ανάπτυξης λογισμικού πρέπει να μοχθήσει, ώστε να μπορέσει τελικά να συμβαδίσει με την ιδέα της τεχνικής αυτής. Αυτά που πρέπει να εφαρμοσθούν, πάντα με ιδιαίτερη προσοχή, είναι:

- Έλεγχος του κώδικα.(Tested)
- Η συγγραφή του κώδικα. (Coded)
- Ο σχεδιασμός του κώδικα. (Designed)
- Η ενοποίηση του κώδικα. (Integrated)
- Η δόμηση και μεταγλώττιση. (Build)
- Η δημιουργία του αρχείου εγκατάστασης της εφαρμογής. (Installs)
- Ο πελάτης έχει επανεξετάσει το έργο και έχει συμφωνήσει ότι τον ικανοποιεί. (Reviewed)
- Ο εντοπισμός και εξάλειψη όλων των σφαλμάτων. (Fixed)
- Ο πελάτης συμφωνεί ότι το έργο έχει τελειώσει. (Accepted)

3.7.2 Χωρίς Σφάλματα (No bugs)

Όπως όλοι γνωρίζουμε, κατά το στάδιο ανάπτυξης λογισμικού, και για να είμαστε πιο συγκεκριμένοι κατά το στάδιο ανάπτυξης κώδικα, η εμφάνιση σφαλμάτων-δυσλειτουργιών (bugs) είναι αναπόφευκτη και ιδίως αν πρόκειται για μεγάλα και πολύπλοκα έργα. Ο εντοπισμός βέβαια των σφαλμάτων αυτών, αποτελεί τον ακρογωνιαίο λίθο καθώς στις παλαιότερες προγραμματιστικές μεθοδολογίες, ο έλεγχος του κώδικα όχι μόνο γινόταν για αρκετές γραμμές κώδικα, αλλά και αρκετά μεταγενέστερα από τη συγγραφή του.

Λύση ήρθε να προσφέρει ο ευέλικτος προγραμματισμός και ειδικότερα η τεχνική “χωρίς σφάλματα”, όπου η ομάδα ανάπτυξης εργάζεται μεθοδικά και προσεχτικά, με κύριο σκοπό να προκύψουν όσο το δυνατό λιγότερα σφάλματα. Αυτό βέβαια, για να επιτευχθεί θα πρέπει η ομάδα ανάπτυξης να τηρεί με μεγάλη αυστηρότητα μερικούς κανόνες-τεχνικές όπως:

- Συγγραφή με λιγότερα σφάλματα, χρησιμοποιώντας μια ευρεία ποικιλία τεχνικών και οργανωτικών πρακτικών.
- Περιορισμός των λαθών με την ανακατασκευή κομματιών κώδικα με φτωχό σχεδιασμό.
- Εξάλειψη των σφαλμάτων γρήγορα, ώστε να μειωθεί ο βαθμός επίδρασής τους.
- Συνεχής έλεγχος των λειτουργιών, που κατασκευάζονται.
- Επιδιόρθωση των λειτουργιών από επιπόλαια λάθη, που ίσως αποβούν κρίσιμα.

3.7.3 Έλεγχος Έκδοσης (version control)

Μια ομάδα ανάπτυξης, αν θέλει να δουλεύει ως ομάδα, θα πρέπει να επινοήσει ένα αποτελεσματικό τρόπο συντονισμού του πηγαίου κώδικά της, των ελέγχων της, όπως επίσης και άλλων σημαντικών πηγών του έργου. Ο έλεγχος έκδοσης του συστήματος παρέχει ένα χώρο εναπόθεσης, που βοηθά στο συντονισμό των αλλαγών μέσω της καταχώρησής τους σε αρχεία, παρέχοντας ένα ιστορικό αλλαγών. Η ομάδα η οποία χρησιμοποιεί την τεχνική του ελέγχου έκδοσης, έχει τη δυνατότητα του συντονισμού των αλλαγών που διαπράττονται. Αυτή είναι μια λειτουργία, η οποία εκτελείται καθημερινά από μερικά μέλη της ομάδας ανάπτυξης, τα οποία παίρνουν τον πιο πρόσφατο κώδικα, εκτελούν τους απαραίτητους ελέγχους επί του κώδικα αυτού με σκοπό την καταγραφή των όποιων αλλαγών έχουν γίνει κατά το διάστημα της ημέρας, στην συνέχεια αποθηκεύουν τον νέο αυτό κώδικα σε κάποιο εξυπηρετητή, για τη χρήση του από άλλους προγραμματιστές, όποτε καταστεί αναγκαίο.

3.7.4 Δεκάλεπτη Ανάπτυξη (ten minute build)

Τα περισσότερα εξειδικευμένα προγράμματα, που κυκλοφορούν σήμερα, απαιτούν σύνδεση και επικοινωνία με άλλα ώστε να εκπληρώσουν τις λειτουργίες για τις οποίες έχουν προορισθεί, όπως για παράδειγμα μια βάση δεδομένων ή κάποιος εξυπηρετητής ιστοσελίδων. Έτσι οι προγραμματιστές όλο και συχνότερα καλούνται να κατασκευάσουν τέτοιου είδους εφαρμογές, κάτι που δυσχεράνει σε σημαντικό βαθμό τη διαδικασία ανάπτυξης, καθιστώντας τη σαφώς μη επιθυμητή λόγω αυξημένων καθυστερήσεων. Όπως γίνεται κατανοητό, λόγω του καταμερισμού της εργασίας στις ομάδες των προγραμματιστών, παρουσιάζονται αρκετά προβλήματα, που αφορούν όχι μόνο την εγκατάσταση των προγραμμάτων που χρησιμοποιεί το λογισμικό προς ανάπτυξη, αλλά και την ενημέρωσή

τους. Ακόμη, ο έλεγχος του κώδικα καθυστερεί λόγω της χρήσης εφαρμογών τρίτων, κάτι που δεν είναι αποδεκτό, ειδικά κατά την ανάπτυξη πολύπλοκων συστημάτων. Αυτό λοιπόν το πρόβλημα, ώθησε τους δημιουργούς του ακραίου προγραμματισμού στο να συμπεριλάβουν τη δεκάλεπτη ανάπτυξη σαν ξεχωριστή τεχνική, η οποία πραγματεύεται την ταχύτερη ανάπτυξη του κώδικα χρησιμοποιώντας, όσο το δυνατόν, περισσότερους αυτοματισμούς μειώνοντας με τον τρόπο αυτό τον μη παραγωγικό χρόνο. Σκοπός της δεκάλεπτης ανάπτυξης είναι:

- Δημιουργία εκτελέσιμων αρχείων με τη χρήση πηγαίου κώδικα.
- Έλεγχος παραγόμενου κώδικα.
- Παραμετροποίηση λειτουργικού συστήματος για τη σωστή λειτουργία του αναπτυσσόμενου λογισμικού.
- Παραμετροποίηση των προγραμμάτων που χρησιμοποιεί το αναπτυσσόμενο λογισμικό, όπως οι βάσεις δεδομένων.
- Ενημέρωση των τερματικών, με την νέα έκδοση του πηγαίου κώδικα του επιθυμητού προγράμματος.
- Αυτόματη εγκατάσταση προγραμμάτων σε κάθε τερματικό.

3.7.5 Συχνή Ενοποίηση (Continuous integration)

Στα περισσότερα έργα ανάπτυξης λογισμικού υπάρχει μια κρυφή καθυστέρηση ανάμεσα στο σημείο, που η ομάδα ισχυρίζεται ότι το έργο έχει τελειώσει, μέχρι το σημείο που πραγματικά το έργο είναι έτοιμο προς παράδοση στον πελάτη. Τέτοια θέματα συνήθως προσδίδουν περισσότερο άγχος στην ομάδα, καθώς αυτή δεν μπορεί να γνωρίζει το χρονικό διάστημα που θα χρειαστεί, ώστε να φέρει εις πέρας τις απαραίτητες αυτές λειτουργίες. Η αναγνώριση του συγκεκριμένου προβλήματος επέφερε την ενσωμάτωση της συχνής ενοποίησης στις ευέλικτες προγραμματιστικές μεθοδολογίες. Η

συχνή ενοποίηση κρατά όλο τον κώδικα της ομάδας ενοποιημένο με την υπόλοιπη εφαρμογή, με αποτέλεσμα τη γρήγορη αναγνώριση των τυχών παραλείψεων κατά το σχεδιασμό του κώδικα. Ο απώτατος στόχος της τεχνικής αυτής έγκειται στο να είναι σε θέση να παρέχει ανά πάσα στιγμή, κώδικα ολοκληρωμένο, χωρίς δυσλειτουργίες και έτοιμο προς χρήση.

Για να μπορέσει η ομάδα ανάπτυξης να εφαρμόσει με επιτυχία την τεχνική της συχνής ενοποίησης θα πρέπει να δώσει ιδιαίτερη προσοχή στις εξής δυο δραστηριότητες:

- Ενσωμάτωση του παραγόμενου κώδικα ανά τακτά χρονικά διαστήματα, αλλά όχι παραπάνω από μερικές ώρες.
- Συνεχής έλεγχος του κώδικα, που δημιουργείται καθώς και ενημέρωση των αλλαγών που πραγματοποιούνται.

Με την ενσωμάτωση του παραγόμενου κώδικα εννοούμε την ενσωμάτωση του νέου, που μόλις παρήχθη στον ήδη υπαρκτό κώδικα. Όσον αφορά την ενσωμάτωση του παραγόμενου κώδικα η ομάδα θα πρέπει να εκτελεί τα ακόλουθα βήματα:

- Ακολουθούμε τα δύο βήματα που αναφέρθηκαν προηγουμένως. Μεταγλωττίζουμε και εκτελούμε τον ήδη υπάρχοντα κώδικα, ώστε να διαπιστώσουμε ότι η τελευταία ενημέρωση δουλεύει σωστά.
- Παίρνουμε την εξουσιοδότηση για νέα ενημέρωση του κώδικα και αυτήν τη φορά ελέγχουμε το δικό μας κώδικα, τον οποίο πρόκειται να ενσωματώσουμε.
- Πηγαίνουμε στο μηχάνημα όπου εκτελούνται διαδικασίες ενημέρωσης(Ενσωμάτωσης). Ενσωματώνουμε τον κώδικά μας.
- Δίνουμε εξουσιοδότηση στο επόμενο ζεύγος προγραμματιστών, που πρόκειται να προβεί στην επόμενη ενημέρωση.

Σε αυτό το σημείο να αναφέρουμε ότι αν προκύψει κάποιο σφάλμα στον κώδικα, τον οποίο πρόκειται να ενσωματώσουμε, θα πρέπει να το επιδιορθώσουμε πριν δώσουμε την εξουσιοδότηση για ενσωμάτωση κώδικα σε κάποιο άλλο ζεύγος προγραμματιστών. Ο πιο εύκολος τρόπος είναι να μην προβούμε σε αλλαγές στον ήδη υπάρχοντα κώδικα, εκτός και αν κανένα άλλο ζεύγος δεν επιθυμεί ενσωμάτωση, οπότε διορθώνουμε το πρόβλημα και ενσωματώνουμε τον κώδικα.

3.7.6 Κοινή δικαιοδοσία κώδικα(Collective code ownership)

Η κοινή δικαιοδοσία κώδικα είναι μια τεχνική του ακραίου προγραμματισμού, η οποία έχει ως απώτερο σκοπό τη διατήρηση της ποιότητας του κώδικα, σε όσο το δυνατό υψηλότερο επίπεδο, επιτρέποντας σε διαφορετικά άτομα την τροποποίηση του κώδικα. Έτσι, κάθε προγραμματιστής ο οποίος εντοπίζει κάποιο πρόβλημα που έχει να κάνει με τον κώδικα, το διορθώνει χωρίς να τον ενδιαφέρει από ποιον προήλθε το πρόβλημα αυτό. Αυτό που διαπιστώνουμε είναι ότι όλοι οι προγραμματιστές έχουν την ευθύνη για οποιοδήποτε πρόβλημα προκύψει μέσα στο κώδικα και όλοι θα έχουν τις ίδιες ευθύνες σε περίπτωση που κάτι συμβεί. Όμως, πέρα από αυτό, έχουμε αναφέρει ότι ο κώδικας συχνά υπόκειται σε αναδόμηση(refactoring), ώστε να επιτευχθεί η ποιότητα.

Με τη χρήση του ακραίου προγραμματισμού, αν κάποιος αναλάβει να υλοποιήσει μια λειτουργία, η οποία περιλαμβάνει κώδικα που δε γνωρίζουμε, ζητούμαι εθελοντικά να γίνουμε ζεύγος με αυτόν. Έτσι, καθώς εργαζόμαστε με τον άνθρωπο αυτό, μπορούμε εύκολα να ρωτήσουμε για θέματα που δε γνωρίζουμε και σαφώς να συλλέξουμε εμπειρία από το συγκεκριμένο προγραμματιστή. Σίγουρα, δεν είναι δυνατό να μπορέσουμε να κατανοήσουμε τι συμβαίνει σε κάθε γραμμή κώδικα, όμως σίγουρα μπορούμε να έχουμε μια γενική εικόνα του τι επιτυγχάνει ο συγκεκριμένος κώδικας. Επιπροσθέτως, κάτω από συγκεκριμένες συνθήκες, η τεχνική της κοινής δικαιοδοσίας κώδικα

μπορεί να εύκολα να εφαρμοσθεί, καθώς όλοι οι προγραμματιστές θα έχουν επίγνωση επί του κώδικα και θα μπορούν να επωμιστούν τις ευθύνες, που προβλέπει η συγκεκριμένη τεχνική.

4 Μέθοδος Συνωστισμού (SCRUM)

Οι πρώτες αναφορές στον όρο αυτό έγιναν μέσω ενός άρθρου των Tacheuchi και Nonaka, με σκοπό την παρουσίαση μιας ευέλικτης προγραμματιστικής διαδικασίας, που είχε επινοηθεί έχοντας ως στόχο να προσδώσει ταχύτητα, υψηλό επίπεδο προσαρμογής και ευελιξία στη διαδικασία ανάπτυξης λογισμικού. Θα πρέπει να αναφέρουμε ότι η έννοια χρησιμοποιήθηκε αρχικά στο δημοφιλέστερο άθλημα των Η.Π.Α, το ράγκμπι.

Η μέθοδος του Scrum προβλέπει ένα εντελώς διαφορετικό μοτίβο οργάνωσης και διοίκησης έργου σε σχέση με τις υπόλοιπες μεθόδους. Πιο συγκεκριμένα, η μέθοδος αυτή επιδιώκει όσο το δυνατό μικρότερο χρονικό διάστημα κύκλου ανάπτυξης. Επίσης για τον εκάστοτε κύκλο ανάπτυξης, προβλέπεται η παράδοση τμημάτων κώδικα του συστήματος, που έχουν ήδη συμφωνηθεί πριν την έναρξη του. Ένα σημαντικό σημείο που θα πρέπει να αναφερθεί, είναι η καθημερινή συνεδρίαση των μελών της ομάδας ανάπτυξης, με σκοπό την επίτευξη όσο το δυνατόν καλύτερης συνεργασίας και συντονισμού των εργασιών τους.

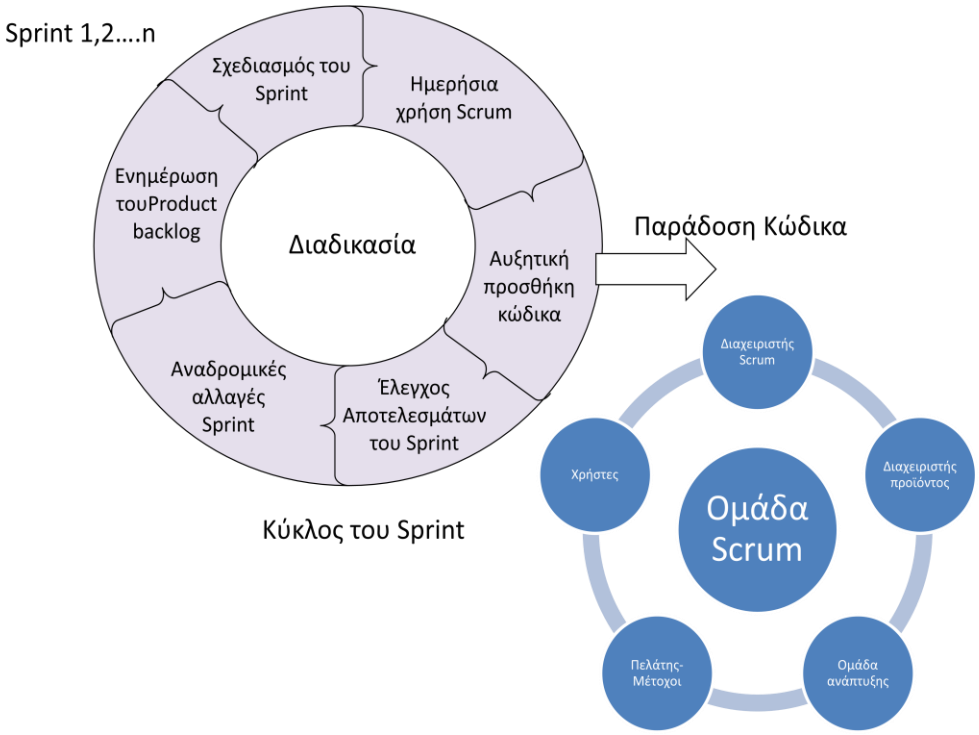
Αρχικά, η μέθοδος αυτή χρησιμοποιήθηκε με σκοπό την καλύτερη οργάνωση και διαχείριση της διαδικασίας ανάπτυξης λογισμικού, προσδίδοντας ευελιξία και αυξάνοντας την παραγωγικότητα της ομάδας ανάπτυξης. Επίσης, η ικανότητα προσαρμογής της υπό ανάπτυξη εφαρμογής, σύμφωνα με τις εκάστοτε ανάγκες που προκύπτουν, ήταν μια σημαντική πτυχή της συγκεκριμένης μεθόδου. Η εφαρμογή της μεθόδου αυτής επέφερε θεαματικά αποτελέσματα και γινόταν ολοένα και πιο διάσημη, λόγω της αποτελεσματικότητας της.

Μυστικό της επιτυχίας της, όπως έχει χαρακτηριστεί από διάσημες προσωπικότητες της ανάπτυξης λογισμικού όπως ο Schwaber και ο Beedle,

είναι ο τρόπος με τον οποίο προσεγγίζει την έννοια της διαδικασίας ανάπτυξης και κατ' επέκταση της διαχείρισης της διαδικασίας αυτής. Πιο συγκεκριμένα, η μέθοδος αυτή δίνει μεγαλύτερη έμφαση στο πώς θα πρέπει η ομάδα ανάπτυξης να συνεργαστεί, ώστε να μπορέσει να επιτύχει το μέγιστο βαθμό αποτελεσματικότητας σε ένα διαρκώς μεταβαλλόμενο περιβάλλον, που συνεχώς προκύπτουν προβλήματα, όπως αλλαγές στα πλάνα ανάπτυξης, ενσωμάτωση νέων απαιτήσεων από την πλευρά του πελάτη και δυσκολίες που αφορούν την πάντα απρόβλεπτη διαδικασία ανάπτυξης κώδικα.

Η Μεθοδολογία Scrum διαχωρίζεται σε 3 σημαντικές φάσεις οι οποίες είναι οι εξής:

1. Αρχική Διερεύνηση
2. Σχεδιασμός
3. Ολοκλήρωση



4.1 Φάση αρχικής διερεύνησης

Αυτή η αρχική φάση περιλαμβάνει δύο υπό-φάσεις. Τη φάση της ανάλυσης και τη φάση του υψηλού επιπέδου σχεδιασμού.

4.1.1 Φάση Ανάλυσης

Η φάση της ανάλυσης περιλαμβάνει τον καθορισμό και την πιστοποίηση του συστήματος που πρόκειται να αναπτυχθεί. Πιο συγκεκριμένα, περιλαμβάνει τον καθορισμό των προδιαγραφών του συστήματος, καθώς και των απαιτήσεων του χρήστη από αυτό. Έτσι, σημαντικό ρόλο στη διαδικασία αυτή διαδραματίζει ο πελάτης, ο οποίος καλείται πλέον να ορίσει με μεγάλη λεπτομέρεια το τι θέλει να κάνει το σύστημα, ώστε η ομάδα ανάπτυξης να μπορεί να ορίσει το δικό της πλάνο ανάπτυξης με όσο το δυνατό μικρότερη πιθανότητα εμφάνισης προβλημάτων. Στη συνέχεια, η ομάδα ανάπτυξης έχει το δύσκολο έργο του καθορισμού προτεραιοτήτων και του βαθμού σημαντικότητας των επιμέρους λειτουργιών του συστήματος που πρόκειται να αναπτυχθεί. Οι λειτουργίες αναλύονται μία προς μία και εξετάζονται εξονυχιστικά, ώστε να διαπιστωθεί ο βαθμός δυσκολίας υλοποίησής τους, όπως και η τυχόν ενσωμάτωση επιπλέον τεχνογνωσίας στη διαδικασία ανάπτυξης. Έτσι, αν η ενσωμάτωση τεχνογνωσίας κρίνεται απαραίτητη, γίνεται αντιληπτό ότι το γεγονός αυτό προσδίδει επιπλέον δυσκολία στην ομάδα ανάπτυξης, καθώς πλέον η ομάδα θα πρέπει να σπαταλήσει ένα μεγάλο χρονικό διάστημα στην εκπαίδευση κάποιων μελών της, ώστε να μπορέσει να αντεπεξέλθει πλήρως στις ανάγκες του έργου.

4.1.2 Φάση Υψηλού Επιπέδου Σχεδιασμού

Κατά τη διάρκεια αυτής της φάσης, επιτελείται η ανάπτυξη του συστήματος συμπεριλαμβανομένου της αρχιτεκτονικής του με βάση τις

αρχικές προδιαγραφές του συστήματος, οι οποίες έχουν καθοριστεί στην προηγούμενη φάση.

4.2 Φάση Ανάπτυξης

Η φάση αυτή αποτελεί την κρισιμότερη της μεθόδου Scrum και είναι ο λόγος που αυτή αποκαλείται ευέλικτη προγραμματιστική μέθοδος. Η εν λόγω φάση απαιτεί ιδιαίτερη προσοχή από τα μέλη της ομάδας, διότι είναι πολύ απρόβλεπτη. Ο λόγος του χαρακτηρισμού αυτού, έγκειται στο γεγονός ότι η επιτυχία της εξαρτάται κατά ένα μεγάλο βαθμό σε παραμέτρους όπως: χρόνος, ποιότητα, απαιτήσεις, πηγές, ενσωμάτωση επιπλέον τεχνογνωσίας κ.α., που ίσως διαφοροποιηθούν κατά τη διάρκεια της ανάπτυξης του συστήματος. Έτσι, η ομάδα θα πρέπει να προβεί στους απαραίτητους ελέγχους, όπως και να λάβει τα κατάλληλα μέτρα, ώστε να μπορέσει να αποφύγει, όσο αυτό είναι εφικτό, τους κινδύνους που παραμονεύουν, όπως επίσης να αντιμετωπίσει προβλήματα που ίσως προκύψουν όσο το δυνατόν ανώδυνα.

Στη φάση της σχεδίασης, το σύστημα αναπτύσσεται σε επαναλήψεις (Sprints). Οι επαναλήψεις αποτελούνται από επαναληπτικούς κύκλους, με απώτατο σκοπό την αύξηση της λειτουργικότητας του συστήματος στα υψηλότερα εφικτά επίπεδα. Η κάθε επανάληψη περιλαμβάνει όλες τις παραδοσιακές φάσεις ανάπτυξης συστημάτων, όπως για παράδειγμα τη φάση της ανάλυσης των απαιτήσεων, τη φάση σχεδιασμού, τη φάση παράδοσης κ.α.. Η αρχιτεκτονική της σχεδίασης του συστήματος αναπτύσσεται σταδιακά κατά την επαναληπτική σχεδίαση. Σε αυτό το σημείο θα πρέπει να τονίσουμε ότι η κάθε επανάληψη διαρκεί ένα συγκεκριμένο χρονικό διάστημα και σίγουρα όχι περισσότερο από ένα μήνα, ενώ το ελάχιστο χρονικό διάστημα είναι μία βδομάδα. Μια ακόμη σημαντική πληροφορία είναι ότι ένα έργο ανάπτυξης λογισμικού δεν περιέχει παραπάνω από οκτώ, αλλά ούτε και λιγότερες από τρεις επαναλήψεις, έως ότου είναι έτοιμο για παράδοση στον πελάτη.

4.3 Φάση Ολοκλήρωσης

Κατά τη φάση της ολοκλήρωσης εκτελούνται όλες εκείνες οι διαδικασίες που απαιτούνται για την περάτωση του έργου και την τελική παράδοσή του στον πελάτη. Η ομάδα ανάπτυξης προχωρά στη φάση αυτή, αφού προηγουμένως έχει συμφωνήσει με τον πελάτη ότι όλες απαιτήσεις οι οποίες είχαν αρχικά ορισθεί, έχουν ικανοποιηθεί πλήρως και το σύστημα είναι έτοιμο προς χρήση. Έτσι, ο πελάτης δεσμεύεται ότι δεν πρόκειται να προκύψουν εκ νέου απαιτήσεις από τη δική του πλευρά και ότι το σύστημα είναι έτοιμο για παράδοση. Στη συνέχεια, η ομάδα ανάπτυξης κινεί τις διαδικασίες, ώστε το σύστημα να παραδοθεί στον πελάτη.

4.4 Ρόλοι και ευθύνες

Υπάρχουν έξι αναγνωρίσιμοι ρόλοι στην ευέλικτη μεθοδολογία SCRUM, καθένας από τους οποίους επιτελεί κάποιες συγκεκριμένες εργασίες και εξυπηρετεί διαφορετικούς σκοπούς. Οι ρόλοι αυτοί είναι οι εξής: ο διαχειριστής του SCRUM, ο διαχειριστής προϊόντος, η ομάδα SCRUM, ο πελάτης, ο χρήστης και η διαχείριση. Στη συνέχεια, θα παρουσιάσουμε αυτούς τους ρόλους, σύμφωνα με τους ορισμούς, που έχουν προσδώσει σε αυτούς οι Schwaber και Beedle.

4.4.1 Ο διαχειριστής του SCRUM (Scrum master)

Ο διαχειριστής του Scrum είναι ένας νέος αλλά σημαντικός ρόλος που εισήγαγε η μεθοδολογία Scrum. Ο διαχειριστής του SCRUM είναι υπεύθυνος για τη διασφάλιση της ορθής ανάπτυξης του εκάστοτε έργου, σύμφωνα πάντα

με τις πρακτικές, τις αξίες και τους κανόνες που είχαν καθοριστεί κατά το σχεδιασμό του έργου. Για να επιτευχθεί αυτό, αλληλεπιδρά τόσο με την ομάδα ανάπτυξης, όσο και με τον πελάτη κατά τη διάρκεια ανάπτυξης του έργου. Είναι επίσης, υπεύθυνος να διασφαλίσει ότι, τυχόν εμπόδια κατά τη διάρκεια του έργου θα εξαλειφθούν, ώστε η ομάδα ανάπτυξης να διατηρήσει όσο το δυνατόν υψηλότερα το επίπεδο παραγωγικότητάς της.

4.4.2 Διαχειριστής Προϊόντος (Product owner)

Ο διαχειριστής προϊόντος είναι ο επίσημα υπεύθυνος για το έργο, τη διαχείριση, τον έλεγχο και την εμφάνιση της λίστας των backlog, που αφορούν το έργο. Ο διαχειριστής προϊόντος επιλέγεται από το διαχειριστή του Scrum, τον πελάτη και τη διαχείριση. Είναι αυτός που παίρνει τις τελικές αποφάσεις για διαδικασίες που σχετίζονται με το backlog του έργου, συμμετέχει στην εκτίμηση της προσπάθειας ανάπτυξης στοιχείων του backlog και συμβάλει, όσο είναι δυνατό, στην ανάπτυξή τους παρέχοντας σημαντικές λύσεις.

4.4.3 Ομάδα ανάπτυξης Scrum (Scrum team)

Η ομάδα Scrum είναι η ομάδα ανάπτυξης του έργου, η οποία έχει την εξουσιοδότηση να αποφασίσει για τις κατάλληλες δράσεις που πρέπει να επιτελέσει, ώστε να αναπτυχθεί το εκάστοτε έργο. Η ομάδα θα πρέπει να είναι σε θέση να οργανώνει αποτελεσματικά το εσωτερικό της, όπως και τις εργασίες της, με σκοπό την επίτευξη των καθορισμένων στόχων για κάθε επανάληψη (Sprints). Η ομάδα ανάπτυξης για παράδειγμα, εμπλέκεται σε διαδικασίες, όπως στην εκτίμηση του χρόνου για τη δημιουργία μιας επανάληψης για τα κομμάτια που έχουν καθυστερήσει, αλλά και στην επανεξέταση της λίστας καθυστερήσεων του έργου. Επίσης, η ομάδα

ανάπτυξης συμβάλει στην ανακάλυψη των εμποδίων, που θα πρέπει να εξαλειφθούν από το έργο, ώστε η διαδικασία ανάπτυξης να συνεχιστεί ομαλά.

4.4.4 Πελάτης

Οι πελάτες συμμετέχουν ποικιλοτρόπως κατά την ανάπτυξη του λογισμικού και αυτό φαίνεται άλλωστε και στις προϋποθέσεις των ευέλικτων προγραμματιστικών λειτουργιών. Η σημαντικότερη, ίσως, συμβολή τους στο έργο, αφορά τον έλεγχο της καλής λειτουργίας του λογισμικού. Ο πελάτης είναι συνήθως και ο χρήστης του λογισμικού, οπότε εκτός από την γνώση του τομέα, μπορεί άμεσα να αναγνωρίσει το παραγόμενο αποτέλεσμα και να ζητήσει την τροποποίησή του, όποτε χρειασθεί. Επίσης, είναι λογικό οι προγραμματιστές να μη γνωρίζουν τις πτυχές του τομέα στον οποίο θα εφαρμοσθεί το παραγόμενο λογισμικό, οπότε ο πελάτης απαντάει άμεσα στις ερωτήσεις της ομάδας, αυξάνοντας έτσι την παραγωγικότητα. Τέλος, μην ξεχνάμε πως ο πελάτης καθορίζει τη σημαντικότητα της κάθε πτυχής του προϊόντος, και κατ' επέκταση τη σειρά δημιουργίας τους.

4.4.5 Διαχείριση-Μέτοχοι

Η διαχείριση είναι επιφορτισμένη με το βάρος της τελικής απόφασης που πρέπει να παρθεί, σύμφωνα με τα στάνταρ και τις συμβάσεις που θα πρέπει να ακολουθεί το εκάστοτε έργο. Η διαχείριση, συμμετέχει στη διαδικασία του καθορισμού των στόχων, όπως και των απαιτήσεων του έργου. Για παράδειγμα, η διαχείριση εμπλέκεται σε διαδικασίες που έχουν να κάνουν με την επιλογή του διαχειριστή προϊόντος και τη μείωση των καθυστερήσεων, με τη βοήθεια του διαχειριστή του Scrum.

4.5 Πρακτικές

Η μεθοδολογία Scrum δεν απαιτεί ούτε παρέχει κάποια συγκεκριμένη μέθοδο ανάπτυξης λογισμικού. Αντ' αυτού, απαιτεί ορισμένες πρακτικές διαχείρισης και μερικά εργαλεία στις διάφορες φάσεις της, ώστε να αποφευχθεί το χάος, που ίσως δημιουργηθεί από κάθε τι απρόβλεπτο, όπως και από την πολυπλοκότητα.

Στη συνέχεια, αναλύουμε κάποια από τα χαρακτηριστικά της μεθόδου Scrum, σύμφωνα πάντα με τους Schwaber and Beedle.

4.5.1 Product backlog

Το backlog του εκάστοτε έργου ορίζει οτιδήποτε είναι απαραίτητο στον τελικό προϊών, σύμφωνα με την τρέχουσα τεχνογνωσία. Έτσι, το product backlog ορίζει τη δουλειά που πρέπει να γίνει στο έργο. Περιλαμβάνει ένα διαρκώς ενημερωμένο κατάλογο προτεραιοτήτων, που αφορούν ένα σύνολο εργασιών, όπως επίσης και τις τεχνικές απαιτήσεις του εκάστοτε συστήματος που αναπτύσσεται. Μπορεί να περιλαμβάνει, για παράδειγμα, λειτουργίες, επιδιορθώσεις σφαλμάτων, ελαττώματα, ζητούμενες βελτιώσεις όπως και τεχνολογικές αναβαθμίσεις. Οι περισσότεροι ρόλοι της μεθοδολογίας Scrum συμβάλουν στη δημιουργία του backlog, όπως οι πελάτες, η ομάδα ανάπτυξης, μάρκετινγκ και πωλήσεις, η διαχείριση και η υποστήριξη πελατών. Ο ιδιοκτήτης του προϊόντος είναι υπεύθυνος για τη διατήρηση του backlog.

4.5.2 Εκτίμηση προσπάθειας (Effort estimation)

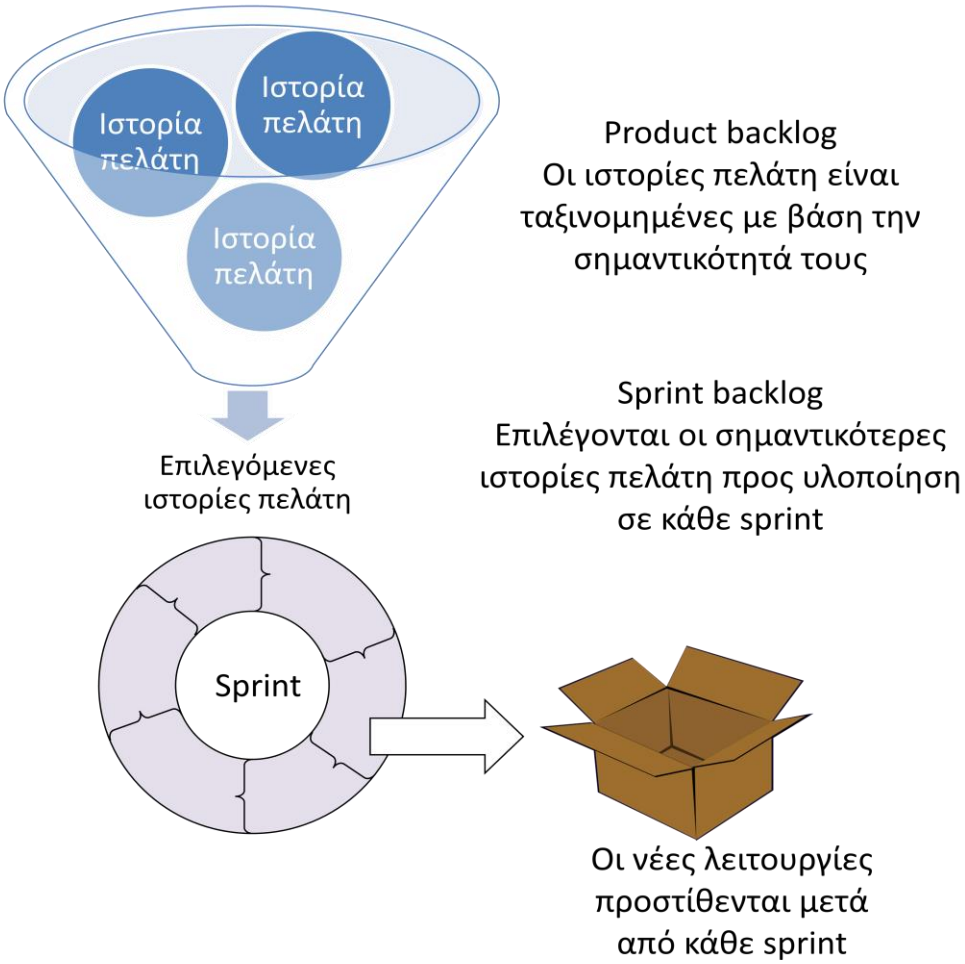
Η εκτίμηση της προσπάθειας είναι μια επαναληπτική διαδικασία, κατά την οποία τα στοιχεία του backlog εκτιμώνται με μεγαλύτερη ακρίβεια, καθώς

στο στάδιο αυτό συλλέγονται περισσότερες πληροφορίες για το καθένα από αυτά. Ο διαχειριστής προϊόντος σε συνεργασία με την ομάδα ανάπτυξης είναι υπεύθυνοι για τη διαδικασία αυτή.

4.5.3 Επανάληψη (Sprint)

Είναι η διαδικασία προσαρμογής στο συνεχώς μεταβαλλόμενο αναπτυξιακό περιβάλλον, που περιλαμβάνει παράγοντες όπως οι απαιτήσεις, ο χρόνος, οι πηγές, η τεχνογνωσία και η τεχνολογία κ.α. Η ομάδα ανάπτυξης οργανώνει τις εργασίες της, έτσι ώστε να παράγει μια νέα εκτελέσιμη και προσαυξητική εφαρμογή σε ένα χρονικό διάστημα, περίπου τριάντα ημερολογιακών ημερών. Οι διαδικασίες, ή αλλιώς τα εργαλεία, που χρησιμοποιεί η ομάδα ανάπτυξης για την επίτευξη αυτού του στόχου είναι:

- Οι συνεδριάσεις για το σχεδιασμό των επαναλήψεων (Sprint planning meetings)
- Sprint backlog
- Οι καθημερινές συνεδριάσεις Scrum (Daily Scrum meetings).



4.5.4 Συνεδριάσεις για το σχεδιασμό των επαναλήψεων (Sprint planning meetings)

Μια συνεδρίαση για το σχεδιασμό μιας επανάληψης αποτελείται από δύο φάσεις και οργανώνεται από το διαχειριστή του Scrum διαδικασίας. Οι πελάτες, οι χρήστες, η διαχείριση, ο διαχειριστής του προϊόντος, όπως και η ομάδα ανάπτυξης συμμετέχουν στην πρώτη φάση της συνεδρίασης, κατά την οποία καθορίζονται οι στόχοι και η λειτουργικότητα της επόμενης επανάληψης. Στη δεύτερη φάση της συνεδρίασης, συμμετέχουν μόνο ο διαχειριστής του Scrum και η ομάδα ανάπτυξης. Οι δύο πλευρές εστιάζουν στο πώς ακριβώς θα αναπτυχθεί το έργο, κατά τη διάρκεια της επανάληψης.

4.5.5 Sprint Backlog

Το Sprint backlog αποτελεί το σημείο εκκίνησης όλων των επαναλήψεων. Είναι μια λίστα των στοιχείων του backlog που έχουν επιλεχθεί ώστε να υλοποιηθούν στην επόμενη επανάληψη. Τα στοιχεία αυτά επιλέγονται από την ομάδα ανάπτυξης σε συνεργασία με το διαχειριστή του Scrum και το διαχειριστή του προϊόντος. Η επιλογή αυτή γίνεται κατά τη διάρκεια της συνεδρίασης για το σχεδιασμό της επανάληψης, με βάση την προτεραιότητα του κάθε στοιχείου και τους στόχους που έχουν καθοριστεί για την επανάληψη. Σε αντίθεση με το backlog του έργου, το Sprint backlog είναι σταθερό μέχρι την ολοκλήρωση της συγκεκριμένης επανάληψης. Όταν όλα τα στοιχεία του Sprint backlog υλοποιηθούν, τότε μια νέα έκδοση του συστήματος παραδίδεται στον πελάτη.

4.5.6 Καθημερινή συνεδρίαση Scrum

Οι καθημερινές συνεδριάσεις Scrum οργανώνονται, ώστε να παρακολουθείται συνεχώς η ομάδα ανάπτυξης και πιο συγκεκριμένα η πορεία και η απόδοση της. Επίσης, αυτές οι συνεδριάσεις μπορούν να θεωρηθούν και ως συνεδριάσεις σχεδιασμού, οι οποίες ελέγχουν τι έχει υλοποιηθεί στο χρονικό διάστημα που μεσολάβησε από την προηγούμενη συνεδρίαση και τι θα πρέπει να υλοποιηθεί στο διάστημα, μέχρι την επόμενη συνεδρίαση. Επίσης, προβλήματα και άλλα απρόβλεπτα θέματα που ίσως προκύψουν, συζητιούνται και αντιμετωπίζονται εγκαίρως σε αυτή τη μικρή περίπου δεκαπεντάλεπτη συνεδρίαση, που λαμβάνει χώρα καθημερινά. Τυχόν ελλείψεις ή εμπόδια στη διαδικασία ανάπτυξης εντοπίζονται και εξαλείφονται συμβάλλοντας έτσι στην ορθή και επιτυχημένη ανάπτυξη του συστήματος. Ο διαχειριστής του Scrum διεξάγει τη συγκεκριμένη συνεδρίαση. Εκτός από την

ομάδα ανάπτυξης και η διαχείριση μπορεί να συμμετέχει στην εν λόγω συνεδρίαση.

4.5.7 Συνεδρίαση για την επανεξέταση της επανάληψης (Sprint Review Meeting)

Στην τελευταία μέρα της επανάληψης, ο διαχειριστής του Scrum και η ομάδα ανάπτυξης παρουσιάζουν τα αποτελέσματα στη διαχείριση, στους πελάτες, στους χρήστες και το διαχειριστή του προϊόντος σε μια ανεπίσημη συνεδρίαση. Οι συμμετέχοντες αξιολογούν την ανάπτυξη του έργου και λαμβάνουν αποφάσεις που αφορούν τις επόμενες λειτουργίες που θα πρέπει να υλοποιηθούν. Αυτή η συγκεκριμένη συνεδρίαση, ίσως να επιφέρει νέα στοιχεία στο backlog του έργου, όπως επίσης μπορεί να σηματοδοτήσει την αλλαγή της κατεύθυνσης του.

5 Ιστορική Ανασκόπηση Επιχειρησιακών Συστημάτων

Πρόγονος των ΕΣ είναι τα συστήματα προγραμματισμού απαιτήσεων Υλικών ή MRP (Materials Requirement Planning) και τα MRP II (Manufacturing Resource Planning), τα οποία χρησιμοποιούσαν οι βιομηχανικές επιχειρήσεις κυρίως για τον αποτελεσματικό προγραμματισμό της παραγωγής και των διαδικασιών προμηθειών, σύμφωνα με τη ζήτηση που προέβλεπαν ότι θα είχαν τα προϊόντα τους, και λαμβάνοντας υπ΄ όψιν τα αποθέματά τους. Τα συστήματα ERP (EntERPrise Resource Planning) αναφέρονται στο σχεδιασμό και την αξιοποίηση όλων των πόρων(ή καλύτερα όλων των διαδικασιών) της επιχείρησης και όχι μόνο της παραγωγικής διαδικασίας/λειτουργίας εφοδιασμού. Εν αντιθέσει με τα συστήματα MRP/MRPII, τα οποία ήταν ειδικά κατασκευασμένα για το συγκεκριμένο υπολογιστικό σύστημα που χρησιμοποιούσε η κάθε επιχείρηση, τα

συστήματα ERP βασίζονται στην αρχιτεκτονική client/server και είναι ανεξάρτητα της χρησιμοποιούμενης υπολογιστικής πλατφόρμας. Επομένως, τα συστήματα MRP/MRPII και ERP διαφέρουν ως προς την αρχιτεκτονική τους.

Παρά το γεγονός ότι ο πρόγονος των ERP ήταν τα συστήματα προγραμματισμού της παραγωγής, σήμερα τα ERP έχουν σαφώς δομηθεί με βάση τη χρηματοοικονομική λειτουργία και τις αρχές του ελέγχου και προγραμματισμού της επιχειρηματικής δράσης, σύμφωνα με τη λογιστική/χρηματοοικονομική θεωρία (π.χ. λογιστική οντότητα επιχειρηματικής μονάδας, λογιστικό σχέδιο, προϋπολογισμοί, κέντρα κόστους κ.λπ.). Στο SAP R/3, π.χ. , όλη η πληροφόρηση του συστήματος αναφέρεται στον κωδικό εταιρίας, ο οποίος με τη σειρά του συνδέεται με το επικρατές νομικό, φορολογικό και λογιστικό σύστημα της χώρας, στην οποία είναι εγκατεστημένο το σύστημα. Το χρηματοοικονομικό module αυτών των συστημάτων είναι ο συνδετικός κρίκος όλων των άλλων modules.

Στην πιο πολυσύνθετη μορφή τους τα συστήματα ERP επεκτείνονται πέρα από τα όρια μιας συγκεκριμένης επιχείρησης και επικοινωνούν με παρόμοια συστήματα άλλων συνδεδεμένων επιχειρήσεων, ώστε να καθίσταται αποτελεσματικότερη η ροή των πληροφοριών. Επιδέχονται δε επιπρόσθετες εφαρμογές και συστήματα, τα οποία αυξάνουν κατά πολύ τη λειτουργικότητα τους. Τέτοια συστήματα είναι, π.χ., τα παρακάτω:

- Ηλεκτρονικού Εμπορίου (e-commerce)
- Διαχείρισης Πελατειακών Σχέσεων (CRM-Customer Relationship Management)
- Διαχειριστής Εφοδιαστικής Αλυσίδας (SCM-Supply Chain Management)
- Συστήματα Υποστήριξης Αποφάσεων (DSS-Decision Support Systems)

6 Ορισμός και Χρησιμότητα των Επιχειρησιακών Συστημάτων

Ένα σύστημα επιχειρησιακών πόρων είναι ένα ολοκληρωμένο σύνολο παραμετροποιήσιμων, και στενά συνεργαζόμενων εφαρμογών πραγματικού χρόνου βασισμένων στην υπολογιστική αρχιτεκτονική πελάτη/εξυπηρετητή, οι οποίες διαμοιράζονται μια κοινή βάση δεδομένων και υποστηρίζουν βασικές επιχειρηματικές, παραγωγικές και διοικητικές λειτουργίες, όπως είναι οι πωλήσεις, η παραγωγή, ο εφοδιασμός, η λογιστική, η κοστολόγηση και διοίκηση ανθρωπίνων πόρων.

Το βασικό χαρακτηριστικό των ΕΣ υποδηλώνεται από τον όρο WIDE ENTERPRISE SYSTEMS με τον οποίο επίσης είναι γνωστά. Ο χαρακτηρισμός Wide υποδηλώνει ότι τα ΕΣ αναφέρονται σε ολόκληρη την επιχείρηση, επιδιώκουν δηλαδή να τυποποιήσουν όλες τις επιχειρησιακές διαδικασίες της, οι οποίες διαχέονται συνήθως σε περισσότερες από μία λειτουργίες (π.χ. πωλήσεις, παραγωγή και διάθεση προϊόντος), μέσω της χρησιμοποίησης ενός μοναδικού λογισμικού προγράμματος, βασισμένου στο υπολογιστικό αρχιτεκτονικό υπόδειγμα client/server.

6.1 EntERPrise Resource Planning

Το ERP είναι ένα λογισμικό πακέτο μορφής ολοκληρωμένου πληροφοριακού συστήματος, σχεδιασμού και υποστήριξης των επιχειρησιακών και διοικητικών διαδικασιών μιας επιχείρησης που αφορούν κυρίως:

- Πωλήσεις-Διανομές
- Παραγωγή

- Εφοδιασμό
- Χρηματοοικονομικά

Το ERP αντικαθιστά τα ξεχωριστά, αυτόνομα υπολογιστικά συστήματα, που είχαν δομηθεί κυρίως με βάση τις λειτουργίες της επιχείρησης (όπως π.χ. χρηματοοικονομικά, αποθήκευση, πωλήσεις, παραγωγή), με ένα μόνο ενοποιημένο ή ολοκληρωμένο σύστημα. Το σύστημα αυτό αποτελείται από επιμέρους ενότητες, αλλά αυτές είναι στενά συνδεδεμένες μεταξύ τους και κυρίως χρησιμοποιούν μια κοινή βάση δεδομένων, η οποία ενημερώνεται σε πραγματικό χρόνο επιτρέποντας στο χρήστη του συστήματος να έχει μια συνολική και άμεση εικόνα για όλη την επιχείρηση.

Αυτό καθιστά σαφέστερο, αν αναλογισθούμε για παράδειγμα την καθημερινή, εξαιρετικά συνηθισμένη και απλή διαδικασία λήψης και εκτέλεσης μιας παραγγελίας πελάτη. Σε ένα μη ενοποιημένο υπολογιστικό σύστημα, η εκτέλεση της παραγγελίας θα βασιζόταν κυρίως σε μη αυτοματοποιημένες ενέργειες, όπου ο παράγοντας άνθρωπος και οι χειρονακτικές ενέργειες θα

έπαιζαν σημαντικό ρόλο στην εξυπηρέτηση και κατά συνέπεια στην ικανοποίηση του πελάτη. Το τμήμα πωλήσεων θα αγνοούσε σε πιο στάδιο θα βρισκόταν η εκτέλεση της παραγγελίας, αν δεν ενημερωνόταν με κάποιο τρόπο από τους υπευθύνους της αποθήκης, και φυσικά η άγνοια αυτή θα επικρατούσε τόσο στο λογιστήριο όσο και σε στα άλλα τμήματα της εταιρίας. Με ένα σύστημα ERP, η διαχείριση της διαδικασίας και η ποιότητα των παρεχόμενων υπηρεσιών θα ήταν εξαιρετικά βελτιωμένη. Το σύστημα θα ήλεγχε (μεταξύ άλλων) τα εξής:

- Πιστωτικό όριο πελάτη
- Ύψος αποθέματος αιτουμένων αγαθών
- Σημεία φόρτωσης και καταγραφής της διαδρομής των αγαθών

Όλοι οι χρήστες που θα είχαν τη σχετική εξουσιοδότηση, ανεξαρτήτως του τμήματος στο οποίο θα εργάζονταν, θα είχαν τη δυνατότητα να ελέγξουν άμεσα την κατάσταση της παραγγελίας και να ενημερώσουν αξιόπιστα και έγκαιρα τον πελάτη, σε μια ενδεχόμενη αίτησή του.

Ένα δεύτερο παράδειγμα για την κατανόηση της έννοιας αλλά και της χρησιμότητας των ΕΣ, αποτελεί η εισαγωγή ενός ατόμου σε ένα νοσοκομείο για νοσηλεία ή εξετάσεις ρουτίνας. Ένα κατακερματισμένο υπολογιστικό σύστημα, που πιθανώς τρέχει και σε διαφορετικές υπολογιστικές πλατφόρμες (π.χ. προσωπικοί υπολογιστές με Windows ή μεγάλοι υπολογιστές τύπου Mainframes με Unix) αποτελείται από ξεχωριστές, αυτόνομες, εφαρμογές λογισμικού. Μια εφαρμογή παρακολουθεί την εισαγωγή, μια άλλη το κοστολόγιο της περίθαλψης ή των εξετάσεων, μια άλλη, ξεχωριστή εφαρμογή, τη λογιστική αντιμετώπιση της όλης διαδικασίας και μια άλλη αυτόνομη εφαρμογή, την παρακολούθηση της κατανάλωσης φαρμάκων και υγειονομικού υλικού από τις αποθήκες του νοσοκομείου. Ενώ κάθε εφαρμογή παρέχει πιθανώς επαρκή πληροφόρηση για τη λειτουργία που παρακολουθεί, είναι δύσκολο να υπάρχει μια συνολική εικόνα για τον οργανισμό ή έστω για μια συγκεκριμένη επιχειρηματική διαδικασία, η οποία πολλές φορές, όπως

τονίσαμε και προηγουμένως, διαχέεται σε περισσότερες από μία λειτουργίες της επιχείρησης. Αντιθέτως, ένα πληροφοριακό σύστημα ERP θα αντιμετώπιζε την εισαγωγή του ατόμου ως ένα μόνο μέρος μιας ολοκληρωμένης διαδικασίας. Η διαδικασία αυτή θα περιελάμβανε την παρακολούθηση του εισαχθέντος, από τη στιγμή της εισαγωγής του στο νοσοκομείο μέχρι και το εξιτήριό του, διαμορφώνοντας μια πλήρη ηλεκτρονική εγγραφή ασθενούς. Η ERP θα έδινε σαφώς μια πιο ολοκληρωμένη εικόνα, τόσο για τα ιατρικά θέματα που αφορούν τον εισαχθέντα (ιατρικό ιστορικό, φαρμακευτική αγωγή, αποτελέσματα εξετάσεων κ.λπ.), όσο και για τα μη ιατρικά (συνολικό κόστος περίθαλψης, οικονομική συμμετοχή του εισαχθέντος, πληρωμές, κ.λπ.). Παραλλήλως, η τιμολόγηση των παρεχόμενων υπηρεσιών θα πραγματοποιείτο αυτόματα, όπως και οι απαιτούμενες ενημερώσεις στο λογιστικό/ χρηματοοικονομικό κύκλωμα. Μια εκτεταμένη εκδοχή του συστήματος ERP θα περιελάμβανε και ένα σύστημα διαχείρισης πελατειακών σχέσεων, που θα λειτουργούσε στο διαδίκτυο. Αυτό θα επέτρεπε στον εισαχθέντα, μετά την έξοδο του από το νοσοκομείο και από οποιοδήποτε σημείο βρισκόταν σε σύνδεση στο διαδίκτυο, να έχει πρόσβαση σε στοιχεία που τον αφορούν, να ενημερώνεται σχετικά με διαιτητική ή θεραπευτικές αγωγές και να διαχειρίζεται τις μελλοντικές συναντήσεις με τους ιατρούς του.

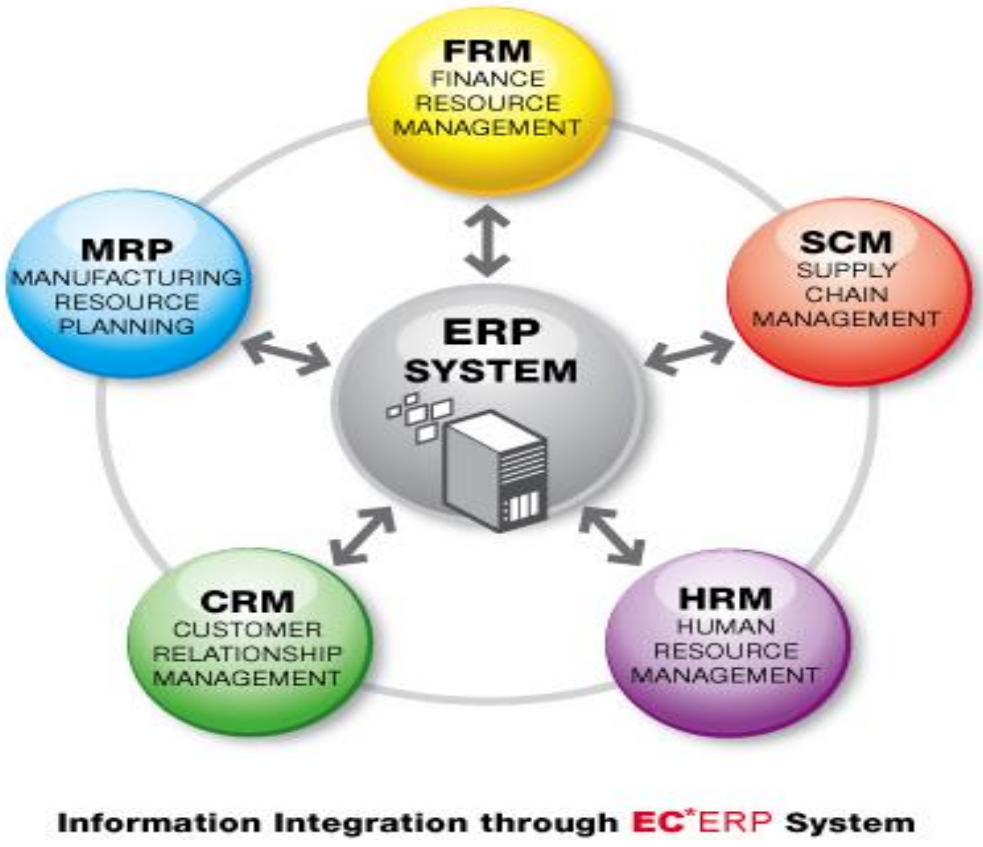
Στη συνέχεια αναφέρονται μερικά από τα σημαντικότερα οφέλη που παρέχουν τα συστήματα ERP στις επιχειρήσεις:

- Μείωση της σπατάλης και του χρόνου τιμολόγησης και επομένως καλύτερη διαχείριση χρηματικών διαθεσίμων (cash management) και εξοικονόμηση χρηματικών πόρων.

- Μείωση του κόστους παραγωγής και παροχής υπηρεσιών και αύξηση της παραγωγικότητας, λόγω αναδιοργάνωσης και εκλέπτυνσης των διαδικασιών και της ροής εργασιών(workflow).

- Μείωση του όγκου των αποθεμάτων των υλικών, και ημιτελών και των τελικών προϊόντων και επομένως των συνακόλουθων δαπανών διατήρησής τους.

- Βελτίωση της ποιότητας παραγομένων προϊόντων, λόγω λιγότερων λαθών στον προγραμματισμό της παραγωγής και των ενσωματωμένων δυνατοτήτων ποιοτικού ελέγχου του συστήματος.
- Μείωση του χρόνου που απαιτείται για να διατεθούν τα αγαθά στην αγορά (time to market)



6.2 Οι επιχειρήσεις που επωφελούνται από τέτοια συστήματα

Οι οικονομικές μονάδες κάποιου μεγέθους, οι οποίες έχουν κατακερματισμένα πληροφοριακά συστήματα, είναι πολύ πιθανό να χαρακτηρίζονται από καθημερινές, συνηθισμένες καθυστερήσεις στην

εκτέλεση των παραγγελιών τους προς τους πελάτες τους και να είναι ασυνεπείς στις υποχρεώσεις τους γενικότερα. Είναι επίσης, πολύ συχνό το να μη διαθέτουν την απαραίτητη πληροφόρηση, που αφορά τα οικονομικά αποτελέσματα, στοιχεία του κόστους παραγωγής και προσφοράς υπηρεσιών και γενικότερα την οικονομική πορεία προς την επίτευξη των τεθέντων στόχων. Διακοπή της ροής πληροφόρησης σε οποιοδήποτε επίπεδο ή παραγωγικό στάδιο, αποτελεί ένδειξη ανυπαρξίας ενός ολοκληρωμένου πληροφοριακού συστήματος, που να καλύπτει το σύνολο των δραστηριοτήτων μιας εταιρίας. Συνοπτικά, σημαντικές ενδείξεις ανάγκης προμήθειας συστημάτων ERP έχουν εκείνες οι επιχειρήσεις, οι οποίες:

- Δέχονται ολοένα και περισσότερα παράπονα από τους πελάτες τους.
- Διαμαρτύρονται οι εργαζόμενοι τους ότι τίποτε δε λειτουργεί σωστά σε αυτές.
- Τα διευθυντικά στελέχη μάταια ζητούν άμεση πληροφόρηση σχετικά με τα τρέχοντα χρηματοοικονομικά και κοστολογικά στοιχεία της.
- Έχουν υψηλότερο κόστος λειτουργίας σε σχέση με άλλες ομοειδείς επιχειρήσεις.
- Διακρίνονται από χαμηλή παραγωγικότητα και σπατάλες.
- Επιθυμούν να επεκταθούν γρήγορα σε νέες αγορές, με διαφορετικά λογιστικά συστήματα, γλώσσες και νομίσματα.
- Έχουν ανάγκη καλύτερου προγραμματισμού και διαχείρισης της παραγωγικής τους διαδικασίας.
- Έχουν υψηλό κόστος διατήρησης αποθεμάτων.
- Δεν έχουν κανένα τυποποιημένο τρόπο ή μέθοδο προγραμματισμού του ανθρώπινου δυναμικού τους, ενός εκ των σημαντικότερων πόρων των σύγχρονων επιχειρήσεων.
- Η διαχείριση των λογιστικών πάσχει, αλλά είναι άμεσης προτεραιότητας για τη διατήρηση της ανταγωνιστικότητας τους.

- Χαρακτηρίζονται από άναρχη λειτουργία, καθυστερήσεις στην εκπλήρωση των φορολογικών τους υποχρεώσεων και από πληθώρα εγγράφων, τα οποία κανείς δεν μπορεί να βρει όταν τα χρειάζεται.

- Και τέλος, αλλά όχι λιγότερο, οι περισσότερες μεγάλες Ελληνικές Επιχειρήσεις και όλοι ανεξαρτήτως οι οργανισμοί, τα Υπουργεία, τα Ν.Π.Δ.Δ. και οι επιχειρήσεις του δημοσίου τομέα στην Ελλάδα.

Ασφαλώς, δεν πρέπει σε καμιά περίπτωση να αγνοηθεί το υψηλό κόστος αγοράς, εγκατάστασης και λειτουργίας/ συντήρησης ενός συστήματος ERP. Η εγκατάσταση αυτών των συστημάτων είναι συνήθως αρκετά χρονοβόρα και δαπανηρή, απαιτεί μεγάλης διάρκειας εκπαίδευση των χρηστών από εξωτερικούς συμβούλους και η συμβολή τους στην επιχειρηματική επιτυχία δεν είναι πάντοτε εμφανής. Στην ουσία, η απόφαση για εγκατάσταση ενός συστήματος ERP σε μια εταιρία είναι μια στρατηγική απόφαση, μπορεί να οδηγήσει σε πλεονεκτήματα, αλλά και σε σημαντικές δυσλειτουργίες και επομένως πρέπει να ληφθεί μετά από σοβαρή μελέτη του ενδεχομένου κινδύνου και των εναλλακτικών λύσεων.

6.3 Αρχιτεκτονική Συστημάτων ERP

Τα πρώτα συστήματα ERP είχαν κατασκευασθεί για μεγάλους υπολογιστές τύπου mainframes, όπως για παράδειγμα το σύστημα SAP R/2 στην αρχή της δεκαετίας του 1990. Προσαρμόστηκαν όμως, πολύ νωρίς στην ευέλικτη αρχιτεκτονική Πελάτη/Εξυπηρετητή, η έννοια της οποίας υπήρχε από πολύ παλιά, αλλά μόνο κατά τη δεκαετία του 1990 επικράτησε, σχεδόν πλήρως, για τις επιχειρησιακές εφαρμογές με την κατάτμηση σύνθετων επιχειρησιακών συστημάτων σε συνεργαζόμενα τμήματα.

Το υπολογιστικό υπόδειγμα των mainframes κατένειμε το χρόνο επεξεργασίας στους χρήστες, οι οποίοι ήταν συνδεδεμένοι με αυτά, διαμέσου απλών τερματικών, χωρίς δυνατότητες επεξεργασίας. Με αυτόν τον τρόπο,

εκμεταλλεύονταν πολλοί χρήστες μια κοινή εφαρμογή, η οποία λειτουργούσε σε ένα κεντρικό υπολογιστή.

Με την επανάσταση των προσωπικών υπολογιστών, η ανάγκη για δικτύωση των χρηστών δημιούργησε το τοπικό δίκτυο, στο οποίο όμως κάθε χρήστης είχε τοπική δύναμη επεξεργασίας, διότι πλέον δεν χρησιμοποιούσε μια απλή οθόνη επικοινωνίας με τον κεντρικό υπολογιστή, αλλά διέθετε ένα υπολογιστή με επεξεργαστή, μνήμη και δυνατότητες αποθήκευσης δεδομένων. Στην αρχιτεκτονική προσωπικός υπολογιστής-τοπικό δίκτυο, η έννοια «κατανεμημένη επεξεργασία» σήμαινε κυρίως ότι ο κεντρικός υπολογιστής ήταν ένας εξυπηρετητής αρχείων, περιείχε δηλαδή τα κοινά αρχεία δεδομένων, αλλά η επεξεργασία και όλη η λογική του συστήματος κατανεμόταν στους προσωπικούς υπολογιστές του δικτύου. Με το παράδειγμα της υπολογιστικής αρχιτεκτονικής πελάτης/εξυπηρετητής, έγινε δυνατή η κατάτμηση σύνθετων επιχειρησιακών συστημάτων σε τμήματα, με αποτέλεσμα η επεξεργασία να κατανέμεται σε διαφορετικούς εξυπηρετητές και επομένως να καθίσταται περισσότερο αποτελεσματική. Διακρίνουμε κυρίως, δύο βασικά συστήματα πελάτη/εξυπηρετητή. Το σύστημα δύο διαζωμάτων (2-tier) και το σύστημα τριών διαζωμάτων (3-tier).

Στο αρχιτεκτονικό υπόδειγμα δύο διαζωμάτων, ο εξυπηρετητής αποτελεί έναν εξυπηρετητή βάσης δεδομένων (database server), ο οποίος διαχειρίζεται ένα σύστημα βάσεων δεδομένων, με τη λογική του συστήματος να κατανέμεται στους συνδεδεμένους με αυτόν Η/Υ, τους πελάτες (clients).

Πολλά συστήματα ERP βασίζονται στο αρχιτεκτονικό υπόδειγμα τριών διαζωμάτων (3-tier), το οποίο αποτελείται από τα εξής διαζώματα:

- Εξυπηρετητής Βάσης Δεδομένων (database server): Διαχειρίζεται τα δεδομένα της εφαρμογής (π.χ. λογιστικά στοιχεία, δεδομένα παραγωγής και πωλήσεων, στοιχεία πελατών και προμηθευτών, κ.λπ.), με τη χρησιμοποίηση κάποιας εφαρμογής διαχείρισης βάσης δεδομένων.

- Εξυπηρετητής εφαρμογών (application server): Διαχειρίζεται τη λογική του συστήματος, αποτελεί δηλαδή την εφαρμογή, το πρόγραμμα, το οποίο διαχειρίζεται τα επιχειρηματικά δεδομένα.
- Εξυπηρετητής Παρουσίασης (presentation server): Διαχειρίζεται την αλληλεπίδραση, την επικοινωνία των χρηστών με το σύστημα, αποτελεί δηλαδή τον πελάτη (client) του συστήματος.

7 Ευέλικτο ERP (Εισαγωγή)

Εφαρμόζοντας τα κοινώς αποδεκτά πρότυπα για τη διεκπεραίωση ενός έργου, μειώνεται σημαντικά το επίπεδο συντονισμού μεταξύ των εργαζομένων στην πορεία προς την ολοκλήρωση του. Οι εργαζόμενοι έχουν να αντιμετωπίσουν ένα μεγάλο όγκο εισερχόμενων πληροφοριών, καθώς το έργο εξελίσσεται και όπως συμπεραίνουμε αποτελεί επιτακτική ανάγκη η όσον το δυνατόν καλύτερη συνεργασία μεταξύ των μελών της ομάδας ανάπτυξης. Επίσης, είναι αναγκαία η χρήση μιας εφαρμογής, που θα αποτελέσει σύμμαχο στην προσπάθεια διαχείρισης αυτού του μεγάλου όγκου εισερχόμενων πληροφοριών. Το ERP παρέχει όλα τα μέσα για τη διαχείριση και το συντονισμό αυτών των πληροφοριών, ενσωματώνοντας επιχειρησιακές πληροφορίες και πρακτικές λειτουργίες.

Η διαχείριση ενός έργου ERP δεν είναι το ίδιο με τη διαχείριση ενός μεγάλου IT (Information Technology) έργου. Τα IT έργα δίνουν έμφαση στην απόσπαση των απαιτήσεων, το λεπτομερή σχεδιασμό και την εκτέλεση καθορισμένων διεργασιών, παρέχοντας μια πλήρη αναφορά της εργασιακής λειτουργικότητας. Παρ' όλα αυτά, η IT μεθοδολογία αντιμετωπίζει αρκετά προβλήματα στα μεγάλα έργα όπως τα ERP. Το περιβάλλον του ERP αντιμετωπίζει τις συνεχόμενες αλλαγές, που ενδεχομένως να απαιτούνται, όπως επίσης προσδίδει τη δυνατότητα επανεκτίμησης λειτουργιών και τεχνολογιών. Η μεθοδολογία διαχείρισης έργου, που χρησιμοποιείται σε μια ERP ανάπτυξη, θα πρέπει να παρέχει ικανότητα προσαρμογής και την

ευστροφία να υποστηρίξει αυτές τις καινοτόμες λειτουργίες και τεχνολογίες. Η χρήση των ευέλικτων μεθοδολογιών στο ERP παρέχει:

- Αυξημένη συμμετοχή από τους ενδιαφερόμενους. (Πελάτες)
- Στοιχειώδη και επαναληπτική παράδοση της πορείας ολοκλήρωσης του έργου.
- Μέγιστη αξιοποίηση των πόρων.

7.1 Το πρόβλημα που έχουμε να αντιμετωπίσουμε

Η έννοια των έργων COTS (Commercial, off-the-shelf) αποτελεί τη λύση σε πολλά επαγγελματικά προβλήματα. Η εφαρμογή των επιστημονικών αρχών διαχείρισης για τα έργα αυτά είναι κατανοητή. Η χρήση καινοτόμων στρατηγικών σε αυτό το περιβάλλον όμως είναι ακατάλληλη, όπως επίσης και αρκετά αναποτελεσματική, δεδομένου ότι δεν μπορούν να διευθύνουν τις ανερχόμενες και, μερικές φορές, χαοτικές συμπεριφορές της αγοράς, των ενδιαφερομένων και των πωλητών. Αυτή η εργασία περιγράφει μια μέθοδο, που συγκαταλέγεται σε αυτές των δομημένων μεθόδων ανάπτυξης ενός έργου, με την ευελιξία να παραχθεί μια νέα προσέγγιση στη διαχείριση των ERP έργων. Αυτή η ευέλικτη προσέγγιση απαιτεί εργαλεία ανάλυσης, ώστε να παρθούν κάποιες αμετάκλητες αποφάσεις για την αντιμετώπιση όλων των αβέβαιων σημείων στο περιβάλλον του ERP. Η συγκεκριμένη προσέγγιση, επίσης, παρέχει μεθόδους που βοηθούν στην επίτευξη συμφωνιών, όσον αφορά τις διαπροσωπικές συμφωνίες, συμφωνίες με τους ενδιαφερομένους, όπως επίσης και συμφωνίες σε θέματα που έχουν να κάνουν με επαγγελματικές διεργασίες, που ανακύπτουν συνεχώς στο ταχέως μεταβαλλόμενο περιβάλλον του ERP.

Οι ευέλικτες μέθοδοι παρέχουν τα μέσα για την παράδοση, όχι απλώς μιας υποτιθέμενης προόδου, αλλά μιας πραγματικής προόδου, γεγονός που

ικανοποιεί όλους τους συμμετέχοντες (αγοραστή, πωλητή, πάροχο υπηρεσιών), καθώς τους παρέχει μια πραγματική επιχειρηματική αξία.

7.2 Τι είναι ένα ERP έργο

Ο όρος διαχείριση επιχειρησιακών πόρων (EntERPrise resource planning), που επινοήθηκε στις αρχές του 1990, αντικατοπτρίζει μια εφαρμογή λογισμικού που ενσωματώνει πληροφορίες και διάφορες επαγγελματικές διαδικασίες μιας επιχείρησης, επιτρέποντας σε αυτή να τις διαμοιράζει σε όλο το εσωτερικό της. Ενώ το ERP χρησιμοποιούταν ως ένα σύστημα , που θα παρείχε τη δυνατότητα σε όλες τις επιχειρήσεις να κάνουν ένα καλύτερο προγραμματισμό, είχε παράλληλα επεκταθεί και σε “back-office” λειτουργίες, όπως η διαχείριση παραγγελιών, οικονομικών μεγεθών, περιουσιακών στοιχείων, πελατειακών σχέσεων και του ανθρώπινου δυναμικού. Αν θεωρούσαμε ένα ERP έργο ως μία μεγάλη IT ανάπτυξη, τότε θα συναντούσαμε αρκετές μη αποδεκτές καταστάσεις όπως:

- Δαπάνη 2 εκατομμυρίων δολαρίων ή 20 ή 200 για μια τεχνολογία με 50% έως 70% πιθανότητα μερικής ή πλήρους αποτυχίας της επένδυσης.
- Με σκοπό την απόσβεση της επένδυσης, θα πρέπει να διπλασιάσει την αρχική δαπάνη, ώστε να ολοκληρωθεί το έργο με επιτυχία.

7.3 Διαχείριση ενός ERP έργου

Η σύγχρονη διαχείριση ενός έργου επηρεάζεται σε μεγάλο βαθμό από την πεποίθηση, ότι μια λειτουργία διαχείρισης έργου μπορεί να βελτιωθεί με επιστημονικές μεθόδους. Έτσι, δημιουργείται ο μύθος σύμφωνα με τον οποίο:

- Σαφείς επενδυτικές ευκαιρίες με καθορισμένο σκοπό, αρχή, διάρκεια και τέλος μπορούν εύκολα να εντοπισθούν στο έργο.
- Χαμηλό κόστος για κάθε εγχείρημα, που αφορά το έργο και έχει να κάνει με λήψη μιας απόφασης σχετικά με την πορεία του έργου.
- Εφικτά, κατάλληλα και πλήρως αποδεκτά χαρακτηριστικά ενός έργου μπορούν εύκολα να προσδιοριστούν.
- Ακριβείς προβλέψεις, όσον αφορά τη διάρκεια του έργου, όπως επίσης και τις απαιτήσεις σε πόρους, είναι εφικτές εφόσον οι απαιτήσεις έχουν ορισθεί.
- Οι συνέπειες μιας λανθασμένης απόφασης, κατά τη διάρκεια του έργου, μπορούν να καθορισθούν εκ των προτέρων.
- Η κάθε αποτυχία οφείλεται στην έλλειψη κατάλληλων δεξιοτήτων και όχι στην ακατάλληλη αποδοχή μιας λύσης που εκ των προτέρων έπρεπε να χαρακτηρισθεί ως μη εφικτή.

Αυτή είναι μια επιστημονική όψη της διαχείρισης ενός έργου. Στο περιβάλλον του ERP αυτή η επιστημονική όψη αντικαθίσταται από μια πιο μοντέρνα σύμφωνα με την οποία υπάρχουν:

- Εξαιρετικά αβέβαια γεγονότα, που πρέπει να ληφθούν υπόψη κατά τη διάρκεια του καθορισμού των χαρακτηριστικών του έργου.
- Διαφωνίες όσον αφορά τις αξίες και τις προσδοκίες για κάθε έργο.
- Ανεπανόρθωτες συνέπειες αν παρθεί μια λάθος απόφαση στην πορεία του έργου.
- Αποτελέσματα που επηρεάζουν όλους τους ενδιαφερομένους.

Η χρήση των ευέλικτων μεθόδων σε τέτοιου είδους έργα, σε καμιά περίπτωση δεν αναιρεί τη χρησιμότητα των επιστημονικών μεθόδων, απλά αυτές θα πρέπει να χρησιμοποιούνται σε έργα, όπου το κόστος μιας λανθασμένης απόφασης θα είναι μικρό και οι συνέπειες μηδαμινές.

7.4 Τα έργα ERP είναι νέα εγχειρήματα

Οι ευέλικτες μέθοδοι, που χρησιμοποιούνται για τη διαχείριση έργων ERP, μπορούν να θεωρηθούν περισσότερο ως εγχειρήματα μιας καπιταλιστικής προσέγγισης, πάρα μιας προσέγγισης IT διαχείρισης. Αυτές οι μέθοδοι περιλαμβάνουν:

- Οργανωμένες επενδύσεις – το κεφάλαιο πρέπει να διατηρηθεί
- Διαχείριση του κίνδυνου ή του ρίσκου – όλοι οι συμμετέχοντες πρέπει να μοιράζονται το ρίσκο.
- Η σύνθεση των συμμετεχόντων αποτελεί σημαντικό παράγοντα για την επιτυχία του έργου.

7.5 Ο μετασχηματισμός μιας επιχείρησης

Ανασχεδιασμός διεργασίας: Πρόκειται για την αντικατάσταση κάποιων διεργασιών, οι οποίες έχουν αναπτυχθεί ιστορικά μέσα στην επιχείρηση με νέες και καινοτόμες διεργασίες, που ενσωματώνονται στα συστήματα ERP. Αν οι ανάγκες των επιχειρήσεων δεν καλύπτονται από το υπάρχον σύστημα ERP, τότε γεννιέται ο πειρασμός για αναπροσαρμογή του συστήματος. Εάν αυτό επιτευχθεί, τότε παράγεται ένα νέο σύστημα, απόγονος του ήδη υπάρχοντος, το οποίο πλέον καλύπτει τις πρόσθετες απαιτήσεις.

Στροφή προς τις επιχειρηματικές διαδικασίες με βάση τα Modules: Πρόκειται για τη διαμόρφωση της αρχιτεκτονικής μιας επιχείρησης και του λογισμικού της. Υπάρχει η τεχνική αρχιτεκτονική, η αρχιτεκτονική των δεδομένων, η αρχιτεκτονική των εφαρμογών και η αρχιτεκτονική της

επιχείρησης. Η ανάπτυξη των ERP επηρεάζει και τις τέσσερις αυτές αρχιτεκτονικές.

7.6 Η αρχιτεκτονική και το όφελός της

Μια προσέγγιση στην ευέλικτη ανάπτυξη συστημάτων ERP μας καθοδηγεί να ξεκινήσουμε από την αρχιτεκτονική του συστήματος. Αυτό επιφέρει πολλά και σημαντικά πλεονεκτήματα όπως:

- **Απλοποιούνται οι επιχειρηματικές διεργασίες**- Με την ανακάλυψη και εξάλειψη των πλεονασμών μέσα σε αυτές τις διεργασίες.
- **Η πολυπλοκότητα των δεδομένων του συστήματος μειώνεται**- Με τον εντοπισμό και την εξάλειψη των πλεονασμών στα δεδομένα και το λογισμικό.
- **Επιχείρηση**- Η ενοποίηση της επιχείρησης επιτυγχάνεται μέσω του αποτελεσματικού τρόπου διαμοιρασμού των δεδομένων στο εσωτερικό της, όπως επίσης και μέσω του εντοπισμού των σημείων, για την ανάπτυξη προτύπων που αφορούν τα διαμοιραζόμενα δεδομένα και διαδικασίες.
- **Προωθείται η ταχεία εξέλιξη στις νέες τεχνολογίες**- Απομονώνοντας δεδομένα από διεργασίες, που δημιουργούν και έχουν πρόσβαση σε αυτά τα δεδομένα.

Η αρχιτεκτονική συντίθεται από ένα σύνολο κανόνων, που ορίζουν πως μια ενοποιημένη και συνεκτική δομή από συνιστώσες και συνδέσεις, εναρμονίζονται και συνεργάζονται. Η μορφή, η λειτουργία, η ορθή αξιοποίηση των πόρων και των υλικών, η ανθρώπινη αλληλεπίδραση με τους πόρους αυτούς, η μακροβιότητα των αποφάσεων σχεδιασμού, όπως και η ευρωστία της προκύπτουσας οντότητας είναι όλα χαρακτηριστικά τα οποία συντελούν

στη σωστή ανάπτυξη τόσο ενός κτιρίου όσο και ενός επιτυχημένου συστήματος.

Η αρχιτεκτονική δε διευκρινίζει τις λεπτομέρειες για κάθε εφαρμογή. Ωστόσο, προβλέπει τη θέσπιση κατευθυντήριων γραμμών και όρων, που θα πρέπει να τηρούνται στις επιλογές της εφαρμογής. Οι όροι αυτοί είναι ιδιαίτερα σημαντικοί στον τομέα των ERP, δεδομένου ότι η ανάπτυξη ERP ενσωματώνει επεκτάσιμα χαρακτηριστικά που επιτρέπουν στην εκάστοτε ανάπτυξη να επεκτείνεται, όποτε αυτό επιβάλλεται για διάφορους λόγους.

Στον τομέα των COTS, η αρχιτεκτονική παρέχει την καθοδήγηση της ομάδας ανάπτυξης και κατευθύνει τη δημιουργικότητά της.

8 Ανάπτυξη συστήματος ERP

Τα έργα IT παραδοσιακά χρησιμοποιούν τυποποιημένες διαδικασίες διαχείρισης για την απόκτηση, την ανάπτυξη, την εγκατάσταση και λειτουργία του συστήματος. Αυτή η προσέγγιση οργανώνει τις εργασίες σε φάσεις οι οποίες καθορίζονται αρκετά νωρίς στο έργο και πριν την έναρξη της ανάπτυξης του. Οι ειδικοί της προσέγγισης αυτής τονίζουν ότι οι αλλαγές που λαμβάνουν χώρα στην αρχή του έργου είναι λιγότερο ακριβείς από αυτές που ενδεχομένως θα χρειαστούν αργότερα και κατά τη διάρκεια αυτού.

Στο παρελθόν, η προσέγγιση αυτή είχε ονομαστεί καταρράκτης. Η προσέγγιση του καταρράκτη περιέχει ορισμένες λανθασμένες πρακτικές, που επηρεάζουν αρνητικά την ανάπτυξη των έργων ERP.

Σχεδιασμός- Δεν είναι ανθρωπίνως δυνατό να καθοριστεί ένα σαφές σχέδιο το οποίο θα τυποποιήσει όλες τις εργασίες ανάπτυξης του έργου.

- Σχέδια για περίπλοκα έργα σπανίως αποδεικνύονται αποτελεσματικά.
- Τα απρόβλεπτα προβλήματα αποτελούν, συνήθως, κανόνα παρά εξαίρεση.

Αλλαγή – Δεν είναι εφικτή η προστασία από αργοπορημένες αλλαγές.

- Όλες οι εταιρίες προσπαθούν να ανταποκρίνονται στο θέμα της αργοπορημένης αλλαγής.

8.1 Ο δρόμος προς την κόλαση είναι στρωμένος με καλές προθέσεις

Τα εσφαλμένα στοιχεία στην παράγραφο 7.3 δημιουργούν μια δυσλειτουργική σχέση εντός του έργου, η οποία υπονομεύει την αποτελεσματικότητά του. Αυτή η δυσλειτουργική σχέση δημιουργείται όταν:

- Ο πελάτης προσποιείται ότι είναι δυνατό να καθοριστούν ορόσημα και παραδοτέα κομμάτια έργου εκ των προτέρων.
- Ο πελάτης, στη συνέχεια δημιουργεί ένα σχέδιο έργου στο οποίο επισημοποιεί τα συγκεκριμένα ορόσημα.
- Η αντίστοιχη εταιρία προσποιείται ότι μπορεί να ικανοποιήσει αυτά τα ορόσημα, προκειμένου να πάρει τη δουλειά.

Και οι δύο πλευρές διατηρούν την ψευδαίσθηση της ορθής διαχείρισης του έργου, ισχυριζόμενοι ότι μπορούν να ικανοποιήσουν τα συγκεκριμένα ορόσημα. Όμως, και οι δύο γνωρίζουν ότι οδηγούνται με μαθηματική ακρίβεια στην αποτυχία.

8.2 Σχεδιάζοντας με αβεβαιότητα

Οι κανόνες για την εφαρμογή ευέλικτων διεργασιών είναι βασισμένοι γύρω από τα αυξανόμενα επίπεδα αβεβαιότητας που βιώνουν οι εφαρμογές όπως:

- **Ένα σαφές μέλλον**- Μια ενιαία συνεκτική άποψη για το αποτέλεσμα.
- **Μέλλον με εναλλακτικές λύσεις**- Ένα μικρό σύνολο από αποτελέσματα, ένα εκ των οποίων θα προκύψει.
- **Μια σειρά από μελλοντικές λύσεις**- Πολλά πιθανά αποτελέσματα.
- **Αληθινή ασάφεια**- Δεν προσδιορίζεται ένα φάσμα αποτελεσμάτων.

Όσο μεγαλύτερος είναι ο βαθμός αβεβαιότητας ενός έργου, τόσο μεγαλύτερη είναι η αποτελεσματικότητα των ευέλικτων διεργασιών. Με την παρουσία τους, η δυσκολία του σχεδιασμού δεν αίρει την ανάγκη για σχεδιασμό, απλά αλλάζει το σκοπό του:

- Σχέδιο, που έχει ως σκοπό να κερδίσει την κατανόηση.
- Σχέδιο για απρόβλεπτα γεγονότα, που αποσκοπεί στη μείωση του κινδύνου.
- Δεν παίρνουμε τον αρχικό σχεδιασμό στα σοβαρά. Ο αρχικός σχεδιασμός είναι απλά ένας οδηγός για το μέλλον, δεν είναι το μέλλον.

8.3 Αποφυγή δυσλειτουργικών σχέσεων

Χρησιμοποιώντας τις τρεις βασικές πτυχές της μεθοδολογίας Venture Capital, που έχουμε εξετάσει στην παράγραφο 7.4, τα έργα ERP μπορούν να χαρακτηριστούν ως επιχειρησιακά εγχειρήματα. Χρησιμοποιώντας μια μεταγενέστερη μεθοδολογία, η διαχείριση των ERP περιλαμβάνει:

- Σταδιοποίηση- Η ανάπτυξη όλων των χαρακτηριστικών ενός ERP, ταυτόχρονα με σκοπό να προσκομίσουμε άλλα οφέλη, δεν είναι μια καλή Venture Capital απόφαση.

- Κίνητρο εναρμόνισης και διαμοιραζόμενο ρίσκο- Η λύση ενός προβλήματος με συνεργασία από όλα τα μέλη, αποτελεί βασικό παράγοντα επιτυχίας.
- Οι άνθρωποι αποτελούν το κλειδί της επιτυχίας. Κάθε επιτυχημένο εγχείρημα μερίζει ένα μεγάλο ποσοστό της επιτυχίας του στους κατάλληλους ανθρώπους, οι οποίοι ανέλαβαν να το εκπληρώσουν.

9 Ευέλικτες μέθοδοι και συστήματα ERP

Η ευελιξία, μας προσδίδει την ικανότητα να ανταποκριθούμε στις αλλαγές που συχνά είναι αναγκαίες. Όλοι οι οργανισμοί που επιλέγουν τις ευέλικτες μεθόδους, αντιμετωπίζουν την αλλαγή ως ευκαιρία και όχι ως απειλή.

Κατά την εφαρμογή ευέλικτων μεθόδων διαχείρισης έργου στον τομέα των ERP, θα πρέπει να έχουμε πάντα στο μυαλό μας, σε γενικό πλαίσιο, τρία βασικά στοιχεία:

- Πώς μπορούν αυτές οι μινιμαλιστικές προσεγγίσεις να εφαρμοστούν σε ένα περιβάλλον COTS (Commercial, off-the-shelf), διατηρώντας επίσης την αναγκαία πληρότητα του παραδοθέντος προϊόντος, το κόστος, τον έλεγχο, τις λειτουργικές ικανότητες, τη διαχείριση των πόρων και την έγκαιρη παράδοση;
- Ποιες απλουστεύσεις της διαδικασίας διαχείρισης έργου είναι κατάλληλες για τον τομέα των ERP και ποιες όχι;
- Είναι όλα τα βήματα της ευέλικτης διαδικασίας διαχείρισης έργου εφαρμόσιμα στο περιβάλλον του ERP; Αν όχι, ποια μέτρα θα μπορούσαν να εφαρμοστούν;

Σε γενικές γραμμές κάθε ευέλικτη διαδικασία θα πρέπει να περιλαμβάνει τρία βασικά χαρακτηριστικά, σύμφωνα με τα οποία, η διαδικασία θα πρέπει να είναι:

- Επαναληπτική και εξελικτική, επιτρέποντας την προσαρμογή στα εσωτερικά ή εξωτερικά συμβάντα της διαδικασίας.
- Προσαρμόσιμη και υπολογίσιμη, επιτρέποντας στοιχεία της διαδικασίας να εισέρχονται και να εξέρχονται, αναλόγως τις καθορισμένες ανάγκες των συμμετεχόντων στη διαδικασία, όπως επίσης και των πελατών.
- Βασιζόμενη σε ένα συγκεκριμένο χρονικό διάστημα, δομημένη σε κύκλους εργασιών οι οποίοι περιέχουν βρόγχους ανάδρασης, σημεία ελέγχου, καθώς και καθοδήγηση του επόμενου κύκλου εργασιών με βάση τις πληροφορίες που παρήχθησαν από τον προηγούμενο.

Είδαμε παραπάνω, τι θα πρέπει να προσέξουμε σε γενικές γραμμές εφαρμόζοντας μια ευέλικτη διαδικασία, αναφέροντας τα τρία βασικά χαρακτηριστικά των διαδικασιών αυτών. Σε αυτό το σημείο, θα προσπαθήσουμε να εισάγουμε μια ευέλικτη διαδικασία στον τομέα των ERP, να διαπιστώσουμε ποια είναι τα σημεία που χρειάζονται ιδιαίτερη προσοχή, καθώς επίσης και τις προϋποθέσεις για την εφαρμογή της.

1. Ευέλικτες αξίες του έργου (Agile Project Values)

Μερικές από αυτές τις αξίες του έργου είναι οι εξής:

- Επικοινωνία- η διακίνηση των πληροφοριών στο εσωτερικό και το εξωτερικό των ευέλικτων διαδικασιών θα πρέπει να είναι αδιάκοπη. Ένα από τα βασικά στοιχεία της επιτυχίας είναι η επικοινωνία γεγονός που σημαίνει ότι πρέπει να δίνεται μεγάλη βαρύτητα σε αυτή.

- Απλότητα- ορίζει την προσέγγιση της αντιμετώπισης των κρίσιμων παραγόντων επιτυχίας ενός έργου, όσον αφορά την απλούστερη δυνατή λύση.

- Ανατροφοδότηση- ¨Η αισιοδοξία αποτελεί κίνδυνο στην ανάπτυξη λογισμικού, η ανατροφοδότηση είναι η θεραπεία¨

- Θάρρος- οι σημαντικές αποφάσεις που αφορούν σε αλλαγές στην κατεύθυνση ενός έργου, πρέπει να λαμβάνονται με θάρρος από τους αρμόδιους. Όλοι γνωρίζουμε ότι μια λάθος απόφαση μπορεί να επιφέρει μεγάλα έξοδα στην ανάπτυξη του έργου, οπότε είναι εύκολο να διαπιστώσουμε πόσο σημαντικές είναι αυτές.

- Ταπεινότητα- Ακόμη και οι καλύτεροι διαχειριστές έργων θα πρέπει να καταλάβουν ότι δεν είναι δυνατό να γνωρίζουν τα πάντα για την εκάστοτε εφαρμογή και έτσι θα πρέπει συχνά να απευθύνονται στους πελάτες, ώστε να διευκρινίζονται όλες οι ασάφειες που ενδεχομένως να υφίστανται.

2. Εφαρμόζοντας τις ευέλικτες αρχές (Applying Agile Principles)

Οι ακόλουθες αρχές δημιουργούν τα θεμέλια για τη διαχείριση των ERP έργων με μια ευέλικτη προσέγγιση.

- Δεχόμαστε την απλότητα- καθώς το έργο εξελίσσεται θεωρείται ότι η απλούστερη λύση είναι και η καλύτερη. Η αναδόμηση του συστήματος, όπως και οποιοδήποτε τεχνούργημα πάνω σε αυτό, θα πρέπει να αποφεύγεται.

- Αγκαλιάζουμε την αλλαγή- καθώς το έργο εξελίσσεται και οι απαιτήσεις των ενδιαφερομένων (πελατών) παίρνουν σάρκα και οστά, υπάρχουν πολλές πιθανότητες κάποιος από αυτούς να επιχειρήσει να τις διαφοροποιήσει. Σε αυτό το σημείο, η ομάδα ανάπτυξης πρέπει να είναι έτοιμη να αντεπεξέλθει στη νέα αυτή πρόκληση. Αυτές οι διαφοροποιήσεις αποτελούν σύνηθες φαινόμενο στην ανάπτυξη των ERP.

- Ενεργοποίηση της επόμενης προσπάθειας- Υπάρχουν πολλές πιθανότητες το έργο να χαρακτηρισθεί ως αποτυχία, ακόμη και αν η ομάδα παραδώσει ένα άκρος λειτουργικό σύστημα στον πελάτη. Ο πελάτης θα είναι αρκετά ικανοποιημένος, αν εξασφαλισθεί ότι το σύστημα είναι αρκετά ανθεκτικό, ώστε να επεκταθεί με την πάροδο του χρόνου. Μπορεί δηλαδή, να ζητηθεί η άμεση παραγωγή μιας νέας έκδοσης του συστήματος με κάποιες πρόσθετες δυνατότητες ή την αναβάθμιση κάποιων υπαρχόντων λειτουργιών του συστήματος.

- Αυξητική αλλαγή- Η πίεση να τα κάνουμε όλα σωστά από την πρώτη φορά, μπορεί να επιφέρει την καταστροφή στο έργο. Αντί να προσπαθούμε ματαίως να αναπτύξουμε ένα έργο συνολικά, θα πρέπει αυτό να χωρισθεί σε επιμέρους τμήματα και να αναπτυχθεί καθ' αυτό τον τρόπο.

- Αξία στον πελάτη- Οι πελάτες επενδύουν σε πόρους, χρόνο, χρήματα, εγκαταστάσεις κλπ., αναμένοντας ένα σύστημα που θα καλύπτει τις ανάγκες τους, και όπως είναι κατανοητό, την αξιοποίηση των επενδύσεών τους με τον καλύτερο τρόπο.

- Διαχείριση με ένα βασικό σκοπό- τη δημιουργία ενός συστήματος, που θα έχει αξία για τον πελάτη και αυτό θα επιτευχθεί αν προσδιορισθεί σε ποιους θα απευθύνεται αυτό.

- Πολλαπλές προβολές του έργου- λαμβάνοντας υπόψη την πολυπλοκότητα της κάθε σύγχρονης τεχνολογίας ανάπτυξης ενός έργου, γεννιέται η ανάγκη για ένα ευρύ φάσμα μορφών παρουσίασης ώστε η επικοινωνία με τους πελάτες να είναι αποτελεσματικότερη και να διευκρινιστούν όλες οι ασάφειες, που έχουν σχέση με την ορθή ανάπτυξη του εκάστοτε έργου.

- Ταχεία ανατροφοδότηση- Το χρονικό διάστημα μεταξύ της δράσης και της ανατροφοδότησης, σε αυτή τη δράση, πρέπει να ελαχιστοποιηθεί. Η λύση είναι να δουλεύουμε κοντά στον πελάτη, να κατανοούμε τις απαιτήσεις του, να τις αναλύουμε διεξοδικά και να παράγουμε ένα

αγώγιμο πλάνο, που θα μας παρέχει πολυάριθμες ευκαιρίες ανατροφοδότησης.

- Το λειτουργικό λογισμικό είναι ο πρωταρχικός στόχος- όχι στην παραγωγή μίας εξωτερικής τεκμηρίωσης ή λογισμικού. Κάθε δραστηριότητα, η οποία δε συμβάλει στην επίτευξη του βασικού μας στόχου, και στην προκειμένη περίπτωση την ανάπτυξη ενός λειτουργικού λογισμικού, θα πρέπει να εξετασθεί ώστε να προσδιορισθεί η αξία της.

9.1 ERP Λειτουργικοί τομείς

Οι τομείς δραστηριότητας στους οποίους το ERP διαδραματίζει ένα κρίσιμο ρόλο είναι οι εξής:

Πεδίο	Λειτουργίες
Διαχείριση παραγωγής προϊόντων	Διαχείριση προγράμματος, διαχείριση δεδομένων, διαχείριση ποιότητας, διαχείριση περιουσιακών στοιχείων
Διαχείριση εφοδιαστικής αλυσίδας	Δίκτυο εφοδιασμού, σχεδιασμός, συντονισμός, εκτέλεση
Διαχείριση σχέσεων με τους πελάτες	Δέσμευση πελάτη, επιχειρησιακές συναλλαγές, εκτέλεση παραγγελιών, εξυπηρέτηση πελατών
Οικονομικών	Χρηματοοικονομικές πράξεις, λογιστικά, εταιρικές υπηρεσίες
Ανθρωπίνου δυναμικού	Διαχείριση, καταβολή μισθών, διαχείριση χρόνου, στρατηγικές, νομικές εκθέσεις, διαχείριση οργανισμού και ανάπτυξης
Προμηθειών	Έμμεση προμήθεια υλικών, Άμεση προμήθεια υλικών, ηλεκτρονική δημοπρασία, Ολοκληρωμένες αναλύσεις

9.2 Σχέσεις μεταξύ του πεδίου PDM και ERP

Ένας λειτουργικός τομέας, ο οποίος επηρεάζει αρκετά πολλές από τις επιχειρηματικές δραστηριότητες, είναι ο τομέας PDM, ο οποίος αναλύεται ως εξής, διαχείριση πληροφοριών έργου (product data management). Ο συγκεκριμένος τομέας διαχειρίζεται τις οντότητες έργου, από το στάδιο του σχεδιασμού μέχρι και την τελική παράδοση του στον πελάτη.

Οι μηχανικές λειτουργίες βοηθούν ένα έργο να αναπτυχθεί. Έτσι τα εργαλεία της μηχανικής αποτελούν την καρδιά της αλληλεπίδρασης των PDM συστημάτων με την κοινωνία των χρηστών. Αυτές οι μηχανικές λειτουργίες αποτελούν καλές πρακτικές για ευέλικτη ανάπτυξη, εφόσον οι πτυχές ανάπτυξής τους συνήθως ανακαλύπτονται στην πράξη κατά την εφαρμογή τους, πειραματίζοντας με διάφορα εργαλεία και αλληλεπιδρώντας με τους χρήστες, όπως επίσης εξελίσσοντας αυτές τις λειτουργίες, ώστε να είναι σε θέση να αντιμετωπίζουν άγνωστες και μερικές φορές ακατάληπτες απαιτήσεις της αγοράς.

9.3 Ευέλικτες πρακτικές PDM

1.Δεχόμαστε την απλότητα

- Τα έργα COTS (Commercial of the self) καθορίζουν τις απαιτήσεις τους, περισσότερο από ότι οι χρήστες. Μην προβείτε σε αλλαγές στο σύστημα, εφόσον αυτό μπορεί να αποφευχθεί. Ξεκινήστε με το έτοιμο-βασικό σύστημα, ώστε να ανακαλύψετε τα κενά τα οποία θα καλυφθούν με παραπλήσια συστήματα εφόσον είναι εφικτό.

- Ο διαχωρισμός των κινδύνων είναι ένας κρίσιμος παράγοντας τόσο για το έργο όσο και για τις λειτουργίες. Βάσει αυτών των διαχωρισμών, στη συνέχεια θα εφαρμόσουμε την τεχνική δομή του.

- Η διαχωρισμένη εργασία προσδίδει απλότητα στην αρχιτεκτονική του έργου.

- Προσπάθεια σμίκρυνσης των επιμέρους προγραμματιστικών λειτουργιών από την αρχή του κύκλου ανάπτυξης. Με αυτό τον τρόπο, προσπαθούμε να έχουμε όσο το δυνατό μικρότερες λειτουργίες ανεξαρτήτως του πλήθους τους.

- Παρέχει μέσα για την ενσωμάτωση νέων λειτουργιών και σχέσεων μεταξύ αυτών, αργότερα στον κύκλο ανάπτυξης.

- Πάντα ξεκινάμε με την ανάπτυξη των πιο εύκολων λειτουργιών και στη συνέχεια προχωρούμε στις δυσκολότερες, προσπαθώντας όμως να επιλέξουμε τις απλούστερες λύσεις για την ανάπτυξή τους.

2.Αγκαλιάζουμε την αλλαγή

- Η αρχιτεκτονική, με βάση το αντικείμενο, επιδέχεται αλλαγές αλλά πολύ δύσκολα διατηρεί την ορθή ιδιότητα των αντικειμένων.

- Επεκτασιμότητα, κάλυψη πληροφοριών και διάφορες άλλες ιδιότητες μπορεί να πληρώσουν μεγάλα μερίσματα κατά την πάροδο του χρόνου. Με πιο απλά λόγια, μπορεί να επιφέρουν μεγάλα έξοδα.

- Ένα μοντέλο βασιζόμενο, κατά ένα μεγάλο ποσοστό, στην εστίαση των απαιτήσεων.

- Απομόνωση χαρακτηριστικών, ώστε να παράγουμε μια αναντικατάστατη αρχιτεκτονική.

3.Ενεργοποίηση της επόμενης προσπάθειας

- Ο σχεδιασμός με βάση την αρχιτεκτονική, αποτελεί το κύριο μέλημα του διαχειριστή του έργου.

- Χρησιμοποίηση ενός παραδείγματος σχεδιασμού σε κάθε καθημερινή δραστηριότητα του έργου.

- Δίνουμε έμφαση στις αξίες του παρόντος, όσον αφορά το έργο μας, διατηρώντας όμως στο μυαλό μας σκέψεις για μελλοντική δημιουργία αξίας από την ήδη υπάρχουσα.

- Συνεχής αξιολόγηση του κόστους μίας μελλοντικής αναβάθμισης ή επέκτασης.

4.Επαυξητική αλλαγή

- Σχεδιάζουμε μαζικά, δηλαδή όλοι οι ενδιαφερόμενοι και οι ομάδες ανάπτυξης, υλοποιούμε τοπικά, έχοντας πάντα ως γνώμονα την αρχιτεκτονική του έργου.

- Ο ταχύς και εις βάθος σχεδιασμός δεν είναι οξύμωρος.

5.Μεγιστοποιούμε την αξία του πελάτη

- Παραδώστε εργαλεία στα χέρια των χρηστών. Θα πρέπει να ανακαλύψουμε τι μπορούμε να κάνουμε εμείς για αυτούς τους ανθρώπους, ώστε και αυτοί με τη σειρά τους να κάνουν εύκολα τη δουλειά τους.

- Παραδώστε αυτά τα εργαλεία γρήγορα, αποτελεσματικά, με έναν επωφελή τρόπο, με τη χρήση του ελάχιστου των πόρων. Δίνουμε στο χρήστη συνεχείς εκδόσεις του συστήματος, καθώς το έργο αναπτύσσεται και προστίθενται νέες λειτουργίες.

- Οι πελάτες καθορίζουν τις διαστάσεις της αξίας, θα πρέπει να τους ρωτήσουμε τι θέλουν, πότε το θέλουν και πόσα είναι διατεθειμένοι να πληρώσουν.

6. Διαχείριση με ένα βασικό σκοπό

- Πάντα θα πρέπει να καθορίζεται το αποτέλεσμα μιας δράσης, ποιος επωφελείται; Πώς μπορεί αυτό το όφελος να αναγνωριστεί; Τι θα κοστίσει αυτό;

- Ποτέ μην συγχέετε την προσπάθεια με τα αποτελέσματα.

7. Πολλαπλές προβολές του έργου

- Αντικείμενα στατικά ή δυναμικά είναι πολύ καλά, αλλά δε δείχνουν την επιχειρηματική διαδικασία.

- Η ροή δεδομένων είναι καλή, αλλά δε δείχνει τη βασική αρχιτεκτονική αντικειμένων μιας επιχείρησης.
- Ο έλεγχος ροής είναι χρήσιμος στη βελτίωση επιχειρηματικών διαδικασιών, αλλά πρέπει να είμαστε προσεκτικοί σχετικά με πλεονάζοντα δεδομένα και εμμένουσες οντότητες.

8. Ταχεία ανατροφοδότηση

- Συνεχής επικοινωνία με τον πελάτη.
- Εργασία σε ένα περιβάλλον όπου όλοι μάχονται για την ολοκλήρωση του έργου και όχι αναμεταξύ τους.
- Συνεχής παράδοση της λειτουργικότητας.

9. λειτουργικό λογισμικό

- Τα έργα COTS τείνουν να εξαλείψουν αυτή την έννοια, αλλά η προσπάθεια για την ολοκλήρωση του συστήματος είναι εξίσου δύσκολη και σημαντική.
- Συνεχής παράδοση χρησιμοποιώντας στάνταρ έργα με την ελάχιστη παραμετροποίηση.
- Αποφύγετε την προσαρμογή ενός έργου COTS, αν είναι δυνατό.

9.4 Κρίσιμοι παράγοντες επιτυχίας για ευέλικτα ERP

Υποστήριξη από τους διαχειριστές	Πρωταθλητές του έργου
Διαχείριση των προσδοκιών	Σχέσεις ομάδας με πελάτες
Χρήση εργαλείων πελάτη	Προσεκτική επιλογή πακέτων
Διαχείριση έργου	Οργανωτική επιτροπή

Χρήση συμβούλων	Ελάχιστη προσαρμογή
Επιχειρησιακή διαδικασία ανασχεδιασμού	Ορισμός της αρχιτεκτονικής
Αποκλειστική χρήση πόρων	Διακριτές αρμοδιότητες στην ομάδα
Αλλαγή διαχείρισης	Ξεκάθαροι στόχοι
Εκπαίδευση στις διαδικασίες	Διυπηρεσιακή επικοινωνία
Διυπηρεσιακή συνεργασία	Συνεχή υποστήριξη πωλητή

10 Related work (AGILE ERP)

Σε αυτό το σημείο θα αναφέρουμε μερικές σημαντικές εμπειρικές μελέτες, που έχουν γίνει στο παρελθόν, με θέμα την ευέλικτη ανάπτυξη και κατ' επέκταση το ευέλικτο ERP.

Θα ξεκινήσουμε με ένα paper δημοσιευμένο από την Elsevier με τίτλο «Το αντίκτυπο της εξίσωσης εικονικών επιχειρήσεων και της τεχνολογίας πληροφοριών, όσον αφορά την απόδοση, σε ένα ευέλικτο περιβάλλον ανάπτυξης» (The impact of alignment between virtual entERPrise and Information technology on business performance In an agile manufacturing environment), με συντάκτες τους καθηγητές πανεπιστημίου του Μισούρι Qing Cao και Shad Dowlatshahi (Qing Cao και Shad Dowlatshahi, 2005). Σύμφωνα με το συγκεκριμένο paper, οι εταιρίες κατασκευής λογισμικού έχουν να αντιμετωπίσουν ταχείες και απρόβλεπτες αλλαγές στο εργασιακό περιβάλλον τους. Η λύση για την ορθή αντιμετώπιση των αλλαγών αυτών είναι η ευέλικτη ανάπτυξη, η οποία εστιάζει σε μικρότερη κλίμακα όσον αφορά την ανάπτυξη, και παρέχει γρήγορες και άμεσες δράσεις ικανές να παράγουν ένα πολύ σημαντικό αποτέλεσμα σε ένα συνεχώς μεταβαλλόμενο περιβάλλον εργασίας. Από τους πολλούς συντελεστές της ευέλικτης ανάπτυξης επιλέχθηκαν η εικονική επιχείρηση (virtual entERPrise, VE) και η τεχνολογία πληροφοριών (information technology, IT) και στο εν λόγω paper υφίσταται μια εμπειρική μελέτη, ώστε να ανακαλυφθούν οι επιπτώσεις της εξίσωσής τους στην απόδοση μιας εταιρίας που εφαρμόζει ευέλικτες μεθοδολογίες ανάπτυξης.

Με βάση τη συγκεκριμένη εμπειρική μελέτη και ύστερα από την κατάθεση των στοιχείων και απόψεών τους, οι Qing Cao και Shad

Dowlatshahi κατέληξαν σε μερικά συμπεράσματα, εκ των οποίων τα σημαντικότερα αναφέρονται παρακάτω :

- Η εξίσωση των VE και IT είχε μεγαλύτερο αντίκτυπο στην απόδοση μιας εταιρίας από ό,τι η VE ή η IT ξεχωριστά. Αυτό, σαφώς δείχνει ότι η συνεργασία και τα αποτελέσματα της αλληλεπίδρασης μεταξύ των συντελεστών της ευέλικτης ανάπτυξης, θα μπορούσαν να αποτελέσουν ένα πιο καθοριστικό παράγοντα επιτυχίας της, από ό,τι ο κάθε συντελεστής της ξεχωριστά. Το γεγονός αυτό, επίσης, μας βοήθα να συμπεράνουμε ότι η ευέλικτη ανάπτυξη είναι μια διεπιστημονική προσπάθεια.
- Κατανομή Πόρων (Resource Sharing). Η κατανομή των πόρων είναι μια διάσταση των VE, η οποία επηρεάζεται περισσότερο από τη χρήση των συστημάτων IT. Αυτό μας δείχνει ότι τα συστήματα IT διευκολύνουν τη χρήση της κατανομής πόρων σε VE περιβάλλοντα, πράγμα που είναι πολύ σημαντικό, αν αναλογιστούμε ότι η κατανομή των πόρων είναι ένα από τα σημαντικότερα χαρακτηριστικά της VE.
- Η αύξηση μεριδίου αγοράς και η βελτίωση της ποιότητας είναι δύο διαστάσεις της VE, που έχουν μετρίως επηρεαστεί από τη χρήση των IT. Στην περίπτωση της βελτίωσης της ποιότητας, τα IT συστήματα θα μπορούσαν να αναδείξουν την ποιότητα των συστημάτων μιας επιχείρησης, ενώ όσον αφορά την αύξηση του μεριδίου αγοράς, τα IT θα μπορούσαν να χρησιμοποιηθούν για να προωθήσουν και να αναπτύξουν στρατηγικές μάρκετινγκ μιας επιχείρησης, με σεβασμό στους εκάστοτε ανταγωνιστές.

Στη συνέχεια θα αναφερθούμε σε ένα εξίσου σημαντικό paper, το οποίο αποτελεί έκδοση της Elsevier. Προέρχεται από τη χώρα της Σουηδίας και έχει συνταχθεί από τους καθηγητές Jan Olhager και Erik Selldin του πανεπιστημίου Linkoping με τίτλο : «Έρευνα για τη χρήση των συστημάτων ERP σε Σουηδικές επιχειρήσεις» (EntERPrise resource planning survey of Swedish manufacturing firms) (Jan Olhager και Erik Selldin, 2003). Σύμφωνα με τους συντάκτες, τα ERP αποτελούν ένα νέο είδος πληροφοριακών

συστημάτων που βοηθούν στην ολοκλήρωση μιας επιχείρησης, εισάγοντας επιπλέον λειτουργικότητα στα ήδη υπάρχοντα πληροφοριακά συστήματα. Στο παρόν paper παρουσιάζεται μια έρευνα, που αφορά την εφαρμογή των συστημάτων ERP στη βιομηχανία της Σουηδίας και πιο συγκεκριμένα σε επιχειρήσεις που ασχολούνται με τη διείσδυση στα συστήματα ERP, τη διαδικασία πριν την εφαρμογή του συστήματος, την εμπειρία που χρειάζεται για την εφαρμογή του συστήματος και τη διαμόρφωση του συστήματος.

Με ένα ποσοστό ανταπόκρισης 37,2% των εταιριών της Σουηδίας, που οι Jan Olhager και ο Erik Selldin ήρθαν σε επαφή, σίγουρα μπορεί να παραχθεί ένα αξιόπιστο αποτέλεσμα και μια ακριβής εικόνα των ζητημάτων εφαρμογής των συστημάτων ERP. Μερικές από τις σημαντικότερες διαπιστώσεις των δύο καθηγητών είναι οι εξής:

- Οι περισσότερες Σουηδικές κατασκευαστικές εταιρίες, σε γενικές γραμμές, υιοθετούν τα συστήματα ERP. Το 83% των εταιριών έχουν υλοποιήσει ή είναι στη διαδικασία υλοποίησης τέτοιων συστημάτων.
- Το κόστος υλοποίησης αυτών των συστημάτων, κατά μέσο όρο, είναι το 0,5% των ετήσιων εσόδων για τις μεγάλες επιχειρήσεις και περίπου 3,5% για τις μικρότερες επιχειρήσεις.
- Οι Σουηδικές επιχειρήσεις επιλέγουν συχνά την υλοποίηση συστημάτων ERP για Σουηδούς πελάτες.
- Οι πιο συχνά αναφερόμενες βελτιώσεις που αφορούν το σύστημα, έχουν να κάνουν με την πρόσβαση σε πληροφορίες και την διαεπιχειρησιακή αλληλεπίδραση και επικοινωνία, δηλαδή, την όσο το δυνατό καλύτερη αλληλεπίδραση των διαφόρων τμημάτων της επιχείρησης.

Σε αυτό το σημείο θα αναφερθούμε σε ένα paper, το οποίο έχει εκδοθεί από την IEEE και έχουν συγγράψει οι Gerard Meszaros και Janice Aston με τίτλο: «Δεν ξέρετε τι έχετε κάνει, πριν το κάνετε» (Agile ERP: you don’t know what you’ve got “till it’s gone!”) (Gerard Meszaros και Janice Aston, 2007). Σύμφωνα πάντα με το συγκεκριμένο paper, τα συστήματα ERP είναι ευρέως γνωστά για τον υψηλό βαθμό ολοκλήρωσης όπως επίσης και για τα εργαλεία και τις διεργασίες ανάπτυξης. Υπάρχουν ορισμένα πράγματα τα οποία

θεωρούμε απλά όταν ασχολούμαστε με .NET ή JAVA, αλλά την ίδια στιγμή αυτά θεωρούνται κρίσιμης σημασίας όταν μιλάμε για ευελιξία. Στο παρόν paper οι Gerard Meszaros και Janice Aston περιγράφουν τον αντίκτυπο που έχουν αυτές οι παραδοχές στην ευέλικτη ανάπτυξη και πιο συγκεκριμένα στην ανάπτυξη του SAP NetWeaver 2004s. Περιγράφονται τα προβλήματα που διαπιστώθηκαν και πώς η ομάδα ανάπτυξης κατάφερε να τα αντιμετωπίσει.

Μετά από ανάλυση των προβλημάτων και των λύσεων τους, οι αναφερόμενοι συγγραφείς, μας συνιστούν να είμαστε προετοιμασμένοι να αντιμετωπίσουμε πολλά θέματα, τα οποία ίσως να μην παίρναμε ποτέ στα σοβαρά, αν ασχολούμασταν με .NET ή JAVA. Τόσο η φύση ενός βασιζόμενου σε διακομιστή (server-based) περιβάλλοντος, όσο και ενός περιβάλλοντος με νοοτροπία καταρράκτη, μπορεί να προκαλέσει σοβαρά εμπόδια στην εφαρμογή μιας πρακτικής πλήρους ευέλικτης ανάπτυξης, όπως η πρακτική του οδηγούμενου από ελέγχους κώδικα (test-driven development). Ορισμένες ευέλικτες πρακτικές, όπως ο προγραμματισμός σε ζεύγη (Pair Programming) δεν επηρεάζονται δραματικά, αλλά η εφαρμογή τους σε ένα server-based περιβάλλον ανάπτυξης κάνει την αυτόματη δοκιμή (automated testing) και την ανάπτυξη με βάση τις ιστορίες πελάτη (Story test-driven) πρόκληση.

Ακολούθως, θα αναφέρουμε ένα επίσης σημαντικό whitepaper με θέμα το ευέλικτο ERP, το οποίο έχει δημοσιευθεί από την coServe extending trust, μια εταιρία που ειδικεύεται στον τομέα ανάπτυξης συστημάτων ERP με τίτλο : «Μια ευέλικτη προσέγγιση για επιτυχή ανάπτυξη συστημάτων ERP» (An Agile Approach to Successful ERP Implementations) (coServe extending trust, 2008). Σύμφωνα με το συγκεκριμένο whitepaper, οι επιτυχείς υλοποιήσεις ενός ERP είναι αποτέλεσμα καλής ανάλυσης, σχεδίασης και εφαρμογής και όχι απλώς αποτέλεσμα εργασίας μιας καλής και έμπειρης ομάδας, χωρίς να υποβαθμίζουμε τη σημαντικότητά της. Μια, βήμα προς βήμα, δοκιμασμένη μεθοδολογία υλοποίησης σαφώς θα πρέπει να εκτελείται από μια ταλαντούχα ομάδα ανάπτυξης, ώστε να εκμεταλλεύονται πλήρως τα οφέλη της. Η coServe ακολουθεί μια ευέλικτη μεθοδολογία ανάπτυξης χωρισμένη σε φάσεις (Sprints) και αποτελεί μια από τις πιο αποτελεσματικές στην υλοποίηση

συστημάτων ERP. Η συγκεκριμένη μεθοδολογία αποτελείται από πέντε φάσεις, τις οποίες και θα αναφέρουμε σε τίτλους.

- 1η φάση: “Κατανόηση του προβλήματος” (Understand the problem).
- 2η φάση: “Καθορίστε τη λύση” (Define the solution).
- 3η φάση και σαφώς πιο ενδιαφέρουσα: “Υλοποίηση” (Put hands to the task).
- 4η φάση: “Κάντο να συμβεί” (Make it happen).
- 5η φάση: “Συνέχεια της προσπάθειας” (Keep on – going).

΄Υστερα από μία αναλυτική παρουσίαση της συγκεκριμένης ευέλικτης μεθοδολογίας ανάπτυξης, η coServe διαπίστωσε ότι αυτή η λεπτομερής και συνάμα πλήρης ευέλικτη διαδικασία ανάπτυξης συστημάτων ERP, διασφαλίζει το γεγονός ότι οι διαχειριστές μπορούν να κάνουν ένα βήμα πίσω και να δώσουν έμφαση σε κρίσιμες, όσον αφορά την ανάπτυξη του έργου, διαδικασίες.

Στη συνέχεια, θα αναφέρουμε ένα πολύ σημαντικό paper από το agile συνέδριο του 2008, που έχει εκδοθεί από την IEEE και έχει ως συγγραφέα τον αντιπρόεδρο τεχνολογίας Roger Valade με τίτλο: «Τα μεγάλα έργα πάντα αποτυγχάνουν» (The Big Projects Always Failed) (Roger Valade, 2008). Σύμφωνα με το Roger Valade και μέσω της οκταετούς εμπειρίας του στην ευέλικτη ανάπτυξη λογισμικού, το εκάστοτε λογισμικό θα πρέπει να αναπτύσσεται σε όσο το δυνατό μικρότερα τμήματα, καθώς με αυτό τον τρόπο, σύμφωνα με τη γνώμη του, εντοπίζουμε την αποτυχία πιο γρήγορα όταν κατευθυνόμαστε προς λάθος κατεύθυνση. Με άλλα λόγια, μπορούμε με μεγαλύτερη ευκολία να διαπιστώσουμε τα λάθη μας, χωρίς αυτά να μας κοστίσουν ακριβά. Αν και όχι τόσο ασυνήθιστο, το μοντέλο αυτό αποτελεί ένα από τα πιο αξιόπιστα ευέλικτα μοντέλα ανάπτυξης.

Στο εν λόγω paper ο Roger Valade κάνει λόγο για τα μαθήματα που αυτός πήρε μέσω της πολυετούς ενασχόλησης του με την ευέλικτη ανάπτυξη, καθώς και ποια σημεία θα πρέπει να προσέξουμε και τί να αποφύγουμε, ώστε να μη βρεθούμε προ εκπλήξεων. Επίσης, μας εξηγεί γιατί θα πρέπει να

επιλέγουμε την ευέλικτη ανάπτυξη και μας παρέχει παραδείγματα εφαρμογής της σε διάφορα πεδία, όπως για παράδειγμα στην ανάπτυξη συστημάτων ERP (Agile ERP). Παρακάτω αναφέρονται βραχυλογικά, μερικές από τις διαπιστώσεις του:

- “Το να υποστηρίζουμε το παραγωγικό περιβάλλον όπως θα τρέχαμε ένα ευέλικτο έργο, είναι αρκετά εύκολο”. Η πλειοψηφία της εργασίας, που εμπίπτει στην ομάδα ανάπτυξης του ERP, είναι καθορισμένη. Η ομάδα δε βλέπει τα δεδομένα, τα οποία αναμένει στην αναφορά.
- “Προβλήματα του παραγωγικού περιβάλλοντος μπορούν να σε εκτροχιάσουν εάν δεν έχουν ρητά σχεδιασθεί”. Όταν το έργο μας αντιμετωπίζει προβλήματα, η κρίσιμη διαδικασία αποτυγχάνει, ενώ η ανάπτυξη θα πρέπει να αναιρεθεί ή να γυρίσει μερικά βήματα πίσω και τα σχέδια, τα οποία είχαν τεθεί, αρχίζουν να αμφισβητούνται. Η διαδικασία μας δίνει τη δυνατότητα της εύκολης αλλαγής , όμως , δεν είναι αυτό το πρόβλημα, αλλά οι αρχικές προσδοκίες, οι οποίες θα πρέπει τώρα να αναπροσδιοριστούν.
- “Η διάσπαση μίας μεγάλης διεργασίας σε επιμέρους διεργασίες, σαφώς ορισμένες, είναι εξαιρετικά δύσκολη στην ανάπτυξη των ERP”. Οι ομάδες μας επικεντρώνονται στις ειδικές απαιτήσεις των συστημάτων ERP.
- “Οι δοκιμές και ο έλεγχος του πηγαίου κώδικα αποτελεί πρόκληση στη ανάπτυξη ERP”. Στο διαδικτυακό κομμάτι η πρακτική ανάπτυξής μας βασίζεται σε παραδοσιακές τεχνολογίες ανοιχτού κώδικα, που έχουν γίνει το «στάνταρ» για πολλές ομάδες. Java, Apache, JBoss, Hibernate, Subversion, Eclipse, CruiseControl και JUnit.
- “Τελικά είναι όλα σχετικά με την επικοινωνία και τις σχέσεις των ατόμων”. Όσο καλύτερη είναι η σχέση της ομάδας ανάπτυξης με τους εκάστοτε πελάτες, τόσο καλύτερα αποτελέσματα επιτυγχάνουμε. Η αποτελεσματική επικοινωνία είναι το κλειδί της επιτυχίας.

Κλείνοντας θα αναφερθούμε σε ένα paper που έχουν συγγράψει οι Aidas SMAIZYS και Olegas VASILECAS με τίτλο: «Ανάπτυξη συστημάτων

ERP ενσωματώνοντας τους κανόνες των επιχειρήσεων» (Business Rules Based Agile ERP Systems Development) (Aidas SMAIZYS και Olegas VASILECAS, 2009).Σύμφωνα με το εν λόγω paper, οι επιχειρησιακοί κανόνες (Business rules) είναι μια σχετικά νέα προσθήκη στον τομέα ανάπτυξης των συστημάτων ERP. Προσφάτως, εισήχθησαν μερικές νέες μέθοδοι βελτίωσης των υφισταμένων συστημάτων ERP, παρ' όλα αυτά υπάρχουν ακόμη ανοιχτά θέματα για το πώς αυτοί οι επιχειρησιακοί κανόνες μπορούν να χρησιμοποιηθούν, για να αναδείξουν ποιοτικά και ποσοτικά αυτά τα χαρακτηριστικά των συγκεκριμένων πληροφοριακών συστημάτων. Σε αυτό το paper, οι Aidas SMAIZYS και Olegas VASILECAS συζητούν για τα θέματα, τα οποία προκύπτουν από την εισαγωγή αυτών των επιχειρησιακών κανόνων στον τομέα ανάπτυξης των ευέλικτων συστημάτων ERP, μέσα από την εφαρμογή τους σε υφιστάμενα συστήματα, όπως επίσης και τρόπους κατά τους οποίους αυτοί οι επιχειρησιακοί κανόνες αναδεικνύουν την ευέλικτη ανάπτυξη. Ακόμη, οι συγγραφείς έχουν καταγράψει την εμπειρία, που αποκόμισαν ύστερα από την εποπτεία μιας ευέλικτης μεθόδου ανάπτυξης συστημάτων ERP, που εισάγει επιχειρησιακούς κανόνες.

Οι Aidas SMAIZYS και Olegas VASILECAS κατέληξαν ότι υπάρχουν τρεις προσεγγίσεις, που χρησιμοποιούνται στην ανάπτυξη τέτοιου είδους συστημάτων. Αυτές οι τρεις προσεγγίσεις έχουν να κάνουν με το επίπεδο ωριμότητας και πολυπλοκότητας των διαδικασιών, που χρησιμοποιούνται στο εσωτερικό τους. Ανάλογα με την προσέγγιση η οποία θα χρησιμοποιηθεί, είναι δυνατό να πετύχουμε διαφορετικά επίπεδα ευελιξίας μετά από μια στιγμιαία αλλαγή στην πολιτική των επιχειρήσεων και μετατροπή των ήδη υφισταμένων κανόνων τους. Οι προσεγγίσεις αυτές είναι:

- Διαχωρισμός του μοντέλου της επιχειρησιακής λογικής κατά το σχεδιασμό από τις διαδικασίες των επιχειρησιακών κανόνων.
- Συγκεντρωτισμός της εκτέλεσης των επιχειρησιακών κανόνων.
- Χρήση τεχνητής νοημοσύνης, που θα επιτρέπει την προσαρμογή του ERP.

Παρά το γεγονός ότι τα συστήματα ERP που ενσωματώνουν κανόνες επιχειρήσεων στη διαδικασία ανάπτυξης τους είναι αρκετά πιο πολύπλοκα κατά το στάδιο του σχεδιασμού, είναι πιο αποτελεσματικά σε περίπτωση περαιτέρω συντήρησης, οι τροποποιήσεις είναι πιο απλοποιημένες και αυτό αποτελεί ένα μεγάλο πλεονέκτημα, λόγω του ότι οι επιχειρήσεις μεταβάλουν συνεχώς τους κανονισμούς και την πολιτική τους.

10.1 Συμπεράσματα

Όπως θα δούμε και στην εφαρμογή του πειράματος, κατά τη διαδικασία ανάπτυξης του λογισμικού με τη χρήση μικρών ιστοριών πελάτη, επιτυγχάνεται όχι μόνο η ταχύτερη δυνατή εύρεση των σφαλμάτων, αλλά και η αναγνώριση της κατάστασης του έργου. Με τον τρόπο αυτό, μειώνουμε τον κίνδυνο της αποτυχίας του έργου, ειδικά όταν πρόκειται για την κατασκευή ενός ERP. Επίσης, με τη χρήση μικρών ιστοριών πελάτη, ο σχεδιασμός και η ανάλυση παρουσιάζονται ευκολότερες στη διαχείριση, με αποτέλεσμα να αναγνωρίζονται οι κίνδυνοι του έργου σε πρώιμο στάδιο. Έκτος, όμως, από αυτά τα οφέλη, παρατηρούμε ότι η ενσωμάτωση μιας νέας ιστορίας πελάτη δεν παρουσίασε αρκετά προβλήματα κατά την ανάπτυξη του λογισμικού, όπως ακριβώς παρατήρησαν και οι καθηγητές Qing Cao και Shad Dowlatshahi.

Επίσης, όπως αναφέρεται από την coServe extending trust, οι επιτυχείς υλοποιήσεις ενός ERP είναι αποτέλεσμα καλής ανάλυσης, σχεδίασης και εφαρμογής και όχι απλώς αποτέλεσμα εργασίας μιας καλής και έμπειρης ομάδας. Αυτό παρατηρήθηκε έντονα κατά τη διάρκεια της ανάπτυξης του ERP καθώς, αν και η ομάδα δεν κατείχε την απαραίτητη εμπειρία λόγω του σωστού σχεδιασμού, εμφανίσθηκε μειωμένος αριθμός σφαλμάτων.

Ένα άλλο σημαντικό κομμάτι της ανάπτυξης του λογισμικού είναι η επικοινωνία και αυτός είναι ο λόγος, που οι περισσότερες τεχνικές του

ευέλικτου προγραμματισμού βασίζονται σε αυτή. Όμως, λόγω της μικρής ομάδας, που πήρε μέρος στο πείραμα, δεν μπορούμε να αποδείξουμε την παρατήρηση του Roger Valade, που αφορά τη σχέση των πελατών με την ομάδα ανάπτυξης.

11 Περιγραφή Προγράμματος

Πριν αναλύσουμε το πείραμα, καλό είναι να αναφέρουμε το είδος του προγράμματος, που αναπτύξαμε. Επειδή λοιπόν, ασχολούμαστε με την εφαρμογή του ευέλικτου προγραμματισμού στην ανάπτυξη ERP λογισμικού, θα δημιουργήσουμε βασικές πτυχές ενός σύγχρονου ERP, όπως είναι η λειτουργία της αποθήκης.

Η λειτουργία της αποθήκης περιλαμβάνει τις καταγραφές προϊόντων, τη δημιουργία παραστατικών, την εύρεση κόστους αποθέματος, στατιστικά στοιχεία για τις εξαγωγές-εισαγωγές αποθέματος κ.α. Επειδή ο χρόνος που μπορεί να αφιερωθεί για το πείραμα είναι αρκετά μικρός, αποφασίστηκε να υλοποιηθούν οι παρακάτω πτυχές:

- Δημιουργία παραστατικών.
- Απογραφή προϊόντων με βάση το είδος.
- Απογραφή προϊόντων με βάση την αποθήκη.

Όμως, πέρα από τις προαναφερόμενες λειτουργίες, ο πελάτης μας θα πρέπει να έχει τη δυνατότητα να εισάγει καινούρια προϊόντα εφόσον υπάρχει περίπτωση να τα προμηθευτεί. Αντιθέτως, νέα είδη των προϊόντων δε θα δημιουργούνται, ούτε θα υπάρχει δυνατότητα τροποποίησης τους. Γι’ αυτό το λόγο, θα θεωρήσουμε πως ο πελάτης μας ασχολείται με εμπορία τροφίμων, ώστε να καθοριστούν τα είδη που θα εισαχθούν στη βάση δεδομένων.

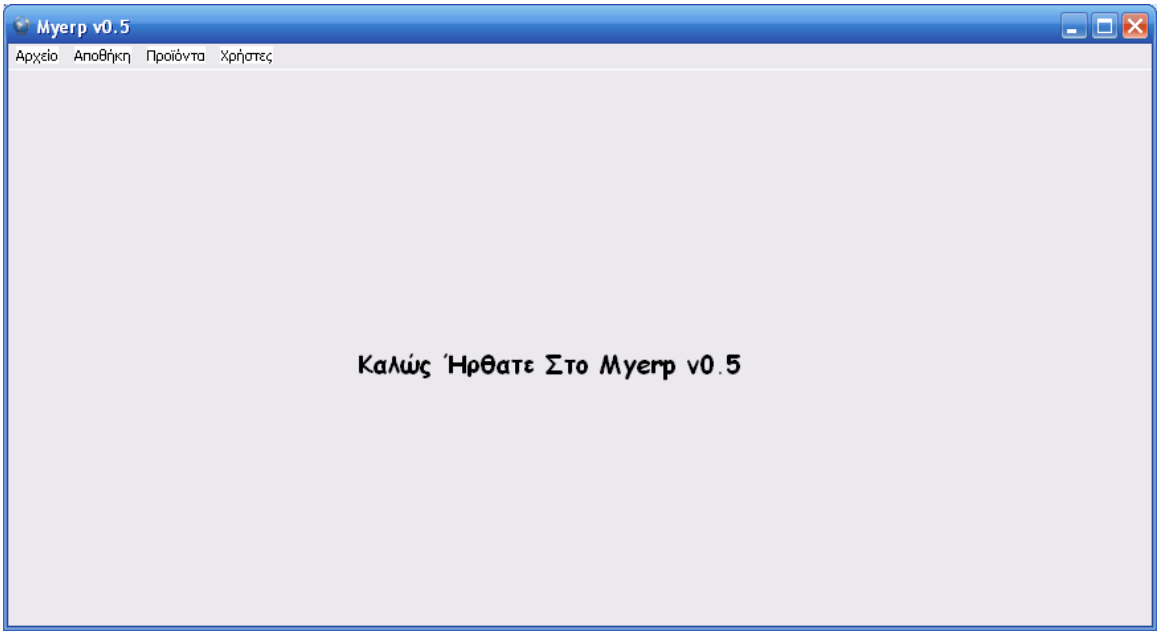
Μία επίσης σημαντική λειτουργία, που υπάρχει σε κάθε ERP, είναι η ασφάλεια. Με τον όρο ασφάλεια εννοούμε την εισαγωγή χρηστών με

καθορισμένα δικαιώματα. Έτσι, θα δημιουργηθούν αρκετοί χρήστες με διαφορετικά δικαιώματα, όσον αφορά την πρόσβασή τους στις πτυχές του λογισμικού. Φυσικά θα υπάρχει και ένας διαχειριστής (administrator), που θα έχει τη δυνατότητα να προσθέτει και να αφαιρεί δικαιώματα από τους χρήστες, εκτός του λογαριασμού του, για αποφυγή κλειδώματος από το πρόγραμμα.

Αυτές επιγραμματικά, είναι οι λειτουργίες, οι οποίες θα δημιουργηθούν στο πείραμά μας. Στις επόμενες σελίδες θα αναλύσουμε την κάθε πτυχή εμπλουτισμένη πάντα με εικόνες από το παραχθέν πρόγραμμα.

11.1 Κεντρική φόρμα

Πριν αναλύσουμε τις διακριτές λειτουργίες του λογισμικού, θα παρουσιάσουμε την κεντρική φόρμα, από την οποία ο χρήστης αποκτά πρόσβαση σε κάθε λειτουργία. Η κεντρική φόρμα λοιπόν, είναι αρκετά απλή, περιέχοντας μόνο το κεντρικό μενού όπως φαίνεται στην παρακάτω εικόνα.



Το κεντρικό μενού, αποτελείται από τις παρακάτω επιλογές:

1. Αρχείο
    - Είσοδος.
    - Ρυθμίσεις.
    - Έξοδος.
2. Αποθήκη
    - Παραστατικά.
    - Απογραφή ανά είδος.
    - Απογραφή ανά κατάστημα.
3. Προϊόντα
    - Εισαγωγή.
    - Τροποποίηση.
4. Χρήστες
    - Εισαγωγή Χρηστών.
    - Διαγραφή Χρηστών.
    - Τροποποίηση Χρηστών.

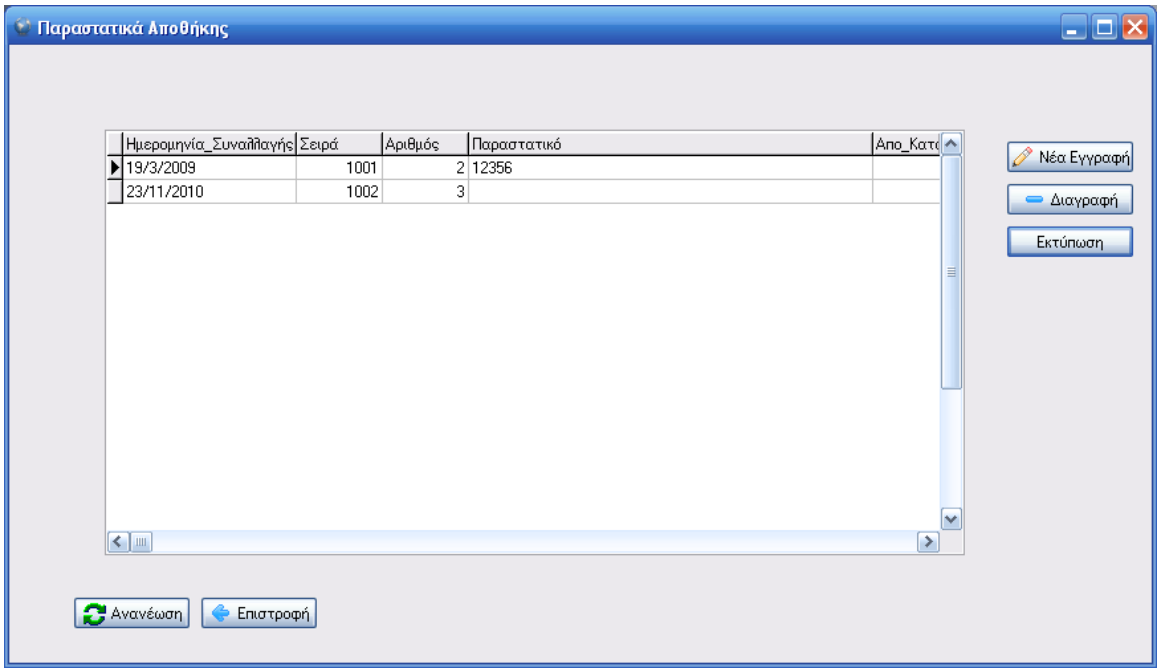
Από το κεντρικό μενού λοιπόν, εφόσον υπάρχουν τα κατάλληλα δικαιώματα, ο χρήστης επιλέγει τη λειτουργία που επιθυμεί και μια νέα φόρμα εμφανίζεται. Αυτές οι φόρμες αναλύονται στα επόμενα κεφάλαια.

11.2 Λειτουργίες αποθήκης

Για να δημιουργηθούν οι λειτουργίες αποθήκης ώστε να προσεγγίζουν όσο το δυνατόν τα ERP προγράμματα του εμπορίου, μελετήθηκε ένα αρκετά γνωστό ERP, από το οποίο αντλήθηκαν αρκετές ιδέες, που αφορούν όχι μόνο το περιβάλλον αλληλεπίδρασης, αλλά και του λογιστικού περιεχομένου.

Η πρώτη πτυχή της αποθήκης, που συναντά κανείς στο παραγόμενο λογισμικό, αφορά τα παραστατικά. Ο πελάτης μας, μπορεί να δημιουργεί νέα παραστατικά, να εμφανίζει γενικές πληροφορίες από παλαιά παραστατικά, και τέλος να τα διαγράφει.

Για τη δημιουργία των παραστατικών, από την κεντρική φόρμα επιλέγουμε το μενού αποθήκη κι ύστερα τα παραστατικά. Στη νέα φόρμα, επιλέγουμε το κουμπί που αναγράφει το «νέα εγγραφή», όπως φαίνεται στην παρακάτω εικόνα.



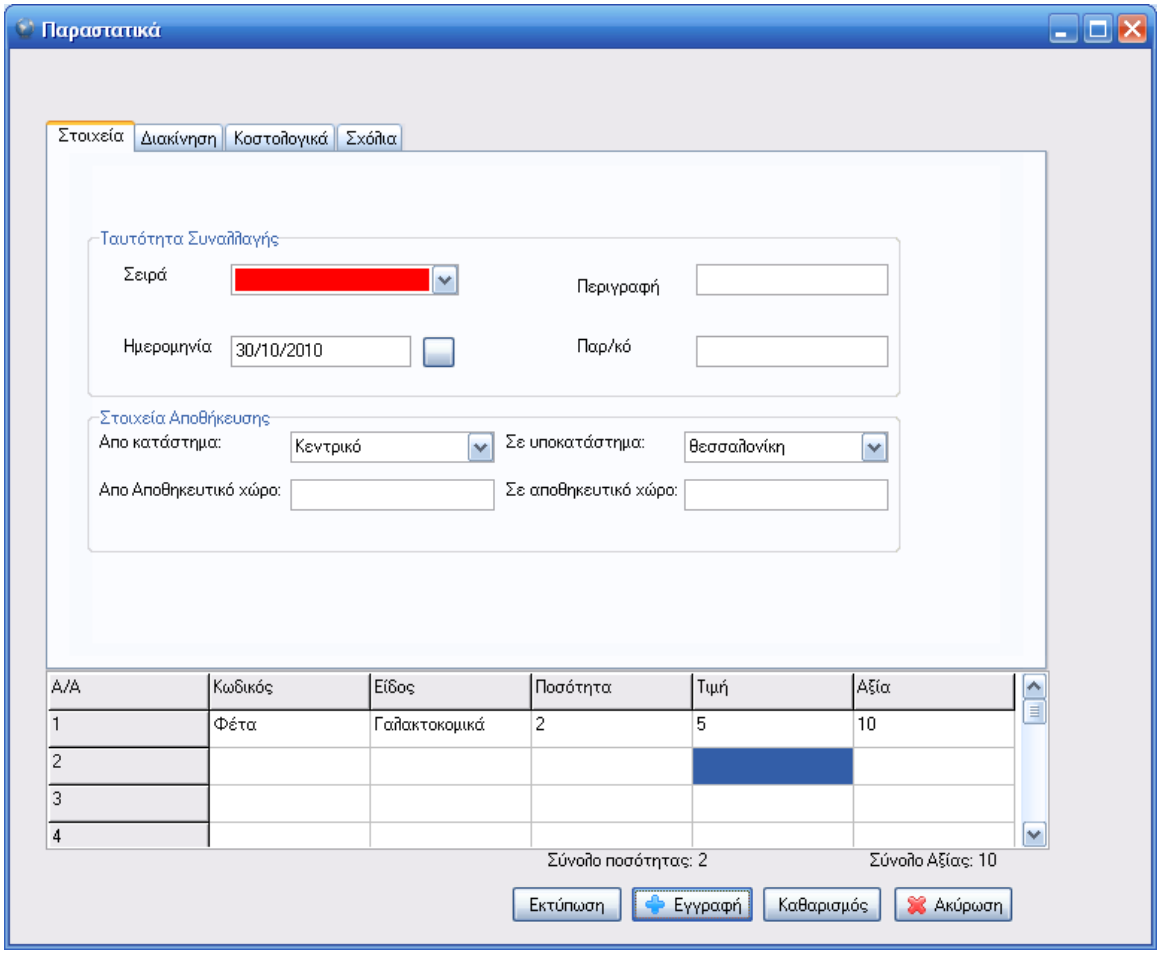
Εικόνα 1

Κατά τη δημιουργία νέων παραστατικών, εμφανίζονται αρκετές επιλογές, που καθορίζουν τα χαρακτηριστικά του παραστατικού. Μερικές από αυτές είναι:

- Αριθμός Παραστατικού
- Ημερομηνία Παραστατικού
- Σειρά
- Περιγραφή
- Επιλογή καταστήματος
- Διεύθυνση
- Σκοπός διακίνησης
- Τρόπος αποστολής
- Σχόλια

Φυσικά, ο χρήστης εισάγει σε πίνακα τα προϊόντα του παραστατικού. Αυτό γίνεται είτε με επιλογή από τη λίστα των προϊόντων, είτε με την εισαγωγή κειμένου, ώστε να υπάρχει δυνατότητα απογραφής προϊόντων, που δεν υπάρχουν στις αποθήκες. Κατά την εισαγωγή των προϊόντων, ο χρήστης επιλέγει την ποσότητα και την αξία τους. Με αυτό τον τρόπο, υπολογίζεται αυτόματα η συνολική αξία και η ποσότητα όλων των προϊόντων και αναγράφονται κάτω από τον πίνακα των προϊόντων. Να τονισθεί, πως κατά τη μεταφορά προϊόντων υπολογίζεται αυτόματα το απόθεμα που απομένει στο κάθε κατάστημα, το οποίο εάν είναι μικρότερο του μηδενός, εμφανίζεται μια προειδοποίηση στο χρήστη και καλείται να επιβεβαιώσει την ολοκλήρωση μεταφοράς ή να την ακυρώσει.

Το είδος του παραστατικού, καθορίζεται από τη σειρά, η οποία επιλέγεται από μία λίστα. Κατά την επιλογή της σειράς των παραστατικών, ο πελάτης έρχεται αντιμέτωπος με αρκετές επιλογές, όπου άλλες επιτρέπουν την εισαγωγή παραστατικού, κι άλλες όχι, όπως δηλαδή στα επαγγελματικά ERP. Επειδή σε κάθε παραστατικό πρέπει να ορισθεί μια σειρά, έχει δημιουργηθεί έλεγχος κατά την αποθήκευση του παραστατικού, ώστε ο χρήστης να ενημερώνεται για τη μη ύπαρξη αυτής. Παράλληλα, το παραστατικό δεν αποθηκεύεται, και το πεδίο της σειράς αλλάζει σε κόκκινο φόντο, που επιτρέπει με αυτό τον τρόπο την εύκολη εύρεσή του.



Μη επιλογή σειράς από το χρήστη

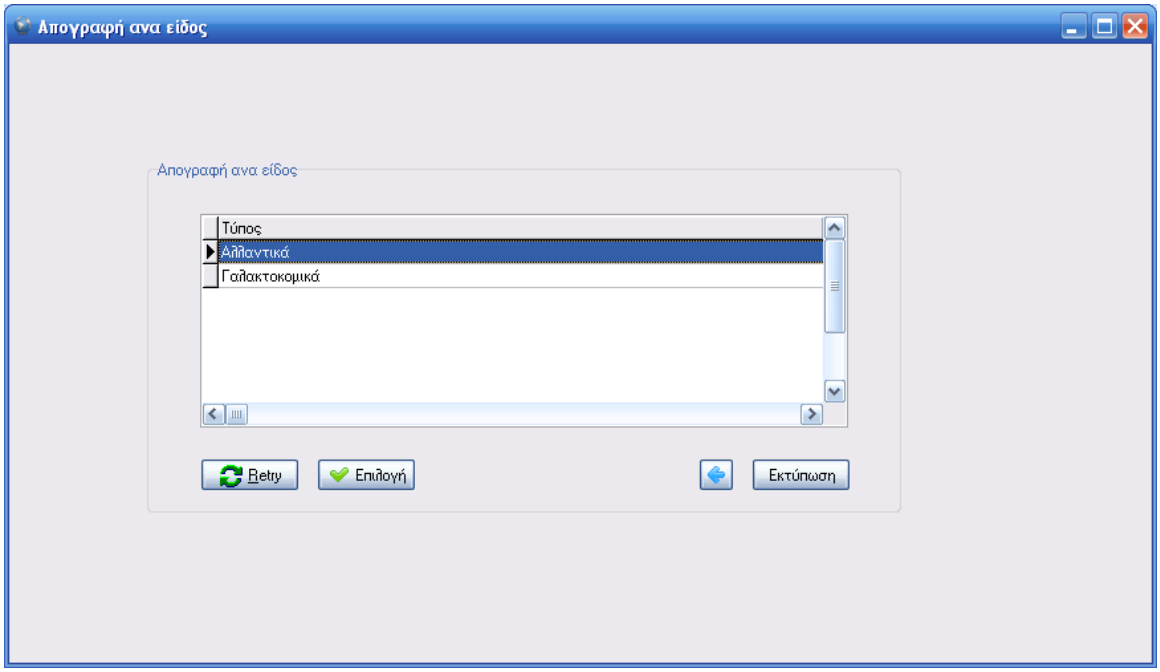
Κατά την αποθήκευση του παραστατικού, τα καταστήματα ενημερώνονται για τα νέα αποθέματα, τα οποία εμφανίζονται και στις απογραφές που θα αναλύσουμε παρακάτω. Όμως, ο πελάτης δεν επιδιώκει την αποθήκευση της πληροφορίας για τα προϊόντα που υπάρχουν σε κάθε παραστατικό, με αποτέλεσμα να μην ενημερώνεται κάποιος πίνακας με την πληροφορία αυτή. Αντιθέτως, επειδή δίνεται η δυνατότητα να εισάγει ο χρήστης στο παραστατικό, προϊόντα που δεν υπάρχουν στη βάση δεδομένων μας, για αυτά τα προϊόντα αποθηκεύουμε την πληροφορία σε ξεχωριστό πίνακα. Καλό είναι να τονισθεί, πως λόγω του περιορισμένου χρόνου ανάπτυξης, αρκετά πεδία, όπως αυτά που αφορούν την κοστολόγηση, όχι μόνο δεν εμφανίζουν επιλογές, αλλά και δεν αποθηκεύονται.

Ο χρήστης μας εκτός από την αποθήκευση, έχει τη δυνατότητα της επισκόπησης των παραστατικών που έχει δημιουργήσει. Η ενέργεια αυτή γίνεται στη φόρμα, που εμφανίζεται όταν από την κεντρική φόρμα επιλέξει το μενού αποθήκες – παραστατικά (βλ εικόνα 1). Οι πληροφορίες που αντικρίζει είναι:

- Ημερομηνία συναλλαγής.
- Αριθμός σειράς.
- Αύξων αριθμός παραστατικού.
- Κωδικός παραστατικού.
- Τα δύο καταστήματα, από τα οποία έγινε η συναλλαγή.

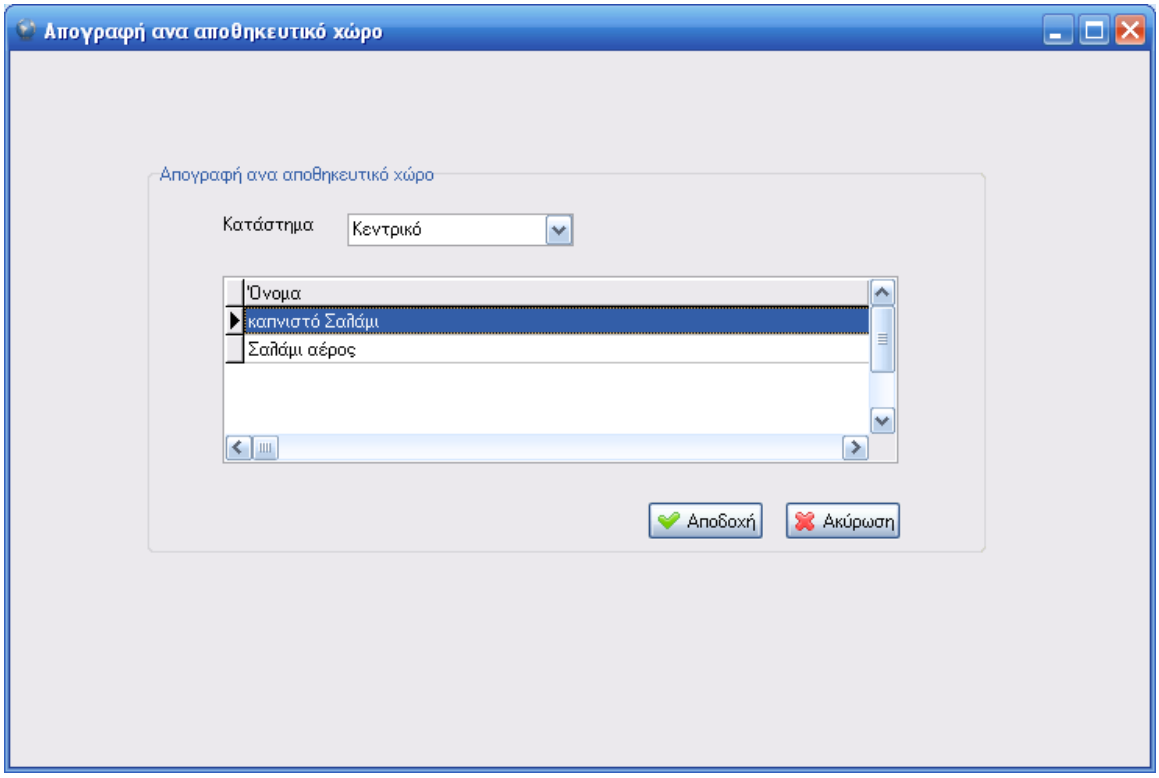
Στην ίδια φόρμα, επιλέγοντας το παραστατικό, μπορούμε να το διαγράψουμε, αφού πρώτα επιβεβαιώσουμε τη συγκεκριμένη ενέργεια. Πρέπει να τονισθεί όμως, ότι λόγω του ότι δεν υπάρχει πίνακας που να καταγράφει τις μεταφορές των προϊόντων ανά παραστατικό, τα αποθέματα δεν τροποποιούνται.

Η δεύτερη πτυχή που συναντά ο χρήστης στις λειτουργίες της αποθήκης, αφορά τις απογραφές ανά είδος. Ο πελάτης από την κεντρική φόρμα, επιλέγοντας από το μενού της αποθήκης τη δεύτερη επιλογή, συναντά μία καινούρια φόρμα, που με μινιμαλιστικό τρόπο, εμφανίζει την καταγραφή αυτή. Στην νέα φόρμα, παρουσιάζεται το σύνολο των προϊόντων, που υπάρχουν στις αποθήκες του πελάτη, καθώς και οι ποσότητές τους και η αξία τους. Ο πελάτης, με αυτό τον τρόπο, ανακτά την κατάσταση των αποθεμάτων του με ένα γρήγορο και εύχρηστο τρόπο, όπως φαίνεται στην επόμενη εικόνα.



Απογραφή ανά είδος

Η τρίτη και τελευταία πτυχή του προγράμματος, που αφορά τις λειτουργίες της αποθήκης, είναι η απογραφή με βάση τις αποθήκες. Ο πελάτης, εκτός από το να ανακτά πληροφορίες για τα προϊόντα που έχει, θα πρέπει να γνωρίζει και σε ποιες αποθήκες βρίσκονται. Η νέα φόρμα λοιπόν, προσπαθεί να καλύψει τη συγκεκριμένη ανάγκη. Όπως μπορούμε να διακρίνουμε, η φόρμα που αφορά τις καταγραφές με βάση τις αποθήκες, μοιάζει σε μεγάλο βαθμό με την προηγούμενη της καταγραφής ανά είδος. Η μόνη διαφορά βρίσκεται σε μια λίστα από όπου ο χρήστης επιλέγει το κατάστημα που θέλει να καταγράψει. Μετά την επιλογή του λοιπόν, εμφανίζονται τα προϊόντα του καταστήματος, εμφανίζοντας παράλληλα την ποσότητα και τη συνολική τους αξία.



Απογραφή ανά χώρο

11.3 Λειτουργίες Διαχείρισης Προϊόντων

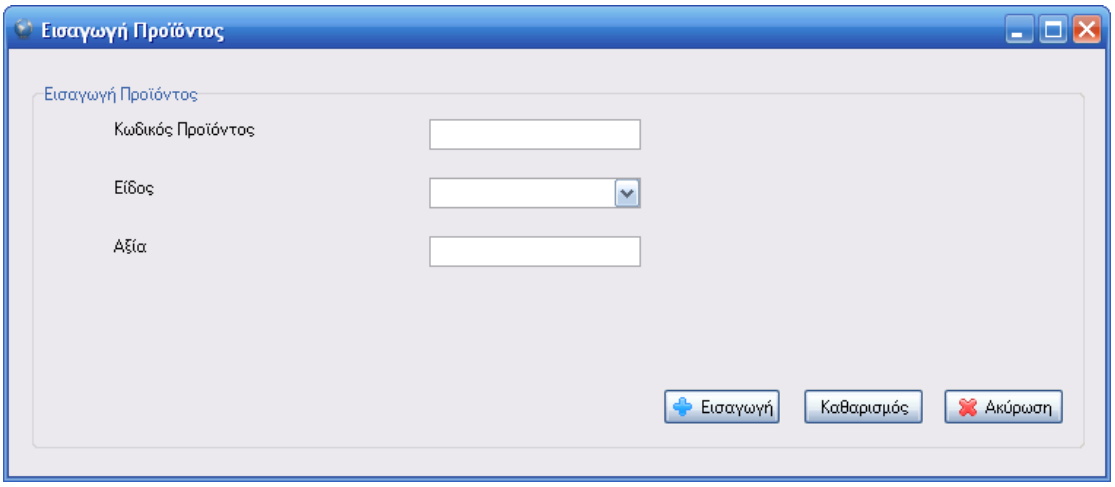
Επειδή το ERP δίνει αρκετή βαρύτητα στις λειτουργίες της αποθήκης, ο πελάτης μας, με κάποιο τρόπο, θα πρέπει να χειρίζεται τα προϊόντα του. Γι' αυτό το λόγο, δημιουργήθηκαν οι λειτουργίες διαχείρισης προϊόντων, που αφορούν τη δημιουργία και την τροποποίηση των προϊόντων.

Για να μπορέσει κάποιος χρήστης να χρησιμοποιήσει τις λειτουργίες αυτές, θα πρέπει να έχει τα κατάλληλα δικαιώματα. Έτσι, εάν ισχύει η προαναφερόμενη συνθήκη, τότε από την κεντρική φόρμα η τρίτη επιλογή του μενού είναι ενεργοποιημένη. Επιλέγοντάς τη βλέπουμε:

- Εισαγωγή
- Τροποποίηση

Η πρώτη επιλογή αφορά τη δημιουργία νέων προϊόντων, από όπου ο χρήστης μπορεί να εισάγει στοιχεία για το καινούριο προϊόν. Έτσι, επιλέγοντας τη συγκεκριμένη επιλογή, μια νέα φόρμα εμφανίζεται στην οποία μπορούν να εισαχθούν τα παρακάτω στοιχεία:

- Κωδικός προϊόντος.
- Είδος.
- Αξία.



Φόρμα εισαγωγής νέου Προϊόντος

Ο κωδικός προϊόντος αφορά το όνομα, με το οποίο θα εμφανίζεται το νέο προϊόν. Για τον πελάτη μας, που είναι έμπορος τροφίμων θα μπορούσε αυτό το πεδίο να συμπληρωθεί με το όνομα μιας σοκολάτας. Αντιθέτως, το είδος δεν εισάγεται από το πληκτρολόγιο αλλά από μία λίστα, και αφορά το είδος του προϊόντος. Καλό είναι να τονισθεί, πως το είδος δεν τροποποιείται, ούτε δίνεται η δυνατότητα να δημιουργηθεί ένα νέο. Τέλος, η αξία αφορά την ανά μονάδα αξία του προϊόντος, η οποία χρησιμοποιείται και στις καταγραφές.

Όταν ένα νέο προϊόν δημιουργηθεί, το παρόν πρόγραμμα, μας ενημερώνει για την επιτυχία της συγκεκριμένης ενέργειας, ενώ παράλληλα μια νέα φόρμα εμφανίζεται. Τώρα, ο χρήστης καλείται να εισάγει την ποσότητα αλλά και σε ποιο κατάστημα υπάρχει το νεοεισαχθέν προϊόν. Η επιλογή του

καταστήματος, γίνεται πάντα με την επιλογή από λίστα, ενώ ένα μήνυμα λάθους εμφανίζεται εάν δεν εισαχθεί η ποσότητα, αλλάζοντας παράλληλα το χρώμα της σε κόκκινο.

Η τροποποίηση των προϊόντων, γίνεται με τη δεύτερη επιλογή στο μενού προϊόντα. Αφού λοιπόν, ο χρήστης επιλέξει να τροποποιήσει κάποιο προϊόν, μια νέα φόρμα εμφανίζεται και ο χρήστης καλείται να επιλέξει το προϊόν που θέλει να τροποποιήσει. Όταν το κάνει, η αξία και το είδος προϊόντος ενημερώνονται αυτόματα, δείχνοντας έτσι την τωρινή κατάσταση του προϊόντος. Με αυτό τον τρόπο, ο χρήστης ενημερώνεται για την αξία και το είδος του προϊόντος, και μπορεί να ενημερώσει τα πεδία αυτά, χωρίς ιδιαίτερο κόπο.

11.4 Λειτουργίες Διαχείρισης Χρηστών

Σε ένα ERP πρόγραμμα, είναι αρκετά σημαντικό να καθορίζουμε τα δικαιώματα του κάθε χρήστη, ώστε να αυξήσουμε με αυτό τον τρόπο την ασφάλεια και τη σταθερότητα του προγράμματος μας. Έτσι, στο λογισμικό μας έχει ενσωματωθεί η πτυχή της διαχείρισης των χρηστών. Έτσι έχουμε:

- Εισαγωγή χρηστών.
- Διαγραφή χρηστών.
- Τροποποίηση χρηστών.

Όπως είναι λογικό, στο συγκεκριμένο μενού, μόνο ο διαχειριστής του προγράμματος μπορεί να έχει πρόσβαση, οπότε είναι απαραίτητη η δημιουργία του. Αφού, λοιπόν, δημιουργήσουμε ένα διαχειριστή και συνδεθούμε με το λογαριασμό του στο ERP, το μενού χρήστες ξεκλειδώνει κι εμφανίζονται οι τρείς προαναφερόμενες επιλογές.

Στην πρώτη, δημιουργούμε νέους χρήστες, καθορίζοντας το όνομα κι έναν κωδικό ασφαλείας, που χρησιμοποιείται για την είσοδό του στο ERP. Με αυτό τον τρόπο, δημιουργούμε έναν χρήστη χωρίς καθόλου δικαιώματα οπότε

δεν έχει πρόσβαση σε καμία επιλογή του κεντρικού μενού. Πριν δούμε τον τρόπο εισαγωγής δικαιωμάτων, ας ασχοληθούμε με τη δεύτερη επιλογή, που είναι η διαγραφή.

Το να διαγράψουμε ένα χρήστη, είναι αρκετά εύκολο, αφού μόνο η επιλογή του ονόματός του απαιτείται από το πρόγραμμα. Γι' αυτό το σκοπό, στη φόρμα της διαγραφής των χρηστών, υπάρχει μια λίστα με τα ονόματα όλων των λογαριασμών που μπορούν να διαγράφουν. Αξίζει να τονισθεί, πως στο ERP έχει προβλεφθεί η πιθανότητα σφάλματος κατά την εισαγωγή ή διαγραφή εγγραφών από τη βάση δεδομένων, ώστε να μην υπάρξει ασυνέπεια της βάσης. Έτσι, σε περίπτωση σφάλματος κατά τη διαγραφή λογαριασμού, ένα μήνυμα σφάλματος εμφανίζεται και η βάση δεδομένων επιστρέφει στην κατάσταση, πριν της διαγραφής(rollback).

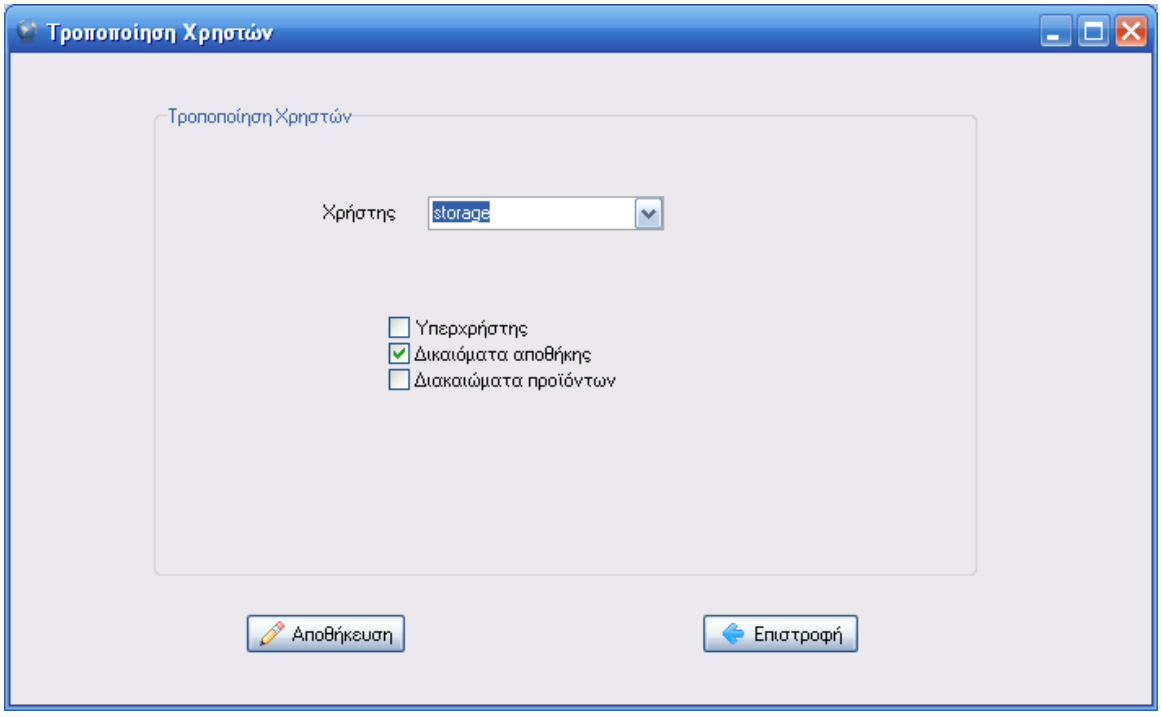
Η τρίτη πτυχή των λειτουργιών διαχείρισης των χρηστών, αφορά την τροποποίησή τους και πιο συγκεκριμένα την αλλαγή δικαιωμάτων, αφού στους λογαριασμούς, δεν υπάρχει δυνατότητα να τροποποιηθεί ούτε το όνομά τους, ούτε ο κωδικός ασφαλείας τους. Τα είδη δικαιωμάτων που ζητήθηκαν από τον πελάτη είναι τα:

- Δικαιώματα διαχειριστή.
- Δικαιώματα αποθήκης.
- Δικαιώματα προϊόντων.

Οι λογαριασμοί που έχουν δικαιώματα διαχειριστή, μπορούν να έχουν πρόσβαση σε κάθε λειτουργία του προγράμματος. Έτσι, μπορούν να τροποποιούν τους υπολοίπους χρήστες, αφαιρώντας ή προσθέτοντας δικαιώματα. Αντιθέτως, οι λογαριασμοί με μόνο δικαιώματα αποθήκης, έχουν πρόσβαση μόνο στο μενού της αποθήκης. Επίσης, τα δικαιώματα προϊόντων, αφορούν την πρόσβαση στο ομώνυμο μενού. Μπορεί να υπάρχουν λογαριασμοί, που να έχουν ταυτόχρονα τα δύο τελευταία δικαιώματα οπότε αποκτούν πρόσβαση και στις δύο λειτουργίες. Όπως έχουμε αναφέρει, κατά τη δημιουργία ενός λογαριασμού, δεν καθορίζονται τα δικαιώματά του. Αυτό σημαίνει πως δεν έχει πρόσβαση σε καμία λειτουργία και χρησιμοποιείται μόνο για τον έλεγχο συνδεσιμότητας με τη βάση

δεδομένων. Στην επόμενο πίνακα σας παρουσιάζουμε τους λογαριασμούς που δημιουργηθήκαν, καθώς και τα δικαιώματά τους.

Όνομα Χρήστη	Δικαιώματα
sa	Διαχειριστή
Sales	Προϊόντων
Storage	Αποθήκης
test	-



Τα δικαιώματα του χρήστη storage

11.5 Άλλες λειτουργίες

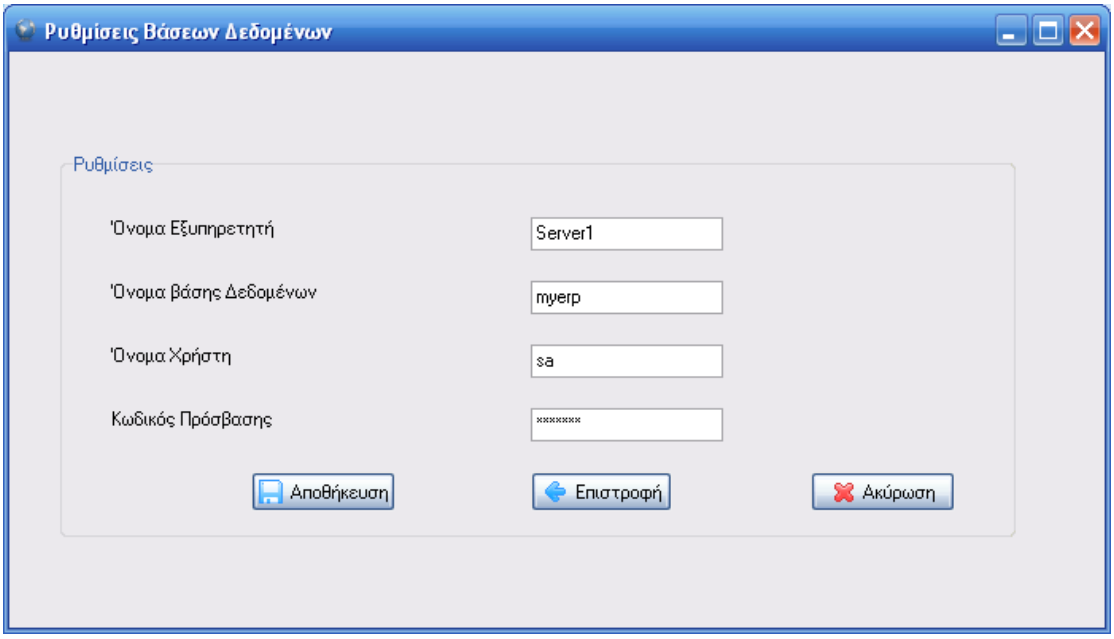
Πριν ολοκληρώσουμε την αναφορά μας για το παραγόμενο λογισμικό, θα πρέπει να επισημάνουμε άλλες δύο λειτουργίες, που είναι αρκετά σημαντικές κι έχουν ενσωματωθεί μετά από απαίτηση του πελάτη.

Η πρώτη, αφορά την είσοδο του χρήστη στο λογισμικό μας. Μπορεί να έχουμε δημιουργήσει αρκετούς λογαριασμούς χρηστών, όμως για να μπορούν να χρησιμοποιήσουν τα δικαιώματα που τους παρέχονται, πρέπει με κάποιο τρόπο να συνδεθούν στο πρόγραμμα. Για να καλυφθεί η συγκεκριμένη ανάγκη, από την κεντρική φόρμα, επιλέγοντας από το μενού αρχείο την πρώτη επιλογή, εμφανίζεται στο χρήστη μια νέα φόρμα, που εισάγει το όνομα του λογαριασμού και τον κωδικό του.

Όμως, υπάρχει περίπτωση ο πελάτης μας να επιθυμεί είτε να αλλάξει τη βάση δεδομένων του, είτε τον εξυπηρετητή του. Επίσης, σε περίπτωση βλάβης στον εξυπηρετητή, να έχει τη δυνατότητα χρήσης κάποιου εφεδρικού. Έτσι στο ERP, δίδεται η δυνατότητα αλλαγής των ρυθμίσεων της βάσης δεδομένων μας. Συγκεκριμένα:

- Εισαγωγή ονόματος εξυπηρετητή.
- Εισαγωγή ονόματος βάσης δεδομένων.
- Εισαγωγή χρήστη βάσης δεδομένων.
- Εισαγωγή κωδικού βάσης δεδομένων.

Φυσικά, όταν ο χρήστης αποθηκεύσει τις επιθυμητές ρυθμίσεις, γίνεται προσπάθεια σύνδεσης με τη βάση δεδομένων, ώστε να αναγνωριστεί η ορθή ή μη, ρύθμιση. Και στις δύο περιπτώσεις, εμφανίζεται ένα μήνυμα επιβεβαίωσης, που ενημερώνει το χρήστη.



11.6 Μηνύματα Σφάλματος

Σε αρκετά σημεία, αναφέρουμε ότι εμφανίζονται μηνύματα σφαλμάτων, όταν καταστεί αναγκαίο ώστε να ενημερωθεί ο χρήστης. Γι' αυτό το λόγο, παρουσιάζουμε τα μηνύματα σφάλματος που μπορούμε να έρθουμε αντιμέτωποι κατά τη χρήση του λογισμικού, καθώς και τις ενέργειες που απαιτούνται.

Λεκτικό Σφάλματος	Επεξήγηση	Ενέργειες
Λάθος Χρήστης ή κωδικός	Είναι λανθασμένα τα στοιχεία του χρήστη	Χρειάζεται επανάληψη εισαγωγής στοιχείων του χρήστη για την εισαγωγή του στο ERP
Cannot perform this operation on an open dataset	Η αλλαγή ρυθμίσεων της βάσης δεδομένων δεν είναι εφικτή αφού	Πρέπει να τερματίσουμε το πρόγραμμα και πριν συνδεθούμε με κάποιο

	έχει γίνει σύνδεση	λογαριασμό, να αλλάξουμε τις ρυθμίσεις
Sql server has been paused	Η βάση δεδομένων είναι σε paused mode	Αλλάξτε την κατάσταση της βάσης δεδομένων σε ενεργή
Sql server does not exist or access denied / Παρακαλώ ελέγξτε τις ρυθμίσεις	Οι ρυθμίσεις της βάσης δεδομένων είναι λανθασμένες η ο διακομιστής δεν ανταποκρίνεται	Ελέγξτε τις ρυθμίσεις της βάσης δεδομένων και το διακομιστή
Παρακαλώ συμπληρώστε τη σειρά	Κατά τη δημιουργία του παραστατικού δεν έχει γίνει επιλογή της σειράς	Επιλέξτε τη σειρά του παραστατικού και δοκιμάστε ξανά
Παρακαλώ συμπληρώστε το είδος	Κατά τη δημιουργία νέου προϊόντος δεν έχει εισαχθεί το είδος	Επιλέξτε το είδος και δοκιμάστε ξανά
Cannot insert duplicate key in object poionta	Το όνομα του προϊόντος έχει καταχωρηθεί σε υπάρχον	Αλλάξτε το όνομα του προϊόντος και δοκιμάστε ξανά
Cannot insert duplicate key in object xrhstes	Το όνομα του χρήστη έχει καταχωρηθεί σε υπάρχον	Αλλάξτε το όνομα του χρήστη και δοκιμάστε ξανά
Σφάλμα διαγραφής. Παρακαλώ δοκιμάστε ξανά	Παρουσίαση γενικού σφάλματος κατά τη διαγραφή χρήστη	Δοκιμάστε ξανά τη διαγραφή χρήστη
Σφάλμα καταχώρισης χρήστη. Παρακαλώ δοκιμάστε ξανά	Παρουσίαση γενικού σφάλματος κατά τη δημιουργία χρήστη	Δοκιμάστε ξανά τη δημιουργία χρήστη

Παρακαλώ συμπληρώστε όλα τα πεδία	Τα απαραίτητα πεδία της φόρμας δεν έχουν συμπληρωθεί	Συμπληρώστε τα πεδία με κόκκινο χρώμα
Παρακαλώ εισάγετε ποσότητα	Κατά τη δημιουργία προϊόντων δεν έχει εισαχθεί η ποσότητα	Εισάγετε ποσότητα και δοκιμάστε ξανά
Δεν υπάρχουν Αποθέματα. Θέλετε να συνεχίσετε;	Κατά τη δημιουργία παραστατικού, η ποσότητα προϊόντος ξεπερνά το απόθεμα	Ακυρώστε το παραστατικό ή επιλέξετε συνέχεια.

Στον παραπάνω πίνακα, έχουμε τα βασικά είδη σφαλμάτων που έχουν προκύψει κατά τους ελέγχους καλής λειτουργίας του λογισμικού, χωρίς να αποκλείεται η πιθανότητα εμφάνισης νέων μηνυμάτων. Όμως, όπως έχουμε τονίσει, έχει προβλεφθεί η κατάσταση μη ομαλής λειτουργίας του λογισμικού, ώστε να μην προκαλείται πρόβλημα στη βάση δεδομένων.

12 Το πείραμα

Εκτός από την ανάλυσή μας πάνω στον τρόπο λειτουργίας και οργάνωσης των τεχνικών του ακραίου προγραμματισμού, είναι σημαντικό να αποδείξουμε τα οφέλη, που παρουσιάζονται κατά τη χρήση κάποιας ευέλικτης προγραμματιστικής μεθοδολογίας. Για το συγκεκριμένο σκοπό, θα δημιουργήσουμε ένα πολύ απλό πρόγραμμα ERP και συγκεκριμένα μια βασική λειτουργία της αποθήκης, όπως είναι η καταγραφή. Στο σημείο αυτό, θα πρέπει να τονίσουμε ότι τα αποτελέσματα του συγκεκριμένου πειράματος δεν αποτελούν τεκμήρια, όσον αφορά την αποτελεσματικότητα του ακραίου προγραμματισμού και κατ' επέκταση του ευέλικτου προγραμματισμού, αφού οι συνθήκες διεξαγωγής του πειράματος δεν αντιπροσωπεύουν το

πραγματικό-επιχειρηματικό τομέα. Για το λόγο αυτό, παρουσιάζονται οι παρακάτω σημαντικές παραδοχές που καθορίζουν σε μεγάλο βαθμό την πορεία του πειράματος.

Βασική προϋπόθεση του ευέλικτου προγραμματισμού είναι ότι κάθε ομάδα θα πρέπει να αποτελείται από τουλάχιστον οκτώ με δέκα άτομα. Η συγκεκριμένη παρατήρηση, σε συνδυασμό με το ότι μόλις δύο άτομα συμμετείχαν στο συγκεκριμένο πείραμα, μας προτρέπει στη διαμοίραση περισσοτέρων του ενός ρόλου καθήκοντος σε κάθε φοιτητή. Ο πρώτος φοιτητής ανέλαβε τα καθήκοντα του:

- Διαχειριστή προϊόντος.
- Διαχειριστή έργου.
- Προγραμματιστή.

Ο δεύτερος φοιτητής ανέλαβε τα παρακάτω καθήκοντα:

- Του χρήστη.
- Του σχεδιαστή.
- Του ελεγκτή.
- Του ειδικού τομέα (μέσα από τα καθήκοντα του χρήστη).

Στο σημείο αυτό, θα πρέπει να τονίσουμε ότι οι παραπάνω ρόλοι αρκετές φορές εναλλάσσονταν, ώστε να επιτευχθούν όσο το δυνατόν πλησιέστερα αποτελέσματα με την πραγματικότητα. Για παράδειγμα, ο πρώτος φοιτητής κατά περιόδους αναλάμβανε για μικρό χρονικό διάστημα τα καθήκοντα του πελάτη, όταν ο δεύτερος σχεδίαζε το σύστημα αλληλεπίδρασης του λογισμικού. Επιπροσθέτως, λόγω του μικρού αριθμού ατόμων που συμμετείχαν στο πείραμα, δεν επιτεύχθηκε πτυχή του πειράματος που αφορά τις ομάδες, δηλαδή την εναλλαγή ατόμων σε διαφορετικές ομάδες ή τον τρόπο καταμερισμού εργασίας σε αυτές.

Έχουμε αναφερθεί συχνά, στην τεχνική του προγραμματισμού σε ζεύγη, καθώς και στα αποτελέσματα της χρήσης του. Δεδομένου όμως του μικρού χρονικού πλαισίου, και του μειωμένου προσωπικού, η χρήση της έγινε σε σχετικά περιορισμένο βαθμό. Το αποτέλεσμα είναι μη εκτενής αναφορά για τα αποτελέσματα της χρήσης της συγκεκριμένης τεχνικής, καθώς και η επιφύλαξή μας για τις ευεργετικές ικανότητες που προσφέρει. Αυτή είναι και η δεύτερη μας παραδοχή. Αν και υπάρχουν κι άλλες παραδοχές, θα τονισθούν κατά τη διάρκεια παρουσίασης του πειράματος, για την επίτευξη ευκολότερης κατανόησης τους.

12.1 Η πρώτη Επανάληψη

12.1.1 Πρώτη μέρα

Ήρθε η ώρα να ξεκινήσει το πείραμα. Όμως από πού; Κατ' αρχάς, δε θα δημιουργήσουμε ομάδες, παρά μόνο θα κατανέμουμε τα καθήκοντα του κάθε φοιτητή, όπως αναφέρθηκαν πρωτύτερα. Μετά από τη συγκεκριμένη ενέργεια, είμαστε έτοιμοι να χρησιμοποιήσουμε τις τεχνικές του ακραίου προγραμματισμού ξεκινώντας φυσικά, αφού γνωρίζουμε ότι θέλουμε να κατασκευάσουμε μερικές λειτουργίες των ERP, από το παιχνίδι της κυκλοφορίας. Πριν όμως το αναλύσουμε, θα πρέπει να τονίσουμε πως αποφασίστηκε από την ομάδα ανάπτυξης λογισμικού, ότι θα εργάζονται καθημερινώς μόλις δύο ώρες, κάτι που αποτελεί την επόμενη παραδοχή μας.

Στο πρώτο στάδιο της προαναφερθείσας τεχνικής, ο πελάτης σε συνεργασία με το διαχειριστή του προϊόντος αποφασίζουν για τις λειτουργίες, που θα ενσωματωθούν στο επιθυμητό λογισμικό. Λόγω της μικρής έκτασης του πειράματος, η συγκεκριμένη διαδικασία διήρκησε μόλις δέκα λεπτά, επιφέροντας τις παρακάτω ιστορίες πελάτη:

1. Παραστατικά
2. Απογραφές
3. Εισαγωγή Χρηστών
4. Διαγραφή Χρηστών
5. Τροποποίηση Χρηστών
6. Καταστήματα
7. Εισαγωγή προϊόντων
8. Διαγραφή προϊόντων
9. Τροποποίηση προϊόντων

Με μια πρώτη ματιά, φαίνεται ότι η δημιουργία των ιστοριών πελάτη δεν είναι η ορθότερη, όμως ο ευέλικτος προγραμματισμός μας διορθώνει, όταν θα εκτιμηθούν οι χρόνοι της κάθε ιστορίας πελάτη που χρειάζεται για την πλήρη ολοκλήρωση της. Το επόμενο βήμα είναι η εύρεση της σημαντικότητας της κάθε ιστορίας πελάτη. Κατά τη συγκεκριμένη διαδικασία, ο ένας φοιτητής είχε αναλάβει τα καθήκοντα του πελάτη, ενώ ο δεύτερος του διαχειριστή προϊόντος ώστε η χρήση του σταδίου αυτού να γίνει με βάση τις πραγματικές καταστάσεις. Μετά από είκοσι λεπτά, το αποτέλεσμα ήταν:

id	όνομα	περιγραφή	Προτεραιότητα	εκτίμηση	σημειώσεις
1	Παραστατικά		55		
2	Απογραφές		35		
3	Εισαγωγή Χρηστών		30		
4	Διαγραφή Χρηστών		25		

5	Τροποποίηση Χρηστών		27		
6	Καταστήματα		10		
7	Εισαγωγή προϊόντων		40		
8	Διαγραφή προϊόντων		45		
9	Τροποποίηση προϊόντων		42		

Οι αριθμοί, που είναι στη στήλη προτεραιότητα, είναι η σημαντικότητα της κάθε μίας από αυτές και φυσικά, όσο πιο μεγάλος ο αριθμός, τόσο πιο σημαντική θεωρείται η κάθε ιστορία. Μια ερώτηση που γεννάται αφορά τις μεγάλες αριθμητικές διαφορές στη σημαντικότητα. Για παράδειγμα, γιατί να μην ήταν συνεχόμενοι οι αριθμοί; Η απάντηση στη συγκεκριμένη ερώτηση είναι πολύ απλή, εάν αναλογισθούμε, ότι οι ευέλικτες προγραμματιστικές μεθοδολογίες, επιτρέπουν στον πελάτη την άμεση τροποποίηση των προδιαγραφών του προγράμματος, κάτι που συνήθως συνεπάγεται της δημιουργίας μιας νέας ιστορίας πελάτη. Αυτή η ιστορία θα έχει με τη σειρά της κάποιο αριθμό σημαντικότητας, οπότε θα πρέπει να τοποθετηθεί στη σωστή σειρά. Ο αριθμός της όμως, θα συμπίπτει με άλλης ιστορίας πελάτη, οπότε είχαμε σπατάλη χρόνου για την τροποποίηση αρκετών ιστοριών κι αν αναλογισθούμε τη συχνότητα τροποποίησης των προδιαγραφών, μια τέτοια κίνηση δεν είναι αποδεκτή. Έτσι, ανάμεσα στις ιστορίες μας, έχουμε σχετικά μεγάλες διαφορές στον αριθμό της σημαντικότητας, ώστε ανά πάσα στιγμή να τοποθετηθεί μια νέα ιστορία πελάτη.

Αφού έχουν εκτιμηθεί όλες, τοποθετούνται σε μία σειρά με βάση τη σημαντικότητά τους, κάτι που θα εξοικονομήσει αρκετό χρόνο στην επιλογή των ιστοριών προς υλοποίηση, αφού πρώτα επιλέγονται οι πιο σημαντικές.

Το τελευταίο βήμα της συγκεκριμένης τεχνικής είναι η εκτίμηση του χρόνου των ιστοριών για την ολοκλήρωσή τους. Όπως είναι λογικό, για να επιτευχθεί το συγκεκριμένο εγχείρημα, ο φοιτητής-πελάτης συναντήθηκε με τον φοιτητή-προγραμματιστή, για να αποφασίσουν από κοινού το χρόνο που θα χρειαστεί η κάθε ιστορία για την ολοκλήρωσή της. Ο χρόνος, όπως έχουμε επισημάνει, θα μετρηθεί με βάση τις ημέρες κι όχι τις ώρες που απαιτούνται για την ολοκλήρωση της κάθε ιστορίας ξεχωριστά. Όμως, για να επιτύχουμε πλησιέστερες προσεγγίσεις του χρόνου ολοκλήρωσης, απαιτείται αρκετός χρόνος ώστε να καθορισθούν οι διαστάσεις της κάθε ιστορίας, δηλαδή ο πελάτης επεξηγεί το τι επιθυμεί από την κάθε ιστορία. Λόγω του ότι η ομάδα μας είναι αρκετά μικρή, οι διαφωνίες μεταξύ προγραμματιστών και χρηστών ήταν αρκετά λιγότερες από ό,τι σε ένα επαγγελματικό περιβάλλον. Κατά τη διαδικασία αυτή, εκτιμούμε τις πιο σημαντικές λειτουργίες, οι οποίες εφόσον χρειάζονται πάνω από μισή μέρα η κάθε μία για την ολοκλήρωσή της, παραμένουν ως έχουν δηλαδή δεν τις συγχωνεύουμε. Παρακάτω παρουσιάζουμε τις ιστορίες πελάτη με τις εκτιμήσεις τους:

id	όνομα	περιγραφή	Προτεραιότητα	εκτίμηση	σημειώσεις
1	Παραστατικά		55	6	
2	Απογραφές		35	3	
3	Εισαγωγή Χρηστών		30	1,5	
4	Διαγραφή Χρηστών		25	1,5	
5	Τροποποίηση Χρηστών		27	1,5	
6	Καταστήματα		10	1	

7	Εισαγωγή προϊόντων		40	1	
8	Διαγραφή προϊόντων		45	1	
9	Τροποποίηση προϊόντων		42	1	

Παρατηρούμε ότι η πρώτη ιστορία πελάτη χρειάζεται έξι μέρες για να ολοκληρωθεί. Αυτό μεταφράζεται, ότι με δύο προγραμματιστές θα χρειασθούμε τρείς ημέρες, όσες ακριβώς διαρκεί και η επανάληψη μας που έχουμε συμφωνήσει. Η συγκεκριμένη κατάσταση δεν είναι αποδεκτή γιατί:

1. Εάν δεν ολοκληρωθεί η ιστορία πελάτη, τότε δε θα παραδώσουμε στο τέλος της επανάληψης έναν έτοιμο και πλήρως χρησιμοποιήσιμο κώδικα.
2. Η μη ολοκλήρωση σημαίνει μηδενική ταχύτητα ανάπτυξης, άρα το έργο δεν αναπτύχθηκε καθόλου.

Εξαιτίας των προαναφερθέντων λόγων, έχουμε δύο επιλογές. Ή να διασπάσουμε τη συγκεκριμένη ιστορία πελάτη σε δύο μικρότερες, ή να μεγαλώσουμε το χρόνο κατά τον οποίο θα παραδίδουμε τον έτοιμο κώδικα, κάτι που δεν είναι και η πλέον αποδοτικότερη λύση. Έτσι, την ιστορία πελάτη που ονομάζεται «παραστατικά», τη διασπούμε στα παραστατικά και στα είδη παραστατικών. Για το λόγο αυτό, πρέπει εκ νέου να υπολογίσουμε τη σημαντικότητα της νέας ιστορίας πελάτη και να υπολογίσουμε το χρόνο για την ολοκλήρωσή της. Η σημαντικότητα της ιστορίας αυτής καθορίστηκε στο πενήντα δύο, ενώ ο χρόνος ολοκλήρωσής της σε δύο μέρες. Το σύνολο της διαδικασίας διήρκησε σαράντα λεπτά, οπότε απέμειναν πενήντα λεπτά για τη συγκεκριμένη μέρα.

Η επόμενη τεχνική που χρησιμοποιήθηκε είναι ο σχεδιασμός της επανάληψης. Κατά την τεχνική αυτή, πρέπει να καθορισθεί η ταχύτητα

ανάπτυξης(velocity), η οποία υπολογίζεται με βάση τη συγκέντρωση των ατόμων που συμμετέχουν στην ανάπτυξη λογισμικού, και την προηγούμενη ταχύτητα ανάπτυξης. Όπως γίνεται κατανοητό, δε γνωρίζουμε κανέναν από τους δύο αριθμούς κατά την πρώτη επανάληψη, οπότε θα πρέπει να υποθέσουμε έναν αριθμό. Είναι η πρώτη επανάληψη, οπότε δε μας επηρεάζει σε μεγάλο βαθμό η συγκεκριμένη επιλογή, αλλά είναι σωστό να προσεγγίσουμε, κατά το δυνατόν, τη λογικότερη τιμή. Στο πείραμά μας επιλέξαμε τον αριθμό έξι, ώστε να μπορέσουμε να ενσωματώσουμε τις δύο πρώτες ιστορίες πελάτη στην επανάληψη μας. Εάν είχαμε επιλέξει, για παράδειγμα, τον αριθμό τέσσερα, τότε θα μπορούσαμε να επιλέξουμε μόνο την πρώτη ιστορία πελάτη στην επανάληψή μας, κάτι που μπορούσε να αποβεί μοιραίο, κατά τον ίδιο τρόπο, πριν τη διάσπαση της αρχική ιστορίας πελάτη. Όπως είναι λογικό, η ταχύτητα ανάπτυξης που έχουμε επιλέξει, δε θα επιτευχθεί διότι έχουμε δύο προγραμματιστές και η κάθε επανάληψη διαρκεί για τρεις μέρες άρα 2 * 3 = 6. Όμως, δεδομένου ότι, εκτός από την ανάπτυξη του λογισμικού θα πρέπει να εκτελεστούν κι άλλες διεργασίες, αλλά και ότι απαιτείται πλήρης συγκέντρωση από τους προγραμματιστές, καταλήγουμε στο συμπέρασμα, ότι είναι ανέφικτη η επίτευξή του. Θα επηρεάσει η λανθασμένη μας επιλογή της ταχύτητας ανάπτυξης τη δημιουργία του λογισμικού; Αυτή η ερώτηση θα απαντηθεί κατά την εκτέλεση του πειράματος.

Αφού έχουμε επιλέξει την ταχύτητα ανάπτυξης, και σύμφωνα με αυτήν και τη σημαντικότητα, επιλέγουμε τις ιστορίες πελάτη που θα αναπτύξουμε κατά τη συγκεκριμένη επανάληψη. Έτσι, παρατηρούμε ότι η ιστορία πελάτη που είναι πιο σημαντική, τα παραστατικά, χρειάζεται τέσσερις ημέρες για την ολοκλήρωση της. Άρα, μας απομένουν 6 – 4=2 πόντοι, που χρησιμοποιούνται για την επιλογή μιας άλλης ιστορίας πελάτη, που θα είναι τα είδη των παραστατικών, αφού καθορίζονται σαν τη δεύτερη πιο σημαντική ιστορία και χρειάζεται δύο πόντους ακριβώς για την ολοκλήρωσή της. Έτσι, μετά από δέκα λεπτά, γνωρίζαμε την ταχύτητα ανάπτυξης και τις ιστορίες πελάτη που θα υλοποιηθούν. Ποιο είναι το επόμενο βήμα πριν την υλοποίηση; Αυτό είναι η διάσπαση των ιστοριών σε προγραμματιστικές λειτουργίες και η εκτίμηση της κάθε μιας από αυτές. Η διαδικασία αυτή διήρκησε μόλις δέκα λεπτά και

λόγω του ότι δεν υπάρχουν διαφορετικές προγραμματιστικές ομάδες, δε χρειάσθηκε να διαμοιραστεί ο φόρτος εργασίας. Από την ιστορία πελάτη, με το όνομα παραστατικά, έχουμε τις παρακάτω προγραμματιστικές λειτουργίες:

1. Δημιουργία παραστατικών (0,5)
2. Διαγραφή παραστατικών (0,5)
3. Δημιουργία βάσης δεδομένων (2)
4. Δημιουργία περιβάλλοντος αλληλεπίδρασης (1)

Οι αριθμοί παρουσιάζουν τις μέρες που χρειάζεται η κάθε μία προγραμματιστική λειτουργία για την ολοκλήρωσή της και εάν τις αθροίσουμε, θα έχουμε τον αριθμό τέσσερα που είναι και ο χρόνος ολοκλήρωσης της ιστορίας πελάτη. Αντίστοιχα, για την ιστορία πελάτη με το όνομα «είδη παραστατικών», θα έχουμε:

1. Δημιουργία πίνακα στη βάση δεδομένων (1)
2. Αλλαγές στο περιβάλλον αλληλεπίδρασης (1)

Αφού έχουμε ολοκληρώσει κι αυτό το βήμα, θα πρέπει στα τριάντα λεπτά που απομένουν να ξεκινήσουμε την κωδικοποίηση και λόγω του μικρού χρόνου, αποφασίστηκε η δημιουργία περιβάλλοντος αλληλεπίδρασης, μιας και είναι εύκολη και ανήκει στην πρώτη ιστορία πελάτη, που πρέπει να ολοκληρωθεί πριν τη δεύτερη. Ο πρώτος φοιτητής ανέλαβε το ρόλο του πελάτη, ενώ ο δεύτερος του σχεδιαστή. Όμως, ο χρόνος της μισής ώρας δεν ήταν αρκετός. Για το λόγο αυτό, θεωρούμε πως δεν αναπτύχθηκε καθόλου το πρόγραμμα την πρώτη μέρα, αλλά έγινε μια αρκετά καλή χρήση του ακραίου προγραμματισμού. Έχουμε αναφέρει πως σε αρχάριες ομάδες, όπως η δική μας, θα χρειασθούν αρκετές επαναλήψεις ώστε να επιτύχουν τον επιθυμητό ρυθμό της επανάληψης. Ωστόσο, από την πρώτη μέρα η συγκεκριμένη προγραμματιστική μεθοδολογία κατάφερε να οργανώσει σε ικανοποιητικό βαθμό τον τρόπο ανάπτυξης του έργου. Παρακάτω παρουσιάζεται ο πίνακας της επανάληψής μας κατά τον τέλος της συγκεκριμένης μέρας:

Δεν έχουν ξεκινήσει	Έχουν ξεκινήσει	Έχουν τελειώσει	Μονάδες
παραστατικά			
Δημιουργία παραστατικών (0,5)			
Διαγραφή παραστατικών (0,5)			
Δημιουργία βάσης δεδομένων (2)			
	Δημιουργία περιβάλλοντος αλληλεπίδρασης (1)		
Είδη Παραστατικών			
Δημιουργία πίνακα βάσεων δεδομένων (1)			
Αλλαγές στο περιβάλλον αλληλεπίδρασης (1)			

Περιληπτικά, ο χρόνος της ομάδας αναλώθηκε στις παρακάτω λειτουργίες:

- Σχέδιο Κυκλοφορίας (70 λεπτά)
    1. Καθορισμός λειτουργιών που επιθυμεί ο πελάτης (10 λεπτά).
    2. Καθορισμός σημαντικότητας των ιστοριών πελάτη (20 λεπτά).

3. Ανάλυση των ιστοριών πελάτη και καθορισμός χρόνου υλοποίησης (40 λεπτά).

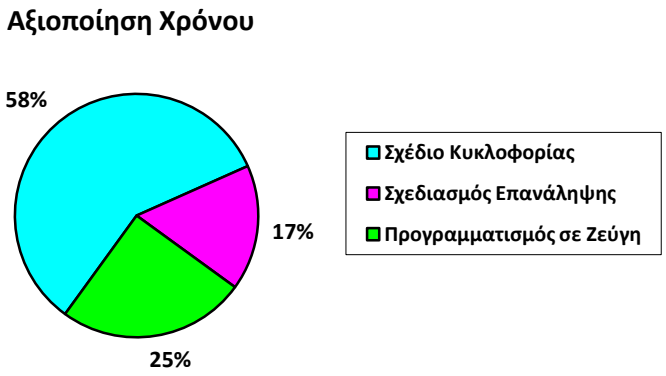
- Σχεδιασμός Επανάληψης (20 λεπτά)

1. Επιλογή ταχύτητας ανάπτυξης(2 λεπτά).
2. Επιλογή ιστοριών πελάτη προς υλοποίηση (8 λεπτά).
3. Διάσπαση ιστοριών πελάτη σε προγραμματιστικές λειτουργίες και εκτίμησή τους (10 λεπτά).

- Προγραμματισμός σε Ζεύγη (30 λεπτά)

1. Δημιουργία περιβάλλοντος αλληλεπίδρασης (30 λεπτά).

Στο επόμενο διάγραμμα μπορούμε να δούμε πώς αξιοποιήθηκε ο χρόνος, σε διάγραμμα πίτας, ώστε να γίνει εύκολα εμφανές το ποσοστό του χρόνου που αναλώθηκε σε κάθε διαδικασία.



12.1.2 Δεύτερη μέρα

Κανονικά, σε καθημερινή βάση θα πρέπει η ομάδα να συγκεντρώνεται και να συζητά για τις εργασίες που έγιναν, ποια προβλήματα συνάντησαν και ποιες οι σημερινές ενέργειές τους. Όμως, στη δική μας περίπτωση, τα άτομα

από τα οποία απαρτίζεται η ομάδα ανάπτυξης είναι μόλις «δύο», οπότε οι λεγόμενες όρθιες συναντήσεις δεν εφαρμόζονται. Άρα, τη δεύτερη μέρα αφιερώνεται συνολικά ένα πεντάλεπτο για την επιλογή των προγραμματιστικών λειτουργιών, με τις οποίες θα ενασχοληθεί η ομάδα μας.

Επειδή η ανάπτυξη του περιβάλλοντος αλληλεπίδρασης είχε ξεκινήσει την προηγούμενη μέρα, οι δύο φοιτητές ασχολήθηκαν πρώτα με αυτή και την ολοκλήρωσαν μετά από είκοσι λεπτά. Ήταν εμφανές πως η ύπαρξη του πελάτη, κατά τη διαδικασία αυτή, αν και προσέθεσε περισσότερο χρόνο για την ολοκλήρωση της συγκεκριμένης πτυχής, προσέθεσε σιγουριά για την αποδοχή του περιβάλλοντος αλληλεπίδρασης, άρα οι πιθανότητες μελλοντικής αλλαγής του είναι ακόμα μικρότερες. Για αυτό το λόγο, θεωρούμε πως τα οφέλη της ύπαρξης του πελάτη είναι πολλά περισσότερα από αυτά που θα είχαμε σε αντίθετη περίπτωση.

Στα ενενήντα λεπτά που απέμειναν, θα έπρεπε να δημιουργηθεί μια κλάση, που θα αναλάμβανε όχι μόνο την επικοινωνία με τη βάση δεδομένων, αλλά θα εξασφάλιζε και την ορθή μεταφορά των αποτελεσμάτων. Όπως γίνεται αντιληπτό, η παρουσία του πελάτη δεν ήταν απαραίτητη, οπότε χρησιμοποιήθηκε ο προγραμματισμός σε ζεύγη, κατά τον οποίο ανά τριάντα λεπτά είχαμε εναλλαγή ρόλων. Κατά τη συγκεκριμένη διαδικασία, η εύρεση σφαλμάτων κώδικα ήταν αρκετά εύκολη, λόγω του ότι δύο προγραμματιστές έλεγχαν τον κώδικα ταυτόχρονα. Ακόμη, η εναλλαγή των ρόλων συνετέλεσε σε μια πιο ξεκούραστη ανάπτυξη του κώδικα. Όπως είναι λογικό, κατά τη συγκεκριμένη διαδικασία, χρησιμοποιήθηκε η τεχνική της αναδόμησης κώδικα, όποτε παρουσιαζόταν αναγκαίο. Στα τελευταία δέκα λεπτά της συγκεκριμένης μέρας, ο ένας εκ των δύο φοιτητών, ανέλαβε την κατασκευή ενός μικρού προγράμματος, στο οποίο ενσωμάτωσε τη νέα μας κλάση και την έλεγξε, ώστε να διαπιστωθεί η καλή της λειτουργία και να θεωρηθεί έτοιμη. Κατά τη διαδικασία αυτή, βρέθηκε ένα μικρό σφάλμα κώδικα, το οποίο διορθώθηκε άμεσα.

Μπορούμε εύκολα να παρατηρήσουμε ότι τη συγκεκριμένη μέρα η ανάπτυξη του κώδικα επιτεύχθηκε σε ικανοποιητικό ρυθμό. Για να

ολοκληρωθεί και η δεύτερη προγραμματιστική λειτουργία, μας απομένει ο σχεδιασμός της βάσης δεδομένων και η δημιουργία της. Ακόμα, απομένουν δυο αρκετά μικρές λειτουργίες που αφορούν τη δημιουργία και τη διαγραφή παραστατικών όμως χρησιμοποιούν τη βάση δεδομένων μας. Γι' αυτό το λόγο, θέλαμε πρώτα να εκπληρώσουμε τη σωστή επικοινωνία με τη βάση μας. Εφόσον κατά τη συγκεκριμένη μέρα ολοκληρώθηκε εξ ολοκλήρου μόνο το σύστημα δημιουργίας περιβάλλοντος αλληλεπίδρασης, το οποίο όπως αναφέραμε χρειάζεται μόλις μία μέρα για την ολοκλήρωση του, μας απομένουν 6 – 1 = 5 πόντοι για ολοκλήρωση της πρώτης επανάληψης. Έτσι, ο πίνακας της επανάληψης μας ήταν:

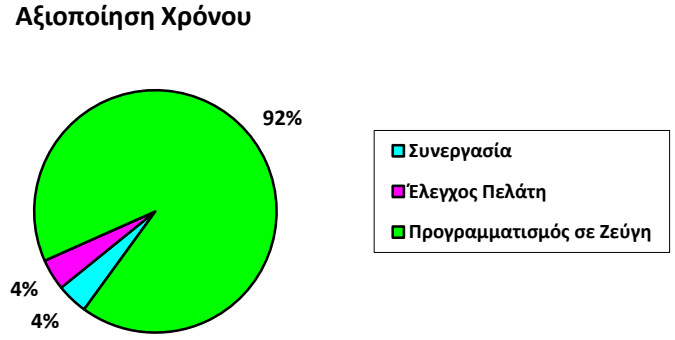
Δεν έχουν ξεκινήσει	Έχουν ξεκινήσει	Έχουν τελειώσει	Μονάδες
παραστατικά			
Δημιουργία παραστατικών (0,5)			
Διαγραφή παραστατικών (0,5)			
	Δημιουργία βάσης δεδομένων (2)		
		Δημιουργία περιβάλλοντος αλληλεπίδρασης (1)	1
Είδη Παραστατικών			
Δημιουργία πίνακα βάσεων δεδομένων			

(1)			
Αλλαγές στο περιβάλλον αλληλεπίδρασης (1)			

Τη δεύτερη μέρα έγιναν τα εξής:

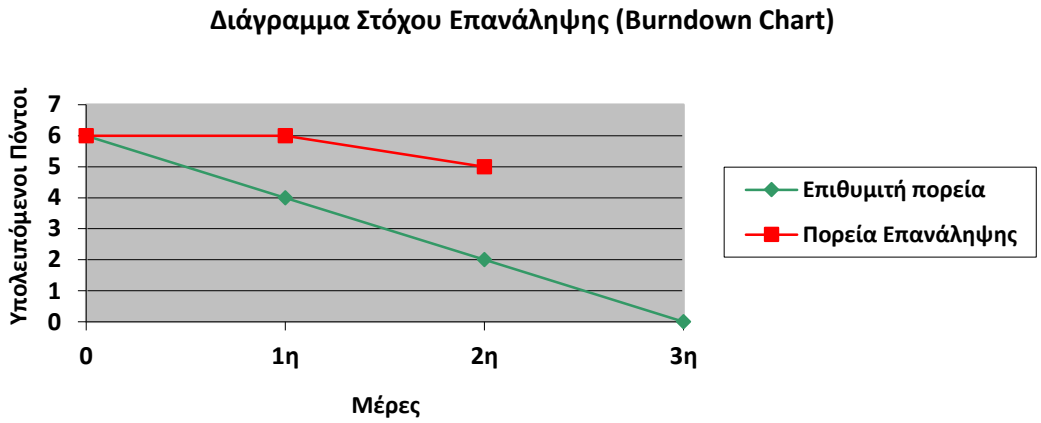
- Συνεργασία (5 λεπτά)
    1. Όρθια συνάντηση για πορεία του έργου και καταμερισμό εργασιών(5 λεπτά).
- Προγραμματισμός σε ζεύγη (110 λεπτά)
    1. Δημιουργία περιβάλλοντος αλληλεπίδρασης (20 λεπτά).
    2. Δημιουργία Βάσης δεδομένων (90 λεπτά).
- Έλεγχος πελάτη (5 λεπτά).
    1. Εύρεση σφάλματος στο περιβάλλον αλληλεπίδρασης.
    2. Διόρθωση σφάλματος.

Το διάγραμμα πίτας, που αφορά τα ποσοστά κατανάλωσης του χρόνου, διαμορφώνεται ως εξής:



Ο έλεγχος πελάτη, είναι αρκετά μικρότερος, από όσο θα έπρεπε. Αυτό έγινε, διότι ο μόνος έλεγχος που μπορεί να γίνει είναι του περιβάλλοντος αλληλεπίδρασης στο οποίο ο πελάτης μπορεί να κάνει περιορισμένες λειτουργίες.

Εφόσον έχουμε ολοκληρώσει την πρώτη λειτουργία, μπορούμε να ενημερώσουμε το διάγραμμα στόχου επανάληψης(burndown chart), που μας παρουσιάζει με «μια ματιά» την πορεία της επανάληψης.



Όπως παρατηρούμε, η πορεία του έργου μας δεν είναι η επιθυμητή με βάση την ταχύτητα ανάπτυξης που έχουμε ορίσει, κάτι που είναι απολύτως λογικό και συναντάται συχνά κατά την πρώτη επανάληψη. Τη δεύτερη μέρα έπρεπε να έχουμε ολοκληρώσει 4 πόντους, όμως μόνο ένας έχει ολοκληρωθεί.

12.1.3 Τρίτη μέρα

Η συγκεκριμένη μέρα είναι και η τελευταία της επανάληψής μας, αφού έχουμε αποφασίσει ότι η κάθε επανάληψη θα περιλαμβάνει τρεις μέρες. Είναι προφανές ότι δε θα προλάβουμε να ολοκληρώσουμε και τις δύο ιστορίες πελάτη και για το λόγο αυτό ασχοληθήκαμε μόνο με την πρώτη. Στο ξεκίνημα της συγκεκριμένης μέρας, ο πρώτος φοιτητής ανέλαβε το σχεδιασμό και την

υλοποίηση της βάσης δεδομένων, ενώ ο δεύτερος την προγραμματιστική λειτουργία της διαγραφής παραστατικών.

Η διαγραφή παραστατικών δημιουργήθηκε σε είκοσι λεπτά, ενώ χρειάστηκε άλλα δέκα λεπτά για να γίνουν οι απαραίτητοι έλεγχοι σωστής λειτουργίας. Στο ίδιο χρονικό διάστημα, ο δεύτερος φοιτητής είχε σχεδιάσει και δημιουργήσει τη βάση δεδομένων, κάτι που μας αφαιρούσε άλλους δύο πόντους από το σύνολο εργασιών απομένοντας 2,5 ακόμα. Στο σημείο αυτό, εάν δεν ολοκληρωθεί και η προγραμματιστική λειτουργία της δημιουργίας παραστατικών, τότε η ανάπτυξη της σημαντικότερης ιστορίας πελάτη δε θα έχει τελειώσει. Αυτό, σημαίνει ότι κατά τη συγκεκριμένη επανάληψη, θα θεωρηθεί πως η ταχύτητα ανάπτυξης είναι μηδενική.

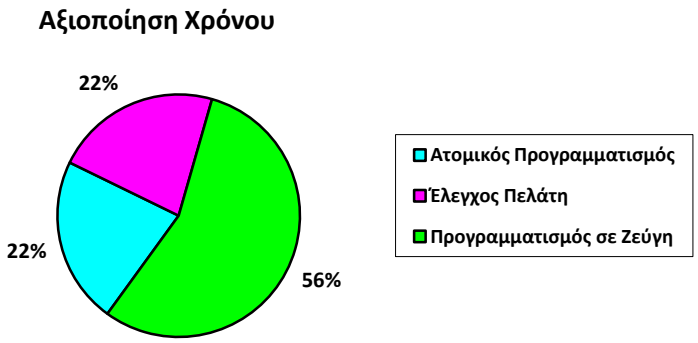
Έχοντας ολοκληρώσει τις άλλες προγραμματιστικές λειτουργίες, οι δύο φοιτητές με τη χρήση του προγραμματισμού σε ζεύγη, ολοκληρώνουν σε πενήντα λεπτά τη δημιουργία παραστατικών, συνεπώς και την ιστορία των παραστατικών που είναι και η πιο σημαντική σύμφωνα με τον πελάτη. Με τον τρόπο αυτό, κατοχυρώθηκαν οι τέσσερις πόντοι. Ωστόσο, η επόμενη ιστορία παραμένει ανολοκλήρωτη, όπως είχαμε προβλέψει. Στο σημείο αυτό μπορούμε να παρατηρήσουμε ότι οι προβλέψεις μας για το χρόνο που χρειάζεται η κάθε προγραμματιστική λειτουργία, ήταν σχεδόν αληθοφανές ασχέτως εάν η ταχύτητα ανάπτυξής μας ήταν μικρότερη από αυτό που είχαμε δηλώσει.

Έτσι την τελευταία μέρα της πρώτης επανάληψης έγιναν:

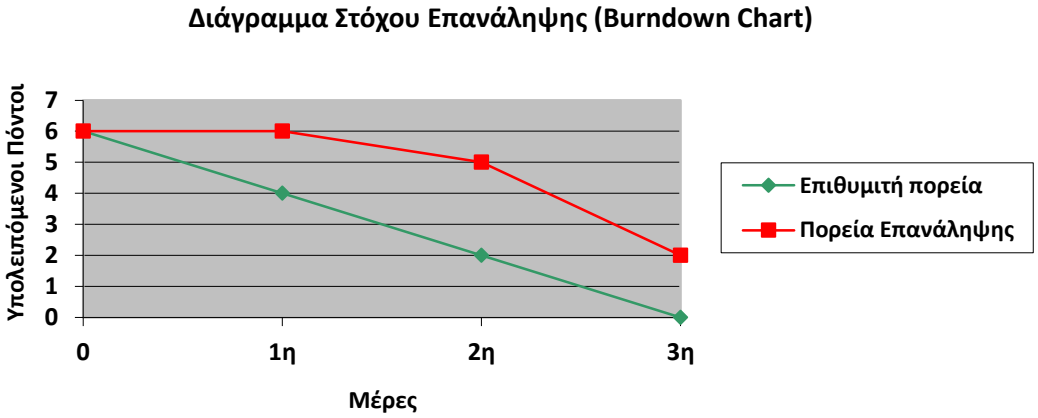
- Προγραμματισμός σε ατομικό επίπεδο (20 λεπτά).
- Έλεγχος πελάτη (20 λεπτά).
- Προγραμματισμός σε ζεύγη (50 λεπτά).

Όπως γίνεται αντιληπτό, το άθροισμα των λεπτών που αναγράφονται πρωτύτερα, δε συμπληρώνουν 2 ώρες. Θα μπορούσαμε να ξεκινήσουμε τη συγγραφή κώδικα για την επόμενη ιστορία πελάτη, όμως κάτι τέτοιο θα αλλοίωνε το αποτέλεσμα στον υπολογισμό της ταχύτητας ανάπτυξης, αφού δε

θα ολοκληρωνόταν. Το διάγραμμα κατανάλωσης του χρόνου παρουσιάζεται στο επόμενο διάγραμμα τύπου πίτας.

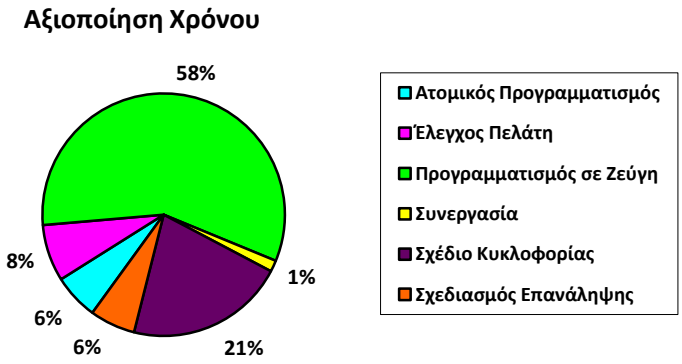


Εφόσον έχουμε ολοκληρώσει την πρώτη επανάληψη, μπορούμε να ενημερωθούμε από το διάγραμμα στόχου επανάληψης για το εάν επιτύχαμε το στόχο μας.



Η απάντηση φυσικά είναι αρνητική, καθώς μια ιστορία πελάτη δεν έχει ολοκληρωθεί. Όμως, λόγω του ότι ήταν η πρώτη επανάληψη, το συγκεκριμένο αποτέλεσμα ήταν αναμενόμενο, καθώς η επιλογή της ταχύτητας ανάπτυξης ήταν αυθαίρετη και δεν βασιζόταν σε παλαιότερες μετρήσεις.

Το επόμενο διάγραμμα παρουσιάζει συνολικά ποιές τεχνικές χρησιμοποιήθηκαν κατά την πρώτη επανάληψη, καθώς και πόσος χρόνος αφιερώθηκε σε κάθε μία.



Όπως είναι κατανοητό, η ομάδα αφιέρωσε αρκετό χρόνο σε προγραμματιστικές λειτουργίες με ποσοστό 63%. Όμως, σε επαγγελματικό επίπεδο, είναι προφανές πως το μέγεθος της ομάδας αυξάνει, οπότε οι υπόλοιπες λειτουργίες, που αφορούν τη συνεργασία και το συντονισμό της ομάδας, καταναλώνουν περισσότερο χρόνο καθώς γίνεται χρήση και άλλων τεχνικών του ευέλικτου προγραμματισμού. Καλό είναι να τονισθεί πως ο έλεγχος πελάτη καταλαμβάνει πολύ μικρό ποσοστό από όσο είναι το επιθυμητό. Αυτό συμβαίνει για δύο λόγους. Ο πρώτος αφορά το μέγεθος της ομάδας, ενώ ο δεύτερος την ολοκλήρωση μόνο μιας ιστορίας πελάτη.

12.2 Η Δεύτερη Επανάληψη

12.2.1 Πρώτη μέρα

Μια νέα επανάληψη ξεκινά. Όμως, τι πρέπει να υπολογίσουμε πρώτα πριν ξεκινήσουμε την κωδικοποίηση; Το βασικότερο βήμα είναι ο υπολογισμός της ταχύτητας ανάπτυξης, ώστε να επιλέξουμε τις ιστορίες πελάτη που θα υλοποιήσουμε κατά τη συγκεκριμένη επανάληψη. Για να υπολογίσουμε το συγκεκριμένο αριθμό θα πρέπει να υπολογίσουμε πρώτα το ποσοστό συγκέντρωσης από τον τύπο:

Προηγούμενη ταχύτητα ανάπτυξης / σύνολο ημερών εργασίας = ποσοστό συγκέντρωσης.

Η προηγούμενη ταχύτητα ανάπτυξης ήταν τέσσερα, αφού ολοκληρώθηκε η ιστορία των παραστατικών που είχε τέσσερις πόντους. Το σύνολο ημερών εργασίας είναι έξι αφού είχαμε δύο φοιτητές που δούλεψαν τρεις μέρες. Άρα, δύο επί τρία ίσον έξι. Έτσι, έχουμε 4/6=0,66 ή 66% ποσοστό συγκέντρωσης. Τώρα, για να βρούμε την καινούρια ταχύτητα ανάπτυξης, θα χρησιμοποιήσουμε τον τύπο:

ποσοστό συγκέντρωσης * σύνολο ημερών εργασίας = εκτιμώμενη ταχύτητα

Οπότε, έχουμε 0,66 * 6 = 4 που είναι η εκτιμώμενη μας ταχύτητα. Εάν κάποιος από τους δύο φοιτητές δεν εργαζόταν μία μέρα, τότε το σύνολο ημερών εργασίας θα άλλαζε από έξι σε πέντε, οπότε θα είχαμε 0,66 * 5 = 3,5 σαν εκτιμώμενη ταχύτητα.

Αφού έχουμε βρει την εκτιμώμενη ταχύτητά μας, σειρά έχει η επιλογή των ιστοριών που θα υλοποιηθούν, κατά την οποία το ένα άτομο αναλαμβάνει καθήκοντα πελάτη, ενώ το δεύτερο του προγραμματιστή. Εφόσον δεν ολοκληρώσαμε την ιστορία των ειδών παραστατικού, τότε είναι σίγουρο πως θα ενσωματωθεί στην επανάληψή μας, μειώνοντας έτσι τους υπολειπόμενους πόντους σε δύο, καθώς χρειάζεται δύο μέρες για την ολοκλήρωσή της. Η επόμενη πιο σημαντική ιστορία πελάτη είναι η διαγραφή των προϊόντων. Όμως, ο πελάτης αποφασίζει πως δε θέλει να διαγράφονται τα προϊόντα του, οπότε η συγκεκριμένη ιστορία απομακρύνεται και η ομάδα ανάπτυξής μας επιλέγει την επόμενη πιο σημαντική που είναι η τροποποίηση των προϊόντων και εκτιμά το χρόνο ολοκλήρωσής της σε μόλις μία μέρα. Αυτό έγινε μετά την επεξήγηση του πελάτη για τις προσδοκίες του. Έτσι λοιπόν, μας μένει μόλις ένας πόντος για να συμπληρώσουμε την ταχύτητα ανάπτυξης και όπως είναι φυσικό, μελετάμε την επόμενη ιστορία σε σημαντικότητα που είναι η εισαγωγή προϊόντων. Επειδή εκτιμήθηκε ότι θα χρειασθεί κι αυτή με τη σειρά της μόλις μία μέρα υλοποίησης, την εισάγαμε στον πίνακα της επανάληψης. Εάν όμως ήθελε περισσότερες τι θα κάναμε; Θα συνεχίζαμε να εκτιμούμε ιστορίες μέχρι

να βρούμε την πιο σημαντική, που να μπορούμε να υλοποιήσουμε στη συγκεκριμένη επανάληψη. Το σύνολο της διαδικασίας αυτής δεν κράτησε πάνω από είκοσι λεπτά κι ήμασταν σχεδόν έτοιμοι να ξεκινήσουμε την επανάληψή μας. Παρακάτω, παρουσιάζεται ο πίνακας του σχεδίου επανάληψης:

id	όνομα	περιγραφή	Προτεραιότητα	εκτίμηση	σημειώσεις
1	Παραστατικά		55	4	
10	Είδη Παραστατικών		52	2	
2	Απογραφές		35	3	
3	Εισαγωγή Χρηστών		30	1,5	
4	Διαγραφή Χρηστών		25	1,5	
5	Τροποποίηση Χρηστών		27	1,5	
6	Καταστήματα		10	1	
7	Εισαγωγή προϊόντων		40	1	
9	Τροποποίηση προϊόντων		42	1	

Σειρά έχει το να διασπάσουμε τις ιστορίες μας σε προγραμματιστικές λειτουργίες και να εκτιμήσουμε το χρόνο ολοκλήρωσης της κάθε μιας από αυτές. Μετά από δέκα λεπτά έχουμε τις παρακάτω προγραμματιστικές λειτουργίες με τις μέρες που θα χρειασθούν. Η ιστορία πελάτη με το όνομα «είδη παραστατικών» δεν τροποποιήθηκε και παρέμεινε ως έχει, αφού είχε

αναλυθεί κατά την προηγούμενη επανάληψη. Έτσι, δημιουργείται ο παρακάτω πίνακας επανάληψης:

Δεν έχουν ξεκινήσει	Έχουν ξεκινήσει	Έχουν τελειώσει	Μονάδες
Εισαγωγή προϊόντων			
Δημιουργία περιβάλλοντος αλληλεπίδρασης (0,5)			
Τροποποίηση βάσης δεδομένων (0,5)			
Τροποποίηση προϊόντων			
Τροποποίηση βάσης δεδομένων (0,5)			
Δημιουργία περιβάλλοντος αλληλεπίδρασης (0,5)			
Είδη Παραστατικών			
Δημιουργία πίνακα βάσεων δεδομένων (1)			
Αλλαγές στο			

περιβάλλον αλληλεπίδρασης (1)			

Στο συγκεκριμένο σημείο, θα πρέπει να επιλέξουμε τις προγραμματιστικές λειτουργίες που θα υλοποιήσουμε και θα διαμεριστούν οι εργασίες. Επειδή έχουμε αρκετό χρόνο, αλλά όχι τόσο ώστε να υλοποιήσουμε την πιο σημαντική ιστορία μας στη συγκεκριμένη επανάληψη, θα υλοποιήσουμε την εισαγωγή προϊόντος. Ο πρώτος φοιτητής αναλαμβάνει καθήκοντα πελάτη και παράλληλα τροποποιεί τη βάση δεδομένων, ενώ ο δεύτερος δημιουργεί το περιβάλλον αλληλεπίδρασης. Όταν ο πελάτης συμφώνησε με το αποτέλεσμα, τότε το περιβάλλον αλληλεπίδρασης είχε ολοκληρωθεί και είχε διαρκέσει τριάντα λεπτά. Αντιθέτως, η τροποποίηση της βάσης δεδομένων καθυστερούσε σε μεγάλο βαθμό και η χρησιμοποίηση και του δεύτερου ατόμου δεν κρίθηκε απαραίτητη, οπότε επιλέχθηκε η ανάπτυξη της δημιουργίας περιβάλλοντος αλληλεπίδρασης της ιστορίας τροποποίησης προϊόντος. Κατά τη συγκεκριμένη διαδικασία, ο φοιτητής που είχε αναλάβει την τροποποίηση της βάσης δεδομένων, συχνά αναλάμβανε καθήκοντα πελάτη ώστε να ολοκληρωθεί και η προγραμματιστική λειτουργία που αφορούσε την τροποποίηση προϊόντων. Μετά από τριάντα λεπτά είχαν ολοκληρωθεί και οι δύο προγραμματιστικές λειτουργίες, οπότε θα έπρεπε να ελεγχθούν. Για το λόγο αυτό, το άτομο που είχε αναλάβει την τροποποίηση της βάσης δεδομένων, αναλαμβάνει την ανάπτυξη της προγραμματιστικής λειτουργίας που απομένει για την ολοκλήρωση της τροποποίησης των προϊόντων, ενώ ο δεύτερος φοιτητής ελέγχει πλήρως τις λειτουργίες, οι οποίες αφορούν την εισαγωγή προϊόντων, στην οποία δε βρέθηκαν σφάλματα.

Λόγω του μικρού χρόνου που είχε απομείνει, δεν ολοκληρώθηκε η ιστορία πελάτη που αφορούσε την τροποποίηση των προϊόντων, οπότε η ομάδα ανάπτυξής μας συνέχισε την επόμενη μέρα με την τροποποίηση της βάσης δεδομένων. Συνολικά, στη συγκεκριμένη μέρα μειώθηκαν οι πόντοι μας κατά 1 + 0,5 = 1,5 πόντους, οπότε μας απομένουν 4 – 1,5 = 2,5 ακόμη για την

επίτευξη της εκτιμώμενης ταχύτητας ανάπτυξης. Έτσι, ο πίνακας της επανάληψής μας ήταν:

Δεν έχουν ξεκινήσει	Έχουν ξεκινήσει	Έχουν τελειώσει	Μονάδες
		Εισαγωγή προϊόντων	1
		Δημιουργία περιβάλλοντος αλληλεπίδρασης (0,5)	0,5
		Τροποποίηση βάσης δεδομένων (0,5)	0,5
Τροποποίηση προϊόντων			
	Τροποποίηση βάσης δεδομένων (0,5)		
		Δημιουργία περιβάλλοντος αλληλεπίδρασης (0,5)	0,5
Είδη Παραστατικών			
Δημιουργία πίνακα βάσεων δεδομένων (1)			

Αλλαγές στο περιβάλλον αλληλεπίδρασης (1)			

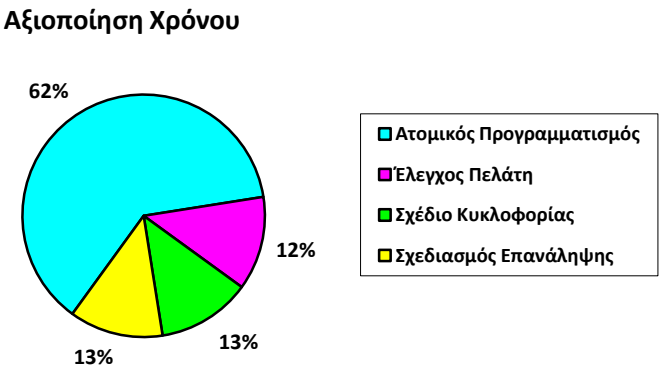
Αναλυτικά η ομάδα ασχολήθηκε με:

- Σχέδιο Κυκλοφορίας (15 λεπτά)
    1. Καθορισμός επιθυμιών πελάτη (15 λεπτά).
- Σχεδιασμός Επανάληψης (15 λεπτά)
    1. Υπολογισμός ταχύτητας ανάπτυξης και επιλογή ιστοριών πελάτη (5 λεπτά).
    2. Διάσπαση ιστοριών πελάτη σε προγραμματιστικές λειτουργίες και εκτίμησή τους (10 λεπτά).
- Προγραμματισμός σε ατομικό επίπεδο (90 λεπτά)
    1. Πρώτος Φοιτητής
        - Δημιουργία του περιβάλλοντος αλληλεπίδρασης για την εισαγωγή προϊόντων (30 λεπτά).
        - Δημιουργία του περιβάλλοντος αλληλεπίδρασης για την τροποποίηση προϊόντων (30 λεπτά).
    2. Δεύτερος Φοιτητής
        - Τροποποίηση της βάσης δεδομένων (60 λεπτά).
        - Τροποποίηση της βάσης δεδομένων για την ιστορία της τροποποίησης προϊόντος (30 λεπτά).
- Έλεγχος πελάτη (30 λεπτά)

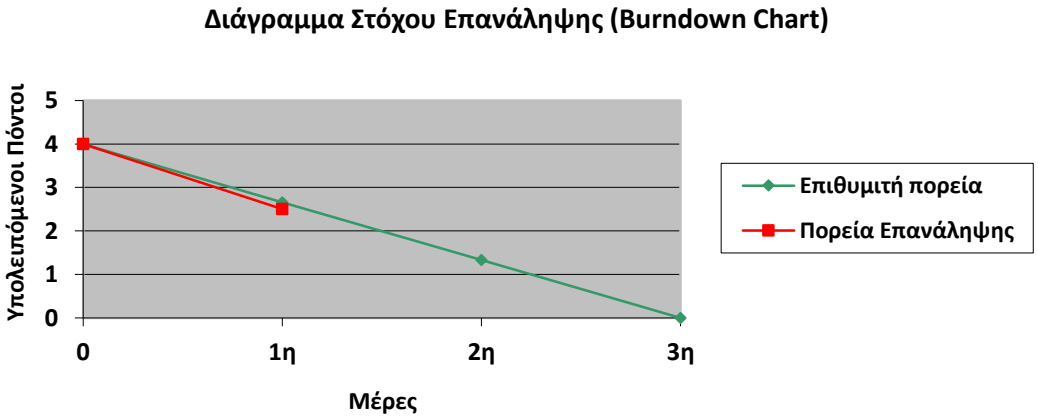
Στον προγραμματισμό, σε ατομικό επίπεδο, παρατηρούμε ότι αναγράφονται ενενήντα λεπτά έναντι εκατό πενήντα λεπτών. Αυτό συμβαίνει

διότι για μία ώρα οι φοιτητές προγραμμάτιζαν παράλληλα, ενώ μετά το πέρας του χρονικού διαστήματος αυτού, μόνο ο ένας συνέχισε τον προγραμματισμό και συγκεκριμένα για μισή ώρα.

Το διάγραμμα που αφορά την αξιοποίηση του χρόνου είναι:



Εφόσον έχουμε ολοκληρώσει προγραμματιστικές λειτουργίες, μπορούμε να ενημερώσουμε το διάγραμμα στόχου επανάληψης, στο οποίο διαφαίνεται πως η ανάπτυξη του λογισμικού γίνεται βάσει των προβλέψεών μας.



12.2.2 Δεύτερη μέρα

Κατά τη δεύτερη μέρα, όπως αναφέραμε και πρωτύτερα, έπρεπε να συνεχισθεί η τροποποίηση βάσης δεδομένων της τροποποίησης προϊόντος, καθώς δεν είχε ολοκληρωθεί. Το δεύτερο άτομο όμως, έπρεπε να αναλάβει την ανάπτυξη της ιστορίας των ειδών παραστατικών και πιο συγκεκριμένα τη δημιουργία νέου πίνακα βάσεων δεδομένων.

Μετά από σαράντα λεπτά, είχαν ολοκληρωθεί και οι δύο προγραμματιστικές λειτουργίες και φυσικά έπρεπε να ελεγχθούν και μάλιστα από άτομο που δε συμμετείχε στη συγγραφή του κώδικα. Τι γίνεται όμως, εάν βρεθεί κάποια δυσλειτουργία; Στη συγκεκριμένη περίπτωση βρέθηκε σφάλμα κατά την τροποποίηση μερικών προϊόντων, οπότε απαιτείται η άμεση επιδιόρθωση της συγκεκριμένης λειτουργίας και μάλιστα ο επανέλεγχος της. Την επιδιόρθωση την ανέλαβε ο φοιτητής που είχε συγγράψει και το συγκεκριμένο κομμάτι κώδικα, σε συνεργασία πάντα με το φοιτητή που ανέδειξε το σφάλμα. Ο δεύτερος ενημέρωσε για τις ενέργειες στις οποίες προέβη, ώστε να ελαχιστοποιηθεί ο χρόνος ανεύρεσης του μη ορθού κώδικα. Όταν είχαν περάσει άλλα σαράντα λεπτά, όλα λειτουργούσαν κανονικά και χωρίς σφάλματα, οπότε έπρεπε να επιλεχθεί μια προγραμματιστική λειτουργία για ανάπτυξη. Αφού δεν υπήρχε άλλη διαθέσιμη, τότε οι αλλαγές στο περιβάλλον αλληλεπίδρασης εμφανίστηκαν στο προσκήνιο και λόγω του ότι τα περιθώρια για καθυστερήσεις και για καινούριες δυσλειτουργίες στενεύουν, χρησιμοποιήθηκε η τεχνική του προγραμματισμού σε ζεύγη, κατά την οποία ανά είκοσι λεπτά είχαμε εναλλαγή ρόλων.

Σύντομα, ο χρόνος για ανάπτυξη του λογισμικού ολοκληρώθηκε και οι πόντοι που μας απομένουν είναι ελάχιστοι. Επειδή μια μόνο προγραμματιστική λειτουργία μας απομένει για την ολοκλήρωση των ιστοριών πελάτη που έχουμε επιλέξει, και συγκεκριμένα έχουμε εκτιμήσει πως χρειάζεται μία μέρα για να λειτουργήσει ορθά, τότε μας απομένει ένας πόντος. Στο σημείο αυτό, μπορούμε εύκολα να παρατηρήσουμε ότι έχουμε μια ολόκληρη μέρα για υλοποίηση και θα πρέπει να την εκμεταλλευτούμε κατάλληλα. Έτσι, θα πρέπει να αποφασισθεί για τον αποδοτικότερο τρόπο κατανάλωσης του αυριανού ελεύθερου χρόνου, μετά φυσικά από την

ολοκλήρωση της ιστορίας, που αφορά τα είδη παραστατικών. Οι περιπτώσεις είναι δύο:

1. Να εισαχθεί στην επανάληψή μας μια νέα ιστορία πελάτη.
2. Να χρησιμοποιήσουμε την τεχνική της χαλάρωσης.

Μπορούμε εύκολα να αναλογισθούμε ότι εάν δε βρεθεί κάποια ιστορία πελάτη, που να μπορούμε να αναπτύξουμε σε τόσο μικρό χρονικό διάστημα, τότε η τεχνική της χαλάρωσης παρουσιάζεται σαν την πιο αξιόπιστη λύση. Στο τέλος της συγκεκριμένης μέρας, ο πίνακας επανάληψης είχε διαμορφωθεί ως εξής:

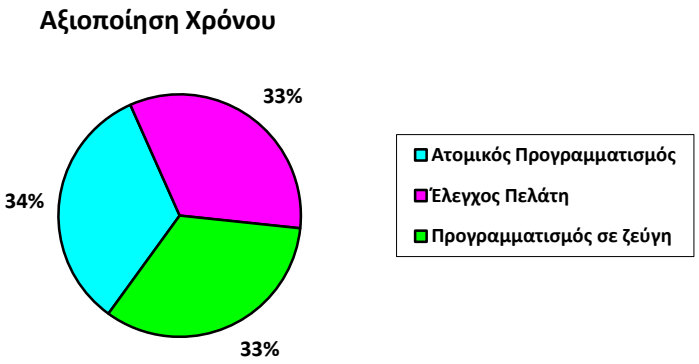
Δεν έχουν ξεκινήσει	Έχουν ξεκινήσει	Έχουν τελειώσει	Μονάδες
		Εισαγωγή προϊόντων	1
		Δημιουργία περιβάλλοντος αλληλεπίδρασης (0,5)	0,5
		Τροποποίηση βάσης δεδομένων (0,5)	0,5
		Τροποποίηση προϊόντων	
		Τροποποίηση βάσης δεδομένων (0,5)	0,5
		Δημιουργία περιβάλλοντος αλληλεπίδρασης	0,5

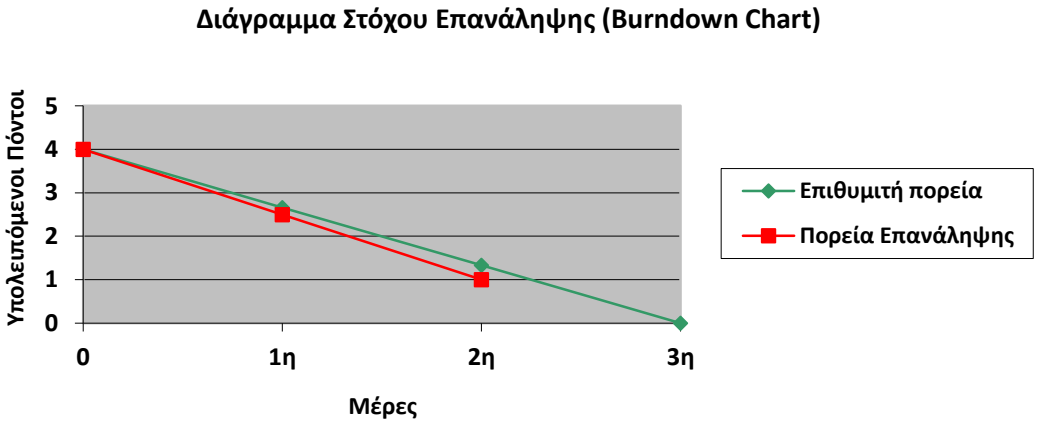
		(0,5)	
Είδη Παραστατικών			
		Δημιουργία πίνακα βάσεων δεδομένων (1)	1
	Αλλαγές στο περιβάλλον αλληλεπίδρασης (1)		

Οι ενέργειες που έγιναν τη συγκεκριμένη μέρα, ήταν:

- Προγραμματισμός σε ατομικό επίπεδο (40 λεπτά)
    1. Πρώτος Φοιτητής
        - Τροποποίηση βάσης δεδομένων (40 λεπτά).
    2. Δεύτερος Φοιτητής
        - Δημιουργία νέου πίνακα βάσεων δεδομένων (40 λεπτά).
- Έλεγχος πελάτη (40 λεπτά)
    1. Έλεγχος λειτουργιών της τροποποίησης προϊόντος(15 λεπτά).
    2. Επιδιόρθωση του σφάλματος (25 λεπτά).
- Προγραμματισμός σε ζεύγη (40 λεπτά).

Έτσι, τα σημερινά διαγράμματα αξιοποίησης χρόνου και στόχου επανάληψης διαμορφώνονται ως:





12.2.3 Τρίτη μέρα

Ξεκινώντας την τρίτη μέρα της επανάληψής μας, είχαμε να ολοκληρώσουμε την τελευταία, αλλά και πιο σημαντική ιστορία πελάτη. Με την τεχνική του προγραμματισμού σε ζεύγη και την εναλλαγή των ρόλων των δύο ατόμων ανά δεκαπέντε λεπτά, επιτεύχθηκε ο συγκεκριμένος στόχος. Μέσα σε πενήντα λεπτά είχαμε ελέγξει και ουσιαστικά ολοκληρώσει κάθε εργασία που είχε ανατεθεί στη συγκεκριμένη επανάληψη. Στο σημείο αυτό επανέρχεται το ερώτημα που θέσαμε στο τέλος της δεύτερης μέρας, δηλαδή το πώς θα εκμεταλλευτούμε την περίπου μία ώρα που μας απομένει.

Για το λόγο αυτό, ο ένας εκ των δύο φοιτητών ανέλαβε καθήκοντα διαχειριστή προϊόντος, ο οποίος θα αναδείξει τις προτεινόμενες ιστορίες

πελάτη, οι οποίες θα εκτιμηθούν και εάν είναι εφικτό κάποια από αυτές θα αναπτυχθούν στη συγκεκριμένη επανάληψη. Επειδή υπάρχει μια τέτοια ιστορία και ονομάζεται «καταστήματα», θα ενσωματωθεί στον πίνακα επανάληψης, αφού εκτιμήθηκε πως χρειάζεται μόλις μισή μέρα για την ολοκλήρωσή της. Η μόνη προγραμματιστική λειτουργία, για τη συγκεκριμένη ιστορία, είναι η τροποποίηση της βάσης δεδομένων, αφού τον πελάτη τον ενδιαφέρει να μπορεί να διαχειρισθεί τις αποθήκες του από διαφορετικά καταστήματα. Ακόμη δεν έχουμε υλοποιήσει τις απογραφές και αυτός είναι ένας από τους παράγοντες που συμβάλλει στην ελαχιστοποίηση του χρόνου για την ανάπτυξη της συγκεκριμένης ιστορίας.

Για ακόμα μια φορά, ο προγραμματισμός σε ζεύγη και η βάση δεδομένων μας εμφανίζεται στο προσκήνιο, μετά από την προσθήκη νέων πινάκων και την τροποποίηση άλλων, και έτσι ολοκληρώνεται η συγκεκριμένη ιστορία πελάτη. Όμως, λόγω του ότι τροποποιήθηκαν κι άλλοι πίνακες της βάσης δεδομένων μας, ελέγχθηκε κάθε πτυχή του προγράμματος μας για τυχόν νέες δυσλειτουργίες που τελικά δεν εμφανίστηκαν.

Με την προσθήκη και ολοκλήρωση της ιστορίας, που αφορούσε τα καταστήματα, η ταχύτητα ανάπτυξης υπερέβη τις προσδοκίες μας και μετατράπηκε σε τεσσεράμισι έναντι του τέσσερα, που είχαμε προβλέψει. Βέβαια, η συγκεκριμένη διαφορά δημιουργήθηκε λόγω των μη ορθών προβλέψεων ολοκλήρωσης των ιστοριών πελάτη και των αρκετών τεχνικών, που δε χρησιμοποιήθηκαν κατά τη συγκεκριμένη επανάληψη.

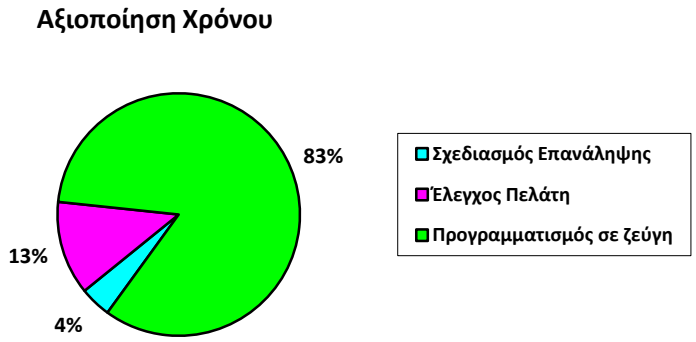
Τη συγκεκριμένη μέρα έγιναν οι παρακάτω ενέργειες.

- Προγραμματισμός σε ζεύγη (100 λεπτά).
    1. Ολοκλήρωση της ιστορίας των ειδών παραστατικών (50 λεπτά)
    2. Ολοκλήρωση της ιστορίας των Καταστημάτων (50 λεπτά).
- Σχεδιασμός Επανάληψης (5 λεπτά)

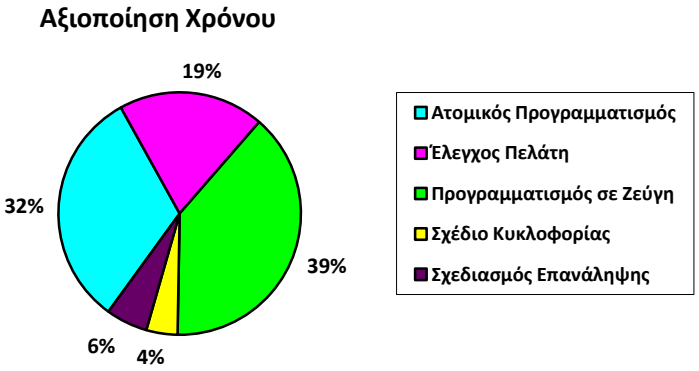
1. Επιλογή νέας ιστορίας πελάτη (5 λεπτά).

- Έλεγχος Πελάτη (15 λεπτά).

Στο τέλος κάθε μέρας παρουσιάζουμε το διάγραμμα αξιοποίησης του χρόνου. Το σημερινό διάγραμμα διαμορφώνεται ως:



Εφόσον ολοκληρώθηκε η δεύτερη επανάληψη, καλό είναι να διευρύνουμε το προηγούμενο διάγραμμα για όλη την επανάληψη, ώστε να μελετηθεί ποιες τεχνικές καταναλώνουν περισσότερο χρόνο και σε ποιες πρέπει να εστιάσει η ομάδα των δύο φοιτητών. Έτσι, καταλήγουμε στο επόμενο διάγραμμα:



Ο έλεγχος πελάτη παρουσιάζει αισθητή αύξηση σε σχέση με την προηγούμενη επανάληψη, ενώ ο προγραμματισμός συνολικά, είτε σε ατομικό

επίπεδο είτε με χρήση του προγραμματισμού σε ζεύγη, παρουσιάζει μια μικρή αύξηση. Όπως ήταν αναμενόμενο, το σχέδιο κυκλοφορίας και ο σχεδιασμός της επανάληψης καταλαμβάνουν αρκετά μικρότερο χρόνο από ό,τι στην προηγούμενη επανάληψη. Αυτό συμβαίνει, διότι κατά την πρώτη επανάληψη χρησιμοποιήθηκαν τεχνικές του ακραίου προγραμματισμού, όπως ο καθορισμός των λειτουργιών που επιθυμεί ο πελάτης, στις οποίες ανατρέχουμε μόνο όταν καταστεί αναγκαίο.

12.3 Η Τρίτη Επανάληψη

12.3.1 Πρώτη μέρα

Όπως έχουμε αναφέρει, όλες οι ευέλικτες προγραμματιστικές τεχνικές, επιτρέπουν στον πελάτη την τροποποίηση των απαιτήσεών του, οποτεδήποτε θελήσει. Για να δούμε τι γίνεται κατά τη συγκεκριμένη επανάληψη, που ο πελάτης αποφασίζει πως το λογισμικό του θα πρέπει να επικοινωνεί με αρκετές βάσεις δεδομένων κι όχι με μία. Για το λόγο αυτό, ο φοιτητής που ανέλαβε καθήκοντα διαχειριστή προϊόντος, μετά από συχνή επικοινωνία με τον πελάτη, ώστε να αποσαφηνισθούν όλες οι προσδοκίες του, αποφασίζει τη δημιουργία μιας νέας ιστορίας πελάτη, με όνομα «ρυθμίσεις βάσεων δεδομένων». Σειρά λοιπόν, έχει η αξιολόγηση της σημαντικότητας της συγκεκριμένης ιστορίας πελάτη, οπότε ο δεύτερος φοιτητής με τα καθήκοντα του πελάτη, της αποδίδει μόλις δεκαπέντε μονάδες. Άρα, η συγκεκριμένη ιστορία δε θεωρείται άμεσα απαραίτητη και για το λόγο αυτό τοποθετείται κι αυτή στον πίνακα ιστοριών πελάτη και μάλιστα με βάση τη σημαντικότητά της. Ο πίνακας του σχεδίου κυκλοφορίας, μετά την προσθήκη της συγκεκριμένης ιστορίας πελάτη, παρουσιάζεται παρακάτω:

id	όνομα	περιγραφή	Προτεραιότητα	εκτίμηση	σημειώσεις
1	Παραστατικά		55	4	
10	Είδη Παραστατικών		52	2	
2	Απογραφές		35	3	
3	Εισαγωγή Χρηστών		30	1,5	
4	Διαγραφή Χρηστών		25	1,5	
5	Τροποποίηση Χρηστών		27	1,5	
6	Καταστήματα		10	0,5	
7	Εισαγωγή προϊόντων		40	1	
9	Τροποποίηση προϊόντων		42	1	
11	Ρυθμίσεις βάσεων δεδομένων		15	1	

Πέρασαν δέκα λεπτά και η ώρα για την εύρεση της εκτιμώμενης ταχύτητας ανάπτυξης είχε φτάσει. Χρησιμοποιώντας τη μεθοδολογία που είχαμε αναφέρει, θα πρέπει πρώτα να υπολογίσουμε το ποσοστό συγκέντρωσης, οπότε έχουμε:

Προηγούμενη ταχύτητα ανάπτυξης / σύνολο ημερών εργασίας = ποσοστό συγκέντρωσης.

4,5 / 6 = 0,75 ή 75%, που είναι ένα αρκετά ικανοποιητικό ποσοστό. Έτσι, η εκτιμώμενη ταχύτητά μας είναι: 0,75 * 6 = 4,5. Όμως, όπως αναφέραμε, η επίτευξη του συγκεκριμένου ρυθμού οφειλόταν, κατά κύριο λόγο, στη μη ορθή εκτίμηση ολοκλήρωσης των προηγούμενων ιστοριών. Οπότε, θα θεωρήσουμε για ακόμα μια φορά, ότι ο αριθμός τέσσερα είναι η ταχύτητά μας.

Σειρά λοιπόν, έχει η επιλογή ιστοριών πελάτη και η τοποθέτησή τους στον πίνακα επανάληψης. Ο φοιτητής που έχει αναλάβει το ρόλο του διαχειριστή προϊόντος, επιλέγει την πρώτη ιστορία, που ονομάζεται «απογραφή» και την τοποθετεί. Η συγκεκριμένη ιστορία είχε εκτιμηθεί ότι χρειάζεται τρείς μέρες για την ολοκλήρωσή της, οπότε από τους τέσσερις πόντους που καθορίστηκαν από την ταχύτητα ανάπτυξης, μας έχει απομείνει μόλις ένας πόντος για να επιλέξουμε άλλη μία ιστορία. Η ιστορία αυτή δεν είναι άλλη από αυτή που δημιουργήσαμε προηγουμένως, αφού εκτιμήθηκε ότι μόλις μια μέρα αρκεί για την ολοκλήρωσή της.

Το επόμενο λογικό βήμα ήταν, όπως πάντα, η δημιουργία προγραμματιστικών λειτουργιών και η εκτίμησή τους, μετά από επικοινωνία μεταξύ προγραμματιστή και πελάτη. Έτσι, ο πίνακας της επανάληψης ήταν:

Δεν έχουν ξεκινήσει	Έχουν ξεκινήσει	Έχουν τελειώσει	Μονάδες
Απογραφή			
Απογραφή ανά χώρο (0,5)			
Απογραφή ανά είδος (0,5)			

Τροποποίηση βάσης δεδομένων (1,5)			
Δημιουργία περιβάλλοντος αλληλεπίδρασης (0,5)			
Ρυθμίσεις βάσεων δεδομένων			
Δημιουργία περιβάλλοντος αλληλεπίδρασης (0,5)			
Τροποποίηση κλάσης βάσης δεδομένων (0,5)			

Αφού έχουμε διασπάσει τις ιστορίες μας σε προγραμματιστικές λειτουργίες, για τις οποίες έχουμε εκτιμήσει και το χρόνο που χρειάζονται για να ολοκληρωθεί η ανάπτυξή τους, έπρεπε να επιλέξουμε το ποιες από αυτές θα υλοποιήσουμε τη συγκεκριμένη μέρα. Έτσι, ο πρώτος φοιτητής επέλεξε την ιστορία των ρυθμίσεων των βάσεων δεδομένων και μάλιστα τη δημιουργία περιβάλλοντος αλληλεπίδρασης. Ο δεύτερος φοιτητής επέλεξε την τροποποίηση της βάσεως δεδομένων από την απογραφή, διότι όχι μόνο απαιτείται αρκετός χρόνος για την υλοποίησή της, αλλά και για να επέμβει στην κλάση, που οφείλεται για την επικοινωνία με τη βάση πριν ο πρώτος φοιτητής τη χρειασθεί και εμπλουτίσει σε αυτή νέες λειτουργίες. Μέσα σε άλλα δέκα λεπτά, επιτύχαμε την έναρξη της συγγραφής κώδικα για τη συγκεκριμένη επανάληψη.

Ο πρώτος φοιτητής, μετά από ένα μισάωρο, ολοκλήρωσε την κωδικοποίηση της προγραμματιστικής λειτουργίας που του είχε ανατεθεί και ήταν έτοιμος να αναλάβει την τροποποίηση της κλάσης βάσης δεδομένων, αφού ο δεύτερος φοιτητής είχε ολοκληρώσει τις επεμβάσεις του στη συγκεκριμένη πτυχή του λογισμικού. Ο δεύτερος φοιτητής όμως, αντιμετώπιζε προβλήματα, που δεν είχαν εκτιμηθεί. Τα προβλήματα αυτά, τον ανάγκαζαν να επανασχεδιάσει ολόκληρη τη βάση δεδομένων, οπότε και να τροποποιήσει πτυχές του προγράμματός μας που είχαν θεωρηθεί ολοκληρωμένες.

Μετά από άλλο ένα μισάωρο, η ιστορία πελάτη με το όνομα «ρυθμίσεις βάσεων δεδομένων», είχε ολοκληρωθεί και ελεγχθεί. Έτσι, το άτομο που είχε αναλάβει τη συγκεκριμένη ιστορία, έπρεπε να αναπτύξει κάποια άλλη προγραμματιστική λειτουργία και τελικά αυτή είναι η δημιουργία περιβάλλοντος αλληλεπίδρασης της απογραφής.

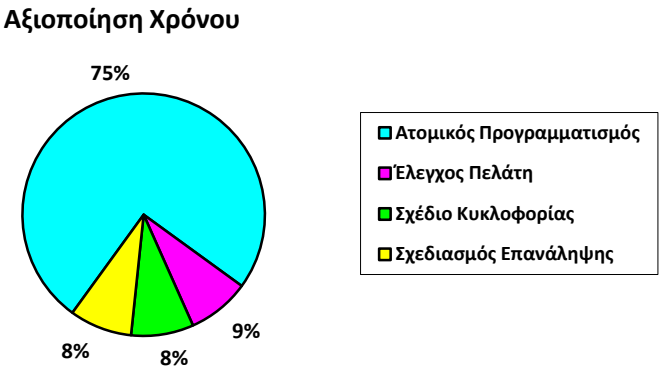
Στο τέλος της συγκεκριμένης μέρας μας, είχαν απομείνει 4 – 1 = 3 πόντοι ακόμη. Στο σημείο αυτό, θα πρέπει να παρατηρήσουμε το εξής μεγάλο πρόβλημα που δημιουργήθηκε το οποίο δεν είναι άλλο από την επαναδημιουργία της βάσης δεδομένων. Όπως έχουμε αναφέρει, ο πελάτης ανά πάσα στιγμή μπορεί να τροποποιήσει τις πτυχές του προγράμματος, κι έτσι σε αρκετές περιπτώσεις έχουμε άμεση επίπτωση στη βάση δεδομένων μας. Αυτό είναι και το μεγαλύτερο πρόβλημα που έχουμε συναντήσει κατά τη διαδικασία του συγκεκριμένου πειράματος, διότι θα πρέπει να τροποποιηθούν αρκετές πτυχές του λογισμικού μας που τις θεωρούσαμε ολοκληρωμένες. Επιπροσθέτως, θα πρέπει να ελεγχθεί το πρόγραμμα από την αρχή, κάτι που θα καθυστερήσει σε μεγάλο βαθμό την ανάπτυξη του. Ας αναλογισθούμε και την περίπτωση κατά την οποία ο πελάτης χρησιμοποιούσε το μη ολοκληρωμένο μας πρόγραμμα και τον αναγκάζουμε να διαγράψει τα δεδομένα, που βρίσκονται στη βάση δεδομένων του. Η συγκεκριμένη κατάσταση από μόνη της μας αποτρέπει από τη συχνή τροποποίηση της βάσης μας, οπότε ένας αρκετά καλός σχεδιασμός της καθίσταται σαν την πιο αξιόπιστη λύση, για την όσο το δυνατόν αποφυγή της παραπάνω κατάστασης. Για να δούμε πως είχε διαμορφωθεί ο πίνακας της επανάληψης κατά το τέλος της συγκεκριμένης μέρας.

Δεν έχουν ξεκινήσει	Έχουν ξεκινήσει	Έχουν τελειώσει	Μονάδες
Απογραφή			
Απογραφή ανά χώρο (0,5)			
Απογραφή ανά είδος (0,5)			
	Τροποποίηση βάσης δεδομένων (1,5)		
	Δημιουργία περιβάλλοντος αλληλεπίδρασης (0,5)		
		Ρυθμίσεις βάσεων δεδομένων	
		Δημιουργία περιβάλλοντος αλληλεπίδρασης (0,5)	0,5
		Τροποποίηση κλάσης βάσης δεδομένων (0,5)	0,5

Αναλυτικά, η ομάδα ασχολήθηκε με:

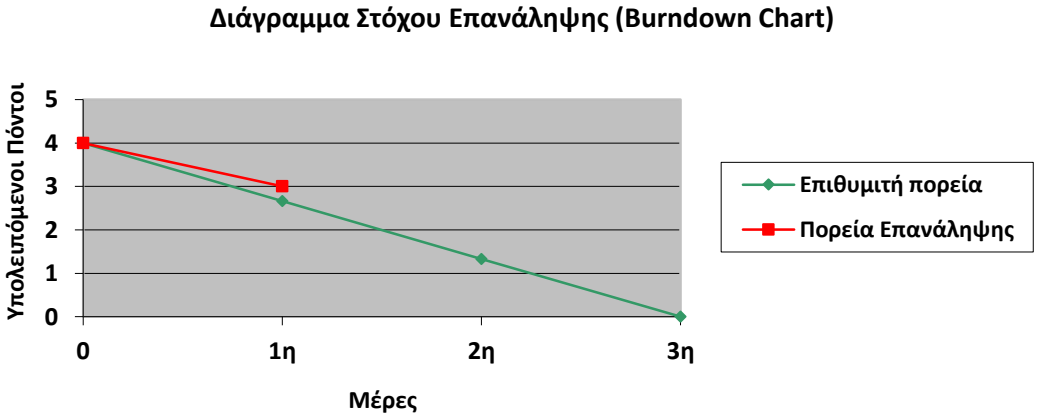
- Σχέδιο Κυκλοφορίας (10 λεπτά)
    1. Καθορισμός επιθυμιών πελάτη (10 λεπτά).
- Σχεδιασμός Επανάληψης (10 λεπτά)
    1. Υπολογισμός ταχύτητας ανάπτυξης και επιλογή ιστοριών πελάτη (2 λεπτά).
    2. Διάσπαση ιστοριών πελάτη σε προγραμματιστικές λειτουργίες και εκτίμησή τους (8 λεπτά).
- Προγραμματισμός σε ατομικό επίπεδο (90 λεπτά)
    1. Πρώτος Φοιτητής
        - Δημιουργία του περιβάλλοντος αλληλεπίδρασης για την νέα ιστορία πελάτη (30 λεπτά).
        - Τροποποίηση κώδικα για ορθή επικοινωνία με τη βάση (20 λεπτά).
        - Δημιουργία του περιβάλλοντος αλληλεπίδρασης για την απογραφή (40 λεπτά).
    2. Δεύτερος Φοιτητής
        - Τροποποίηση της βάσης δεδομένων (90 λεπτά).
- Έλεγχος πελάτη (10 λεπτά)

Το διάγραμμα αξιοποίησης χρόνου διαμορφώθηκε ως:



Ο έλεγχος πελάτη, για ακόμα μια φορά, ήταν μικρότερος από τον προσδοκώμενο. Αυτό όμως, πρόκειται να αλλάξει αφού ο επανασχεδιασμός της βάσης επηρεάζει αρκετές πτυχές του προγράμματος μας, οπότε ένας οικουμενικός έλεγχος είναι απαραίτητος.

Όταν ενημερώσουμε το διάγραμμα στόχου επανάληψης, αυτό θα μοιάζει με το επόμενο διάγραμμα.



12.3.2 Δεύτερη μέρα

Επειδή και τα δύο άτομα δεν ολοκλήρωσαν τις προηγούμενες εργασίες που τους είχαν ανατεθεί, ξεκίνησαν τη συγκεκριμένη μέρα με τις ίδιες ακριβώς ενασχολήσεις.

Έχουν περάσει δεκαπέντε λεπτά κι η δημιουργία περιβάλλοντος αλληλεπίδρασης της ιστορίας της απογραφής έχει ολοκληρωθεί. Για το λόγο αυτό, το συγκεκριμένο άτομο αναλαμβάνει την ανάπτυξη της απογραφής ανά είδος. Μόνο που δε δίδεται η δυνατότητα του έλεγχου της συγκεκριμένης προγραμματιστικής λειτουργίας, πριν την ολοκλήρωση της τροποποίησης της βάσης δεδομένων.

Όταν τελικά έχουν απομείνει περίπου είκοσι λεπτά, έχουν ολοκληρωθεί και οι δυο προγραμματιστικές λειτουργίες. Όμως, δεν έχουν ελεγχθεί ακόμη κι αυτό συμβαίνει. Ένα σφάλμα βρέθηκε στην απογραφή ανά είδος και διορθώθηκε άμεσα, χωρίς να ξεπερασθεί ο χρόνος που αφιερώνεται καθημερινά για την ανάπτυξη του λογισμικού.

Στο τέλος της συγκεκριμένης μέρας, μας έχει απομείνει η ανάπτυξη της απογραφής ανά χώρο. Άρα, μας απομένει 0,5 πόντους για την επίτευξη της ταχύτητας ανάπτυξης. Πώς όμως θα εκμεταλλευτούμε το χρόνο που θα μας απομείνει; Στην πραγματικότητα, μας έχει απομείνει κάποια μικρή ιστορία πελάτη, η οποία υλοποιείται σε λιγότερο χρόνο της μίας μέρας, αλλά δεν την είχαμε επιλέξει ούτε στην προηγούμενη επανάληψη αφού ο κίνδυνος λανθασμένης εκτίμησης είναι αρκετά μεγάλος. Επίσης, φάνηκε η ανάγκη για την αναδόμηση του κώδικα, ώστε σε κάθε τροποποίηση της βάσης δεδομένων να ελαχιστοποιήσουμε τον κόπο που μας δημιουργείται σε κάθε τροποποίησή της. Έτσι έχουμε:

- Προγραμματισμός σε ατομικό επίπεδο (100 λεπτά)

    1. Πρώτος Φοιτητής

        - Δημιουργία του περιβάλλοντος αλληλεπίδρασης για την απογραφή (15 λεπτά).

- Δημιουργία προγραμματιστικής λειτουργίας που αφορά την απογραφή ανά είδος (85 λεπτά).

2. Δεύτερος Φοιτητής

- Τροποποίηση της βάσης δεδομένων (100 λεπτά).

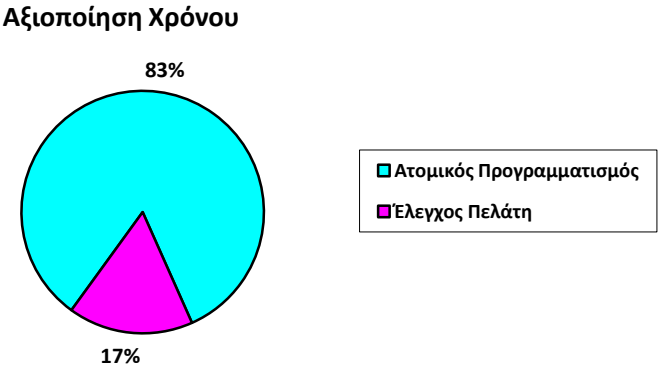
- Έλεγχος πελάτη (20 λεπτά)

1. Έλεγχος καλής λειτουργίας για το σύνολο του λογισμικού (15 λεπτά).

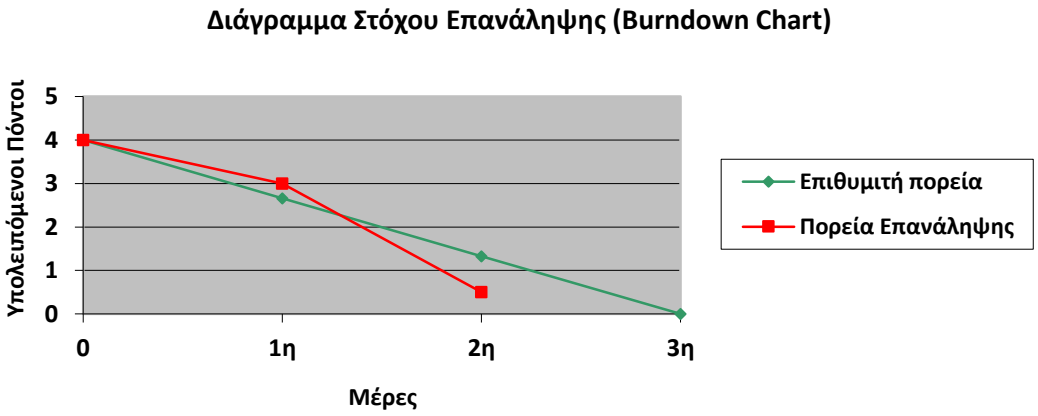
2. Διόρθωση σφάλματος (5 λεπτά).

Τη δεύτερη μέρα της τρίτης επανάληψης, μόνο με δύο τεχνικές ασχολήθηκε η ομάδα. Φυσικά σε μεγαλύτερες ομάδες, θα πρέπει να υπάρχει συντονισμός. Συνεπώς, κι άλλες τεχνικές του ευέλικτου προγραμματισμού θα χρησιμοποιούνταν.

Το σημερινό διάγραμμα αξιοποίησης χρόνου διαμορφώθηκε ως:



Για να μελετήσουμε την πορεία του έργου, θα πρέπει να ελέγξουμε το διάγραμμα στόχου επανάληψης, το οποίο μας προτρέπει να ενσωματώσουμε και άλλη ιστορία πελάτη στην επανάληψή μας. Κάτι τέτοιο, όμως, δε θα γίνει λόγω ανάγκης για βελτιστοποίηση του κώδικα.



12.3.3 Τρίτη μέρα

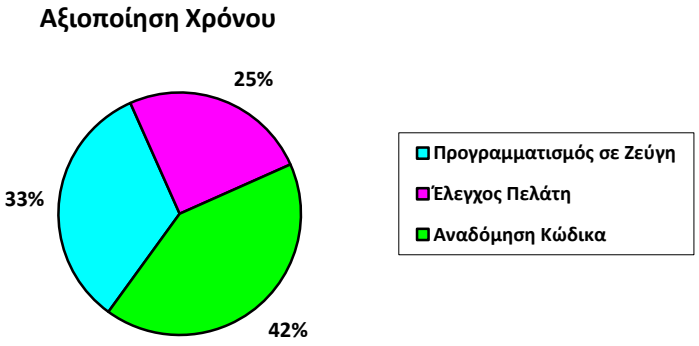
Κατά τη συγκεκριμένη μέρα, οι δύο φοιτητές αποφάσισαν τη χρήση του προγραμματισμού σε ζεύγη για την ανάπτυξη της τελευταίας προγραμματιστικής λειτουργίας, που είναι η απογραφή ανά χώρο αποθήκευσης. Έτσι λοιπόν, με την εναλλαγή των ρόλων ανά είκοσι λεπτά, σε κάτι λιγότερο από σαράντα λεπτά είχαμε ολοκληρώσει και ελέγξει την ιστορία της απογραφής. Με τον τρόπο αυτό, είχαμε επιτύχει το στόχο μας, δηλαδή την εκτιμώμενη ταχύτητα ανάπτυξης κι είχαμε αρκετό χρόνο για την άκρως σημαντική αναδόμηση του κώδικα.

Στο τέλος της συγκεκριμένης μέρας και μάλιστα στο τέλος της τρίτης επανάληψης, είχαμε όχι μόνο επιτύχει την επιθυμητή ταχύτητα ανάπτυξης λογισμικού, αλλά είχαμε αναδομήσει τον κώδικα που είναι ίσως από τις πιο σημαντικές ενέργειες που πρέπει να γίνονται σε κάθε ανάπτυξη λογισμικού. Ακόμη, είναι σύνηθες στις ομάδες που χρησιμοποιούν ευέλικτες προγραμματιστικές μεθοδολογίες, να παραδίδουν ανά τρείς εβδομάδες τον έτοιμο και χωρίς σφάλματα κώδικα, τον οποίο ο πελάτης μπορεί να χρησιμοποιήσει άμεσα.

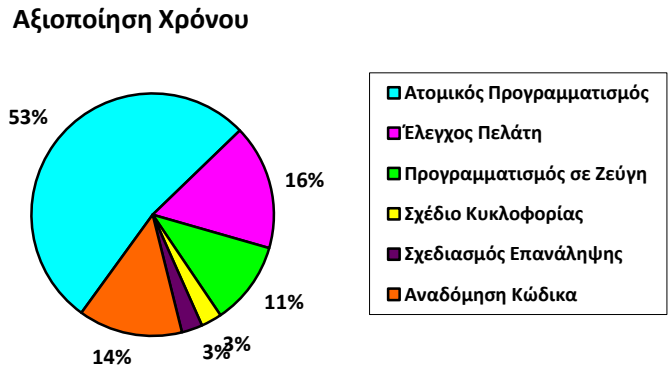
Καλό είναι να τονισθεί πως η αναδόμηση κώδικα επιφέρει την αναγκαιότατα του ελέγχου ώστε να διασφαλισθεί πως δεν επηρεάστηκε η λειτουργικότητα του προγράμματός μας. Έτσι έγιναν:

- Προγραμματισμός σε ζεύγη (40 λεπτά)
- Αναδόμηση του κώδικα (50 λεπτά)
- Έλεγχος πελάτη (30 λεπτά)

Το διάγραμμα της αξιοποίησης χρόνου διαμορφώθηκε ως:



Αντίθετα, το διάγραμμα της αξιοποίησης χρόνου που αφορά το σύνολο της επανάληψης παρουσιάζεται παρακάτω.



Στην τρίτη επανάληψη, τα ποσοστά χρήσης των τεχνικών του σχεδίου κυκλοφορίας καθώς και του σχεδιασμού της επανάληψης, παρουσιάζουν μια αισθητή μείωση. Αυτό συμβαίνει, διότι όχι μόνο έχουν απομείνει ελάχιστες ιστορίες πελάτη προς υλοποίηση, αλλά και οι απαιτήσεις του πελάτη δεν έχουν τροποποιηθεί. Επίσης, είναι σημαντικό να παρατηρήσουμε, ότι η αναδόμηση του κώδικα για πρώτη φορά είναι εμφανής. Όσο η υλοποίηση του λογισμικού εξελίσσεται, τόσο μεγαλύτερη ανάγκη για αναδόμηση του κώδικα έχουμε και σαν συνέπεια τον επανέλεγχο του λογισμικού. Το 14% είναι αρκετά μεγάλο ποσοστό. Όμως, μην ξεχνάμε ότι είναι η πρώτη φορά που χρησιμοποιείται.

12.4 Τέταρτη Επανάληψη

12.4.1 Πρώτη μέρα

Στην αρχή κάθε επανάληψης, όπως έχουμε παρατηρήσει, ο διαχειριστής προϊόντος υπολογίζει την ταχύτητα επανάληψης πριν επιλεγούν οι ιστορίες πελάτη προς ανάπτυξη. Όμως, επειδή η ταχύτητα ανάπτυξης μας έχει σταθεροποιηθεί και δεν έχουμε κάποια απουσία του προσωπικού, είναι ανούσιο να υπολογίσουμε ξανά το συγκεκριμένο αριθμό. Οπότε, θεωρούμε πως είναι για άλλη μια φορά ίδιος με της προηγούμενης επανάληψης.

Εφόσον ο πελάτης μας δεν τροποποίησε κάποια πτυχή του αναπτυσσόμενου προγράμματος, ο διαχειριστής προϊόντος επέλεξε τρεις ιστορίες πελάτη που είναι και οι τελευταίες για την παράδοση του έργου. Αυτές είναι η εισαγωγή, η τροποποίηση και η διαγραφή χρηστών. Μετά τη συγκεκριμένη ενέργεια, ο διαχειριστής προϊόντος μετατράπηκε σε πελάτη, ενώ το δεύτερο άτομο, σαν προγραμματιστής, διασπούσε τις συγκεκριμένες ιστορίες μετατρέποντάς τες σε προγραμματιστικές λειτουργίες και εκτιμώντας το χρόνο της κάθε μιας για υλοποίησή της. Έτσι είχαμε:

Δεν έχουν ξεκινήσει	Έχουν ξεκινήσει	Έχουν τελειώσει	Μονάδες
Δημιουργία χρηστών			
Τροποποίηση βάσης δεδομένων (0,5)			
Δημιουργία περιβάλλοντος αλληλεπίδρασης (1)			
Τροποποίηση χρηστών			
Δημιουργία περιβάλλοντος αλληλεπίδρασης (1)			
Εκτέλεση εντολών sql (0,5)			
Διαγραφή Χρηστών			
Δημιουργία περιβάλλοντος			

αλληλεπίδρασης και εκτέλεση εντολών sql (1)			

Όπως μπορούμε εύκολα να διαπιστώσουμε, στην ιστορία που αφορά τη διαγραφή των χρηστών, είχαμε μια σύνθετη προγραμματιστική λειτουργία. Γιατί όμως, τη μετατρέψαμε σε σύνθετη; Επειδή η εκτέλεση εντολών της βάσης δεδομένων για τη διαγραφή χρηστών θα υλοποιηθεί σε λιγότερο από μίση μέρα, είναι ορθό να την ενσωματώσουμε με άλλη. Κι έτσι έγινε. Όλη η παραπάνω διεργασία διήρκησε περίπου δέκα λεπτά και το μόνο που απέμενε μέχρι τη συγγραφή κώδικα, ήταν η ανάθεση των προγραμματιστικών λειτουργιών. Ο πρώτος φοιτητής ανέλαβε τη δημιουργία περιβάλλοντος αλληλεπίδρασης της εισαγωγής χρηστών, ενώ ο δεύτερος την τροποποίηση της βάσης δεδομένων από τη συγκεκριμένη ιστορία.

Μέσα σε είκοσι λεπτά, ο δεύτερος φοιτητής είχε ολοκληρώσει την ανάπτυξη της πτυχής που του είχε ανατεθεί, οπότε ανέλαβε την ανάπτυξη της ιστορίας της διαγραφής χρηστών. Αντίθετα, ο πρώτος ολοκλήρωσε την ανάπτυξη του περιβάλλοντος αλληλεπίδρασης μετά από σαράντα λεπτά, και ανέλαβε αμέσως καθήκοντα ελεγκτή, με σκοπό την αναγνώριση τυχών σφαλμάτων κατά τη δημιουργία χρηστών.

Είκοσι λεπτά πριν την ολοκλήρωση της ημέρας, η ιστορία της διαγραφής των χρηστών είχε ολοκληρωθεί. Όμως, ήταν έτσι στην πραγματικότητα; Ο έλεγχος, μας είχε επιδείξει αρκετά σφάλματα που όπως είναι λογικό θα πρέπει να επιδιορθωθούν άμεσα. Τα σφάλματα αυτά, όπως φάνηκε, επήλθαν από τον κακό σχεδιασμό, σε συνδυασμό με την ορθή συνεννόηση των προγραμματιστών. Ωστόσο, στο τέλος της συγκεκριμένης ημέρας δεν είχαν επιδιορθωθεί όλα τα σφάλματα, οπότε θεωρούμε ότι η διαγραφή χρηστών δεν έχει ολοκληρωθεί. Έτσι, μας απομένουν ακόμα 2,5 πόντοι για την ολοκλήρωση των ιστοριών πελάτη, αλλά και ολόκληρης της ανάπτυξης του έργου. Ο πίνακας επανάληψης είχε διαμορφωθεί ως εξής:

Δεν έχουν ξεκινήσει	Έχουν ξεκινήσει	Έχουν τελειώσει	Μονάδες
		Δημιουργία χρηστών	
		Τροποποίηση βάσης δεδομένων (0,5)	0,5
		Δημιουργία περιβάλλοντος αλληλεπίδρασης (1)	1
Τροποποίηση χρηστών			
Δημιουργία περιβάλλοντος αλληλεπίδρασης (1)			
Εκτέλεση εντολών sql (0,5)			
Διαγραφή Χρηστών			
	Δημιουργία περιβάλλοντος αλληλεπίδρασης και εκτέλεση εντολών sql (1)		

Αναλυτικά έχουμε:

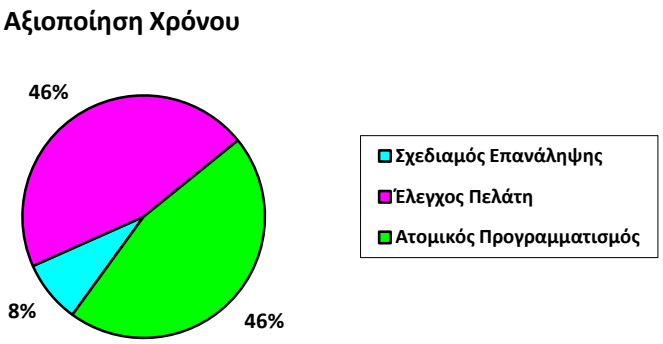
- Σχεδιασμός Επανάληψης(10 λεπτά).

1. Υπολογισμός ταχύτητας ανάπτυξης και επιλογή ιστοριών πελάτη (1 λεπτό).
2. Διάσπαση ιστοριών πελάτη σε προγραμματιστικές λειτουργίες και εκτίμησή τους (9 λεπτά).

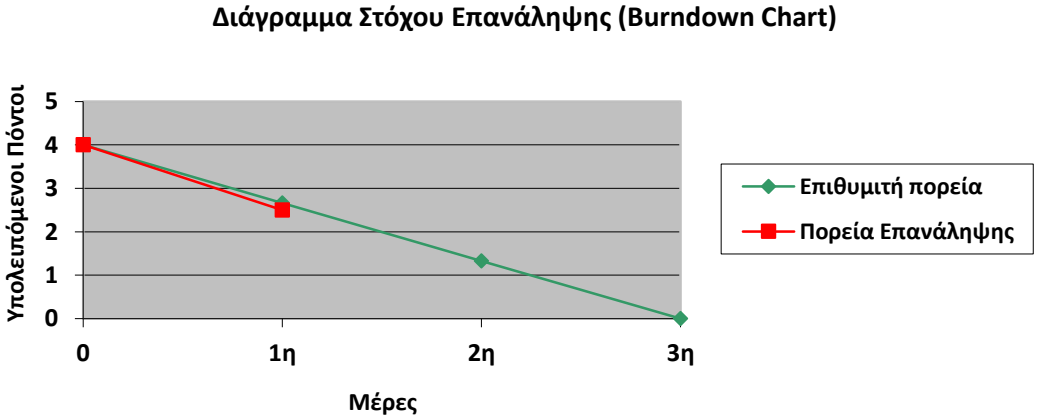
- Προγραμματισμός σε ατομικό επίπεδο (90 λεπτά)
    1. Πρώτος Φοιτητής
        - Δημιουργία του περιβάλλοντος αλληλεπίδρασης για τη δημιουργία χρηστών (40 λεπτά).
    2. Δεύτερος Φοιτητής
        - Τροποποίηση της βάσης δεδομένων για την ιστορία της εισαγωγής χρηστών (20 λεπτά).
        - Ενασχόληση με την ιστορία της διαγραφής των χρηστών (70 λεπτά)
- Έλεγχος πελάτη (70 λεπτά)
    1. Έλεγχος καλής λειτουργίας (10λεπτά).
    2. Διόρθωση σφαλμάτων (60 λεπτά).

Τα σφάλματα κώδικα κοστίζουν στην ανάπτυξη του λογισμικού. Αυτό φαίνεται κι από το χρόνο, που σπαταλήθηκε για την επιδιόρθωση τους. Η ομάδα θα πρέπει να πάρει αποφάσεις για την εξάλειψη νέων σφαλμάτων, αλλά και για την αύξηση του ελέγχου, καθώς όσο νωρίτερα βρίσκονται τα σφάλματα, τόσο λιγότερο χρόνο απαιτούν για την επιδιόρθωσή τους.

Το διάγραμμα αξιοποίησης χρόνου διαμορφώνεται ως εξής:



Επειδή, όπως αναφέραμε, βρέθηκαν σφάλματα στην ιστορία πελάτη που αφορά τη διαγραφή των χρηστών τα οποία δε διορθώθηκαν, θεωρούμε πως η συγκεκριμένη λειτουργία δεν έχει ολοκληρωθεί. Έτσι, το διάγραμμα στόχου επανάληψης διαμορφώνεται όπως παρακάτω:



12.4.2 Δεύτερη μέρα

Εφόσον διορθώναμε από την προηγούμενη μέρα την ιστορία της διαγραφής, ο πρώτος φοιτητής απασχολήθηκε στο συγκεκριμένο κομμάτι. Αντίθετα, στον δεύτερο είχε ανατεθεί η προγραμματιστική λειτουργία της δημιουργίας περιβάλλοντος αλληλεπίδρασης από την ιστορία της τροποποίησης των χρηστών.

Ο πρώτος όμως, ολοκλήρωσε την επιδιόρθωση των σφαλμάτων μετά από μισή ώρα και έπρεπε να ελεγχθεί ξανά η συγκεκριμένη ιστορία πελάτη. Τη διαδικασία αυτή, την ανέλαβε το δεύτερο άτομο, μετά την ολοκλήρωση της δικής του προγραμματιστικής λειτουργίας, που συγκεκριμένα ολοκληρώθηκε μετά από δέκα λεπτά. Όσο, λοιπόν, ελεγχόταν η συγκεκριμένη πτυχή του προγράμματος, η προγραμματιστική λειτουργία, που αφορούσε την τροποποίηση χρηστών και πιο συγκεκριμένα τις εντολές της βάσης δεδομένων, αναπτυσσόταν χωρίς ιδιαίτερα προβλήματα.

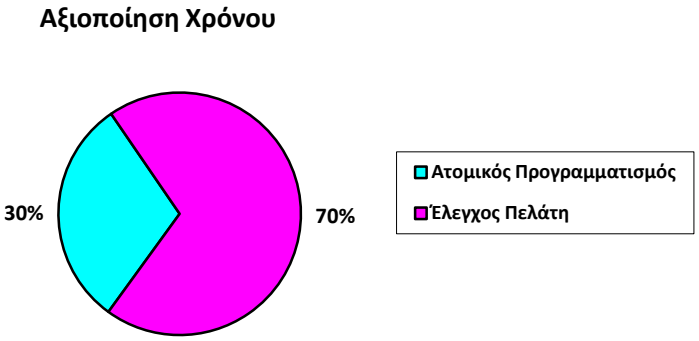
Ευτυχώς, η ιστορία διαγραφής χρηστών λειτουργούσε πια κανονικά, οπότε θεωρήθηκε κι αυτή σαν ολοκληρωμένη ιστορία πελάτη. Μετά από σαράντα λεπτά περίπου, ο έλεγχος θα έπρεπε να γίνει και στην τελευταία μας ιστορία, ώστε να ολοκληρωθεί ολόκληρο το έργο. Ο συγκεκριμένος έλεγχος έδειξε ότι η ανάπτυξη του λογισμικού μας είχε ολοκληρωθεί κι ήταν έτοιμο για παράδοση στον πελάτη.

Αναλυτικά έχουμε:

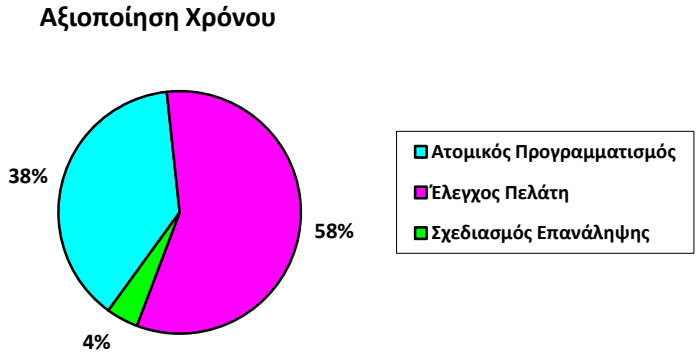
- Έλεγχος πελάτη (80 λεπτά)
    1. Διόρθωση σφάλματος που προκύπτει κατά τη διαγραφή πελάτη. (40 λεπτά)
    2. Έλεγχος των τελευταίων ιστοριών πελάτη (10 λεπτά).
    3. Έλεγχος του λογισμικού πριν παραδοθεί στον πελάτη (30 λεπτά).
- Προγραμματισμός σε ατομικό επίπεδο (40 λεπτά)
    1. Πρώτος Φοιτητής
        - Δημιουργία περιβάλλοντος αλληλεπίδρασης για την τροποποίηση των χρηστών (40 λεπτά).
    2. Δεύτερος Φοιτητής

- Εκτέλεση εντολών βάσης δεδομένων για την τροποποίηση των χρηστών (30 λεπτά).

Το διάγραμμα αξιοποίησης χρόνου για την τελευταία μέρα υλοποίησης παρουσιάζεται παρακάτω:



Η τελευταία επανάληψη ολοκληρώθηκε οπότε μπορούμε να δούμε την αξιοποίηση του χρόνου συνολικά.



Ο έλεγχος πελάτη καταλαμβάνει αρκετά μεγάλο ποσοστό. Αυτό συμβαίνει, διότι έχουμε προσθέσει σε αυτή την τεχνική και το χρόνο επιδιόρθωσης των σφαλμάτων.

Αυτό λοιπόν, ήταν το μικρό μας πείραμα σχετικά με τη χρήση του ευέλικτου προγραμματισμού και πιο συγκεκριμένα του ακραίου προγραμματισμού. Για να δούμε όμως τα αποτελέσματά του.

12.5 Συνολική εικόνα πειράματος

Λόγω του ότι η παρουσίαση του πειράματος έγινε με βάση την κάθε μέρα ανάπτυξης ξεχωριστά, καλό είναι να παρουσιασθούν οι συνολικές λειτουργίες που δημιουργήθηκαν, αλλά και τα ποσοστά αξιοποίησης του χρόνου, ώστε να γίνει σχολιασμός σε αυτά.

Οι λειτουργίες που ενσωματώθηκαν στο πείραμά μας, παρουσιάζονται σε διακριτές ομάδες, που αφορούν την κάθε επανάληψη, δηλαδή σε ποιά επανάληψη ολοκληρώθηκαν. Οι πίνακες που δημιουργήθηκαν, είναι οι παρακάτω.

Πρώτη Επανάληψη		
Όνομα λειτουργίας	Χρόνος Υλοποίησης (Εκτίμηση)	Πραγματικός Χρόνος Υλοποίησης
Παραστατικά	4	4,5

Δεύτερη Επανάληψη		
Όνομα λειτουργίας	Χρόνος Υλοποίησης (Εκτίμηση)	Πραγματικός Χρόνος Υλοποίησης
Είδη παραστατικών	2	1,5
Εισαγωγή προϊόντων	1	1
Τροποποίηση προϊόντων	1	1
Καταστήματα	0,5	0,5

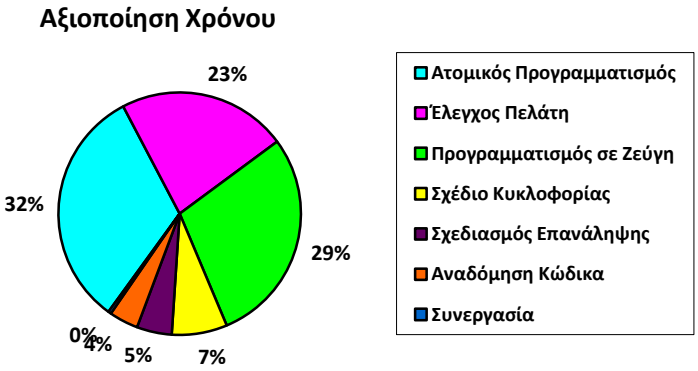
Τρίτη Επανάληψη		
Όνομα λειτουργίας	Χρόνος Υλοποίησης (Εκτίμηση)	Πραγματικός Χρόνος Υλοποίησης
Απογραφή	3	2,5
Εισαγωγή προϊόντων	1	1

Τέταρτη Επανάληψη		
Όνομα λειτουργίας	Χρόνος Υλοποίησης (Εκτίμηση)	Πραγματικός Χρόνος Υλοποίησης
Δημιουργία Χρηστών	1,5	1,5
Τροποποίηση Χρηστών	1,5	1
Διαγραφή Χρηστών	1	1

Από τους παραπάνω πίνακες, μπορούμε να διακρίνουμε, ότι οι εκτιμήσεις μας για την ολοκλήρωση του λογισμικού προσέγγιζαν το πραγματικό χρόνο υλοποίησης, κάτι που είναι πολύ σημαντικό σε επαγγελματικά περιβάλλοντα. Έτσι, με βάση τη συγκεκριμένη παρατήρηση, μπορούμε να υποθέσουμε, με επιφύλαξη λόγω της μικρής έκτασης του πειράματος, ότι ο ευέλικτος προγραμματισμός, προσφέρει μια αρκετά καλή εκτίμηση του χρόνου ολοκλήρωσης του έργου κι αυτό οφείλεται, σε μεγάλο βαθμό, στη διάσπαση των προγραμματιστικών λειτουργιών, καθώς και στη συνεχή επικοινωνία της ομάδας.

Ένα άλλο σημαντικό στοιχείο, είναι η συνολική εικόνα του διαγράμματος της αξιοποίησης του χρόνου. Σε αυτό το διάγραμμα, θα δούμε το χρόνο που

αφιερώθηκε συνολικά στις τεχνικές του ευέλικτου προγραμματισμού, κατά την ανάπτυξη του λογισμικού.



Από το παραπάνω διάγραμμα, μπορούμε να συμπεράνουμε ότι η ομάδα μας κατά 61% ασχολήθηκε με τον προγραμματισμό, ενώ κατά 23% γινόταν έλεγχος στο λογισμικό, που αναπτυσσόταν. Γι' αυτό το λόγο, μπορούμε να πούμε ότι τα ποσοστά του πειράματος, όσον αφορά τον προγραμματισμό και τον έλεγχο του λογισμικού, συμφωνούν σε μεγάλο βαθμό με αυτά που αναφέρονται στις πηγές την πτυχιακής εργασίας. Αντιθέτως, το σχέδιο κυκλοφορίας, ο σχεδιασμός επανάληψης και η συνεργασία, χρησιμοποιήθηκαν σε μικρότερη κλίμακα από ό,τι σε ένα πραγματικό περιβάλλον και γι' αυτό το λόγο, καταλαμβάνουν μικρότερο ποσοστό από το 20%.

12.6 Αποτελέσματα πειράματος

Όπως εύκολα παρατηρήσαμε, η χρήση της συγκεκριμένης ευέλικτης προγραμματιστικής μεθόδου, κατέστη αρκετά εύκολη σε μικρό χρονικό διάστημα, αν και αρκετές τεχνικές της δε χρησιμοποιήθηκαν. Όμως, σύμφωνα με τα άτομα που συμμετείχαν στο πείραμα, ο ευέλικτος προγραμματισμός μπορεί να κάνει τη διαφορά σε αρκετές πτυχές της ανάπτυξης του λογισμικού. Για παράδειγμα, όπως έγινε φανερό, ακόμη και με την τροποποίηση των

απαιτήσεων του πελάτη, η ανάπτυξη του έργου επιβαρύνθηκε σε αρκετά μικρό βαθμό, όπως ακριβώς είχαμε αναφέρει πρωτίστως. Επίσης, όπως παρατηρήθηκε από τις πρώτες επαναλήψεις, η ταχύτητα ανάπτυξης ήταν σταθερή, πράγμα που επιτρέπει τη μερική εκτίμηση του χρόνου ολοκλήρωσης του έργου. Ένα άλλο σημείο στο οποίο θα πρέπει να σταθούμε, είναι η ανεύρεση σφαλμάτων. Όταν χρησιμοποιήθηκε ο προγραμματισμός σε ζεύγη, δεν υπήρξαν σοβαρά σφάλματα κώδικα, όπως στην αντίθετη περίπτωση. Για το λόγο αυτό, μπορούμε με μερική επιφύλαξη να υπογραμμίσουμε ότι με την τεχνική του προγραμματισμού σε ζεύγη, επιτυγχάνεται η συγγραφή κώδικα, ενώ παράλληλα είναι σε μεγάλο βαθμό απαλλαγμένος από σφάλματα.

Όμως, υπήρξε κι ένα σοβαρό πρόβλημα, που λέγεται τροποποίηση της βάσης δεδομένων. Λόγω του ότι δεν ξέρουμε εξ αρχής τις πεποιθήσεις του πελάτη για το προς ανάπτυξη λογισμικό, συχνά η βάση δεδομένων μας τροποποιούταν, προκαλώντας αρκετά προβλήματα στη σωστή λειτουργία του προϊόντος μας. Έτσι, όχι μόνο η βάση δεδομένων μας σχεδιαζόταν από την αρχή, αλλά πτυχές του προγράμματος διορθώνονταν και ελέγχονταν συνεχώς, ώστε να εξασφαλισθεί η σωστή λειτουργία του συστήματος. Εάν φυσικά ο πελάτης χρησιμοποιεί παραδοτέες λειτουργίες του προγράμματος μας, τότε τα πράγματα δυσκολεύουν ακόμα πιο πολύ, διότι θα έπρεπε να διαγραφούν τα δεδομένα της βάσης δεδομένων του και να επανεισαχθούν με τις κατάλληλες αλλαγές.

Αν και το συγκεκριμένο πρόβλημα το συναντούμε και σε μη ευέλικτες προγραμματιστικές λειτουργίες, στον ακραίο προγραμματισμό εμφανίζεται σε μεγαλύτερο βαθμό, λόγω της συχνής τροποποίησης των απαιτήσεων. Όμως, κατά το συγκεκριμένο τρόπο ανάπτυξης λογισμικού, όχι μόνο παρουσιάζουμε στον πελάτη ένα πρόγραμμα που είναι με τις προδιαγραφές που έχει θέσει, αλλά και απαλλαγμένο από σφάλματα. Αυτός είναι ένας από τους λόγους που θα επιλέγαμε τον ευέλικτο προγραμματισμό.

13 Αναφορές (References)

- In *Extreme Programming and Agile Methods: XP/Agile Universe 2002*, pp. 70–88, Springer Verlag, LNCS 2418, Editors, Don Wells and Laurie Williams.

  Agile Project Management Methods for ERP: How to

  Apply Agile Processes to Complex COTS Projects and

  Live to Tell About It

  Glen B. Alleman

- Ambler, Scott, www.agilemodeling.com.
- Ambler, Scott, *Process Patterns* and *More Process Patterns*, Cambridge University Press, 1998 and 1999.
- Amram, Martha, and John Henderson, “Managing Business Risk by IT Investment:
- The Real Options View,” *CIO Magazine*, March 1999.
- Amram, Martha, Nalin Kulatilaka, and John Henderson, “Taking an Option on IT,” *CIO Magazine*, June 15, 1999.
- Amram, Martha and Nalin Kulatilaka, *Real Options: Managing Strategic Investments in a World of Uncertainty*, Harvard Business School Press, 1999.
- Aoyama, Mikio, “New Age of Software Development: How Component Based Software Engineering Changes the Way of Software Development,” *1998 International Workshop on Competent–Based Software Engineering*, 1998.

- James Shore and Shane Warden, “The art of agile development” 2008 O’Reilly media.
- Henrik Kniberg, “Scrum and XP from the Trenches” 2007 C4 Media
- Shad Dowlatshahi , Qing Cao

  “The impact of alignment between virtual entERPrise and information technology on business performance in an agile manufacturing environment”

  Journal of Operations Management 23 (2005) 531–550
- Jan Olhager, Erik Selldin

  “EntERPrise resource planning survey of Swedish manufacturing firms”

  European Journal of Operational Research 146 (2003) 365–373

- Gerard Meszaros, Janice Aston

  Agile ERP: you don’t know what you’ve got “till it’s gone!”

  Agile 2007 Conference

- coServe extending trust

  “An Agile Approach to Successful ERP Implementations”

  2008
- Roger Valade

  “The Big Projects Always Failed”

  Agile 2008 Conference

- Aidas SMAIZYS, Olegas VASILECAS

  “Business Rules Based Agile ERP Systems Development”

  INFORMATICA, 2009, Vol. 20, No. 3, 439–460 © 2009 Institute of Mathematics and Informatics, Vilnius