Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών
Τμήμα Πληροφορικής & Τηλεπικοινωνιών

Τεχνολογία Λογισμικού

8ο Εξάμηνο 2022-23

Διαχείριση έργων λογισμικού (2 / 2)

Δρ. Κώστας Σαΐδης (saiko@di.uoa.gr)

Τεχνική διαχείριση έργων λογισμικού

- Ο ρόλος του επικεφαλής μηχανικού
- Εργαλεία και διαδικασίες
- Συστατικά λογισμικού (Software artifacts)
- Διαχείριση εκδόσεων λογισμικού

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 του πηγαίου κώδικα (version control)
- Αυτόματο "χτίσιμο" του λογισμικού (build automation)
- Διαχείριση των συστατικών του λογισμικού και των εκδόσεών τους (dependencies & artifacts)
- Εκτέλεση σεναρίων ελέγχου (tests)
- Συνεχής ολοκλήρωση (continuous integration)
- Συντήρηση και εξέλιξη λογισμικού (software maintenance & evolution)
- Εγκατάσταση και παράδοση λογισμικού (software deployment & delivery)

Πριν ξεκινήσει το (όποιο) έργο

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

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 X ή πώς να κάνει το βήμα X)
- Αποφυγή "αποκλεισμένου" κώδικα / βήματος (μόνο ο X ξέρει το Ψ)
- Τουλάχιστον δύο πρόσωπα με την ίδια αρμοδιότητα σε κάθε κομμάτι (bus factor > 1)

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/main/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/FooTest.java`

Class files

`build/classes/test/x/y/z/FooTest.class`

Εργαλεία

- Version control (στο μάθημα το git)
- Code editor / IDE -> διαλέγετε κατά βούληση
- Building (στο μάθημα το gradle, κατά κύριο λόγο)
- Testing (στο μάθημα το spock, κατά κύριο λόγο)

Δεν καλύπτουμε στο μάθημα

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

Συνεχής ολοκλήρωση

- Χτίσιμο του λογισμικού μετά από κάθε αλλαγή στο version control (π.χ. μετά από κάθε git push)
- Μεταγλώττιση του κώδικα, εκτέλεση των σεναρίων ελέγχου
- Αναφορές / ενδείξεις ότι "όλα πήγαν καλά"

Συνεχής εγκατάσταση και παράδοση

- Αν όντως "όλα πήγαν κάλα", αυτόματη εγκατάσταση της νέας έκδοσης σε παραγωγική λειτουργία
- Ιδιαίτερη εφαρμογή όταν έχουμε λογισμικό ως υπηρεσία

Δεν θα τα συζητήσουμε στο μάθημα

Καθημερινές διαδικασίες

- Code checkout, commit, branch, tag, merge, push
- Code review (merge pull requests)
- Build
- Test
- Deploy to CI / Pre-release
- Deploy to staging / User acceptance tests
- Release / Deploy to production

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 εκδόσεων του κώδικα ταυτόχρονα
- Ανάκτηση συγκεκριμένης παρελθούσας κατάστασης

Στο μάθημα

Version control

- Φροντιστήριο για το git

Software build automation

Ζητούμενα

- Αυτόματη διαχείριση εξαρτήσεων (dependencies)
- Μεταγλώττιση κώδικα (compilation)
- Εκτέλεση σεναρίων ελέγχου (testing)
- Παραγωγή των software artifacts (assemble)
- Απόθεση/δημοσίευσή τους σε κάποια αποθήκη (software release / publication)

Και

- Συνεχής ολοκλήρωση του λογισμικού (Continuous integration)

Στο μάθημα

- Φροντιστήριο για το Gradle

Software testing

Ζητούμενα

- Αυτοματοποιημένος έλεγχος και επικύρωση του υπό ανάπτυξη λογισμικού, δηλαδή, αν η ομάδα ανάπτυξης
  - Παράγει το λογισμικό σωστά;
  - Παράγει το σωστό λογισμικό;
- Προετοιμασία για τους ελέγχους αποδοχής (user acceptance tests)

Στο μάθημα

- Φροντιστήριο για το Spock

Ο βασικός "τεχνικός" στόχος κάθε έργου λογισμικού είναι η παραγωγή ενός ή περισσότερων software artifacts

Software artifacts

- Αυτοτελή αρχεία έτοιμα προς εκτέλεση ή
- Μερικώς αυτοτελή αρχεία προς ενσωμάτωση σε άλλες εφαρμογές (βιβλιοθήκες)
- Δομή/περιεχόμενα ανάλογα με τη γλώσσα προγραμματισμού, το λειτουργικό σύστημα, την εφαρμογή, κτλ.

Στη Java κοινότητα

- Software artifact = Jar αρχείο (συνήθως)
- Jar αρχείο = Zip αρχείο
- Περιέχει .class αρχεία (JVM κλάσεις) και -ενδεχομένως- metadata (manifests), resources (images), αρχεία ρυθμίσεων κ.ο.κ.

Στην πράξη

Κάθε ξεχωριστή προγραμματιστική/τεχνολογική κοινότητα συνήθως:

- έχει ξεχωριστούς μορφότυπους artifacts
- έχει ξεχωριστά εργαλεία διαχείρισής τους
- έχει -όπως λέγεται- ξεχωριστό technology stack

Ας το δούμε αντίστροφα

Τι είναι ένα software artifact;

- Μπορεί να είναι οτιδήποτε:
  - Application, Library, Component, Server, Client
- Μπορεί να χρησιμοποιείται για ένα μόνο σκοπό (π.χ. standalone app) ή να είναι επαναχρησιμοποιήσιμο για πολλούς σκοπούς (π.χ. library)
- Ας πούμε ότι, γενικά, είναι ένα συστατικό λογισμικού

Διαχείριση συστατικών λογισμικού

Συστατικά λογισμικού

- Επαναχρησιμοποιήσιμα τμήματα λογισμικού (που διατίθενται ως ξεχωριστά software artifacts)
- Αποθήκες συστατικών λογισμικού (software artifact repositories)
- Διαχείριση εκδόσεων συστατικών (software releases, artifact versioning)

Εξαρτήσεις λογισμικού (software dependencies)

- Compile-time dependencis ("static" linking)
- Runtime dependencies ("dynamic" linking)

Μεταβατικές εξαρτήσεις (transitive dependencies)

Οι εξαρτήσεις των εξαρτήσεων (τα συστατικά δεν είναι πάντα αυτοτελή, μπορεί να εξαρτώνται από άλλα συστατικά / artifacts για τη λειτουργία τους)

- `Project -> Lib1, Lib2`
- `Lib1 -> Lib11`
- `Lib2 -> Lib21, Lib22`
- `Lib21 -> Lib211`

Αποθήκες συστατικών λογισμικού (software artifact repositories)

- Τήρηση των software artifacts (files)
- Σε πολλές εκδόσεις
- Τήρηση μεταδεδομένων για τις αλληλο-εξαρτήσεις τους
- Δημόσιες ή ιδιωτικές αποθήκες

Παραδείγματα δημόσιων αποθηκών της Java κοινότητας

https://search.maven.org/

https://bintray.com/bintray/jcenter

https://plugins.gradle.org/

Ιδιωτικές αποθήκες

- Για μεγάλες ομάδες
- Για σύνθετο λογισμικό
- Φιλοξενία της αποθήκης στο εσωτερικό δίκτυο του οργανισμού

Δημοσίευση συστατικού

- Ανέβασμα του συστατικού σε κάποια αποθήκη (artifact publication)
- Αυτοματοποιημένη διαδικασία μέσω του build εργαλείου
- Παράδειγμα `gradle publish` η `mvn release`

Για παράδειγμα

- Εκτέλεση όλων των προαπαιτούμενων για την παραγωγή του jar αρχείου (download dependencies, compile code, run tests, assemble files)
- Παραγωγή του jar αρχείου με κάποια σύμβαση για το όνομα και την έκδοσή του (π.χ. project-name-1.2.jar)
- Ανέβασμα του αρχείου στην αποθήκη
  - σε κάποια συγκεκριμένη -κατά σύμβαση- θέση (π.χ. `/gr/ntua/softeng17b/foo/1.2/project-name-1.2.jar`)
  - μαζί με πληροφορίες/μεταδεδομένα για τις εξαρτήσεις του (π.χ. maven pom, ivy file)

Διαχείριση εκδόσεων λογισμικού

- Versioning
- Releasing

Τυποποίηση εκδόσεων λογισμικού

- Δεν υπάρχει ομοιομορφία και συνέπεια στην ανάθεση εκδόσεων
- Διαφορετικά σχήματα
- Με βάση την ημερομηνία, με βάση κάποια σύμβαση, με βάση εμπορικούς λόγους, κλπ.

Παράδειγμα

[major].[minor].[revision].[build]

- major: Σημαντική αλλαγή στο λογισμικό (major release)
- minor: Προσθήκη ή βελτίωση στο λογισμικό (minor release)
- revision: Patch, διόρθωση bug, επίλυση προβλήματος ασφαλείας, κτλ. (maintenance release)
- build: Αυτόματη αρίθμηση του build (π.χ. αύξων αριθμός, commit id, κτλ.) (internal release)

Semantic Versioning

[major].[minor].[patch]

- major: Breaking change
- minor: Add new backwards compatible functionality
- patch: Apply backwards compatible bug fixes
- Κύρια έννοια: Public API
- http://semver.org/

Επίτευξη του release

Επιμέρους στάδια και ενδιάμεσες εκδόσεις

- Pre-release (internal release)
- Early Access (EA)
  - Alpha
  - Beta
  - Release candidate
- Release (General Availability, GA)

Αυτοματοποίηση μέσω του build εργαλείου