Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Η/Υ

Εθνικό Μετσόβιο Πολυτεχνείο

Τεχνολογία Λογισμικού

7ο Εξάμηνο 2019 - 20

Ν.Παπασπύρου, Καθ. ΣΗΜΜΥ, nickie@softlab.ntua.gr

Β.Βεσκούκης, Αν.Καθ. ΣΑΤΜ, v.vescoukis@cs.ntua.gr

Κ.Σαΐδης, ΠΔ 407, saiko@softlab.ntua.gr

Διαχείριση έργων λογισμικού (1)

Περιεχόμενα

- Διαχείριση έργων (γενικά)
- Διαχείριση έργων λογισμικού (εισαγωγή)

Διαχείριση έργου (γενικά)

- Ορισμοί
- Βασικές έννοιες
- Βασικά εργαλεία

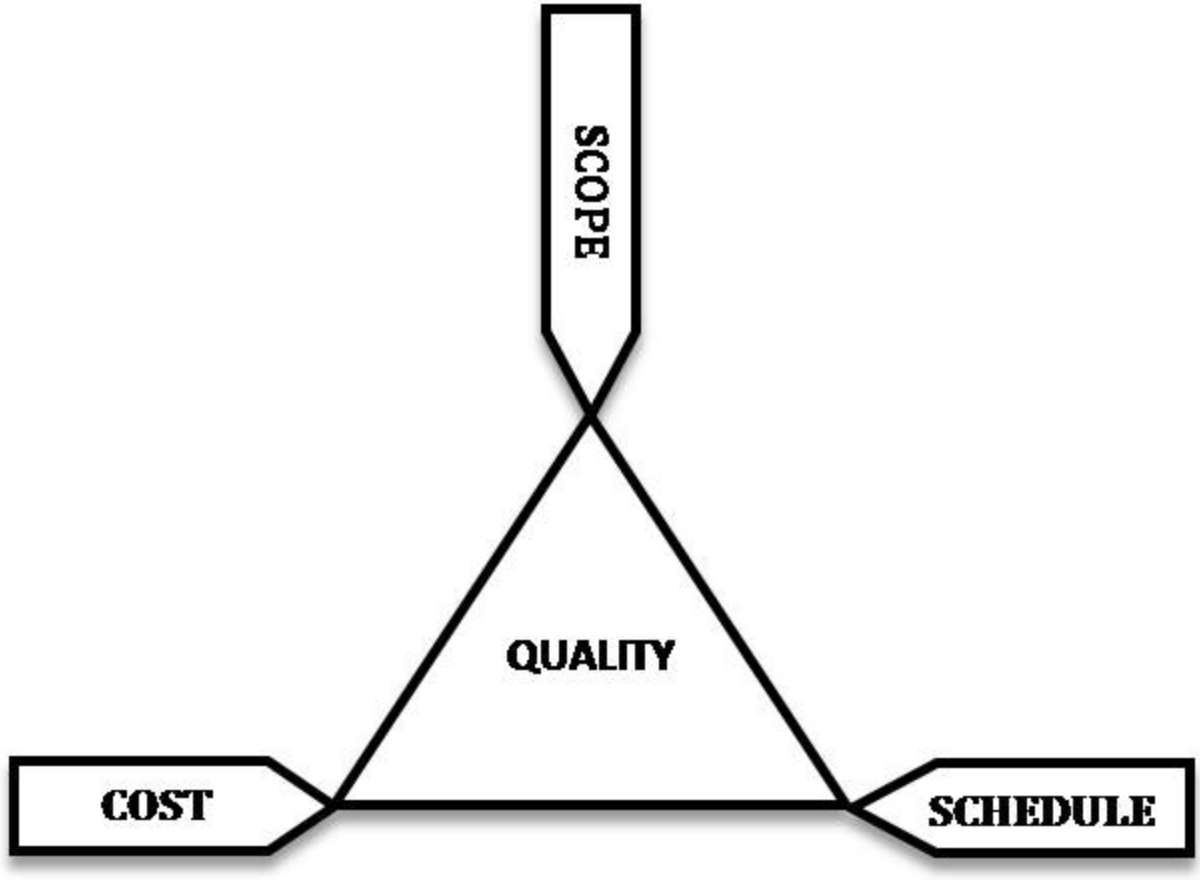
Έργο (project)

Ένα σύνολο ξεχωριστών, σύνθετων και συνδεδεμένων ενεργειών που έχουν ένα στόχο και πρέπει να ολοκληρωθούν εντός συγκεκριμένου χρονικού διαστήματος και προϋπολογισμού με βάση συγκεκριμένες προδιαγραφές.

Περιορισμοί

Σε τέσσερις άξονες:

- Κόστος (cost)
- Χρόνος (schedule)
- Έκταση ζητουμένων (scope)
- Ποιότητα αποτελέσματος (quality)



By I, John Manuel Kennedy T., CC BY-SA 3.0,
https://commons.wikimedia.org/w/index.php?curid=4282986

Στην πράξη

- Συνήθως μπορούμε να ελέγξουμε και να διαχειριστούμε δύο από τους άξονες αυτούς.
- Πρέπει ο πελάτης να αποφασίσει σε ποιους άξονες υπάρχουν περιθώρια ευελιξίας.

Για παράδειγμα

- Μικρότερος χρόνος + Υψηλότερη ποιότητα = Ακριβότερο
- Μικρότερος χρόνος + Φθηνότερο = Χαμηλότερη ποιότητα
- Υψηλότερη ποιότητα + Φθηνότερο = Μεγαλύτερος χρόνος

Διαφορετικές προσεγγίσεις διαχείρισης

- Παραδοσιακή (waterfall)
- Αυξητική (incremental)
- Επαναληπτική (iterative)
- Προσαρμοστική (adaptive)
- κ.ά.

Αντίστοιχες μεθοδολογίες με τα έργα λογισμικού

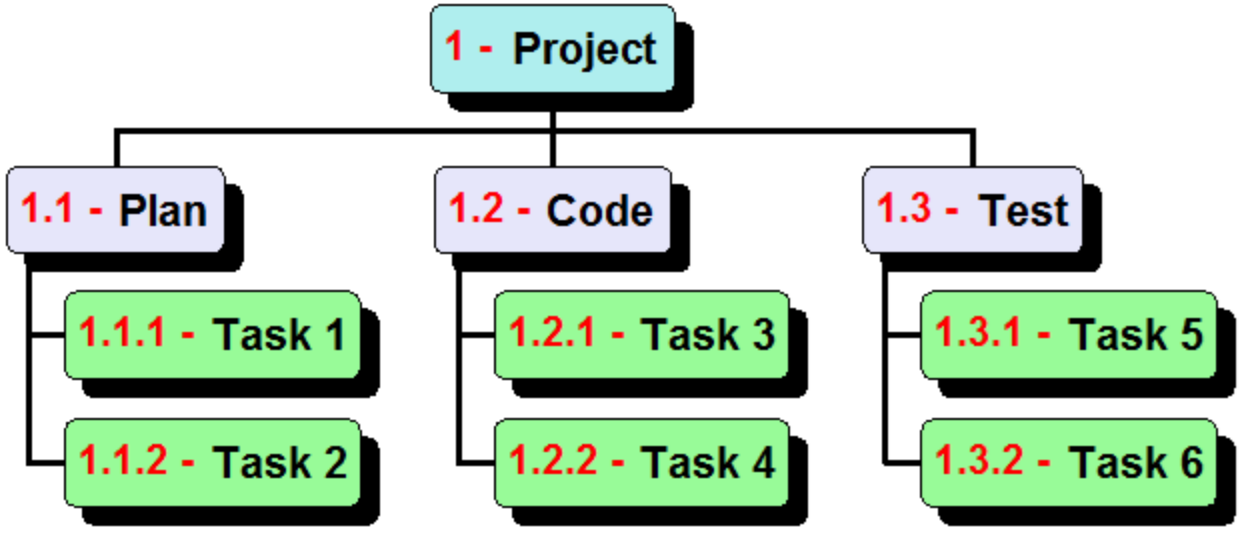
Σχέδιο έργου (Project Plan)

- Διαχωρισμός του έργου σε υποέργα με ενδιάμεσα παραδοτέα (interim deliverables)
- Εκτίμηση προσπάθειας, κόστους και χρόνου
- Χρονοπρογραμματισμός και χρονικά ορόσημα (project milestones)

Ιεραρχική κατανομή εργασιών (Work Breakdown Structure)

- Ιεραρχική περιγραφή όλης της "δουλειάς" που πρέπει να γίνει στο πλαίσιο του έργου για να ικανοποιηθούν οι απαιτήσεις του πελάτη.
  - Πακέτα εργασίας - υποέργα
  - Επιμέρους δράσεις και ενέργειες ανά πακέτο
  - Αλληλοεξαρτήσεις μεταξύ τους
- Δεν πρέπει να είναι ούτε πολύ λεπτομερής, ούτε πολύ περιληπτική (κάπου στη μέση).

WBS



criticaltools.com

Χρονοδιάγραμμα (Timeline)

- Χρονο-προγραμματισμός των εργασιών
- Με βάση το WBS
- Είτε με απόλυτες ημερομηνίες
  - 1ο παραδοτέο στις HH/MM/EEEE
- Είτε με σχετικές ημερομηνίες
  - 1ο παραδοτέο στο δεύτερο μήνα (M2) απο την έναρξη του έργου

Χρήσιμες έννοιες

Κρίσιμο μονοπάτι (critical path)

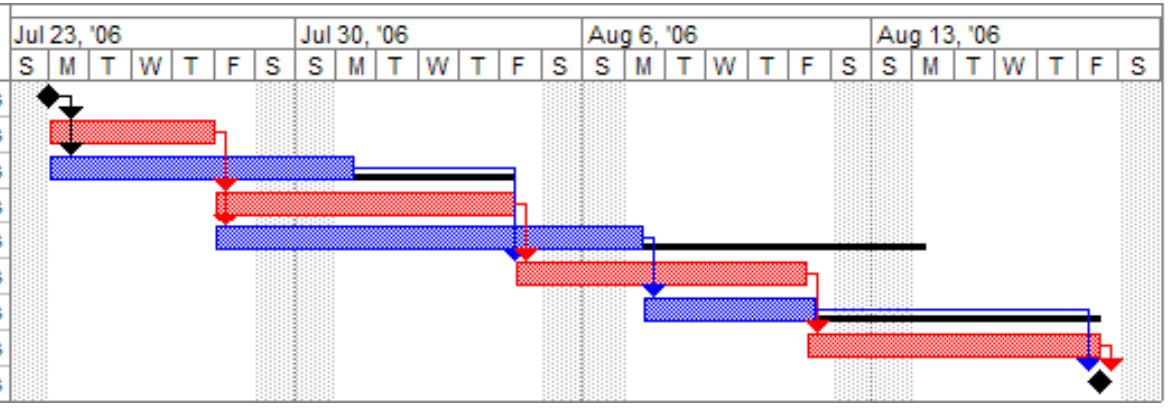
- Η ακουλουθία των ενεργειών που αν αποτύχουν θα αποτύχει το έργο.

Χαλαρό περιθώριο (slack, float)

- Το χρονικό διάστημα που μπορεί να καθυστερήσει μια ενέργεια χωρίς να καθυστερήσει κάποια επόμενή της ή το το έργο συνολικά.

Διάγραμμα Gantt

ID	Task Name	Predecessors	Duration
1	Start		0 days
2	a	1	4 days
3	b	1	5.33 days
4	c	2	5.17 days
5	d	2	6.33 days
6	e	3,4	5.17 days
7	f	5	4.5 days
8	g	6	5.17 days
9	Finish	7,8	0 days



Εξαρτήσεις μεταξύ των επιμέρους εργασιών

- FS: Όταν ολοκληρωθεί η εργασία Α, μπορεί να ξεκινήσει η Β
- FF: Όταν ολοκληρωθεί η Α, μπορεί να ολοκληρωθεί η Β
- SS: Όταν ξεκινήσει η Α, μπορεί να ξεκινήσει η Β
- SF: Όταν ξεκινήσει η Α, μπορεί να ολοκληρωθεί η Β

Χρόνος και Προσπάθεια

Κάθε εργασία στο πλαίσιο του έργου:

- έχει μια χρονική διάρκεια (ο αριθμός των ημερολογιακών ημερών που διαρκεί).
- απαιτεί μια προσπάθεια (ο αριθμός των εργατο-ωρών).

Ανθρωπο-προσπάθεια

- Ανθρωπο-ώρα (ΑΩ)
- Ανθρωπο-ημέρα (ΑΗ) = 8 ΑΩ
- Ανθρωπο-μήνας (ΑΜ) = 20/21 ΑΗ
- Ανθρωπο-έτος (ΑΕ) = 11 ΑΜ

Διαχείριση έργου λογισμικού

- Μοντέλα και μεθοδολογίες ανάπτυξης λογισμικού
- Διοίκηση ομάδας ανάπτυξης λογισμικού
- Τεχνικές και εργαλεία ελέγχου ανάπτυξης λογισμικού
- Διαχείριση συστατικών του λογισμικού

Μεγάλο μέρος του μαθήματος θα αφιερωθεί σε αυτό το θέμα!

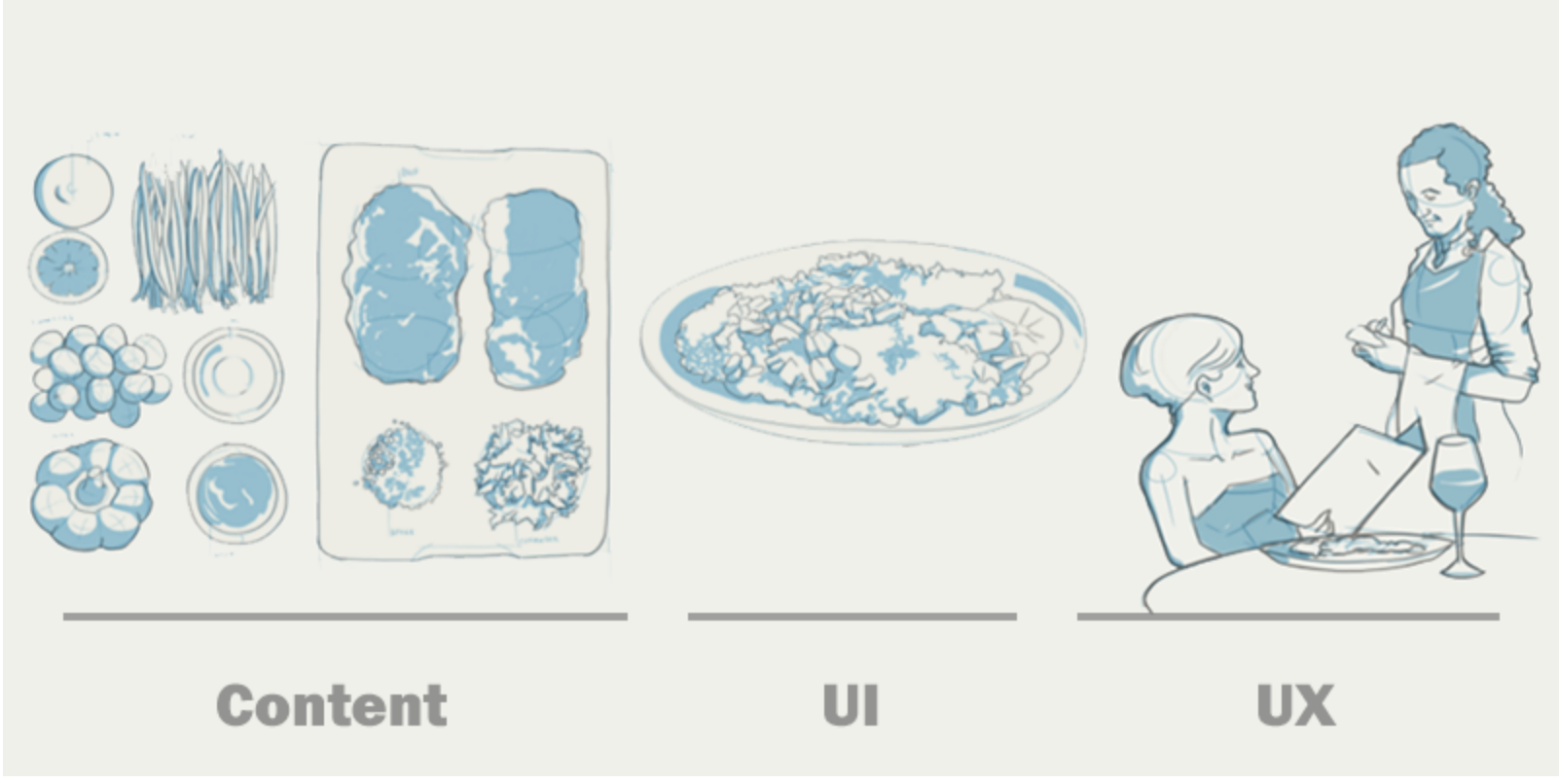
Ρόλοι σε μια ομάδα ανάπτυξης λογισμικού

- Ανάλογα με την εφαρμογή/λογισμικό
- Χρησιμοποιούμε μια εφαρμογή διαδικτύου ως παράδειγμα

Ενδεικτικοί ρόλοι

- Project/Product manager
- Architect
- Team lead
- Senior developer
- Junior developer
- Quality assurance engineer
- User interface designer
- User experience engineer

UX vs UI



Ενδεικτικοί ρόλοι

- Back-end engineer
- Front-end engineer
- Full-stack engineer
- Database developer
- Business analyst
- Data scientist
- Customer support
- DevOps engineer

Συνήθως

- Σε μικρές ομάδες, ένα πρόσωπο έχει πολλούς ρόλους
- Σε μεγάλες ομάδες, ένα πρόσωπο έχει ένα ρόλο

Διοίκηση της ομάδας

Project/Product manager

- Οριοθέτηση στόχων ομάδας και χαρακτηριστικών λογισμικού
- Διαχείριση χρόνου και προσπάθειας των μελών σε μακροσκοπικό επίπεδο

Διοίκηση της ομάδας

Architect

- Λήψη των σημαντικών σχεδιαστικών αποφάσεων (major design decisions)
- Θέσπιση των διεπαφών (interfaces), των συστατικών (components) και των εξαρτήσεων (dependencies)
- Επιλογή εργαλείων (tooling)

Διοίκηση της ομάδας

Team Lead

- Ανάθεση των επιμέρους εργασιών στα μέλη της ομάδας
- Διαχείριση χρόνου και προσπάθειας των μελών σε καθημερινό επίπεδο

Διοίκηση της ομάδας

Επικεφαλής μηχανικός

- Ανάλογα με το έργο, μπορεί να συνδυάζει και τις τρεις ιδιότητες
- Στο μάθημα δίνουμε έμφαση σε αυτόν το "ρόλο"

Το ζητούμενο

Πώς ο επικεφαλής μηχανικός

- Ο οποίος συμμετέχει σε μια ομάδα ανάπτυξης
- Διαχειρίζεται τεχνικά τη διαδικασία ανάπτυξης
- Διασφαλίζοντας την ποιότητα του τελικού αποτελέσματος

Ανεξάρτητα από

- Τους ρόλους των ανθρώπων στην ομάδα
- Τη μεθοδολογία ανάπτυξης
- Την αρχιτεκτονική του υπό ανάπτυξη λογισμικού

Τεχνική διαχείριση έργου λογισμικού (γενικά)

- Διαχείριση εκδόσεων κώδικα (version control)
- Αυτόματο "χτίσιμο" λογισμικού (build automation)
- Στατική ανάλυση κώδικα και αυτόματος εντοπισμός σφαλμάτων (bug detection)
- Εκτέλεση σεναρίων ελέγχου (tests)
- Συνεχής ολοκλήρωση (continuous integration)
- Διαχείριση συστατικών λογισμικού και των εκδόσεών τους

Συναφή με το λεγόμενο "software configuration management"

Καθημερινότητα ομάδας ανάπτυξης

- Λήψη και ενσωμάτωση των αλλαγών που έγιναν στον κώδικα από τα υπόλοιπα μέλη της ομάδας (Version control tool)
- Επανάληψη:
  - Ανάπτυξη νέου κώδικα / διόρθωση υπάρχοντος (Editor, IDE)
  - Μεταγλώττιση κώδικα, "πακετάρισμα" κι εκτέλεση τοπικών σεναρίων ελέγχου (Build automation tool)
- Διαμοιρασμός των αλλαγών / προσθηκών που κάναμε στον κώδικα στα υπόλοιπα μέλη της ομάδας (Version control tool)

Επίσης

- Συνεχής ολοκλήρωση (CI): ένας server που εκτελεί αυτόματα και μηχανικά όλα τα σενάρια ελέγχου και τις δοκιμές μετά από κάθε αλλαγή στον πηγαίο κώδικα (commit)
- Εγκατάσταση λογισμικού σε ενδιάμεσο περιβάλλον (Staging deployment)
- Εγκατάσταση λογισμικού στο παραγωγικό περιβάλλον (Production deployment)
- Δημοσίευση του λογισμικού σε κάποια αποθήκη (artifact repository)

Πριν ξεκινήσει το (όποιο) έργο

- Σύναψη κοινής "τεχνικής κουλτούρας"
- Κοινές συμβάσεις (conventions)
  - Για τον κώδικα, τη μορφοποίηση, την ομματοδοσία, την τοποθεσία, κλπ.
- Κοινά εργαλεία (tools) & διαδικασίες (procedures)
- Κοινά αποδεκτός και σαφώς ορισμένος ο επιμερισμός ευθύνης (ownership)

Επιμερισμός ευθύνης (ownership)

- Ποιος είναι αρμόδιος για ποιο κομμάτι κώδικα / λειτουργικότητας / χαρακτηριστικών ή για ποιο βήμα;
- Αποφυγή "ορφανού" κώδικα / βήματος (κανείς δεν ξέρει πως δουλεύει το component Χ ή πώς να εκτελέσει το βήμα Χ)
- Αποφυγή "αποκλεισμένου" κώδικα / βήματος (μόνο ο Χ ξέρει το Ψ)
- Τουλάχιστον δύο πρόσωπα με την ίδια αρμοδιότητα σε κάποιο κομμάτι

Bus factor

Κι αν τους χτυπήσει λεωφορείο;

> The "bus factor" is the minimum number of team members that have to suddenly disappear from a project before the project stalls due to lack of knowledgeable or competent personnel.

Wikipedia

Συμβάσεις (conventions)

Παραδείγματα

Συμβάσεις για τη δομή του κώδικα

- Τρόπος συγγραφής κώδικα
  - Ονόματα κλάσεων, μεθόδων, μεταβλητών
  - Στοίχιση κώδικα
- Παράδειγμα: Java coding conventions
  - `NameOfClass`
  - `nameOfMethod`, `nameOfVariable` και `nameOfField`
  - `NAME_OF_CONSTANT`

Συμβάσεις για τη δομή των αρχείων

- Διάρθρωση και τοποθέτηση των αρχείων
  - Source files, resources, configuration files, libraries, κτλ.
- Στο JVM τα Maven directory structure conventions αποτελούν το de facto standard
  - `src/main/java`
  - `src/main/resources`
  - `src/test/java`
  - `src/test/resources`
  - `build/classes`

Java Packages

```
package x.y.z;
class Foo {
  ...
}
```

Πηγαίος κώδικας

`src/main/java/x/y/z/Foo.java`

Class files

`build/classes/x/y/z/Foo.class`

Test Packages

```
package x.y.z;
class FooTest {
  //the tests of class Foo
  ...
}
```

Πηγαίος κώδικας

`src/test/java/x/y/z/Foo.java`

Class files

`build/classes/x/y/z/Foo.class`

Εργαλεία

- Version control
- Integrated Development Environments (IDEs)
- Building
- Testing

- Bug detection
- Continuous integration

- Continuous deployment and delivery (DevOps)
- Containers - monitoring - alerting

Version control

Ζητούμενα

- Διαχείριση του source code base (της "βάσης" του κώδικα)
- Καταγραφή των αλλαγών (ιστορικό, ποιος έκανε τι και πότε)
- Συνεργατική ανάπτυξη
- Τήρηση πολλών παράλληλων καταστάσεων του κώδικα ταυτόχρονα
- Ανάκτηση συγκεκριμένης παρελθούσας κατάστασης
- κ.ά.

Στο μάθημα

Εργαλείο git (φροντιστήριο σε λίγο)