Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών
Τμήμα Πληροφορικής & Τηλεπικοινωνιών

Τεχνολογία Λογισμικού

8ο Εξάμηνο 2022-23

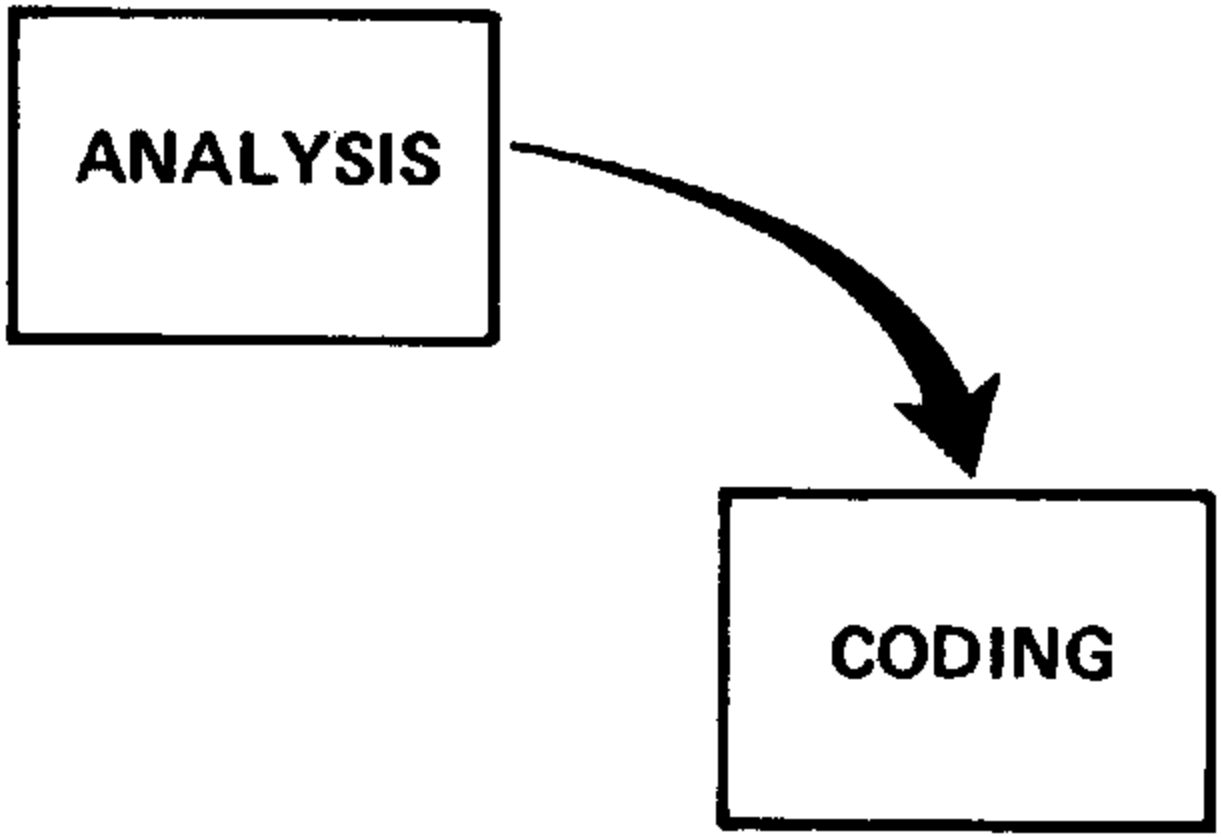
Εισαγωγή

Δρ. Κώστας Σαΐδης (saiko@di.uoa.gr)

Στόχος του μαθήματος

Πραγματιστική εισαγωγή στην Τεχνολογία Λογισμικού

Διαδικασία ανάπτυξης μικρού και απλού λογισμικού



W.W. Royce, www-scf.usc.edu/~csci201/lectures/Lecture11/royce1970.pdf

Ανάπτυξη μεγάλου και σύνθετου λογισμικού;

Τεχνολογία Λογισμικού: μεθοδολογίες, τεχνολογίες, εργαλεία, τεχνικές και πρακτικές για τη βέλτιστη υλοποίηση σύνθετων έργων λογισμικού

Τεχνολογία λογισμικού

- Μεγάλα έργα
- Με πολλές και μεγάλες ομάδες
- Με πολύπλοκες απαιτήσεις
- Με οικογένειες εφαρμογών
- Με πολλές παράλληλες αλλαγές και εκδόσεις
- Που χρησιμοποιούνται για πολλά χρόνια

Κύκλος Ζωής του Λογισμικού

- Καταγραφή και ανάλυση απαιτήσεων.
- Σχεδιασμός.
- Υλοποίηση.
- Επαλήθευση και επικύρωση.
- Εγκατάσταση, έλεγχος, παραμετροποίηση και ολοκλήρωση λογισμικού στο παραγωγικό του περιβάλλον.
- Συντήρηση και επέκταση.

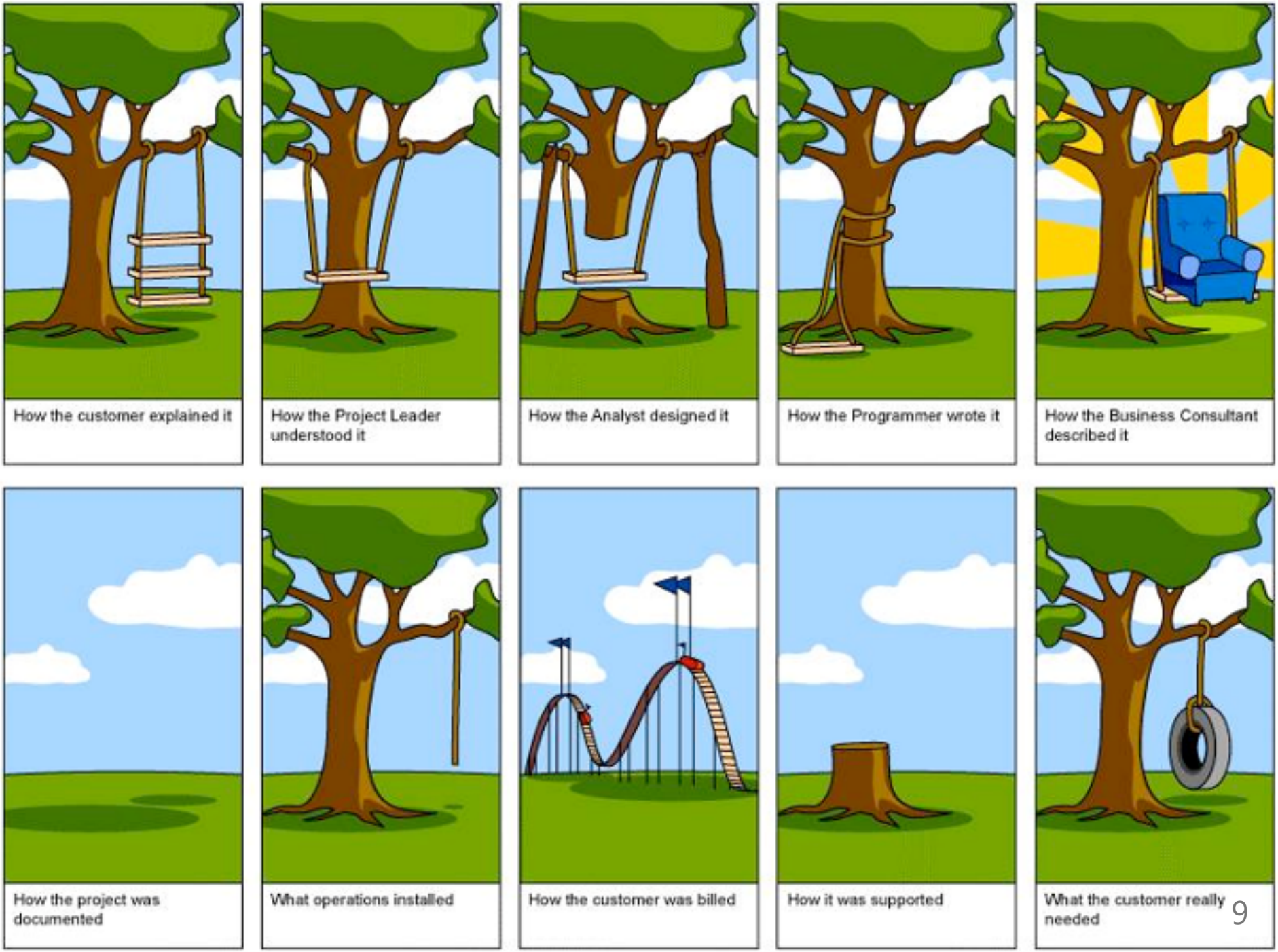
Τεχνολογία Λογισμικού

Γιατί μας απασχολεί η τεχνολογία λογισμικού;

Γιατί το λογισμικό, αν και κατέχει πλέον σημαίνοντα ρόλο σε όλες σχεδόν τις δραστηριότητες της ανθρωπότητας, είναι πολύ δύσκολο να αναπτυχθεί "καλά"!

Τι σημαίνει "να αναπτυχθεί καλά";

- Να ολοκληρωθεί εντός προϋπολογισμού
- Να ολοκληρωθεί έγκαιρα
- Να ικανοποιεί πλήρως τους χρήστες του
- Να μην έχει λάθη και προβλήματα στη λειτουργία του
- Να είναι αξιόπιστο
- Να είναι εύκολο να συντηρηθεί
- Να είναι εύκολο να επεκταθεί
- κ.ά


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

Τι το ιδιαίτερο έχει η ανάπτυξη λογισμικού;

> Most software today is very much like an Egyptian pyramid with millions of bricks piled on top of each other, with no structural integrity, but just done by brute force and thousands of slaves.”
>
> Alan Kay

Alan Kay

- "Alan Curtis Kay is an American computer scientist ... best known for his pioneering work on object-oriented programming and windowing graphical user interface design." (Wikipedia).
- Βραβείο Turing (2003).

Συμμετέχοντες ή εμπλεκόμενοι (stakeholders)

Στα μεγάλα έργα λογισμικού εμπλέκονται πολλοί συμμετέχοντες:

- Χρήστες
- Πελάτες
- Διοίκηση (ακόμα και Μέτοχοι ή Επενδυτές)
- Αναλυτές
- Προγραμματιστές
- Δοκιμαστές
- Σχεδιαστές διεπαφών
- Υπεύθυνοι έργου
- κτλ.

Επομένως

Κάθε ένας από τους συμμετέχοντες έχει διαφορετικές εμπειρίες, βιώματα, αφετηρίες, στόχους και επιδιώξεις από την υλοποίηση του έργου.

Είναι δύσκολο να επιτευχθεί ένας κοινός τόπος, σωστά ιεραρχημένος για το όφελος του έργου και, συνάμα, ικανοποιητικός για όλους τους συμμετέχοντες.

Εκτιμήσεις

> Most software estimates are performed at the beginning of the life cycle. This makes sense until we realize that estimates are obtained before the requirements are defined and thus before the problem is understood. Estimation, therefore, usually occurs at the wrong time.
>
> Robert Glass

Robert Glass

- "Robert L. (Bob) Glass (born 1932) is an American software engineer and writer, known for his works on software engineering, especially on the measuring of the quality of software design and his studies of the state of the art of software engineering research." (Wikipedia)

Robert L. Glass, "Facts and Fallacies of Software Engineering", Addison Wesley, 2002, 0-321-11742-5.

Πολυκλοκότητα

Διακρίνουμε δύο είδη πολυπλοκότητας (complexity):

- τυχαία πολυπλοκότητα (accidental): προκύπτει λόγω εξωγενών του ζητουμένου παραγόντων (επικοινωνία, νομικό πλαίσιο, κ.ά)
- ουσιώδης πολυπλοκότητα (essential): προκύπτει λόγω των εγγενών χαρακτηριστικών του ζητουμένου (το ζητούμενο, αυτό καθ' αυτό, είναι σύνθετο)

Ακόμα κι αν γίνουν σωστές εκτιμήσεις και βρεθεί κοινός τόπος, ένα έργο λογισμικού μπορεί να αποτύχει λόγω της ουδιώδους πολυπλοκότητάς του.

Πολυπλοκότητα

> The function of good software is to make the complex appear to be simple.
>
> Grady Booch

Grady Booch

"Grady Booch is an American software engineer, best known for developing the Unified Modeling Language (UML) with Ivar Jacobson and James Rumbaugh. He is recognized internationally for his innovative work in software architecture, software engineering, and collaborative development environments." (Wikipedia)

Τι το ιδιαίτερο έχει, λοιπόν, η ανάπτυξη λογισμικού;

Με τι μοιάζει και σε τι διαφέρει από τα άλλα ανθρώπινα έργα και προϊόντα;

> Programming is an unnatural act.
>
> Alan Perlis

Alan Perlis

- "Alan Jay Perlis was an American computer scientist and professor at Yale University known for his pioneering work in programming languages and the first recipient of the Turing Award [--To 1966!--]" (Wikipedia).

Alan J. Perlis, "Epigrams on Programming", ACM SIGPLAN Notices, Vol. 17, Issue 9, Sep. 1982, p. 7-13.

www.cs.yale.edu/~perlis-alan/quotes.html

Αν το σκεφτείτε

Το λογισμικό:

- δεν υπάρχει (δεν έχει φυσική υπόσταση)
- δεν τελειώνει/οριστικοποιείται ποτέ (σχεδόν 😄)
- πρέπει να λειτουργεί/διατηρείται σε πολλές παράλληλες εκδόσεις

Και να 'ταν μόνο αυτά...

Ο νόμος του Wirth

Η ταχύτητα με την οποία το λογισμικό γίνεται πιο αργό είναι μεγαλύτερη από την ταχύτητα με την οποία το υλικό γίνεται πιο γρήγορο.

> Were it not for a thousand times faster hardware, modern software would be utterly unusable.
>
> Niklaus Wirth

Niklaus Wirth

- "Niklaus Emil Wirth is a Swiss computer scientist, best known for designing several programming languages, including Pascal, and for pioneering several classic topics in software engineering." (Wikipedia)
- Βραβείο Turing (1984).

N. Wirth, "A Plea for Lean Software", IEEE Computer, Vol 28, Issue 2, Feb. 1995, p. 64-68.

Ωχ!

> The most amazing achievement of the computer software industry is its continuing cancellation of the steady and staggering gains made by the computer hardware industry.
>
> Henry Petroski

- "Henry Petroski is an American engineer specializing in failure analysis. A professor both of civil engineering and history at Duke University, he is also a prolific author." (Wikipedia).

Henry Petroski, "To Engineer Is Human: The Role of Failure in Successful Design", St. Martin's Press, 1985.

Σφαίρα από ασήμι

> So we hear desperate cries for a silver bullet, something to make software costs drop as rapidly as computer hardware costs do. But, as we look to the horizon of a decade hence, we see no silver bullet. There is no single development, in either technology or management technique, which by itself promises even one order of magnitude improvement in productivity, in reliability, in simplicity.
>
> Fred Brooks

Fred Brooks

- "Frederick Phillips Brooks, Jr. is an American computer architect, software engineer, and computer scientist, best known for managing the development of IBM's System/360 family of computers and the OS/360 software support package, then later writing candidly about the process in his seminal book The Mythical Man-Month." (Wikipedia).
- Βραβείο Turing (1999).

Fred P. Brooks, "No Silver Bullet — Essence and Accident in Software Engineering", Proceedings of the IFIP Tenth World Computing Conference, April 1987, p. 1069-1076.
http://www.cs.nott.ac.uk/~pszcah/G51ISS/Documents/NoSilverBullet.html

Ο Μυθικός ανθρωπο-μήνας

Η προσθήκη ανθρωπο-προσπάθειας σε ένα αργοπορημένο έργο λογισμικού το καθυστερεί ακόμα περισσότερο!

Fred P. Brooks, "The Mythical Man Month", Addison-Wesley, 1975, 0-201-00650-2.

Περιεχόμενα διαλέξεων

Κύκλος Ζωής του Λογισμικού

- Καταγραφή και ανάλυση απαιτήσεων.
- Σχεδιασμός.
- Υλοποίηση.
- Επαλήθευση και επικύρωση.
- Εγκατάσταση, έλεγχος, παραμετροποίηση και ολοκλήρωση λογισμικού στο παραγωγικό του περιβάλλον.
- Συντήρηση και επέκταση.

Μοντέλα ανάπτυξης λογισμικού

- Ακολουθιακή διαδικασία (sequential)
- Μοντέλο Καταρράκτη (waterfall)
- Επαναληπτική διαδικασία (iterative)
- Αυξητική διαδικασία (incremental)
- Ευέλικτη διαδικασία (agile)

Μεθοδολογίες ανάπτυξης λογισμικού

- Ταχεία Πρωτοτυποίηση (Rapid prototyping).
- «Ακραίος Προγραμματισμός» (Extreme Programming).
- Ανάπτυξη βασισμένη σε ελέγχους (Test-driven Development).
- Συνεχής παράδοση (Continuous Delivery)
- Ανάπτυξη / Λειτουργία («DevOps»).

Ανάλυση απαιτήσεων

- Ανάλυση και μοντελοποίηση απαιτήσεων
- Σύνταξη προδιαγραφών και παραδοτέων

Σχεδιασμός Λογισμικού

- Βασικές αρχές
- Αντικειμονοστρεφής σχεδιασμός συστημάτων
- Συστατικά λογισμικού

Γλώσσα μοντελοποίησης UML

- Διαγράμματα κλάσεων (Class diagrams)
- Διαγράμματα ακολουθιών (Sequence diagrams)
- Διαγράμματα δραστηριοτήτων (Activity diagrams)
- Διαγράμματα σεναρίων χρήσης (Use-case diagrams)
- κτλ.

Πρότυπα Σχεδίασης (Design Patterns)

- Κατασκευαστικά πρότυπα (Creational patterns)
- Δομικά πρότυπα (Structural patterns)
- Πρότυπα συμπεριφοράς (Behavioral patterns)

Αρχιτεκτονική Λογισμικού

- Αρχιτεκτονικός σχεδιασμός σύνθετων κατανεμημένων συστημάτων
  - Βασικές έννοιες
  - Αρχιτεκτονικά πρότυπα (Architectural patterns)
  - Έμφαση σε εφαρμογές διαδικτύου
  - Αρχιτεκτονική REST (Representation State Transfer)
- Ανάπτυξη RESTful Application Programming Interfaces (APIs)
- Θέματα ασφάλειας
- Θέματα απόδοσης

Σχεδιασμός διεπαφής χρήστη

- Ευχρηστία, διαδραστικότητα και αποκρισιμότητα
- Σύγχρονες μεθοδολογίες ανάπτυξης διεπαφής χρήστη
  - Πρότυπα σχεδίασης Model-View-Controller, Model-View-ViewModel και Observable
  - Έμφαση σε τεχνολογίες εφαρμογών διαδικτύου (HTML5, CSS3, Javascript)
  - Εφαρμογές μιας σελίδας (Single-page applications)
  - Ασύγχρονες τεχνικές (AJAX, Promises)

Διοίκηση και Διαχείριση Έργου Λογισμικού

- Διοίκηση έργου (γενικά)
  - Εισαγωγή και βασικές έννοιες
  - Εκτίμηση κόστους έργου
- Διοίκηση ομάδας ανάπτυξης λογισμικού
  - Συνήθεις ρόλοι σε μια ομάδα ανάπτυξης λογισμικού
  - Αλληλεπιδράσεις μεταξύ ρόλων
- Καλές διεθνείς πρακτικές

Τεχνικές και Εργαλεία Διοίκησης, Παρακολούθησης και Ελέγχου της Ανάπτυξης Λογισμικού

- Έλεγχος εκδόσεων (Version Control)
  - Έμφαση στο σύστημα Git (το πλέον διαδεδομένο Version Control System)
- Αυτοματισμός διαδικασίας «χτισίματος» λογισμικού (Build automation)
  - Έμφαση στο σύστημα Gradle (χρησιμοποιείται στην ανάπτυξη Android εφαρμογών)

- Σενάρια ελέγχου
  - Unit testing, Regression testing, Functional testing, Integration testing, test coverage
- Συνεχής ολοκλήρωση (Continuous integration)
- Αξιοπιστία λογισμικού
- Η έννοια του «τεχνικού/σχεδιαστικού χρέους» (technical debt).

Διαχείριση συστατικών του λογισμικού

- Συστατικά λογισμικού και αλληλεξαρτήσεις τους (software components and dependencies)
- Αποθήκες συστατικών λογισμικού (Software artifact repositories)
- Διαχείριση εκδόσεων λογισμικού (software releases)
- Μέθοδοι αρίθμησης εκδόσεων (versioning schemes)