

Μιχαήλ Ξένος

Ποιότητα Λογισμικού



Περιεχόμενα

Πρόλογος	13
Κεφάλαιο 1	
Εισαγωγή στη Διαχείριση Λογισμικού	17
1.1 Διαχείριση Ανάπτυξης Λογισμικού	18
1.1.1 Βασικές έννοιες	18
1.1.2 Ιδιαιτερότητες στην ανάπτυξη λογισμικού	21
1.1.3 Η κρίση του λογισμικού ως πρόβλημα διαχείρισης	23
1.2 Διαδικασίες Διαχείρισης Έργων	25
1.2.1 Ανάλυση διαδικασιών	26
1.2.1.1 Συγγραφή αρχικής πρότασης	26
1.2.1.2 Προγραμματισμός έργου	29
1.2.1.3 Ανάθεση έργου σε ανθρώπινο δυναμικό	32
1.2.1.4 Επίβλεψη έργου	32
1.2.1.5 Τεκμηρίωση και εκπροσώπηση	34
1.2.2 Τεχνικές διαχείρισης	35
1.2.2.1 Δίκτυο δραστηριοτήτων έργου	36
1.2.2.2 Διάγραμμα αξιολόγησης έργου (διάγραμμα PERT)	37
1.2.2.3 Χρονοδιάγραμμα (διάγραμμα Gantt)	41
1.2.2.4 Διάγραμμα ανάθεσης έργου σε ανθρώπινο δυναμικό	44
1.3 Οι Άνθρωποι	52
1.3.1 Μέλος της διοίκησης	53
1.3.2 Υπεύθυνος έργου ή έργων	54
1.3.3 Ηγέτης ομάδας	55
1.3.4 Μηχανικός ανάπτυξης	56
1.3.5 Προγραμματιστής	57

1.3.6 Τεχνικοί και υπόλοιπο προσωπικό	58
1.3.7 Πελάτες	58
1.3.8 Άλλες κατηγορίες	59
1.4 Εργασία σε Ομάδες	60
1.4.1 Τρόποι οργάνωσης ομάδων	60
1.4.2 Πλεονεκτήματα εργασίας σε ομάδες	61
1.4.3 Οργανόγραμμα ανάπτυξης λογισμικού	63
1.5 Σύνοψη Κεφαλαίου	64
Ασκήσεις	65

Κεφάλαιο 2

Εκτίμηση και Ανάλυση Κινδύνου	67
2.1 Εκτίμηση αναγκών, κόστους και χρόνου	67
2.1.1 Εισαγωγή στην εκτίμηση	68
2.1.2 Παράγοντες που επιδρούν στην εκτίμηση	72
2.1.3 Μέθοδοι εκτίμησης	74
2.2 Τεχνικές Εκτίμησης και Εμπειρικά Μοντέλα	76
2.2.1 Τεχνικές εκτίμησης	76
2.2.1.1 Εκτίμηση από κάτω προς τα πάνω	77
2.2.1.2 Εκτίμηση που βασίζεται στο τελικό κόστος	79
2.2.1.3 Εκτίμηση που βασίζεται σε γραμμές κώδικα ...	79
2.2.1.4 Εκτίμηση που βασίζεται σε λειτουργικά σημεία	82
2.2.2 Εμπειρικά μοντέλα	85
2.2.2.1 Το βασικό μοντέλο	86
2.2.2.2 Το ενδιαμέσο μοντέλο	88
2.2.2.3 Το πλήρες μοντέλο	88
2.2.2.4 Γενικές πληροφορίες για το COCOMO	88
2.3 Ανάλυση Κινδύνου	89
2.4 Σύνοψη Κεφαλαίου	95
Ασκήσεις	96

Κεφάλαιο 3

Εισαγωγή στην Ποιότητα	97
3.1 Ορισμοί και Ιστορική Αναδρομή	97
3.1.1 Ποιότητα και μετρήσεις	98
3.1.2 Βασικές έννοιες	101
3.1.3 Ιστορική αναδρομή σε ποιότητα και μετρήσεις	104
3.2 Ποιότητα στην Παραγωγή Υλικών Αγαθών	106
3.2.1 Διαχείριση ολικής ποιότητας	107
3.2.2 Οι πρώτες απόψεις για την ποιότητα	109
3.2.3 Στατιστικός έλεγχος ποιότητας	110
3.3 Ιδιαιτερότητες στην Ποιότητα Λογισμικού	114
3.4 Σύνοψη Κεφαλαίου	116
Ασκήσεις	117

Κεφάλαιο 4

Ποιότητα Λογισμικού	119
4.1 Ποιοτικά Χαρακτηριστικά Λογισμικού	120
4.1.1 Παράγοντες ποιότητας	121
4.1.2 Το μοντέλο FCM	125
4.1.3 Το μοντέλο του Boehm	127
4.1.4 Το πρότυπο ISO 9126	129
4.2 Σύστημα Ποιότητας Λογισμικού	134
4.2.1 Εφαρμογή του συστήματος ποιότητας	135
4.2.2 Χρήστες του συστήματος ποιότητας	141
4.2.3 Οφέλη από το σύστημα ποιότητας	147
4.3 Σύνοψη Κεφαλαίου	149
Ασκήσεις	150

Κεφάλαιο 5

Μετρήσεις και Μετρικές	153
5.1 Μετρήσεις και Μετρικές στο Λογισμικό	154
5.2 Εσωτερικές Μετρήσεις και Μετρικές	161
5.2.1 Κυκλωματική πολυπλοκότητα	170
5.3 Εξωτερικές Μετρήσεις και Μετρικές	175
5.4 Συσχετίσεις	178
5.5 Σύνοψη Κεφαλαίου	181
Ασκήσεις	182

Κεφάλαιο 6

Πρότυπα Ποιότητας Λογισμικού	185
6.1 Τα Διεθνή Πρότυπα ISO	185
6.1.1 Το πρότυπο ISO 9001 και η οδηγία ISO 9000-3	189
6.2 Το Πρότυπο Αξιολόγησης CMM	195
6.2.1 Ορισμός και εξέλιξη του CMM	196
6.2.2 CMM και ISO 9001: Ομοιότητες και διαφορές	198
6.2.3 Επίπεδα ωριμότητας στο CMM	200
6.3 Το πρόγραμμα TickIT	204
6.3.1 Αρχές και στόχοι του προγράμματος TickIT	205
6.3.2 Σε ποιους απευθύνεται το TickIT	206
6.3.3 Διαδικασία πιστοποίησης με TickIT	206
6.4 Σύνοψη Κεφαλαίου	209
Ασκήσεις	210

Κεφάλαιο 7

Ειδικά Θέματα	213
7.1 Software Audits	213
7.2 Software Inspections	217

7.3 Σύνοψη Κεφαλαίου	224
<i>Βιβλιογραφία</i>	
Ελληνόγλωσση βιβλιογραφία	225
Ξενόγλωσση βιβλιογραφία	226
<i>Γλωσσάριο</i>	233
<i>Απόδοση Όρων</i>	247
<i>Index</i>	263

Κεφάλαιο 1

Εισαγωγή στη Διαχείριση Λογισμικού

Στο κεφάλαιο αυτό θα μιλήσουμε για τη διαχείριση της ανάπτυξης λογισμικού, μία διαδικασία με την οποία επιφορτίζεται ο υπεύθυνος έργου. Στην ενότητα 1.1, θα παρουσιάσουμε τις βασικές έννοιες και ορισμούς και θα συζητήσουμε για τις ιδιαιτερότητες του λογισμικού και για τα προβλήματα διαχείρισης που εντάσσονται στη λεγόμενη κρίση του λογισμικού. Στην ενότητα 1.2 θα αναλύσουμε τις βασικές δραστηριότητες που σχετίζονται με τη διαχείριση έργων λογισμικού και θα μιλήσουμε για μερικές από τις πιο διαδεδομένες τεχνικές που χρησιμοποιούν οι υπεύθυνοι έργων. Θα δούμε επίσης ένα παράδειγμα υπό μορφή εργασίας στην οποία δίνουμε και μία ενδεικτική λύση. Στην ενότητα 1.3 θα μιλήσουμε για τους συμμετέχοντες στην ανάπτυξη λογισμικού το ρόλο τους και συνοπτικά για τα προσόντα που θα πρέπει να έχει ο καθένας από αυτούς. Τέλος, στην ενότητα 1.4 συζητάμε για τις ομάδες εργασίας και πώς πρέπει να οργανωθούν ανάλογα με τις ιδιαιτερότητες κάθε έργου, για τα πλεονεκτήματα της εργασίας σε ομάδες και για το οργανόγραμμα διαχείρισης του έργου.

Για τα περισσότερα σημεία που καλύπτουμε στο κεφάλαιο αυτό σας παραθέτουμε βιβλιογραφία για περαιτέρω μελέτη στο τέλος του βιβλίου, όπου μπορείτε να εμπλουτίσετε τη μελέτη σας, αντλώντας γνώσεις από πολλαπλές πηγές.

1.1 Διαχείριση Ανάπτυξης Λογισμικού

Στην ενότητα αυτή θα δοθούν αρκετοί βασικοί όροι και έννοιες που θα χρησιμοποιηθούν στο βιβλίο. Οι όροι που παρουσιάζονται για πρώτη φορά δίνονται τόσο στα Ελληνικά όσο και στα Αγγλικά (σε παρενθέσεις). Αυτό γίνεται ώστε να σας δοθεί η δυνατότητα να εξοικειωθείτε και με την αγγλική ορολογία, μια και αρκετή από τη βιβλιογραφία που παρατίθεται στο τέλος του βιβλίου είναι στα αγγλικά.

Στο πρώτο μέρος της ενότητας, με τίτλο βασικές έννοιες, θα συζητήσουμε για κάποιους ορισμούς που θα χρησιμοποιηθούν σε όλη την έκταση του βιβλίου. Στο δεύτερο μέρος, με τίτλο ιδιαιτερότητες στην ανάπτυξη λογισμικού, θα παρουσιάσουμε τις διαφορές μεταξύ της διαχείρισης ανάπτυξης λογισμικού και της κλασσικής παραγωγής υλικών αγαθών και, τέλος, στο τρίτο και τελευταίο μέρος της ενότητας θα συζητήσουμε τη λεγόμενη κρίση του λογισμικού, αντιμετωπίζοντάς την ως πρόβλημα διαχείρισης.

1.1.1 Βασικές έννοιες

Όπως θα προσέξατε ο τίτλος της ενότητας είναι *‘Διαχείριση Ανάπτυξης Λογισμικού’*. Με τον όρο *‘διαχείριση’* αποδίδουμε τον αγγλικό όρο *‘management’*. Σήμερα έχει πάντως επικρατήσει να αποδίδεται ο όρος *‘manager’* ως *‘υπεύθυνος διαχείρισης’* και έτσι θα τον αναφέρουμε στη συνέχεια του βιβλίου. Αν αναζητήσουμε στο «Λεξικό της Νέας Ελληνικής Γλώσσας» του Γ. Μπαμπινιώτη τη σημασία της λέξης διαχείριση, θα βρούμε τον ορισμό που δίνεται παρακάτω:

Διαχείριση είναι το σύνολο των ενεργειών που κάνει κανείς, για να τακτοποιήσει, να επιλύσει ή να προωθήσει θέματα της αρμοδιότητάς του, ο τρόπος με τον οποίο τα χειρίζεται.

Σαν ορισμός μπορεί να μοιάζει απλός, αλλά στην πραγματικότητα η διαχείριση ενός έργου είναι μία πολύ δύσκολη δραστηριότητα στην οποία ο υπεύθυνος διαχείρισης πρέπει να δώσει έμφαση σε πολλούς παράγοντες, τόσο ανθρώπινους (κυρίως) όσο και τεχνολογικούς και οικονομικούς. Τους παράγοντες αυτούς θα συζητήσουμε στην ενότητα 1.2 του κεφαλαίου 1.

Ας επανέλθουμε όμως στον τίτλο της ενότητας που μιλά για *‘ανάπτυξη λογισμικού’*. Όπως θα συζητήσουμε στην ενότητα 1.1.2 που ακολουθεί, το λογισμικό δεν κατασκευάζεται με τρόπο αντίστοιχο με τα περισσότερα υλικά αγαθά. Με τον όρο *‘αναπτύσσεται’* αποδίδουμε τον αντίστοιχο αγγλικό όρο *‘engineered’*. Με την εμπειρία που πιθανόν να έχετε ως τώρα (αφού μελετάτε ένα πιο εξειδικευμένο αντικείμενο σχετικό με το λογισμικό), γνωρίζετε καλά τι σημαίνει *τεχνολογία λογισμικού* (software engineering) και γιατί διαφέρει από την παραγωγή υλικών αγαθών. Έχετε πιθανόν γνώσεις για τεχνικές ανάπτυξης λογισμικού ή ακόμα και έχετε εμπλακεί σε ανάπτυξη λογισμικού. Σε αυτό το βιβλίο θα μιλήσουμε λοιπόν για τη **διαχείριση ανάπτυξης λογισμικού**.

Σε μία επιχείρηση ή οργανισμό που αναπτύσσει λογισμικό είναι πολύ πιθανό να αναπτύσσονται παράλληλα αρκετά *έργα λογισμικού* (software projects). Για κάθε ένα από αυτά τα έργα θα υπάρχει κάποιος υπεύθυνος για τη διαχείρισή του, δηλαδή κάποιος υπεύθυνος διαχείρισης έργου λογισμικού (software project manager). **Ο υπεύθυνος διαχείρισης έργου λογισμικού** θα αναφέρεται, στο υπόλοιπο του βιβλίου, καθαρά για λόγους συντομίας και **υπεύθυνος έργου**.

Υπεύθυνος έργου είναι αυτός που έχει την ευθύνη για την πορεία του έργου, δηλαδή την τεχνική, οικονομική και διαχειριστική ευθύνη για το έργο.

Ο ορισμός παραπάνω είναι πολύ συνοπτικός, αφού για τις ευθύνες, αρμοδιότητες και εξουσίες του υπεύθυνου έργου θα μιλήσουμε αναλυτικά στην ενότητα 1.3 του κεφαλαίου 1. Όπως θα δούμε, ο ίδιος άνθρωπος μπορεί να είναι υπεύθυνος για περισσότερα από ένα έργα.

Ένας ακόμα λόγος που επιλέξαμε τον όρο *‘υπεύθυνος διαχείρισης έργου λογισμικού’* για τον άνθρωπο που θα έχει την ευθύνη ενός ή περισσότερων έργων λογισμικού είναι για να τονίσουμε τη διαφορά μεταξύ του ρόλου του υπεύθυνου έργου και του ρόλου του υπεύθυνου για τη διοίκηση της επιχείρησης ή του οργανισμού. Η διοίκηση της επιχείρησης που αναπτύσσει λογισμικό μπορεί να είναι το διοικητικό συμβούλιο ή η συνέλευση των μετόχων (ανάλογα με το είδος της επιχείρησης ή του οργανισμού) και όπως θα συζητήσουμε στην ενότητα 1.3 έχει διαφορετικές αρμοδιότητες, ευθύνες και στόχους από τον υπεύθυνο έργου. Για λόγους απλότητας, στο υπόλοιπο του βιβλίου, θα χρησιμοποιούμε τον όρο **διοίκηση** (administration) για τη διοίκηση της επιχείρησης ή του οργανισμού χωρίς να εισερχόμαστε σε λεπτομέρειες για τη μορφή της επιχείρησης και πώς διοικείται.

Πριν προχωρήσουμε στην επόμενη ενότητα, να διευκρινίσουμε ότι όσα θα αναφέρουμε στο βιβλίο αυτό έχουν καλύτερη εφαρμογή σε επιχειρήσεις πολλών ατόμων, όπου οι ρόλοι και οι αρμοδιότητες καθενός είναι σαφέστατα καθορισμένες. Αυτό όμως δεν σημαίνει ότι όσα διαπραγματευόμαστε δεν ισχύουν στις μικρές επιχειρήσεις που αναπτύσσουν λογισμικό. Στην Ελλάδα υπάρχουν πολλές επιχειρήσεις με δύο έως δέκα άτομα προσωπικό που αναπτύσσουν λογισμικό. Σε τέτοιες περιπτώσεις οι ρόλοι μοιράζονται και κάποιος μπορεί να είναι ταυτόχρονα συνεταίρος, υπεύθυνος έργου και προγραμματιστής. Τότε οι ανάγκες για επικοινωνία και συντονισμό είναι μικρότερες, αλλά οι ανάγκες για οργάνωση της διαδικασίας ανάπτυξης είναι κοινές με τις μεγάλες επιχειρήσεις και οι τεχνικές που θα καλύψουμε ισχύουν και πρέπει να εφαρμόζονται.

1.1.2 Ιδιαιτερότητες στην ανάπτυξη λογισμικού

Με την εμπειρία που έχετε σίγουρα αποκτήσει ήδη στην ευρύτερη περιοχή της Τεχνολογίας Λογισμικού, γνωρίζετε ότι το λογισμικό ως προϊόν έχει αρκετές ιδιαιτερότητες που κάνουν τη διαχείρισή του περίπλοκη.

Το λογισμικό αναπτύσσεται δεν κατασκευάζεται.

Για την πρώτη διαφορά μιλήσαμε ήδη στην προηγούμενη ενότητα. Το λογισμικό, όπως γνωρίζετε πολύ καλά, σχεδιάζεται και αναπτύσσεται και δεν κατασκευάζεται με τρόπο αντίστοιχο της παραγωγής υλικών αγαθών. Βασικές αρχές της παραγωγής υλικών αγαθών δεν εφαρμόζονται στην ανάπτυξη λογισμικού. Τέτοιες αρχές είναι η επεξεργασία πρώτων υλών για τη δημιουργία τμημάτων του προϊόντος, η σύνθεση έτοιμων τμημάτων για τη δημιουργία του τελικού προϊόντος, η δημιουργία ενός πρωτοτύπου και η έμφαση στην πιστή αναπαραγωγή αντιγράφων με ελάχιστες αποκλίσεις από το πρωτότυπο. Αυτές οι αρχές δεν σχετίζονται άμεσα με την παραγωγή λογισμικού. Βέβαια ο αντικειμενοστραφής προγραμματισμός (object – oriented programming) και ακόμα περισσότερο ο προγραμματισμός που είναι βασισμένος σε ψηφίδες (component based programming) έχουν συντελέσει αρκετά στο να θεωρούμε ότι το λογισμικό αναπτύσσεται με σύνθεση τμημάτων, αλλά όχι ακόμα στους ρυθμούς και με την αποτελεσματικότητα παραγωγής υλικών αγαθών.

Για πολλά έργα ανάπτυξης λογισμικού δεν υπάρχουν ιστορικά δεδομένα.

Επειδή η τεχνολογία των ηλεκτρονικών υπολογιστών –και κατά

συνέπεια και του λογισμικού— εξελίσσεται με ταχύτατους ρυθμούς, για αρκετά έργα ανάπτυξης λογισμικού δεν έχουμε καθόλου ιστορικά δεδομένα, ή τα ιστορικά δεδομένα δεν είναι αξιοποιήσιμα. Ιστορικά δεδομένα αποκαλούμε δεδομένα από παρόμοια έργα που αναπτύχθηκαν κάτω από αντίστοιχες συνθήκες. Τέτοια δεδομένα, είτε δεν υπάρχουν γιατί έργα κάποιων κατηγοριών αναπτύσσονται (συνήθως) για πρώτη φορά, είτε δεν μπορούν να χρησιμοποιηθούν (έστω κι αν προέρχονται από αντίστοιχα έργα) γιατί οι μηχανισμοί, οι διαδικασίες και τα εργαλεία ανάπτυξης έχουν αλλάξει σημαντικά.

Η διαδικασία ανάπτυξης λογισμικού είναι σχετικά αδιαφανής.

Η διαδικασία ανάπτυξης λογισμικού δεν είναι τόσο διαφανής όσο είναι η διαδικασία κατασκευής σε αντίστοιχα κατασκευαστικά έργα ή έργα παραγωγής υλικών αγαθών, αλλά δεν είναι (ή δεν θα έπρεπε να είναι) αδιαφανής για τον υπεύθυνο του έργου. Φανταστείτε ένα κατασκευαστικό έργο (π.χ. την κατασκευή ενός σπιτιού) και πόσο εύκολα ακόμα και κάποιος που δεν έχει γνώση της διαδικασίας ανάπτυξης μπορεί να παρακολουθεί την εξέλιξη της κατασκευής και δείτε την αντιστοιχία του με ένα έργο ανάπτυξης λογισμικού. Στην πραγματικότητα ένας από τους στόχους της διαχείρισης της ανάπτυξης λογισμικού είναι η διαφάνεια στην ανάπτυξη. Η διαφάνεια αυτή, όπως θα δούμε και στη διάρκεια της μελέτης σας σχετίζεται, μεταξύ άλλων, με την ωριμότητα της επιχείρησης.

Στην ανάπτυξη λογισμικού οι άνθρωποι είναι σημαντικός παράγοντας της διαδικασίας.

Σε έρευνα του Curtis [Cur88] αναφέρεται ότι η πλειοψηφία των έμπειρων διαχειριστών έργων λογισμικού θεωρεί τους ανθρώπους ως το σημαντικότερο παράγοντα από τον οποίο εξαρτάται η επιτυχία του έργου. Σε αντίθεση με την παραγωγή υλικών αγαθών όπου τα εργαλεία και οι πρώτες ύλες είναι πολύ σημαντικές, στην ανάπτυξη λογισμικού, όπου τα εργαλεία και οι τεχνικές εξελίσσονται ραγδαία, η σημασία των ανθρώπων (με τις ιδιαιτερότητες που αυτό συνεπάγεται) είναι κυρίαρχη. Επίσης, στην ανάπτυξη λογισμικού ένας μεγάλος αριθμός έργων είναι προσαρμοσμένο λογισμικό (custom software), δηλαδή λογισμικό το οποίο αναπτύσσεται για συγκεκριμένο πελάτη ο οποίος εντάσσεται στη διαδικασία ανάπτυξης, γεγονός που εμπλέκει έναν ακόμα ανθρώπινο παράγοντα.

1.1.3 Η κρίση του λογισμικού ως πρόβλημα διαχείρισης

Σίγουρα έχετε ακούσει για τη λεγόμενη **κρίση του λογισμικού**. Η κρίση αυτή προκύπτει από το μεγάλο αριθμό έργων λογισμικού που δεν ολοκληρώθηκαν μέσα στα πλαίσια του χρονοδιαγράμματος, ή ξεπέρασαν κατά πολύ την αρχική εκτίμηση κόστους, ή δεν ικανοποίησαν τις λειτουργικές απαιτήσεις και τις απαιτήσεις ευχρηστίας του πελάτη, ή δεν σχεδιάστηκαν σωστά με αποτέλεσμα να μην μπορούν (τμήματα ή στο σύνολό τους) να ξαναχρησιμοποιηθούν.

Τα προβλήματα αυτά, συνδυαζόμενα με τις συνεχώς αυξανόμενες απαιτήσεις για λογισμικό (που ζητείται συνήθως να παραδοθεί άμεσα για να καλύψει αυτές τις ανάγκες) έχουν οδηγήσει μεγάλο μέρος του ανθρώπινου δυναμικού να απασχολείται με τη λεγόμενη *συντήρηση* (maintenance) –που γνωρίζετε ως φάση της ανάπτυξης λογισμικού– και που είναι η φάση κατά την οποία διορθώνουμε, αλλάζουμε ή βελτιώνουμε λογισμικό που έχει ήδη αναπτυχθεί και παραδοθεί στον πελάτη.

Η κρίση του λογισμικού τις περισσότερες φορές οφείλεται σε λάθη διαχείρισης.

Η λεγόμενη κρίση του λογισμικού είναι κυρίως συνέπεια προβλημάτων διαχείρισης και λαθών των υπευθύνων έργων ή της διοίκησης. Είναι χαρακτηριστικό ότι πολλές φορές στο βωμό της πυροσβεστικής αντιμετώπισης προβλημάτων παραβιάζονται δοκιμασμένες αρχές, με αποτέλεσμα να επιβεβαιώνουν τη σημασία αυτών των αρχών για μία φορά ακόμα. Για παράδειγμα, ενώ ο Brooks [Bro97] τονίζει ότι «προσθέτοντας ανθρώπους σε ένα έργο που έχει καθυστερήσει θα το καθυστερήσουμε περισσότερο», πολλές φορές μια διοίκηση σε κατάσταση πανικού καταφεύγει σε αυτή τη λύση.

Στο βιβλίο του “*The mythical man-month*” ο Brooks για το θέμα της κρίσης του λογισμικού παρουσιάζει πέντε βασικούς λόγους γιατί πολλά έργα ανάπτυξης λογισμικού αποτυγχάνουν στην τήρηση του αρχικού τους χρονοδιαγράμματος:

1. Οι τεχνικές εκτίμησης είναι ανεπαρκείς με αποτέλεσμα οι αρχικές εκτιμήσεις να είναι συνήθως υπερβολικά αισιόδοξες.
2. Συνήθως συγχέεται η *προσπάθεια* (effort) με την *πρόοδο* (progress) και γίνονται αριθμητικές πράξεις με ανθρωπομήνες που συχνά οδηγούν σε σοβαρά λάθη στην εκτίμηση. (Η λογική ότι ένα τμήμα του έργου που έγινε από 5 ανθρώπους σε 6 μήνες θα μπορέσει να γίνει από 10 ανθρώπους σε 3 μήνες είναι τελείως λάθος, γιατί θεωρεί ότι οι μήνες και οι άνθρωποι είναι απλές αριθμητικές μονάδες. Αν δεν σας έπεισα ακόμα, ας επεκτείνουμε στην ίδια λογική, θεωρώντας ότι ένας μήνας έχει 20 εργάσιμες ημέρες: τότε μπορούμε να υπολογίσουμε ότι 600 άνθρωποι θα ανέπτυσαν το ίδιο τμήμα του έργου σε 1 ημέρα!

3. Επειδή δεν υπάρχει βεβαιότητα στις εκτιμήσεις, συχνά οι υπεύθυνοι των έργων δεν έχουν την απαιτούμενη επιμονή στις απαιτήσεις τους.
4. Η διαδικασία ανάπτυξης δεν είναι εύκολα εποπτευόμενη από τους υπευθύνους. Αντίθετα συχνά την αντιμετωπίζουν σαν μία τελειώς σκοτεινή (αδιαφανή) διαδικασία στην οποία περιμένουν να δουν μόνο το τελικό αποτέλεσμα.
5. Τέλος όταν βγούμε εκτός χρονοδιαγράμματος, συχνά προστίθεται νέο προσωπικό. Αυτό είναι σαν να ρίχνει κανείς λάδι στη φωτιά, αυτή φουντώνει περισσότερο και αν συνεχίσουμε να ρίχνουμε κι άλλο λάδι, σύντομα θα έρθει η καταστροφή.

1.2 Διαδικασίες Διαχείρισης Έργων

Έχοντας ολοκληρώσει την ενότητα 1.1, έχετε αποκτήσει γνώσεις για βασικούς ορισμούς, για τις ιδιαιτερότητες του λογισμικού και για τα προβλήματα διαχείρισης του λογισμικού. Σε αυτή την ενότητα θα συζητήσουμε τη διαδικασία (process) διαχείρισης έργων λογισμικού. Θα μιλήσουμε δηλαδή για τις δραστηριότητες που πρέπει να έχει ο υπεύθυνος διαχείρισης έργου και για κάποιες από τις τεχνικές που χρησιμοποιεί.

Ο Pressman [Pre97] αναφέρει ότι η διαχείριση έργων λογισμικού πρέπει να επικεντρώνει το ενδιαφέρον της σε τρία «Ρ»: *People*, *Problem* και *Process*, που μεταφράζονται «*Άνθρωποι*», «*Πρόβλημα*» και «*Διαδικασία*». Για τους *ανθρώπους* θα μιλήσουμε στην ενότητα 1.3, ενώ το *πρόβλημα* σχετίζεται με τις συχνές επαφές με τον πελάτη για τον ακριβή καθορισμό των αναγκών του, ώστε να μην καταλήξουμε να δημιουργήσουμε λογισμικό που να μην ικανοποιεί τις ανάγκες του πελάτη. Για την ανάλυση του προβλήματος θα αναφερθούμε και στο πρώτο τμήμα της ενότητας 1.2.1. Τέλος για τη *διαδι-*

κασία θα μιλήσουμε σε αυτή την ενότητα.

Η ενότητα χωρίζεται σε δύο τμήματα, στο πρώτο τμήμα παρουσιάζουμε τις δραστηριότητες διαχείρισης έργων λογισμικού και στο δεύτερο τμήμα παρουσιάζουμε τεχνικές που χρησιμοποιούνται για τη διαχείριση έργων λογισμικού.

1.2.1 Ανάλυση διαδικασιών

Οι βασικές διαδικασίες που αφορούν τη διαχείριση έργων είναι:

- η συγγραφή της αρχικής πρότασης,
- ο προγραμματισμός του έργου (με όσα αυτός περιλαμβάνει),
- η ανάθεση έργου σε ανθρώπινο δυναμικό,
- η επίβλεψη του έργου και
- η τεκμηρίωση – εκπροσώπηση.

Από αυτές, οι δύο πρώτες γίνονται στα πρώτα στάδια της ανάπτυξης, ενώ οι δύο τελευταίες είναι συνεχείς διαδικασίες που ο διαχειριστής έργων λογισμικού εκτελεί σε όλη τη διάρκεια της ανάπτυξης του λογισμικού. Η ανάθεση έργου σε ανθρώπινο δυναμικό μπορεί είτε να γίνει στα πρώτα στάδια είτε να γίνεται σε συγκεκριμένα χρονικά σημεία του έργου, αν και ο καθορισμός των αναγκών θα έχει γίνει στα πρώτα στάδια.

1.2.1.1 Συγγραφή αρχικής πρότασης

Τις περισσότερες φορές πριν την εκκίνηση ενός έργου έχει προηγηθεί μία αντίστοιχη *αρχική πρόταση*. Η πρόταση αυτή μπορεί να είναι προς τον πελάτη, ή γενικά προς κάποιον φορέα χρηματοδότησης του έργου, ακόμα και προς το διοικητικό συμβούλιο για αυτοχρηματοδοτούμενο έργο. Πολλές φορές αυτή η πρόταση, ξεκινά από μια

αρχική ιδέα και μέχρι να ωριμάσει και να ξεκινήσει ένα έργο απαιτούνται πολλές συναντήσεις με τον πελάτη, συγγραφή πολλών ενημερωτικών και τεχνικών κειμένων, παρουσιάσεις του έργου κτλ. Την εργασία αυτή αναλαμβάνει ο υπεύθυνος έργου συνεπικουρούμενος από κάποιον ή κάποιους τεχνικούς.

Στην εύλογη ερώτηση «πώς ο υπεύθυνος έργου θα έχει πάντα ιδέες για νέα έργα;», η απάντηση είναι ότι ο καλός υπεύθυνος έργων θα μπορεί να αντλεί ιδέες από τη συνεργασία του με το προσωπικό που επιβλέπει στα έργα του. Ο καλός υπεύθυνος έργων θα πρέπει να εμπνέει τους συνεργάτες του ώστε να έχουν ιδέες και προτάσεις για το αντικείμενό τους και αυτός θα πρέπει με την εμπειρία του να εντοπίζει αυτές που έχουν προοπτικές και να τις προωθεί, χωρίς ποτέ να αγνοεί το συνεργάτη του που είχε την αρχική ιδέα (δηλαδή να αναφέρει πάντα την ιδέα ως ιδέα του «*Τάδε*»). Ο καλός υπεύθυνος έργων γνωρίζει ότι είναι επίτευγμα και προσόν του να μπορεί να εντοπίζει και να προωθεί τις καλές ιδέες των υφιστάμενών του και ότι δεν είναι μειωτικό για αυτόν να είχε κάποιος άλλος μία καλή ιδέα.

Ας συζητήσουμε ένα υποθετικό σενάριο ως παράδειγμα για τα παραπάνω: Σε μία επιχείρηση ο Περικλής είναι υπεύθυνος για ένα έργο ανάπτυξης λογισμικού. Μία νεοδιοριζόμενη προγραμματίστρια, η Ασπασία έχει μία πολύ καλή ιδέα ότι ένα τμήμα του έργου που η ίδια ανέπτυξε θα μπορούσε να έχει εμπορική αξία και να πωλείται αυτόνομα σε αρκετούς πελάτες, με κάποια πρόσθετη ανάπτυξη (νέο έργο). Τι από τα παρακάτω θα ήταν το σωστότερο να κάνει ο Περικλής;

1. *Να ετοιμάσει μία πρόταση για το νέο έργο και να την παρουσιάσει ως δική του;* Σίγουρα όχι! Αν κάνει κάτι τέτοιο, τότε μόλις το έργο εγκριθεί θα μαθευτεί από όλους τους υφιστάμενους και πιθανότατα και προϊστάμενους αφού η Ασπασία ίσως να διαμαρτυρηθεί. Ακόμα κι αν δεν το κάνει, ο Περικλής θα γίνει αντι-

παθής στους υφισταμένους του και το σημαντικότερο κανένας από αυτούς δεν θα του ξαναπέι κάποια ιδέα, η ακόμα χειρότερα θα προτιμήσει να τον παρακάμψει.

2. *Να ζητήσει από την Ασπασία να ετοιμάσει μία πρόταση για το νέο έργο και να της προτείνει να την παρουσιάσει αυτή, αφού αυτή είχε και την αρχική ιδέα;* Καλύτερα από την προηγούμενη επιλογή, αλλά και πάλι όχι ότι καλύτερο. Η Ασπασία ως νεοδιοριζόμενη προγραμματίστρια πιθανότατα δεν έχει την εμπειρία του Περικλή στον καθορισμό του έργου, στην εκτίμηση, την ανάλυση ρίσκου, την κοστολόγηση, τη συνεργασία με τον πελάτη και την παρουσίασή του. Σίγουρα η Ασπασία θα ήθελε τη βοήθεια του Περικλή σε αυτά τα θέματα. Εάν δεν την έχει, είναι πολύ πιθανό να αποτύχει να πείσει για το έργο και να απογοητευτεί στο να προτείνει κάτι νέο. Είναι πιθανόν έτσι η καλή της ιδέα να λειτουργήσει ως αντιπαράδειγμα δίνοντας την εντύπωση σε όλους ότι καλύτερη λογική είναι «να μην μπλέκεις με πράγματα που δεν είναι δουλειά σου».
3. *Να συνεργαστεί με την Ασπασία ώστε να ετοιμάσει αυτός μία πρόταση για νέο έργο, έχοντας τη βοήθεια της Ασπασίας για τα τεχνικά θέματα και να την παρουσιάσει μαζί με την Ασπασία, αναφέροντας ότι είναι ιδέα της Ασπασίας;* Είναι το καλύτερο που έχει να κάνει ο Περικλής. Η Ασπασία θα μάθει από την πείρα του Περικλή και θα τον εκτιμήσει περισσότερο, βλέποντάς τον να διορθώνει τα λάθη της και να προβάλλει την ιδέα της. Επίσης θα νιώσει χαρά όταν θα την παρουσιάσει μαζί με τον Περικλή. Τέλος οι άλλοι υφιστάμενοι του Περικλή θα θελήσουν να μιμηθούν την Ασπασία και θα έχουν και οι ίδιοι αντίστοιχες ιδέες. Ίσως ο Περικλής για κάποιο διάστημα να αρχίσει να βομβαρδίζεται με νέες ιδέες που θα απορρίπτει τις περισσότερες (στην

προσπάθεια όλων να μιμηθούν την Ασπασία), αλλά σίγουρα αξίζει τον κόπο να τις ακούσει όλες.

4. *Να βγάλει την Ασπασία για δείπνο;* Ίσως να σας προξενήσει κατάπληξη, αλλά και αυτό εμπεριέχει μία δόση αλήθειας! Δεν μιλάμε για προσωπικές βλέψεις του Περικλή για την Ασπασία, αλλά όταν το έργο (η ιδέα της Ασπασίας) εγκρινόταν, ένας καλός υπεύθυνος έργου θα έβγαζε, όχι μόνο την Ασπασία, αλλά όλη την ομάδα των άμεσων συνεργατών του για δείπνο πιθανότατα πληρωμένο από την εταιρία.

1.2.1.2 Προγραμματισμός έργου

Η αρχική φάση του *προγραμματισμού* (planning) του έργου, την οποία πιθανότατα έχετε μελετήσει στα πλαίσια των γενικότερων γνώσεών σας της τεχνολογίας λογισμικού, εμπεριέχει αρκετές δραστηριότητες διαχείρισης έργου. Στη φάση αυτή καθορίζονται οι βασικές προδιαγραφές του λογισμικού, γίνεται η *κοστολόγηση* του έργου, η *τιμολόγηση* του έργου (εάν είναι έργο που αναπτύσσεται για συγκεκριμένο πελάτη), η *εκτίμηση* των μεγεθών του έργου, η *μελέτη του εφικτού*, η *ανάλυση του ρίσκου*, η *αρχική τμηματοποίηση* και ο *χρονοπρογραμματισμός* του έργου.

Την κοστολόγηση του έργου, την εκτίμηση και την ανάλυση του ρίσκου θα τις συζητήσουμε χωριστά στο κεφάλαιο 2, ώστε να έχουμε την άνεση να τις παρουσιάσουμε πιο αναλυτικά, από τη σκοπιά της διαχείρισης έργων. Η αρχική τμηματοποίηση του έργου (χωρίς να μπαίνουμε σε θέματα σχεδίασης που ήδη έχετε διδαχθεί) και ο χρονοπρογραμματισμός είναι βασικές δραστηριότητες του υπεύθυνου έργου σε αυτή τη φάση.

Κατά την αρχική τμηματοποίηση του έργου ο υπεύθυνος έργου αναλαμβάνει να χωρίσει το έργο σε τμήματα ώστε να υπολογίσει τις

αλληλεξαρτήσεις ανάμεσα στα τμήματα και να εκτιμήσει το απαιτούμενο ανθρώπινο δυναμικό. Χρονοπρογραμματισμός είναι η διαδικασία κατά την οποία γίνεται εκτίμηση του χρόνου που θα πάρει κάθε τμήμα του έργου. Στο τμήμα 1.2.2 της ενότητας αυτής θα μιλήσουμε για τεχνικές που βοηθούν στην αρχική τμηματοποίηση και χρονοπρογραμματισμό του έργου.

Βασικό στοιχείο του χρονοπρογραμματισμού είναι ο καθορισμός των βασικών **ορόσημων** (milestones) του έργου.

Τα **ορόσημα** πήραν το αγγλικό όνομά τους «milestones» από τα πέτρινα κολωνάκια που ήταν τοποθετημένα παλαιότερα στην άκρη των δρόμων και έδειχναν σε ποιο μίλι βρίσκεται ο οδηγός. Σκοπός ενός ορόσημου είναι να κάνει περίπου ότι και το συγκεκριμένο κολωνάκι, να καθορίζει δηλαδή ένα σημαντικό σημείο του έργου που σχετίζεται με την ολοκλήρωση ενός μετρήσιμου στόχου.

Ο καθορισμός των ορόσημων είναι σημαντική και δύσκολη εργασία που θα βοηθήσει στην επίβλεψη του έργου. Ο Metzger [Met73] περιγράφει τη σημασία των ορόσημων και ορίζει ότι ένα ορόσημο θα πρέπει να είναι ξεκάθαρο πότε επιτεύχθηκε. Για παράδειγμα η ολοκλήρωση της κωδικοποίησης είναι ένα μετρήσιμο ορόσημο, αλλά η ολοκλήρωση του 50% του κώδικα δεν είναι καλή επιλογή για ορόσημο. Πώς θα καταλάβουμε ότι είμαστε στο 50% και όχι στο 45%; Η επίτευξη ενός ορόσημου πρέπει να οδηγεί σε τεκμηρίωση με τη μορφή **έκθεσης προόδου** (progress report).

Η έκθεση προόδου (progress report) είναι ένα τεχνικό κείμενο το οποίο συγγράφεται (συνήθως από τον υπεύθυνο έργου) με την επίτευξη κάποιου ορόσημου. Στην έκθεση προόδου γίνεται ανάλυση της συνολικής προόδου του έργου με αφορμή την επίτευξη του ορόσημου.

Εάν λοιπόν σε ένα έργο τοποθετηθούν πολλά ορόσημα τότε θα δημιουργηθεί πρόβλημα, αφού η ομάδα υλοποίησης θα αναλωθεί στη συγγραφή αναφορών προόδου. Ο αριθμός των ορόσημων πρέπει να είναι μικρός και να τοποθετούνται στα σημαντικά σημεία του έργου.

Στο προηγούμενο τμήμα του βιβλίου μιλήσαμε για την έκθεση προόδου (progress report). Υπάρχουν πολλοί δυνατοί τρόποι να γράψει κανείς μία έκθεση προόδου. Έτσι, θα σας δώσουμε υποδείξεις για το τι θα έπρεπε να ενταχθεί σε αυτή.

1. Ανάλυση του χρονοδιαγράμματος του έργου, δηλαδή καταγραφή του πότε (χρονικά) είχαμε προγραμματίσει να «πιάσουμε» το συγκεκριμένο ορόσημο και πότε πραγματικά φτάσαμε σε αυτό.
2. Αιτιολόγηση των αποκλίσεων, δηλαδή επεξήγηση γιατί υπήρξε απόκλιση από το αρχικό χρονοδιάγραμμα και που (σε ποιους παράγοντες) οφείλεται αυτή. Προσοχή, αυτή η ανάλυση θα γίνει ακόμα κι αν έχουμε κερδίσει χρόνο, ώστε να υπάρχει καταγεγραμμένη γνώση γιατί καταφέραμε να ξεπεράσουμε τις αρχικές προσδοκίες.
3. Ανάλυση των παραδοτέων. Συνήθως ένα ορόσημο αντιστοιχεί σε παράδοση (ή ολοκλήρωση) κάποιου σημαντικού τμήματος του έργου. Η ολοκλήρωση αυτού του τμήματος συνεπάγεται κάποια παραδοτέα, που θα πρέπει να αναγράφονται και να αναλύονται και στην έκθεση προόδου.

4. Γενική παρουσίαση της πορείας του έργου και των συνεπειών απόκλισης από το συγκεκριμένο ορόσημο. Συνήθως σε κάθε έργο η μη επίτευξη ενός ορόσημου στο αρχικά καθορισμένο χρονικό πλαίσιο σημαίνει ότι και άλλα ορόσημα πιθανότατα θα καθυστερήσουν αν δεν γίνουν κάποιες ενέργειες. Η έκθεση προόδου πρέπει να τα επισημαίνει.
5. Απαιτούμενες ενέργειες (σε περίπτωση προβλήματος φυσικά) που πρέπει να γίνουν ώστε να εξαλειφθούν (ή να ελαχιστοποιηθούν) οι συνέπειες από την μη επίτευξη αυτού του ορόσημου.

1.2.1.3 Ανάθεση έργου σε ανθρώπινο δυναμικό

Η *ανάθεση έργου σε ανθρώπινο δυναμικό* σχετίζεται με την *εκτίμηση* που θα συζητήσουμε αναλυτικά στο κεφάλαιο 2. Στα πρώτα στάδια του έργου, ο υπεύθυνος θα πρέπει να εντοπίσει τις ανάγκες του έργου σε προσωπικό γενικά, αλλά και να αναθέσει συγκεκριμένα τμήματά του σε κάποιους από τους υφισταμένους του. Προφανώς τμήματα του έργου που προβλέπεται να αρχίσουν πολύ μετά από την έναρξή του θα ανατεθούν σε προσωπικό εκείνη τη χρονική στιγμή, αλλά ο υπεύθυνος έργου θα πρέπει να έχει μεριμνήσει για τη διαθεσιμότητα του προσωπικού. Επειδή η διαθεσιμότητα του προσωπικού είναι σημαντικός παράγοντας για την επιτυχία του έργου, η ανάθεση έργου σε ανθρώπινο δυναμικό σχετίζεται με την αρχική τμηματοποίηση του έργου, όπου θα πρέπει να εξεταστεί ο φόρτος εργασίας που επιφέρει η παράλληλη εκτέλεση κάποιων τμημάτων του έργου. Για όλα αυτά όμως θα μιλήσουμε περισσότερο στην ενότητα 1.2.2.

1.2.1.4 Επίβλεψη έργου

Η *επίβλεψη* του έργου (project monitoring) είναι διαδικασία που εκτελείται σε όλη τη διάρκεια του έργου. Ο υπεύθυνος του έργου

πρέπει να παρακολουθεί όλες τις διαδικασίες του έργου (την πορεία και την πρόοδο όλων των τμημάτων του έργου) και να συγκρίνει το πραγματικό με το αρχικό χρονοδιάγραμμα, το πραγματικό κόστος με την αρχική εκτίμηση και την πραγματική προσπάθεια υλοποίησης με την αρχική εκτίμηση. Οι περισσότερες επιχειρήσεις ή οργανισμοί που παράγουν λογισμικό χρησιμοποιούν συγκεκριμένες τεχνικές (θα μιλήσουμε για μερικές στην ενότητα 1.2.2) και τα αντίστοιχα εργαλεία και μηχανισμούς, αλλά τις περισσότερες φορές ένας ανώτερος υπεύθυνος έργου μπορεί να μάθει πολύ περισσότερα από μία κατ' ιδίαν συνάντηση με το προσωπικό που έχει αναλάβει κάθε τμήμα του έργου. Βέβαια οι συγκεκριμένες συναντήσεις συνήθως θα είναι ενταγμένες στη διαδικασία παρακολούθησης και στο μηχανισμό της επιχείρησης.

Στο βιβλίο τους [Kra95] οι Kraul και Streeter, αναλύουν τους διάφορους τρόπους επικοινωνίας ανάμεσα στα μέλη της ομάδας ανάπτυξης έργου και τον υπεύθυνο του έργου. Οι τρόποι που προτείνουν μπορούν να συνδυαστούν σε δύο βασικούς τύπους επικοινωνίας:

- *Τυπική επικοινωνία.* Περιλαμβάνει την καθορισμένη (τόσο χρονικά, όσο και διαδικαστικά) ανταλλαγή αναφορών όπως τεκμηρίωση, παραδοτέα (κώδικας), αναφορές προόδου (ατομικής και τμήματος που έχει αναλάβει κάποιος), αναφορές λαθών ή προβλημάτων, προτάσεις αλλαγών (στο έργο ή στη διαδικασία παραγωγής) και γενικά ό,τι καθορίζει η επιχείρηση για τη διαδικασία του έργου. Ο τρόπος που γίνεται μπορεί να είναι με απλό ηλεκτρονικό ταχυδρομείο, ή με τη χρήση του ενδοδικτύου (intranet) της επιχείρησης, ή και με απλή ανταλλαγή τυπωμένων αναφορών, αν και οι ηλεκτρονικές μέθοδοι έχουν συνήθως καλύτερα αποτελέσματα.
- *Άτυπη επικοινωνία.* Περιλαμβάνει τη διαπροσωπική επικοινωνία

του υπεύθυνου έργου με κάθε μέλος της ομάδας υλοποίησης, αλλά και τις συναντήσεις ομάδων για ενημέρωση και συλλογική επίλυση προβλημάτων, ακόμα και τη δημιουργία ενός διευρυμένου κύκλου συναντήσεων στις οποίες θα συμμετέχει και προσωπικό έξω από τα πλαίσια του έργου το οποίο θα κληθεί για να βοηθηθεί με την εμπειρία του και τις γνώσεις του.

Η τυπική και η άτυπη επικοινωνία είναι το ίδιο σημαντικοί και συμπληρωματικοί μηχανισμοί επικοινωνίας μεταξύ των μελών μίας ομάδας. Οι περισσότεροι τυπικές προσεγγίσεις συνήθως φέρνουν καλύτερα αποτελέσματα σε επιχειρήσεις ή οργανισμούς με καλή διοικητική συγκρότηση και υψηλό επίπεδο ωριμότητας (θα συζητήσουμε για επίπεδα ωριμότητας μιας επιχείρησης σε επόμενα κεφάλαια, όταν θα μιλήσουμε για ποιότητα λογισμικού).

Τελειώνοντας πρέπει να τονίσουμε, ότι παρόλο που η επίβλεψη του έργου είναι σημαντικότερη δραστηριότητα του υπεύθυνου έργου, αυτός δεν πρέπει να δίνει την εντύπωση στους υφισταμένους του ότι είναι *επιβλέπων*, αλλά να δείχνει και να είναι *δάσκαλος* για αυτούς, να επισημαίνει τα λάθη τους, αλλά και να τους κατευθύνει προς το σωστό δρόμο και να τους διδάσκει με την εμπειρία του.

1.2.1.5 Τεκμηρίωση και εκπροσώπηση

Στα πλαίσια αυτής της ενότητας μιλήσαμε πολλές φορές για αναφορές που είναι ενταγμένες στα πλαίσια των διαδικασιών ανάπτυξης λογισμικού. Όλες αυτές οι αναφορές, καθώς και κάθε εσωτερικό κείμενο που θα διακινηθεί μόνο στα πλαίσια της επιχείρησης και κάθε εξωτερικό κείμενο που θα κοινοποιηθεί ή θα παραδοθεί στον πελάτη συνοψίζονται με τον όρο *τεκμηρίωση* (documentation).

Τεκμηρίωση (documentation) ονομάζεται κάθε είδος κειμένου που παράγεται στα πλαίσια ενός έργου λογισμικού.

Δυστυχώς πολλές φορές η σημασία της τεκμηρίωσης υποβαθμίζεται επειδή το βασικό παραδοτέο ενός έργου λογισμικού για πολλούς θεωρείται –κακώς– ότι είναι μόνο το λογισμικό. Για την αξία της τεκμηρίωσης θα αναφερθούμε συχνά στα πλαίσια αυτού του βιβλίου. Η τεκμηρίωση αν και θα είναι αρμοδιότητα πολλών από τα μέλη της ομάδας ανάπτυξης έργου, θα είναι τελική ευθύνη του υπεύθυνου έργου.

Τέλος, η *εκπροσώπηση* στα πλαίσια ενός έργου είναι και αυτή ενταγμένη στις διαδικασίες ανάπτυξης έργων λογισμικού. Ο υπεύθυνος έργου θα πρέπει να μπορεί να εκπροσωπεί το έργο τόσο στον πελάτη, όσο και στη διοίκηση της επιχείρησης ή του οργανισμού και να παρουσιάζει την πορεία του έργου, τα προβλήματα, καθώς και τις προτάσεις του για το έργο ή και για τις διαδικασίες ανάπτυξης.

1.2.2 Τεχνικές διαχείρισης

Στην ενότητα 1.2.1 μιλήσαμε για τις διαδικασίες του προγραμματισμού έργου, της ανάθεσης έργου σε ανθρώπινο δυναμικό και της επίβλεψης έργου, χωρίς να παρουσιάσουμε τις τεχνικές με τη βοήθεια των οποίων γίνονται όσα περιγράψαμε. Σε αυτή την ενότητα θα μιλήσουμε για τις πιο διαδεδομένες από αυτές τις τεχνικές, δηλαδή το δίκτυο δραστηριοτήτων έργου, το διάγραμμα αξιολόγησης έργου, το χρονοδιάγραμμα και το διάγραμμα ανάθεσης έργου σε ανθρώπινο δυναμικό. Στο τέλος της ενότητας θα παρουσιάσουμε ένα πλήρες παράδειγμα μιας υποθετικής διαχείρισης.

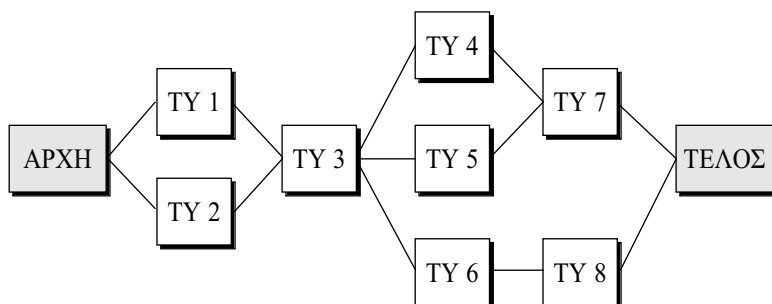
1.2.2.1 Δίκτυο δραστηριοτήτων έργου

Το δίκτυο *Εδραστηριοτήτων έργου* θα το βρείτε στην αγγλική βιβλιογραφία τόσο ως activity network, όσο και ως task network.

Το δίκτυο δραστηριοτήτων έργου είναι μία γραφική αναπαράσταση των διαφόρων δραστηριοτήτων (activities ή tasks) που συνθέτουν ένα έργο.

Στόχος του είναι να δείξει τις σχέσεις και εξαρτήσεις ανάμεσα στις διάφορες δραστηριότητες του έργου. Οι δραστηριότητες αυτές για μεγάλα έργα συχνά ονομάζονται και τυπικά υποέργα. Στο σχήμα 1.1 μπορείτε να δείτε ένα παράδειγμα ενός δικτύου δραστηριοτήτων έργου στο οποίο αναλύεται ένα έργο σε 8 τυπικά υποέργα (στο σχήμα συμβολίζονται ως TY1 ... TY8). Η σχεδίαση του δικτύου δραστηριοτήτων βοηθά στην καλύτερη αντίληψη των συσχετίσεων ανάμεσα στα διάφορα τμήματα στα οποία διασπάται ένα έργο.

Στο σχήμα 1.1 μπορούμε να δούμε ότι με την έναρξη του έργου αρχίζουν παράλληλα δύο τμήματα του έργου, τα TY 1 και TY 2. Για να μπορέσει να αρχίσει το τμήμα TY 3 πρέπει να έχουν ολοκληρωθεί και τα δύο προαπαιτούμενα τμήματα (TY 1 και TY 2). Μόνο μετά από την ολοκλήρωση του τμήματος TY 3 θα μπορέσουν να ξεκινήσουν τα TY 4, TY 5 και TY 6, κ.ο.κ. Προσέξτε ότι στο συγκεκριμένο σχήμα μπορούμε να δούμε μόνο ποιο τμήμα είναι προαπαιτούμενο κάποιου άλλου, αλλά όχι πληροφορίες όπως ο χρόνος που θα χρειαστεί (κατ' εκτίμηση) για την υλοποίηση ενός τμήματος, ή πληροφορίες για τμήματα που αν καθυστερήσουν θα καθυστερήσει και ολόκληρο το έργο. Αυτές οι πληροφορίες υπάρχουν στο διάγραμμα αξιολόγησης έργου το οποίο θα συζητήσουμε ακολούθως και που ουσιαστικά είναι ένα δίκτυο δραστηριοτήτων έργου με περισσότερες πληροφορίες.



Σχήμα 1.1

Παράδειγμα δικτύου δραστηριοτήτων έργου

1.2.2.2 Διάγραμμα αξιολόγησης έργου (διάγραμμα PERT)

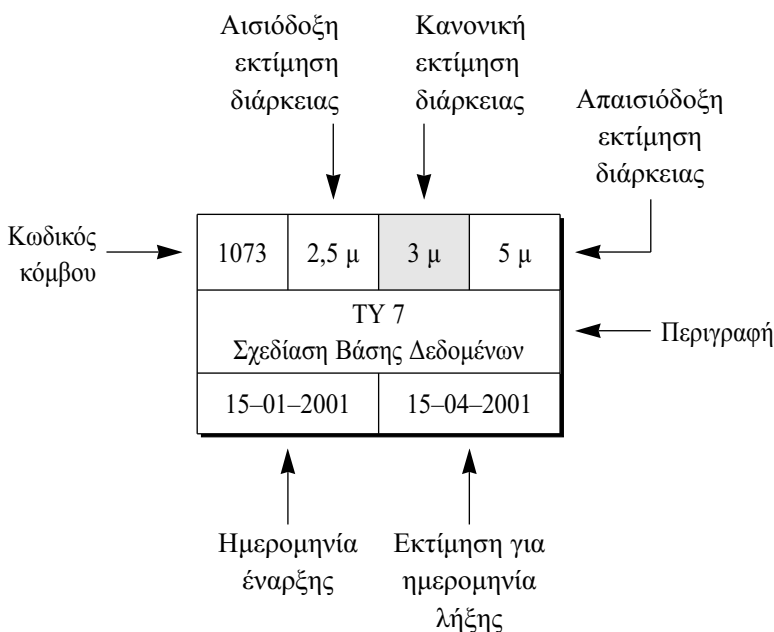
Το *διάγραμμα αξιολόγησης έργου* θα το βρείτε στην αγγλική βιβλιογραφία ως PERT Chart, όπου PERT προκύπτει από το *Program Evaluation and Review Technique*. Το PERT Chart είναι ένα δίκτυο δραστηριοτήτων στο οποίο έχουν προστεθεί ορόσημα και πληροφορίες για τη διάρκεια κάθε τμήματος (έναρξη, λήξη, κανονική διάρκεια, κτλ.).

Το **διάγραμμα αξιολόγησης έργου** (Program Evaluation and Review Technique ή συνοπτικά PERT Chart) είναι μία γραφική αναπαράσταση των διαφόρων **δραστηριοτήτων** (activities ή tasks) που συνθέτουν ένα έργο, εμπλουτισμένη με πληροφορίες όπως εκτιμήσεις διάρκειας και ορόσημα.

Υπάρχουν διάφορες μορφές που χρησιμοποιούνται για να αναπαρασταθεί ένας κόμβος σε ένα PERT Chart. Στις πιο απλές μορφές ένα τμήμα έχει τον κωδικό του και την εκτίμηση για διάρκεια, έναρξη και

λήξη, ενώ στις πιο εμπλουτισμένες μορφές, κάθε τμήμα συνοδεύεται από εκτιμήσεις διάρκειας τόσο κανονικές, όσο αισιόδοξες και απαισιόδοξες. Στο σχήμα 1.2 παρουσιάζεται ένας κόμβος από ένα είδος PERT Chart.

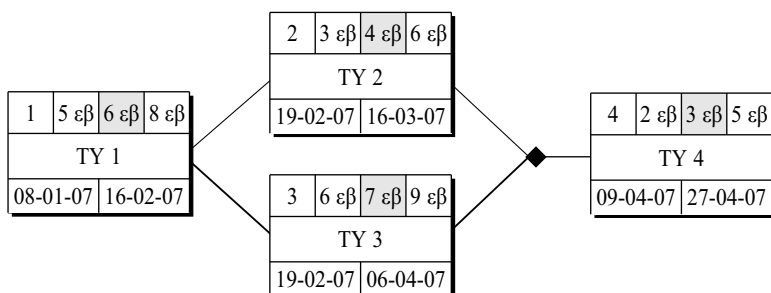
Στο σχήμα 1.2, στην πάνω αριστερή γωνία υπάρχει ο αριθμός αναφοράς του κόμβου (1073 για το παράδειγμα). Οι άλλες τρεις στήλες στο πάνω μέρος είναι εκτίμηση για τη διάρκεια του έργου σε μήνες. Η μεσαία από τις τρεις στήλες (3 μήνες για το παράδειγμα) αντιστοιχεί στην κανονική εκτίμηση, η πρώτη αντιστοιχεί στην αισιό-



Σχήμα 1.2
Κόμβος PERT Chart

δοξη εκτίμηση (2,5 μήνες για το παράδειγμα) και η τελευταία στην απαισιόδοξη εκτίμηση (5 μήνες για το παράδειγμα). Για να είναι πιο φανερή η κανονική ημερομηνία έχει χρωματιστεί. Στη μέση του κόμβου αναγράφεται η περιγραφή του τμήματος στο οποίο αντιστοιχεί ο κόμβος και στο κάτω μέρος η εκτίμηση για την έναρξη (στα αριστερά) και την ολοκλήρωση (στα δεξιά) του τμήματος. Σε πολλά παραδείγματα PERT Chart μπορούν να υπάρχουν πολλές πιθανές ημερομηνίες έναρξης (που εξαρτώνται από τις εκτιμήσεις των προαπαιτούμενων τμημάτων) και πολλές πιθανές ημερομηνίες λήξης (που εξαρτώνται τόσο από την ημερομηνία έναρξης όσο και από τη διάρκεια του τμήματος).

Στο σχήμα 1.3 παρουσιάζεται ένα έργο που έχει χωριστεί σε 4 τμήματα. Η ημερομηνία έναρξης του έργου είναι η 8^η Ιανουαρίου 2007 και ως ημερομηνία ολοκλήρωσης του έργου έχει προγραμματιστεί η 27^η Απριλίου 2007 (είναι δηλαδή ένα έργο με συνολική διάρκεια 16 εβδομάδες). Αυτές οι ημερομηνίες υπολογίζονται με βάση την κανονική εκτίμηση για τη διάρκεια του κάθε τμήματος. Παρατηρήστε ότι έχει υπολογιστεί κάθε εβδομάδα να τελειώνει Παρασκευή (π.χ.



Σχήμα 1.3
Παράδειγμα PERT Chart

16/2/2007)^[1] και η επόμενη να αρχίζει Δευτέρα (π.χ. 19/2/2007). Μετά την ολοκλήρωση των τμημάτων TY 2 και TY 3 υπάρχει ορόσημο (συμβολίζεται με ρόμβο).

Παρατηρήστε ότι το TY 4 που έχει ως προαπαιτούμενα τα TY 2 και TY 3 δεν θα αρχίσει μόλις τελειώσει το TY 2 (που τελειώνει αν όλα πάνε κανονικά νωρίτερα), αλλά μόλις τελειώσουν και τα δύο. Προσέξτε ότι τα τμήματα TY 1, TY 3 και TY 4 συνδέονται με έντονη γραμμή. Αυτό συμβαίνει γιατί συγκροτούν ένα *κρίσιμο μονοπάτι* (critical path).

Κρίσιμο μονοπάτι (critical path) είναι μια αλληλουχία δραστηριοτήτων από τις οποίες αν καθυστερήσει κάποια από αυτές αυτό θα έχει ως συνέπεια την καθυστέρηση όλου του έργου.

Το κρίσιμο μονοπάτι ξεκινά από την αρχή του έργου και τελειώνει με την ολοκλήρωση του έργου, διατρέχει δηλαδή ολόκληρο το έργο. Είναι πιθανό, αλλά όχι σύνηθες, σε ένα έργο να υπάρχουν πολλά κρίσιμα μονοπάτια ή όλα τα τμήματα του έργου να είναι ένα κρίσιμο μονοπάτι. Προσέξτε ότι αν για παράδειγμα καθυστερήσει το TY 3, τότε θα καθυστερήσει και η έναρξη του TY 4 και κατά συνέπεια και όλο το έργο. Το ίδιο θα συμβεί αν καθυστερήσει το TY 1.

Ένα θέμα για το οποίο δεν μιλήσαμε καθόλου είναι η διάρκεια που θα πρέπει να έχει κάθε τμήμα του έργου. Η απάντηση είναι ότι δεν υπάρχει κάποιος «μαγικός» αριθμός που να καθορίζει την ιδανική διάρκεια τμήματος. Είναι πολύ λογικό τα τμήματα ενός έργου συνο-

[1] Θα ήταν επίσης σωστό (αν και λιγότερο διαδεδομένο) να τελειώνει Κυριακή (δηλαδή να είναι 18/2/07 για το TY 1) αλλά το αφήσαμε Παρασκευή σεβόμενοι τις ημέρες αργίας.

λικής διάρκειας 5 ετών να έχουν μεγαλύτερη διάρκεια από τα τμήματα ενός έργου διάρκειας 6 μηνών, αλλά ούτε αυτό είναι απόλυτος κανόνας. Η εμπειρία έχει δείξει ότι συνήθως τμήματα έργων που έχουν διάρκεια 6 έως 14 εβδομάδες (δηλαδή 1,5 μήνα μέχρι 3,5 μήνες) είναι πιο εύκολο να διαχειριστούν και να επιβλέπονται. Πολύ μικρά τμήματα συνήθως δεν έχουν την απαιτούμενη αυτονομία (άρα κακώς ορίστηκαν ως αυτόνομα τμήματα), ενώ πολύ μεγάλα τμήματα συνήθως θα υποχρεωθούμε να τα χωρίσουμε σε μικρότερα.

Για να γίνει ακόμα πιο ξεκάθαρη η έννοια του κρίσιμου μονοπατιού ας συζητήσουμε λίγο ακόμα το σχήμα 1.3. Όπως παρουσιάζεται το διάγραμμα αξιολόγησης έργου (PERT Chart) στο σχήμα 1.3, γιατί δεν θα μπορούσε να είναι κρίσιμο μονοπάτι και η αλληλουχία των τμημάτων *TY 1*, *TY 2*, *TY 4*;

Η απάντηση είναι ότι η αλληλουχία των τμημάτων *TY 1*, *TY 2*, *TY 4* ξεκινά από τη αρχή του έργου και τελειώνει με το τέλος του έργου όπως κάθε κρίσιμο μονοπάτι. Όμως αν και τα *TY 1* και *TY 4* δεν μπορούν να καθυστερήσουν χωρίς να καθυστερήσει και το έργο ως συνέπεια αυτής της καθυστέρησης, αντίθετα το *TY 2* μπορεί να καθυστερήσει χωρίς απαραίτητα αυτό να έχει ως συνέπεια την καθυστέρηση του έργου. Δείτε στο σχήμα 1.3, ότι το *TY 2* έχει περιθώριο να τελειώσει ακόμα και 3 εβδομάδες μετά το προγραμματισμένο χωρίς να επιβραδύνει το έργο, αφού το *TY 4* που έπεται δεν θα ξεκινήσει πριν την ολοκλήρωση του *TY 3*. Για αυτό το λόγο, τα *TY 1* και *TY 4* συμμετέχουν στο κρίσιμο μονοπάτι *TY 1 – TY 3 – TY 4* και όχι μαζί με το *TY 2*.

1.2.2.3 Χρονοδιάγραμμα (διάγραμμα Gantt)

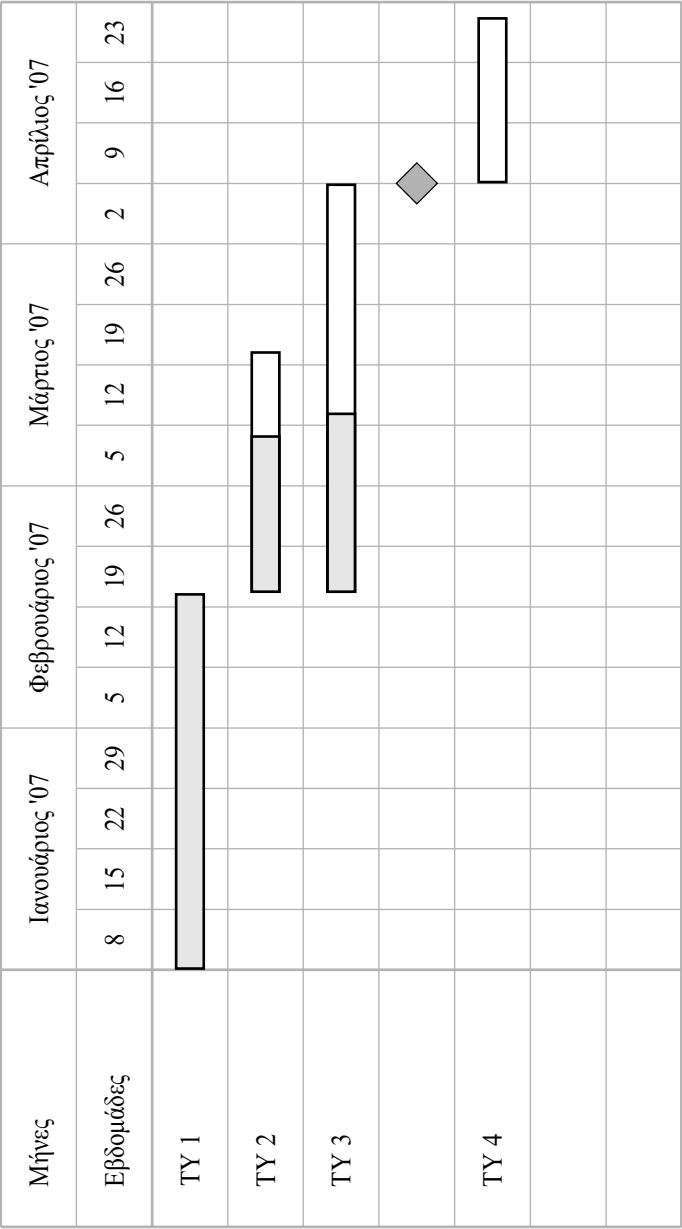
Το *χρονοδιάγραμμα* θα το βρείτε στην αγγλική βιβλιογραφία είτε ως *bar chart*, είτε ως *timeline chart*, είτε ως *Gantt chart*. Σκοπός του

Gantt chart είναι να δείξει, με χρήση οπτικών μέσων, το χρόνο που εκτιμάται ότι θα χρειαστεί κάθε τμήμα του έργου, αλλά και να χρησιμοποιηθεί από τον υπεύθυνο έργου κατά τη διάρκεια της επίβλεψης του έργου για παρακολούθηση της προόδου κάθε έργου και του ποσοστού ολοκλήρωσης κάθε τμήματος του έργου.

Στο σχήμα 1.4 παρουσιάζεται ένα παράδειγμα Gantt chart που αντιστοιχεί στο PERT Chart του σχήματος 1.3. Στο πάνω μέρος του χρονοδιαγράμματος υπάρχει μία κλίμακα που συνήθως είναι μήνες (ή τρίμηνα) για μεγάλα έργα και εβδομάδες για μικρά έργα. Κάθε τμήμα του έργου παρουσιάζεται με μία οριζόντια μπάρα. Το ποσοστό της μπάρας που είναι χρωματισμένη δείχνει το ποσοστό ολοκλήρωσης του κάθε τμήματος. Στο σχήμα 1.4 βλέπουμε ότι το *TY 1* έχει ολοκληρωθεί, ενώ τα *TY 2* και *TY 3* είναι σε εξέλιξη και το *TY 4* δεν έχει ξεκινήσει. Το Gantt chart μπορεί να χρησιμοποιηθεί και για την παρακολούθηση του φόρτου εργασίας του προσωπικού όπως θα δούμε στο επόμενο τμήμα της ενότητας.

Ας συζητήσουμε λίγο περισσότερο το Gantt chart. Αν ο υπεύθυνος έργου έχει μπροστά του το Gantt chart όπως παρουσιάζεται στο σχήμα 1.4 και ας υποθέσουμε ότι η τρέχουσα ημερομηνία είναι η 20/2/2007. Τι σημαίνει για αυτόν αυτό που βλέπει; Το έργο είναι μέσα στους στόχους του όσο αφορά την ολοκλήρωση εργασιών σε σχέση με το χρόνο; Πώς είναι η κατάσταση σε σχέση με τις αρχικές εκτιμήσεις;

Όπως παρουσιάζεται στο Gantt chart (δείτε που είναι η γραμμή στις 19/2/2007 και πόσο την έχουμε ξεπεράσει) το έργο πάει καλύτερα από τις αρχικές εκτιμήσεις. Έχει ήδη ολοκληρωθεί το *TY 1* και το ποσοστό ολοκλήρωσης των *TY 2* και *TY 3* είναι μεγαλύτερο του προγραμματισμένου για την τρέχουσα χρονική στιγμή. Όμως ενώ η ολοκλήρωση του *TY 1* είναι αδιαμφισβήτητο γεγονός η πρόοδος των *TY 2* και *TY 3*



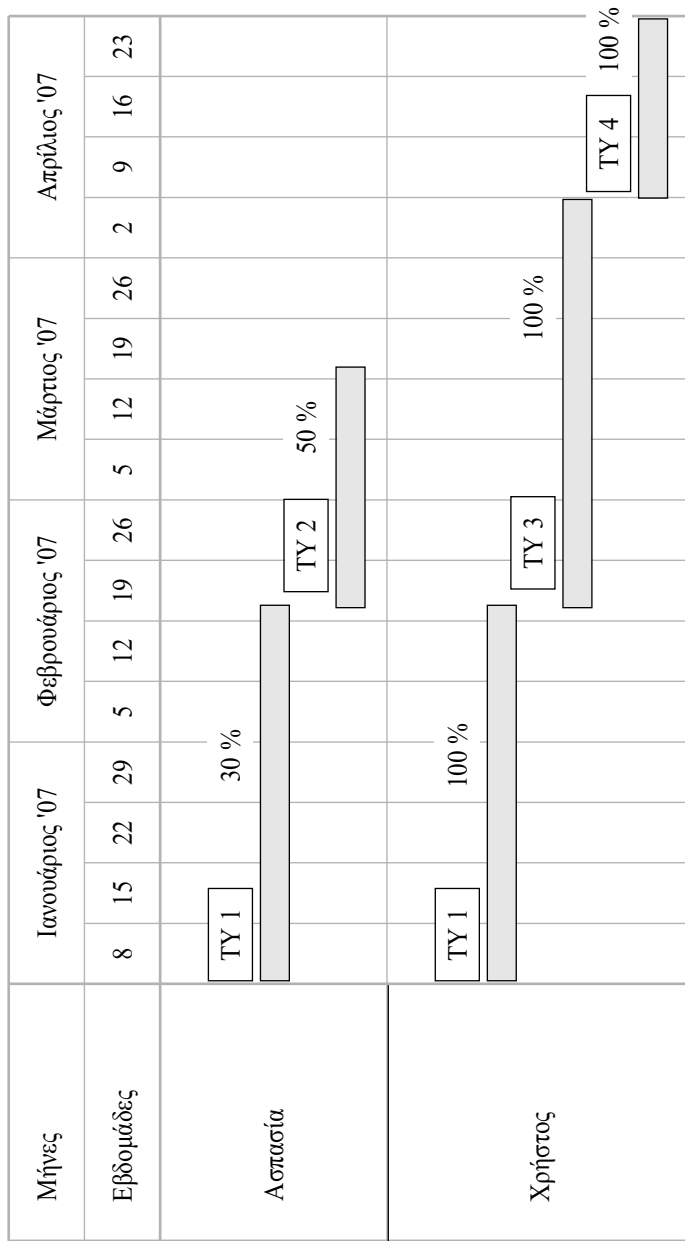
Σχήμα 1.4
Παράδειγμα Gantt chart

δεν θα έπρεπε να είναι λόγος εφησυχασμού για έναν καλό υπεύθυνο έργου. Πολλές φορές τα ποσοστά ολοκλήρωσης υπερεκτιμούνται και παρουσιάζονται υπερβολικά αισιόδοξες εκθέσεις προόδου με αποτέλεσμα η μόνη ουσιαστική ένδειξη να είναι η πραγματική ολοκλήρωση του έργου. Υπάρχει για αυτό και ο λεγόμενος κανόνας του 90–90 [Zah94] που σαρκάζει το γεγονός λέγοντας ότι: «το πρώτο 90% του έργου απαιτεί το 90% του χρόνου, ενώ το υπόλοιπο 10% απαιτεί πάλι το 90% του χρόνου!». Πάντως το γεγονός ότι αυτό δεν συνέβη στο TY 1 (το οποίο και έχει ολοκληρωθεί) σημαίνει ότι πιθανότατα οι εκτιμήσεις είναι σωστές και ότι το έργο προχωράει πραγματικά καλά.

1.2.2.4 Διάγραμμα ανάθεσης έργου σε ανθρώπινο δυναμικό

Το *διάγραμμα ανάθεσης έργου σε ανθρώπινο δυναμικό* (staff allocation chart) σχετίζει το χρονοδιάγραμμα του έργου με το προσωπικό που έχει την ευθύνη της υλοποίησης κάθε τμήματος.

Στο σχήμα 1.5 παρουσιάζεται ένα παράδειγμα διαγράμματος ανάθεσης έργου σε ανθρώπινο δυναμικό, όπου παρατηρούμε ότι για το έργο θα εργαστούν δύο υπάλληλοι: η Ασπασία και ο Χρήστος. Η Ασπασία θα εργαστεί στο TY 1, απασχολούμενη σε ποσοστό 30% του χρόνου της (δηλαδή από 8/1/2007 έως 16/2/2007 θα είναι διαθέσιμη να εργαστεί και κάπου αλλού σε ποσοστό 70% του χρόνου της) και στο TY 2 σε ποσοστό 50% του χρόνου της. Ο Χρήστος θα εργαστεί αποκλειστικά (σε ποσοστό 100%) στο έργο στα TY 1, TY 3 και TY 4. Με το παραπάνω σχήμα προκύπτουν πληροφορίες για την απαιτούμενη προσπάθεια υλοποίησης, όπως για παράδειγμα ότι για το TY 1 θα χρειαστούμε συνολικά 7,8 ανθρωποεβδομάδες ($6 \times 100\% = 6$ του Χρήστου και $6 \times 30\% = 1,8$ της Ασπασίας). Επίσης προκύπτουν πληροφορίες για τη διαθεσιμότητα του προσωπικού, για παράδειγμα ότι ο Χρήστος θα είναι «δεσμευμένος» με το έργο από 8/1/2007 έως 26/4/2007, ή για υπερφόρτωση του προσωπικού (για



Σχήμα 1.5
Παράδειγμα διαγράμματος ανάθεσης έργου σε ανθρώπινο δυναμικό

παράδειγμα αν ο Χρήστος δούλευε και κατά 50% στο ΤΥ 2 τότε στο ίδιο χρονικό διάστημα θα έπρεπε να δουλεύει κατά 150% αφού τα ΤΥ 2 και ΤΥ 3 είναι παράλληλα).

Η συζήτηση που ακολουθεί αφορά την πρακτική εφαρμογή στα θέματα που αναπτύξαμε στην ενότητα 1.2.2. Ας υποθέσουμε, λοιπόν, ότι ο Περικλής έχει αναλάβει υπεύθυνος ενός νέου έργου με κωδικό όνομα «Αρχείο» που αφορά τη δημιουργία ενός συστήματος αρχειοθέτησης για ένα πελάτη με κύρια συστατικά του τη Βάση Δεδομένων και το λογισμικό διεπαφής με τον πελάτη. Μέσα στις δραστηριότητες του έργου περιλαμβάνεται η αρχική επαφή με τον πελάτη και ο έλεγχος και η εγκατάσταση του τελικού συστήματος στον πελάτη. Μπορείτε να θεωρήσετε ότι το έργο αποτελείται από τις παρακάτω φάσεις (τυπικά υποέργα):

ΤΥ1 Ανάλυση αναγκών πελάτη

ΤΥ2 Σχεδίαση Βάσης Δεδομένων

ΤΥ3 Σχεδίαση Λογισμικού και διεπαφής

ΤΥ4 Ανάπτυξη Βάσης Δεδομένων

ΤΥ5 Ανάπτυξη Λογισμικού και διεπαφής

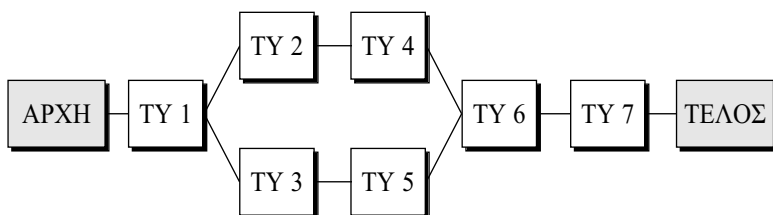
ΤΥ6 Ολοκλήρωση συστήματος και Έλεγχος

ΤΥ7 Εγκατάσταση στον πελάτη και αποδοχή

Το έργο θα έχει διάρκεια ένα χρόνο (ας υποθέσουμε ότι ξεκινά 1/1/2007) και στη διάθεση του Περικλή θα είναι ο *Γιώργος* (ένας έμπειρος αναλυτής συστημάτων), η *Ασπασία* (μία προγραμματίστρια με μεγάλη εμπειρία σε πολλά έργα και εξειδίκευση σε θέματα σχεδίασης και ανάπτυξης Βάσεων Δεδομένων), η *Έλενα* (προγραμματίστρια με ειδίκευση στην ανάπτυξη διεπαφών χρήστη λογισμικού), ο *Σπύρος* (προγραμματιστής με εμπειρία και σε έλεγχο συστημάτων)

και η *Στέλλα* (έμπειρη στον έλεγχο και εγκατάσταση συστημάτων). Η αρχική εκτίμηση της διοίκησης σε συνεργασία με τον Περικλή είναι ότι το έργο θα χρειαστεί λιγότερο από 15 ανθρωπομήνες, που σημαίνει ότι δεν θα εργαστούν όλοι οι παραπάνω για 12 μήνες στο 100% του χρόνου τους.

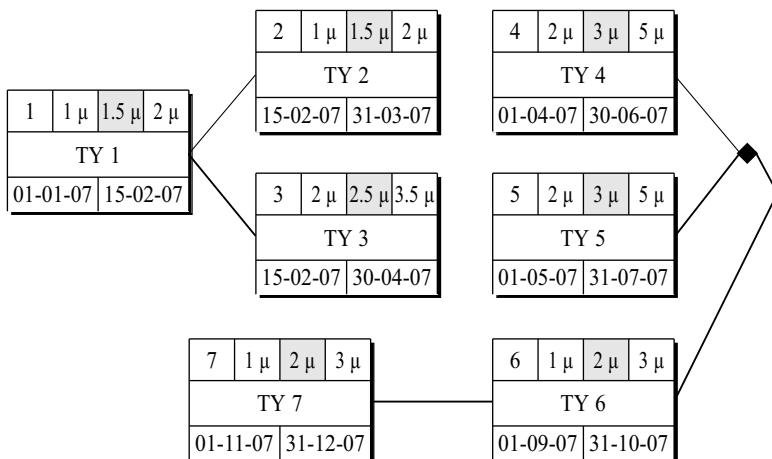
Καλείστε αξιοποιώντας και την εμπειρία που έχετε από τη μελέτη του βιβλίου έως τώρα, να βοηθήσετε τον Περικλή και να σχεδιάσετε το δίκτυο δραστηριοτήτων έργου, το διάγραμμα αξιολόγησης έργου, το χρονοδιάγραμμα έργου και το διάγραμμα ανάθεσης έργου σε ανθρώπινο δυναμικό. Επίσης να υπολογίσετε τους συνολικούς ανθρωπομήνες του έργου και πόσο θα εργαστεί κάθε ένας από το διαθέσιμο προσωπικό. Μη συμπεριλάβετε στους παραπάνω τον Περικλή. Η λύση δεν είναι μονοσήμαντη, άρα η δική σας λύση δεν πρέπει αναγκαστικά να είναι ίδια με την προτεινόμενη, αλλά ούτε και είναι λογικό να έχει σημαντικές διαφορές. Καλύτερα δοκιμάστε να δώσετε τη δική σας λύση και μετά να μελετήστε τη λύση που έδωσε ο Περικλής που ακολουθεί.



Σχήμα 1.6

Δίκτυο δραστηριοτήτων έργου «Αρχείο»

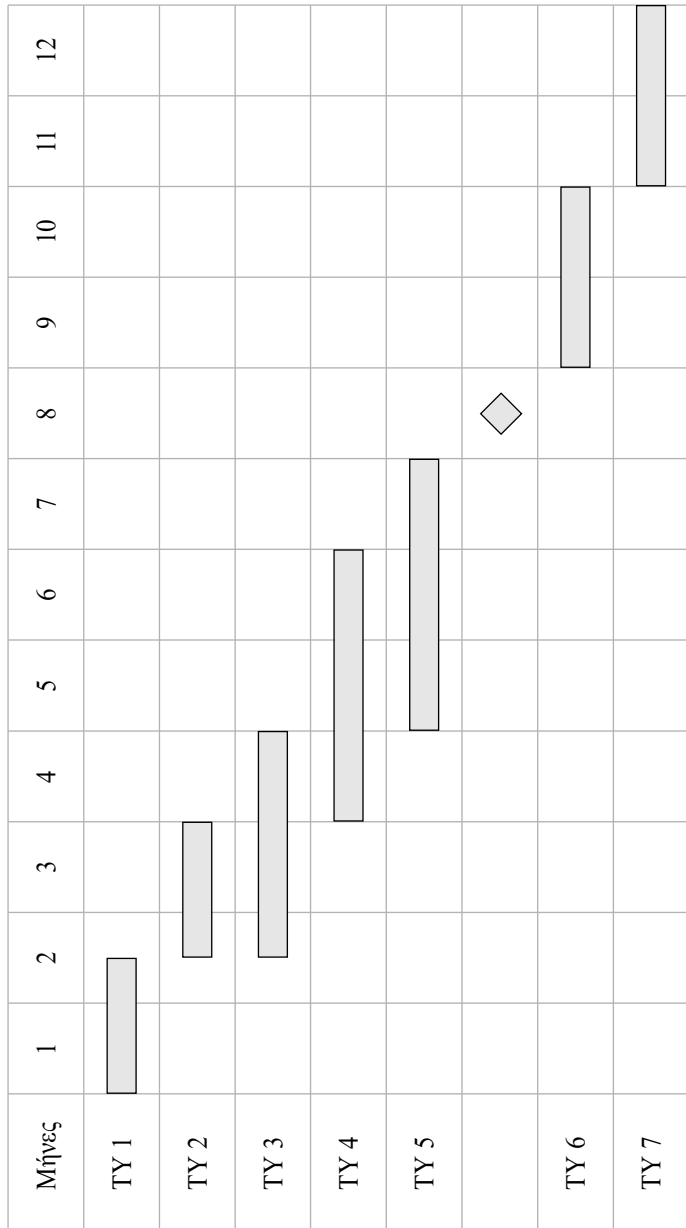
Ο Περικλής, λοιπόν, σχεδίασε το δίκτυο δραστηριοτήτων έργου που φαίνεται στο σχήμα 1.6. Για λόγους απλότητας στη συνέχεια της απάντησης θα χρησιμοποιούμε τις συντομογραφίες αντί για το πλήρες όνομα.



Σχήμα 1.7
PERT Chart έργου «Αρχείο»

Με βάση την εμπειρία του ο Περικλής έκανε τις εκτιμήσεις για τη διάρκεια κάθε τμήματος που παρουσιάζονται στο σχήμα 1.7. Προσέξτε ότι άφησε αρκετό χρόνο για τον έλεγχο του συστήματος και για την εγκατάσταση στον πελάτη, ώστε να εντοπιστούν και να διορθωθούν λάθη καθώς και για να ικανοποιηθούν αλλαγές που θα ζητήσει ο πελάτης μετά την εγκατάσταση του συστήματος και μέχρι την τελική αποδοχή. Επίσης όρισε ως ορόσημο το σημείο που το σύστημα θα έχει ολοκληρωθεί και θα είναι έτοιμο προς τον τελικό έλεγχο συστήματος. Προσέξτε επίσης το κενό που άφησε το μήνα Αύγουστο (για πιθανές διακοπές στο προσωπικό ή για κάλυψη χαμένου χρόνου).

Με βάση το PERT Chart του σχήματος 1.7 το χρονοδιάγραμμα του έργου προκύπτει εύκολα και παρουσιάζεται στο σχήμα 1.8 (Αν και μικρό έργο δίνουμε την κλίμακα σε μήνες για να μην «γεμίσει» πολύ το σχήμα).



Σχήμα 1.8
Χρονοδιάγραμμα έργου «Αρχείο»

Τέλος, με βάση τη διαθεσιμότητα του προσωπικού, τους αρχικούς περιορισμούς σε χρόνο και τα προσόντα κάθε ενός από τους διαθέσιμους συνεργάτες, ο Περικλής ετοίμασε το διάγραμμα ανάθεσης έργου σε ανθρώπινο δυναμικό που παρουσιάζεται στο σχήμα 1.9. Παρατηρήστε ότι προσπάθησε να αξιοποιήσει τη μεγάλη εμπειρία της Ασπασίας, δίνοντάς της μικρή συμμετοχή τόσο στις επαφές με τον πελάτη, όσο και στην ολοκλήρωση και εγκατάσταση του συστήματος.

Το σύνολο των ανθρωπομηνών για κάθε τμήμα, για το σύνολο του έργου, αλλά και η συμμετοχή κάθε εργαζομένου προκύπτει εύκολα αφού γνωρίζουμε τη διάρκεια κάθε τμήματος, το προσωπικό που συμμετέχει και το ποσοστό συμμετοχής. Έχουμε λοιπόν:

TY1: $1,5 \times 50\%$ για το Γιώργο και $1,5 \times 20\%$ για την Ασπασία, σύνολο 1,05 ανθρωπομήνες.

TY2: $1,5 \times 100\%$ για την Ασπασία, σύνολο 1,5 ανθρωπομήνες.

TY3: $2,5 \times 100\%$ για την Έλενα, σύνολο 2,5 ανθρωπομήνες.



TY4: $3 \times 50\%$ για την Ασπασία, σύνολο 1,5 ανθρωπομήνες.

TY5: $3 \times 100\%$ για την Έλενα και $3 \times 50\%$ για το Σπύρο, σύνολο 4,5 ανθρωπομήνες.

TY6: $2 \times 50\%$ για το Σπύρο και $2 \times 50\%$ για τη Στέλλα, σύνολο 2 ανθρωπομήνες.

TY7: $2 \times 20\%$ για την Ασπασία και $2 \times 50\%$ για τη Στέλλα, σύνολο 1,4 ανθρωπομήνες.

Προσθέτοντας τα παραπάνω προκύπτει ότι το σύνολο του έργου είναι 14,45 ανθρωπομήνες κάτι που είναι μέσα στις αρχικές προδιαγραφές. Επίσης με απλούς υπολογισμούς βρίσκουμε τους παρακάτω ανθρωπομήνες: (Γιώργος: 0,75. Ασπασία: 3,7. Έλενα: 5,5. Σπύρος: 2,5. Στέλλα: 2).

Μήνες	1	2	3	4	5	6	7	8	9	10	11	12
Γιώργος		 TY 1		50 %								
Ασπασία							TY 1	20 %			TY 7	20 %
							TY 2	100 %				
							TY 4	50 %				
Έλενα			TY 3	100 %		TY 5	100 %					
Σπύρος												
							50 %		TY 6	50 %		
Στέλλα												
									TY 6	50 %	TY 7	50 %

Σχήμα 1.9

Διάγραμμα ανάθεσης έργου σε ανθρωπινό δυναμικό για το έργο «Αρχείο»

Όπως είπαμε η λύση δεν είναι μονοσήμαντη και σας έλειπαν πολλά στοιχεία για να δώσετε μια πιο τεκμηριωμένη λύση. Στόχος της συζήτησης ήταν να δούμε ένα παράδειγμα όλων των τεχνικών και όχι να παρουσιάσουμε τη λύση του συγκεκριμένου προβλήματος.

1.3 Οι Άνθρωποι

Στην ενότητα 1.1.2 αναφέραμε ότι οι άνθρωποι είναι σημαντικός παράγοντας της διαδικασίας ανάπτυξης λογισμικού. Για τη σημασία των ανθρώπων στη διαδικασία ανάπτυξης λογισμικού μιλήσαμε και στην ενότητα 1.2. Σε αυτή την ενότητα θα αναλύσουμε τους ρόλους κάθε εργαζόμενου σε μία επιχείρηση ή οργανισμό που αναπτύσσει λογισμικό. Θα μιλήσουμε, δηλαδή για το *προσωπικό*.

Προσωπικό (personnel) μίας επιχείρησης ή οργανισμού είναι το σύνολο των ανθρώπων που εργάζονται σε αυτή.

Οι άνθρωποι που εργάζονται σε μία επιχείρηση ή οργανισμό συνήθως αναφέρονται και ως προσωπικό (personnel). Μερικές φορές στην αγγλική βιβλιογραφία θα βρείτε τον όρο *staff* (θυμηθείτε το *staff allocation chart* για το οποίο μιλήσαμε στην ενότητα 1.2.2).

Στην ενότητα αυτή θα μιλήσουμε για όσους συμμετέχουν στην ανάπτυξη λογισμικού. Αυτοί είναι το προσωπικό της επιχείρησης ή οργανισμού, αλλά και ο πελάτης. Δεν θα επεκταθούμε σε τεχνικές λεπτομέρειες για το τι κάνει καθένας από το προσωπικό, αφού αυτό είναι αντικείμενο της Τεχνολογίας Λογισμικού. Επειδή συχνά στα επόμενα κεφάλαια του βιβλίου θα αναφερθούμε στους συμμετέχοντες στην ανάπτυξη λογισμικού καλό θα ήταν να μελετήσετε προσεκτικά αυτό το κεφάλαιο πριν προχωρήσετε. Εκτός από τους ρόλους που αναφέρουμε σε αυτή την ενότητα υπάρχουν πολλές ειδικότητες (εξειδικεύσεις των ρόλων) που δεν τις περιγράφουμε.

1.3.1 Μέλος της Διοίκησης

Είπαμε στην ενότητα 1.1.1 ότι θα διαχωρίσουμε το ρόλο του υπεύθυνου έργου από αυτόν της διοίκησης της επιχείρησης ή του οργανισμού. Σε όλη τη διάρκεια του βιβλίου θα αναφερόμαστε συχνά απρόσωπα στη διοίκηση του οργανισμού (θα μιλήσουμε για παράδειγμα για τη «δέσμευση της διοίκησης στο πρόγραμμα ποιότητας»). Δεν πρέπει να ξεχνάμε ότι και η διοίκηση γίνεται από ανθρώπους και η άποψη της διοίκησης εκπροσωπείται από κάποιον άνθρωπο που συνήθως είναι ο Γενικός Διευθυντής της επιχείρησης ή του οργανισμού. Σκοπός του βιβλίου είναι να μιλήσουμε για τη διαχείριση έργων λογισμικού και όχι για κανόνες διοίκησης επιχειρήσεων, οπότε δεν θα δώσουμε βάρος στους ρόλους, αρμοδιότητες και σκοπούς της διοίκησης. Σκόπιμα στο βιβλίο δεν θα μιλάμε συνήθως σε τρίτο ενικό πρόσωπο για τον εκπρόσωπο της διοίκησης, αλλά απρόσωπα για τη διοίκηση. Πάντως και για το μέλος της διοίκησης ισχύουν όσα θα πούμε και για τον υπεύθυνο έργων, αφού και αυτός είναι ένας άνθρωπος που έχει την ευθύνη να διοικεί ανθρώπους με ό,τι προβλήματα και ιδιαιτερότητες αυτό συνεπάγεται.

Η διοίκηση της επιχείρησης ή του οργανισμού έχει ως μέριμνα την επιβίωση της επιχείρησης και το κέρδος. Θέλει τα έργα ανάπτυξης λογισμικού να αναπτύσσονται μέσα στα χρονοδιαγράμματα (ώστε να συμβάλουν στην καλή εικόνα της επιχείρησης), να κοστίζουν λιγότερο από τα έσοδα που θα αποφέρουν (ώστε να αφήνουν κέρδος) και να δίνουν προοπτική για νέα έργα (είτε με μεταπώληση τμημάτων ή και ολόκληρων έργων, είτε με ιδέες για νέα έργα). Μια καλή διοίκηση θα πρέπει να αντλεί ιδέες από το προσωπικό και να φροντίζει για τη συνεχή εξέλιξη και βελτίωση των συνθηκών ανάπτυξης και των διαδικασιών καθώς και για την ικανοποίηση του προσωπικού. Στο κεφάλαιο 3 θα μιλήσουμε περισσότερο για τη διοίκηση στα πλαίσια της «διαχείρισης της ποιότητας», οπότε ας μην επεκταθούμε άλλο σε αυτό το κεφάλαιο.

1.3.2 Υπεύθυνος έργου ή έργων

Μιλήσαμε λίγο για τον υπεύθυνο διαχείρισης έργου λογισμικού ή υπεύθυνο έργου στην ενότητα 1.1.1. Ουσιαστικά τα δύο πρώτα κεφάλαια του βιβλίου αυτού είναι αφιερωμένα στον υπεύθυνο έργου, αφού όσα είπαμε στην ενότητα 1.2, αλλά και όλο το κεφάλαιο 2 αναλύει τις δραστηριότητες διαχείρισης για τις οποίες έχει την ευθύνη. Η ευθύνη του υπεύθυνου έργου ξεκινά συνήθως από τη συγγραφή της αρχικής πρότασης και συνεχίζεται με τον προγραμματισμό του έργου (στον προγραμματισμό περιλαμβάνονται η εκτίμηση και ανάλυση ρίσκου που θα αναλύσουμε στο κεφάλαιο 2, καθώς και η αρχική τμηματοποίηση, η ανάθεση έργου σε ανθρώπινο δυναμικό και ο χρονοπρογραμματισμός που συζητήσαμε στην ενότητα 1.2). Η ευθύνη του υπεύθυνου έργου είναι ο καθημερινός έλεγχος του έργου, η επίβλεψη του έργου και η εκπροσώπηση και τεκμηρίωση που επίσης συζητήσαμε.

Για να μπορεί να επιτυγχάνει όλα τα παραπάνω ο υπεύθυνος έργου πρέπει να είναι άνθρωπος με ικανότητες οργανωτικές, αλλά και ηγετικές. Πρέπει να μπορεί να εμπνέει και να εμπνυχώνει τους υφισταμένους του, αλλά και να κατανοεί το προβλήματα και τις δυσκολίες τους. Για αυτό το λόγο οι καλύτεροι υπεύθυνοι έργων είναι αυτοί που έχουν εμπλακεί στα διάφορα στάδια της ανάπτυξης και έχουν αποκτήσει την εμπειρία του μηχανικού ανάπτυξης που θα συζητήσουμε στην ενότητα 1.3.4. Επίσης, ο υπεύθυνος έργου πρέπει να μπορεί να επιλύει συστηματικά προβλήματα, να αξιοποιεί την εμπειρία του από προηγούμενα έργα και να διαθέτει ευελιξία ώστε να μπορεί να αλλάζει κατεύθυνση προλαβαίνοντας δύσκολες καταστάσεις. Πρέπει να δείχνει πάντα ότι έχει τον έλεγχο του έργου, αλλά και να αφήνει στους υφισταμένους του την πρωτοβουλία και να επιτρέπει (και να ενθαρρύνει με τις ενέργειές του) την ανάληψη από τους υφισταμένους του πρωτοβουλιών (πάντα με μικρό ή ελεγχόμε-

νο ρίσκο). Τέλος πρέπει να μπορεί να «χτίζει» να μεριμνά για το «δέσιμο» και να επιβλέπει ομάδες εργασίας (για τις οποίες θα μιλήσουμε στην ενότητα 1.4 που ακολουθεί) που θα αναλαμβάνουν τμήματα του έργου. Είναι σαφές ότι τα παραπάνω διδάσκονται, αλλά κυρίως μαθαίνονται στην πράξη και πολλές φορές εξαρτάται από την προσωπικότητα, την ιδιοσυγκρασία και τον χαρακτήρα κάθε ανθρώπου πόσο επιτυχημένος υπεύθυνος έργου θα γίνει.

Υπεύθυνοι έργων ή ανώτεροι υπεύθυνοι έργων (senior project managers) είναι μέλη του προσωπικού μιας επιχείρησης ή οργανισμού που έχουν ανέβει σε ανώτερο διοικητικό επίπεδο από αυτό του υπεύθυνου έργου και έχουν αναλάβει την ευθύνη πολλών έργων.

Συχνά στη βιβλιογραφία θα βρείτε και τον όρο *υπεύθυνοι έργων* ή *ανώτεροι υπεύθυνοι έργων*, που στην αγγλική βιβλιογραφία αποδίδεται ως *senior project managers*. Οι ανώτεροι υπεύθυνοι έργων (όπως θα τους αποκαλούμε) έχουν συνήθως διατελέσει για κάποια χρόνια υπεύθυνοι έργου και έχουν αποδείξει τις ικανότητές τους στην πράξη ώστε να αναλάβουν τη διαχείριση παραπάνω του ενός έργου. Συνήθως είναι λίγοι σε κάθε επιχείρηση ή οργανισμό και οι απόψεις τους επηρεάζουν σε μεγάλο βαθμό την πολιτική της διοίκησης. Είναι δηλαδή οι άνθρωποι αυτοί στους οποίους θα στραφεί η διοίκηση για να ρωτήσει τη γνώμη τους πριν από σημαντικές αποφάσεις.

1.3.3 Ηγέτης ομάδας

Συχνά ο ρόλος του *ηγέτη* (leader) κάποιων ανθρώπων δεν ανήκει αποκλειστικά στον υπεύθυνο έργου. Ο Edgemon αναφέρει ότι «...πολύ συχνά μέλη του προσωπικού θα αποκτήσουν τυχαία το

ρόλο του ηγέτη για κάποια ομάδα ανθρώπων...». Ακριβώς επειδή συχνά στα πλαίσια ενός έργου δημιουργούνται διάφορες ομάδες, προκύπτει η ανάγκη κάποιοι άνθρωποι να ηγηθούν κάποιων άλλων στα πλαίσια της ομάδας. Αυτούς θα τους αποκαλούμε *ηγέτες ομάδας*, αφού έχουν αναγκαστικά ηγετικό (διαχειριστικό ρόλο), χωρίς όμως να έχουν τον τίτλο (και τις ευθύνες) του υπεύθυνου έργου. Η ανάδειξη των καλύτερων μηχανικών ανάπτυξης σε αυτούς τους ρόλους είναι συνήθως φυσική συνέπεια και στην περίπτωση που αποδείξουν τις διαχειριστικές τους ικανότητες φυσική εξέλιξη θα είναι και η μετάβασή τους στο ρόλο του υπευθύνου έργου.

1.3.4 Μηχανικός ανάπτυξης

Με τον όρο *μηχανικός ανάπτυξης* (*software engineer*) θα αποκαλούμε όλους όσους έχουν ενεργό ρόλο σε διάφορα στάδια της ανάπτυξης λογισμικού και που ο ρόλος αυτός απαιτεί γνώσεις τεχνολογίας λογισμικού και δεν περιορίζεται μόνο στον προγραμματισμό συγκεκριμένων και καθορισμένων τμημάτων. Χρησιμοποιούμε τον όρο μηχανικός ως απόδοση του *engineer* και όχι για να υπονοήσουμε τον τίτλο του Μηχανικού (ως απόφοιτος συγκεκριμένων σχολών). Ως μηχανικούς ανάπτυξης θεωρούμε τον *αναλυτή* (*analyst*), το *σχεδιαστή* (*designer*) και το *μηχανικό ελέγχου* (*test engineer*), ρόλους που γνωρίζετε από την Τεχνολογία Λογισμικού.

Ο αναλυτής είναι υπεύθυνος για την επαφή με τον πελάτη και τον καθορισμό των αρχικών προδιαγραφών, ο σχεδιαστής για τον καθορισμό της αρχιτεκτονικής του συστήματος, την τμηματοποίηση του συστήματος και τις ενδοσυσχετίσεις ανάμεσα στα τμήματα και ο μηχανικός ελέγχου έχει την ευθύνη του τελικού ελέγχου ως λευκό κουτί, δηλαδή τον έλεγχο των δομών και λειτουργιών του προγράμματος. Περισσότερα για το θέμα του ελέγχου ως λευκό κουτί μπορείτε να βρείτε σε κάποιο βιβλίο σχετικό με Τεχνολογία Λογισμικού

από τα προτεινόμενα στη βιβλιογραφία. Όλα αυτά τα μέλη του προσωπικού είναι μέλη με σπουδές στην ανάπτυξη λογισμικού (συνήθως έχουν και τον τίτλο του «μηχανικού λογισμικού») και με την εμπειρία τους στην ανάπτυξη θα πρέπει να είναι σε θέση να εξελιχθούν σε ηγέτες ομάδων, αλλά και να συνεισφέρουν με ιδέες και προτάσεις τόσο για νέα έργα όσο και για τη βελτίωση και εξέλιξη της διαδικασίας ανάπτυξης.

1.3.5 Προγραμματιστής

Όπως βλέπετε και στον ορισμό ο *προγραμματιστής (programmer)* που ακολουθεί, είναι αυτός που θα αναλάβει τη βασική εργασία της ανάπτυξης λογισμικού, δηλαδή τη συγγραφή του κώδικα. Τη δραστηριότητα αυτή θα την ακούσετε ως προγραμματισμό (programming), συγγραφή κώδικα (code authoring), κωδικοποίηση (coding), κτλ αν και έχει επικρατήσει η λέξη προγραμματισμός.

Ο **προγραμματιστής** (programmer) είναι αυτός που γνωρίζει μία γλώσσα προγραμματισμού και του δίνονται προδιαγραφές ώστε να υλοποιήσει ένα τμήμα λογισμικού σε αυτή τη γλώσσα.

Πολλοί προγραμματιστές που έχουν σπουδές στην ανάπτυξη λογισμικού και γνώσεις πολλών γλωσσών προγραμματισμού συνήθως καλούνται *ανώτεροι προγραμματιστές (senior programmers)* και πέρα από τον προγραμματισμό έχουν ως ρόλο και την καθοδήγηση των νέων προγραμματιστών στις διαδικασίες και μεθόδους ανάπτυξης. Συχνά άνθρωποι που έχουν σπουδάσει ως μηχανικοί λογισμικού ξεκινούν την εργασία τους σε κάποια επιχείρηση ή οργανισμό ως ανώτεροι προγραμματιστές και σύντομα εξελίσσονται στο ρόλο του μηχανικού ανάπτυξης.

1.3.6 Τεχνικοί και υπόλοιπο προσωπικό

Υπόλοιπο προσωπικό μιας επιχείρησης ή οργανισμού είναι τεχνικό προσωπικό που έχει την ευθύνη θεμάτων όπως έλεγχος μονάδων –συνήθως έλεγχος ως μαύρο κουτί– τεχνικές εγκαταστάσεις, συσκευασίες, υποστήριξη, κτλ. Περισσότερα για το θέμα του ελέγχου ως μαύρο κουτί δείτε σε οποιοδήποτε βιβλίο σχετικό με Τεχνολογία Λογισμικού από τα προτεινόμενα στη βιβλιογραφία. Επίσης μία επιχείρηση ή οργανισμός έχει *διοικητικό προσωπικό, τμήμα προσωπικού* με τους ανθρώπους που είναι υπεύθυνοι για τη μισθοδοσία και τη στελέχωση της επιχείρησης, *τμήμα πωλήσεων και προώθησης προϊόντων* (marketing), *γραμματειακό προσωπικό*, κτλ. Όλοι αυτοί οι άνθρωποι είναι μέλη του προσωπικού μιας επιχείρησης και συχνά έχουν εξίσου σημαντική συμβολή στην επιτυχία (ή αποτυχία) ενός έργου λογισμικού όσο και αυτοί που είναι επιφορτισμένοι με αυτές καθαυτές τις διαδικασίες ανάπτυξης.

1.3.7 Πελάτες

Ο *πελάτης* (customer), που συχνά αναφέρεται και ως *χρήστης* (user) του λογισμικού, δεν είναι μέλος του προσωπικού της επιχείρησης ή οργανισμού. Είναι όμως αναπόσπαστο μέλος της διαδικασίας ανάπτυξης λογισμικού και του προγράμματος ποιότητας για το οποίο θα μιλήσουμε σε επόμενα κεφάλαια του βιβλίου. Όπως θα διδαχθείτε σε αυτό το βιβλίο, ένα από τα σημαντικότερα λάθη της ανάπτυξης λογισμικού είναι να εξαιρέσει τον πελάτη από τη διαδικασία ανάπτυξης και το πρόγραμμα ποιότητας.

Υπάρχουν πολλοί διαφορετικοί ρόλοι του πελάτη όπως: α) ο πελάτης – χρήστης, που θα αγοράσει λογισμικό γενικής χρήσης που έχει ετοιμασθεί βασισμένο σε γενικές προδιαγραφές, β) ο πελάτης – εξειδικευμένος χρήστης, που χρησιμοποιεί λογισμικό που έχει κατα-

σκευαστεί εξειδικευμένα για τις δικές του ανάγκες, γ) ο πελάτης – οριστής, που ορίζει τις προδιαγραφές λειτουργικότητας του λογισμικού και δ) ο πελάτης – χρηματοδότης, που επενδύει στην ανάπτυξη του λογισμικού. Πολλές φορές όλους αυτούς τους εμπλεκόμενους στην ανάπτυξη λογισμικού θα τους αποκαλούμε πελάτες, εκτός αν οι συνθήκες μας επιβάλλουν να το διαχωρίσουμε.

Ας δούμε το εξής παράδειγμα: Έστω ότι ένα Πανεπιστήμιο θέλει να αναπτύξει ένα κόμβο ενημέρωσης των ενδιαφερομένων στο διαδίκτυο. Απευθύνεται στο Υπουργείο Εθνικής Παιδείας και Θρησκευμάτων (πιθανότατα υποβάλλοντας πρόταση για χρηματοδότηση) και αφού πάρει την έγκριση αναθέτει (μετά από διαγωνισμό) σε μια εταιρία την ανάπτυξη. Η εταιρία αυτή θα έχει ως πελάτη – χρηματοδότη το Υπουργείο, ως πελάτη – οριστή το Πανεπιστήμιο, ενώ πελάτες – χρήστες θα είναι όλοι οι ενδιαφερόμενοι που θα επισκέπτονται τον κόμβο.

1.3.8 Άλλες κατηγορίες

Πρέπει να τονιστεί ότι εκτός από τις κατηγορίες που αναφέραμε στην ενότητα 1.3 υπάρχουν και πολλές άλλες κατηγορίες εμπλεκόμενων στην ανάπτυξη λογισμικού που πηγάζουν από τις πιο σύνθετες ανάγκες των επιχειρήσεων που παράγουν λογισμικό. Αν αναζητήσετε τις αγγελίες που ζητούν προσωπικό δεν θα δείτε «*ζητείται προγραμματιστής*», αλλά κάτι σαν «*ζητείται στέλεχος εταιρίας πληροφορικής με γνώσεις Java και εμπειρία σε πρωτόκολλα TCP-IP*». Στην πραγματικότητα και η δεύτερη αγγελία ζητάει προγραμματιστή με εξειδικευμένες γνώσεις για να καλύψει τις ανάγκες κάποιου συγκεκριμένου έργου. Γενικά όμως οι βασικές κατηγορίες είναι αυτές που αναφέραμε.

1.4 Εργασία σε Ομάδες

Στην προηγούμενη ενότητα μιλήσαμε αρκετά για την εργασία σε ομάδες και για τον ηγέτη της ομάδας. Σε αυτή την ενότητα θα μιλήσουμε για τους τρόπους που μπορούν να οργανωθούν οι ομάδες που μετέχουν στην ανάπτυξη λογισμικού και για τα πλεονεκτήματα της ομαδικής εργασίας στην ανάπτυξη λογισμικού και θα παρουσιάσουμε το οργανόγραμμα ανάπτυξης.

1.4.1 Τρόποι οργάνωσης ομάδων

Ο Mantei [Man81] αναφέρει ότι η σύνθεση και ο τρόπος οργάνωσης των ομάδων σχετίζονται με τη δυσκολία και το μέγεθος του προγράμματος, τη διάρκεια ζωής της ομάδας, την ευκολία τμηματοποίησης του έργου, το βαθμό επικοινωνίας που απαιτείται και μία σειρά από άλλους –λιγότερο σημαντικούς– παράγοντες που επηρεάζουν τον τρόπο που θα πρέπει να οργανωθεί μία ομάδα ανάπτυξης. Προτείνει τρεις τρόπους οργάνωσης μίας ομάδας ανάλογα με τους παράγοντες του έργου:

- *δημοκρατικά οργανωμένη,*
- *αποκεντρωμένα διοικούμενη και*
- *κεντρικά διοικούμενη.*

Στις *δημοκρατικά οργανωμένες ομάδες* δεν υπάρχει ηγέτης ομάδας, αν και για κάποιο πρόβλημα μπορεί κάποιος να αναλάβει το ρόλο του ηγέτη, αλλά σύντομα θα αφήσει σε κάποιον άλλο το ρόλο του ηγέτη για το επόμενο πρόβλημα που με τη σειρά του θα τον αφήσει σε κάποιον άλλο. Έτσι όλα τα μέλη της ομάδας θα έχουν την ευκαιρία να την ηγηθούν κατά διαστήματα. Όλα τα προβλήματα θα συζητούνται από κοινού και αυτός που έχει το ρόλο του ηγέτη απλά θα συντονίζει.

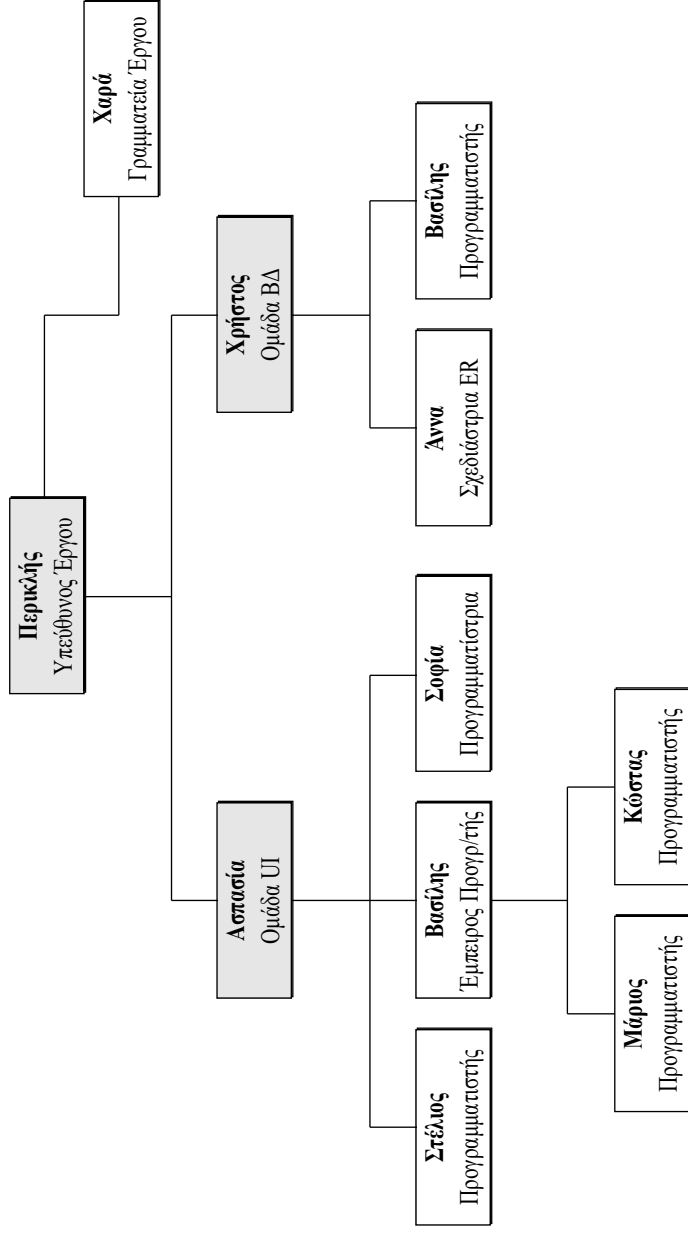
Στις *αποκεντρωμένα διοικούμενες ομάδες* υπάρχει κάποιος μόνιμος ηγέτης, αλλά τα προβλήματα πάλι συζητούνται από κοινού και η επικοινωνία ανάμεσα στα μέλη της ομάδας είναι οριζόντια.

Στις *κεντρικά διοικούμενες ομάδες* υπάρχει μόνιμος ηγέτης και υπάρχουν επιμέρους καθορισμένες εξουσίες. Η διοίκηση και η επικοινωνία γίνεται κάθετα (δηλαδή ο ηγέτης επικοινωνεί με τους άμεσα υφισταμένους του, αυτοί με τους επόμενους στην ιεραρχία, κτλ.

Η επιλογή του τρόπου οργάνωσης εξαρτάται από το είδος του προβλήματος. Τα μεγάλα προβλήματα συνήθως απαιτούν κεντρικά διοικούμενες ομάδες, ενώ τα μικρά δημοκρατικά διοικούμενες, τα προβλήματα που χρειάζονται πολύ χρόνο συνήθως επιλύονται καλύτερα από δημοκρατικά διοικούμενες ομάδες (αφού είναι η καλύτερη οργάνωση σε σχέση με το ηθικό μιας ομάδας που θα εργαστεί για πολύ καιρό μαζί) και τα προβλήματα που διασπώνται εύκολα σε τμήματα είναι καλύτερα να επιλύονται από κεντρικά διοικούμενες ομάδες. Φυσικά ο υπεύθυνος έργου πρέπει να εκτιμήσει όλους τους παράγοντες του έργου και τις ιδιαιτερότητες του προσωπικού πριν αποφασίσει για τον τρόπο οργάνωσης της ομάδας.

1.4.2 Πλεονεκτήματα εργασίας σε ομάδες

Η εργασία σε ομάδες πρώτα από όλα βοηθά στην διευκόλυνση των θεμάτων επικοινωνίας και συντονισμού των συμμετεχόντων στην παραγωγή λογισμικού. Πέρα από τα θέματα επικοινωνίας και συντονισμού βοηθά στο να δουλεύουν τα μέλη της ομάδας κοντά το ένα με το άλλο, αναπτύσσοντας μία κοινή φιλοσοφία ανάπτυξης, να γνωρίζουν ο ένας τη δουλειά του άλλου. Όπως θα δούμε και σε επόμενα κεφάλαια βοηθά επίσης στην εφαρμογή διαδικασιών και προτύπων ποιότητας, αλλά –κυρίως– βοηθά σε αυτό που ο Sommerville [Som89] ονομάζει «μη εγωιστικός προγραμματισμός (*egoless*



Σχήμα 1.10

Παράδειγμα οργανογράμματος έργου

programming)», δηλαδή ο προγραμματιστής να μη θεωρεί τον κώδικά του ως ιδιοκτησία του (δημιουργία του), αλλά ως ιδιοκτησία (δημιουργία) της ομάδας στην οποία έχει ενταχθεί.

1.4.3 Οργανόγραμμα ανάπτυξης λογισμικού

Οι περισσότεροι από εσάς έχετε διδαχθεί ή τουλάχιστον γνωρίζετε το οργανόγραμμα ως το μέσο με το οποίο παρουσιάζονται με σχηματικό τρόπο οι σχέσεις και οι εξαρτήσεις (προϊστάμενοι – υφιστάμενοι) των μελών μίας επιχείρησης. Για μεγάλα έργα ανάπτυξης λογισμικού συνήθως γίνεται και ένα αντίστοιχο *οργανόγραμμα διαχείρισης έργου* (project management structure) που παρουσιάζει τις ομάδες ανάπτυξης και τους ρόλους καθενός που συμμετέχει στην ανάπτυξη λογισμικού.

Το **οργανόγραμμα** διαχείρισης έργου (project management structure) παρουσιάζει σχηματικά τη διοικητική δομή του έργου.

Έτσι ο υπεύθυνος έργου έχει καλύτερη αντίληψη για τους συμμετέχοντες στην ανάπτυξη λογισμικού και τις ευθύνες που έχει αναλάβει ο καθένας από αυτούς. Στο σχήμα 1.10 παρουσιάζεται ένα παράδειγμα οργανογράμματος ενός μικρού έργου. (Το έργο που παρουσιάζεται στο σχήμα 1.10 είναι πολύ μικρό για να γίνει σε πραγματικές συνθήκες οργανόγραμμα, αλλά για λόγους χώρου στο βιβλίο δεν θεωρήσαμε ανάγκη να βάλουμε ένα τεράστιο ρεαλιστικό οργανόγραμμα).

Στο σχήμα 1.10 βλέπουμε ότι ο Περικλής είναι υπεύθυνος ενός έργου με γραμματεία έργου τη Χαρά. Στο έργο υπάρχουν δύο ομάδες που λειτουργούν ως κεντρικά διοικούμενες. Στη μία ομάδα ηγέτης είναι η Ασπασία και στην άλλη ο Χρήστος. Βλέπουμε επίσης τα υπόλοιπα μέλη που εργάζονται στην ανάπτυξη του έργου ποιον έχουν άμεσα προϊστάμενο ο καθένας.

1.5 Σύνοψη Κεφαλαίου

Στο κεφάλαιο μιλήσαμε για τη διαχείριση της ανάπτυξης λογισμικού. Θα συνεχίσουμε και στο επόμενο κεφάλαιο με την εκτίμηση και την ανάλυση ρίσκου, αλλά μέχρι τώρα μιλήσαμε για τις περισσότερες διαδικασίες που αφορούν τη διαχείριση έργων λογισμικού. Πρέπει να γνωρίζετε τις βασικές έννοιες και ορισμούς που σχετίζονται με τη διαχείριση έργων λογισμικού και τις ιδιαιτερότητες του λογισμικού και τα προβλήματα διαχείρισης που εντάσσονται στη λεγόμενη κρίση του λογισμικού.

Δώσαμε ιδιαίτερη έμφαση στις βασικές δραστηριότητες που σχετίζονται με τη διαχείριση έργων λογισμικού και παρουσιάσαμε μερικές από τις πιο διαδεδομένες τεχνικές που χρησιμοποιούν οι υπεύθυνοι έργων. Θα πρέπει να μπορείτε να εφαρμόσετε αυτές τις τεχνικές σε κάποιο παράδειγμα έργου λογισμικού και να λύσετε εργασίες αντίστοιχες με τα παραδείγματα που συζητάμε.

Τέλος μιλήσαμε για τους συμμετέχοντες στην ανάπτυξη λογισμικού το ρόλο τους και συνοπτικά για τα προσόντα που θα πρέπει να έχει ο καθένας από αυτούς, για ομάδες εργασίας και οργανόγραμμα διαχείρισης έργου. Εάν δυσκολευτήκατε στην ενότητα 1.3 του βιβλίου θα πρέπει να κάνετε μία καλή επανάληψη σε θέματα Τεχνολογίας Λογισμικού. Σε αυτό μπορεί να σας βοηθήσει και η βιβλιογραφία που ακολουθεί στο τέλος του βιβλίου. Γενικά η συμβουλή μας είναι να ανατρέχετε και σε βιβλιογραφικές πηγές πέρα από το βιβλίο αυτό, ώστε να εμπλουτίζετε τις γνώσεις σας, αλλά και να διαβάζετε με εναλλακτικό τρόπο την ύλη που καλύψαμε. Στο τέλος του βιβλίου υπάρχει βιβλιογραφία με αρκετά προτεινόμενα βιβλία για περαιτέρω μελέτη στα οποία σας υποδεικνύουμε που να επικεντρώσετε την προσοχή σας.

Ασκήσεις

Ακολουθούν μερικές προτάσεις. Προσπαθήστε να απαντήσετε ποιες από αυτές είναι σωστές και ποιες λάθος;

1. Το λογισμικό δεν κατασκευάζεται ούτε αναπτύσσεται, αλλά διαχειρίζεται.
2. Ο υπεύθυνος έργου είναι αυτός που έχει την ευθύνη για την πορεία του έργου.
3. Ο υπεύθυνος διαχείρισης ενός έργου λογισμικού είναι πάντα μέλος της διοίκησης του οργανισμού.
4. Στα έργα ανάπτυξης λογισμικού δεν υπάρχουν ποτέ ιστορικά δεδομένα.
5. Η διαχείριση των έργων λογισμικού γίνεται με αδιαφανείς διαδικασίες.
6. Η αρχική τμηματοποίηση του έργου είναι βασική διαδικασία σχεδίασης.
7. Χρονοπρογραμματισμός είναι η διαδικασία κατά την οποία γίνεται εκτίμηση του χρόνου που θα χρειαστεί κάθε τμήμα του έργου.
8. Ορόσημα ορίζονται τα σημεία στα οποία θα πρέπει να γραφεί μία έκθεση προόδου.
9. Η επίβλεψη έργου γίνεται από τον υπεύθυνο έργου.
10. Το ενδοδίκτυο προσφέρεται για χρήση σε τυπική επικοινωνία του υπευθύνου έργου με την ομάδα ανάπτυξης έργου.
11. Η τεκμηρίωση του έργου δεν είναι τόσο σημαντική για έργα ανάπτυξης λογισμικού, λόγω του κυρίαρχου ρόλου του λογισμικού.

12. Προσωπικό μίας επιχείρησης είναι όλοι όσοι συμμετέχουν στην ανάπτυξη λογισμικού.
13. Ο ανώτερος υπεύθυνος έργων συμμετέχει και στη διοίκηση του οργανισμού ή της επιχείρησης.
14. Η δημιουργία ομάδων στα πλαίσια ενός έργου δημιουργεί την ανάγκη κάποιοι άνθρωποι να ηγηθούν αυτών των ομάδων, ώστε να είναι πιο εύκολο το έργο του υπεύθυνου έργου.
15. Ο προγραμματιστής αναλαμβάνει και τη σχεδίαση του λογισμικού.
16. Ο πελάτης, αν και δεν ανήκει στο προσωπικό της επιχείρησης, πρέπει να εντάσσεται στη διαδικασία ανάπτυξης λογισμικού.
17. Οι ομάδες ανάπτυξης λογισμικού πρέπει να οργανώνονται δημοκρατικά.
18. Για προβλήματα που απαιτούν επικοινωνία ανάμεσα σε όλα τα μέλη της ομάδας είναι καλύτερη η δημοκρατική οργάνωση των ομάδων.
19. Οι κεντρικά διοικούμενες ομάδες αποδίδουν καλύτερα για έργα που έχουν μικρή διάρκεια.
20. Ο μη εγωιστικός προγραμματισμός υποβοηθείται στα πλαίσια μιας ομάδας.